A

MINI PROJECT REPORT ON

MITIGATING DDOS ATTACK IN IOT NETWORK ENVIRONMENT

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

INTERNET OF THINGS - IOT

By

SINGARA CHAITRAN

Roll No : 20Q91A6941

Under the guidance of

Mr.Venkateswarlu
Assistant Professor, Dept. of IOT

DEPARTMENT OF INTERNET OF THINGS - IOT

MALLA REDDY COLLEGE OF ENGINEERING

(Approved by AICTE-Permanently Affiliated to JNTU-Hyderabad)

Recognized section 2(f) & 12(B) of UGC New Delhi ISO 9001:2015certified Institution

Maisammaguda, Dhulapally (Post via Kompally), Secunderabad- 500100

2023 - 2024

MALLA REDDY COLLEGE OF ENGINEERING

(Approved by AICTE-Permanently Affiliated to JNTU-Hyderabad)

Recognized section 2(f) & 12(B) of UGC New Delhi ISO 9001:2015certified Institution

Maisammaguda, Dhulapally (Post via Kompally), Secunderabad- 500100

DEPARTMENT OF INTERNET OF THINGS - IOT

CERTIFICATE

This is to certify that the Mini Project report on “ MITIGATING DDOS ATTACK IN IOT NETWORK ENVIRONMENT” is successfully done by the following students of Department of Internet of things - IOT of our college in partial fulfilment of the requirement for the award of B.Tech degree in the year 2023-2024. The results embodied in this report have not been submitted to any other University for the award of any diploma or degree.

SINGARA CHAITRAN	**:**	**20Q91A6941**

INTERNAL GUIDE	**HOD**	**PRINCIPAL**
Mr. Sumit Kumar	**Mr. E. Lingappa**	**Dr. Ashok Kumar**
Assistant Professor	**Professor**	**B.Tech.,M.Tech.,Ph.D.**

Submitted for the viva voice examination held on: ________________________________

Internal Examiner **External Examiner**

DECLARATION

I Singara Chaitran with Regd.no. 20Q91A6941 are here by declaring that the mini project report entitled **"MITIGATING DDOS ATTACK IN IOT NETWORK ENVIRONMENT"** has done by usunder the guidance of **Mr. Venkateswarlu** Assistant Professor, Department of CSE is submitted in the partial fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY in INTERNET OF THINGS-IOT.**

The Results embeded in this project report have not been submitted to any other University or institute for the award of any degree or diploma.

Signature of the Candidate

Singara Chaitran **20Q91A6941**

DATE:

PLACE: Maisammaguda

ACKNOWLEDGEMENT

First and foremost, we would like to express our immense gratitude towards our institution Malla Reddy College of Engineering, which helped us to attain profound technical skills in the field of Computer Science & Engineering, there by fulfilling our most cherished goal.

We are pleased to thank **Sri Ch. Malla Reddy** , our Founder, Chairman **MRGI**, **Sri Ch. Mahender Reddy**, Secretary, **MRGI** for providing this opportunity and support throughout the course.

It gives us immense pleasure to acknowledge the perennial inspiration of **Dr. M. Ashok** our beloved principal for his kind co-operation and encouragement in bringingout this task.

We would like to thank **Dr. T. V. Reddy** our vice principal, **Dr. E. Lingappa** HOD,IT Department for their inspiration adroit guidance and constructive criticism for successful completion of our degree.

We would like to thank **Mr. Venkateswarlu** Assistant Professor our internal guide, for his valuable suggestions and guidance during the exhibition and completion of this project.

Finally, we avail this opportunity to express our deep gratitude to all staff who have contribute their valuable assistance and support making our project success.

Singara Chaitran **20Q91A6941**

ABSTRACT

The recent proliferation of Internet of Things (IoT) is paving the way for the emergence of smart cities, where billions of IoT devices are interconnected to provide novel pervasive services and automate our daily life tasks (e.g., smart healthcare, smart home). However, as the number of insecure IoT devices continues to grow at a rapid rate, the impact of Distributed Denial-of-Service (DDoS) attacks is growing rapidly. With the advent of IoT botnets such as Mirai, the view towards IoT has changed from enabler of smart cities into a powerful amplifying tool for cyberattacks. This motivates the development of new techniques to provide flexibility and efficiency of decision making on the attack collaboration in a software defined networks (SDN) context. The new emerging technologies, such as SDN and blockchain, give rise to new opportunities for secure, low-cost, flexible and efficient DDoS attacks collaboration for the IoT environment. In this paper, we propose Co-IoT, a blockchain-based framework for collaborative DDoS attack mitigation; it uses smart contracts (i.e., Ethereum's smart contracts) in order to facilitate the attack collaboration among SDN-based domains and transfer attack information's in a secure, efficient and decentralized manner. Co-IoT's implementation is deployed on the Ethereum official test network Ropsten [1]. The experimental results confirm that Co-IoT achieves flexibility, efficiency, security, cost effectiveness making it a promising scheme to mitigate DDoS attacks in large scale

KEYWORDS:

- Internet of Things (IoT)
- Smart Cities
- Distributed Denial-of-Service (DDoS) attacks
- IoT botnets, Mirai
- Software Defined Networks (SDN)
- Blockchain, Co-IoT
- Smart Contracts
- Ethereum
- Attack Collaboration

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SCREENSHOTS

CHAPTER – 1
INTRODUCTION

The Internet of Things (IoT) represents a rapidly expanding network of interconnected devices, each capable of generating and exchanging data. While this interconnectedness offers significant potential for innovation and efficiency, it also introduces new security challenges. One of the most pressing concerns is the increasing prevalence of Distributed Denial-of-Service (DDoS) attacks targeting IoT devices. These attacks can overwhelm the network with traffic, rendering devices and services unavailable to legitimate users. To effectively mitigate DDoS threats in IoT environments, a comprehensive security strategy is essential. This strategy should encompass proactive measures such as device hardening, network segmentation, and intrusion detection systems, as well as reactive measures such as traffic filtering and rate limiting.

The vast and rapidly growing Internet of Things (IoT) network presents a new frontier for cyberattacks, particularly in the form of Distributed Denial-of-Service (DDoS) attacks. DDoS attacks overwhelm a targeted system with excessive traffic, rendering it unavailable to legitimate users. In the context of IoT, the sheer volume of interconnected devices, often with limited security measures, makes them prime targets for DDoS attacks. These attacks can disrupt critical infrastructure, disrupt business operations, and even endanger public safety. Mitigating DDoS attacks in the IoT environment requires a comprehensive approach that encompasses network security, device hardening, and proactive threat detection.

In the ever-expanding landscape of the Internet of Things (IoT), cybersecurity threats, particularly Distributed Denial-of-Service (DDoS) attacks, pose a significant challenge. Due to the inherent security vulnerabilities of IoT devices, these attacks can easily overwhelm them, rendering them unavailable to legitimate users. This necessitates the implementation of effective mitigation strategies to protect IoT networks from DDoS attacks.

The Internet of Things (IoT) has revolutionized the way we interact with the world around us, connecting billions of devices and enabling a vast array of applications. However, the proliferation of IoT devices has also expanded the attack surface for cybercriminals, making IoT networks a prime target for distributed denial-of-service (DDoS) attacks. DDoS attacks overwhelm a target network with a flood of traffic, rendering it inaccessible to legitimate users. In the context of IoT, the vast number of vulnerable devices makes it easier for attackers to amplify their attacks and cause significant disruptions.

To mitigate DDoS attacks in IoT network environments, a comprehensive approach is required, encompassing both preventive measures and real-time detection and mitigation techniques. Preventive measures include securing IoT devices with strong passwords and firmware updates, implementing network segmentation to isolate critical systems, and educating users about cyber security best practices. Real-time detection and mitigation techniques involve monitoring network traffic for anomalies, identifying and blocking malicious traffic, and utilizing specialized DDoS mitigation services.

One effective DDoS mitigation strategy is to employ rate limiting, which restricts the amount of traffic that can be sent to a particular destination. This can help to prevent attackers from overwhelming the target system with a flood of requests.

Additionally, blackholing malicious trafficcan effectively divert attack traffic away from the target system. In some cases, it may be necessary to temporarily disable vulnerable devices to prevent them from being used in an attack.

Machine learning algorithms can be used to analyze network traffic patterns and identify potential DDoS attacks. By analyzing factors such as traffic volume, packet size distribution, and origin patterns, machine learning models can effectively distinguish between normal traffic and malicious traffic patterns. This information can then be used to trigger mitigation measures, such as rate limiting or blackholing, to stop the attack.

In conclusion, mitigating DDoS attacks in IoT network environments requires a multifaceted approach that combines preventive measures, real-time detection and mitigation techniques, and the use of advanced technologies such as machine learning. By implementing these strategies, organizations can protect their IoT networks from disruption and ensure the continued availability of critical services.

CHAPTER-2

LITERATURE SURVEY

On networking of Internet of Things: Explorations and challenges

AUTHOR:

H. Ma, L. Liu, A. Zhou, and D. ZhaoA

ABSTRACT:

Internet of Things (IoT), as the trend of future networks, begins to be used in many aspects of daily life. It is of great significance to recognize the networking problem behind developing IoT. In this paper, we first analyze and point out the key problem of IoT from the perspective of networking: how to interconnect large-scale heterogeneous network elements and exchange data efficiently. Combining our on-going works, we present some research progresses on three main aspects: 1) the basic model of IoT architecture; 2) the internetworking model; and 3) the sensor-networking mode. Finally, we discuss two remaining challenges in this area

TITLE:

Risks of automation: A cautionary total-system per- spective of our cyberfuture, AUTHORS:

P. G. Neumann

ABSTRACT:

MANY COMPUTER -RELATED RISKS discussed in past Inside Risks columns are still present today. These risks (and new ones) are likely to intensify even further as systems provide extensive automated or semi-automated operation. Significantly greater total-system trustworthiness will be required, encompassing better hardware, system software, and applications that are able to tolerate human limitations and environmental factors. Risks will continue to result from inadequate reliability, security, and privacy, as well as gullibility and general inability of users to cope with complex technology. We repeatedly discover unexpected risks resulting from lashing subsystems together (for example, see Beurdouche 2), because of unexpected system behavior. Many advances in research, system development, and user friendliness are urgent- Ly needed literature survey on mitigating Distributed Denial of Service (DDoS) attacks in Internet of Things (IoT) network environments reveals a growing body of research focused on developing effective strategies to safeguard connected devices from this pervasive threat.

Researchers have emphasized the need for a proactive defense approach, highlighting the significance of intrusion detection systems, firewalls, and traffic filtering tools tailored specifically for the unique characteristics of IoT devices. These defense mechanisms play a pivotal role in identifying and thwarting malicious traffic before it reaches vulnerable devices, mitigating the impact of potential DDoS attacks.

Traffic monitoring and anomaly detection have emerged as critical components in the literature, with studies emphasizing the importance of regularly scrutinizing network traffic patterns. Leveraging machine learning algorithms, researchers aim to enhance the ability to swiftly identify and respond to abnormal activities, contributing to more robust DDoS mitigation strategies.

Authentication and authorization processes for IoT devices are extensively explored in the literature. Establishing stringent device access controls and ensuring that only authenticated and authorized devices can communicate within the network are recognized as fundamental measures to prevent unauthorized access and participation in DDoS attacks.

Studies also underscore the importance of scalable bandwidth and redundancy in mitigating DDoS attacks. By investing in scalable infrastructure and distributing network traffic across multiple servers, organizations can absorb the impact of volumetric attacks, ensuring the continuous functionality of IoT devices even under duress.

Collaboration among stakeholders is a recurring theme, emphasizing the need for information sharing and collective efforts in addressing DDoS threats. The literature encourages collaboration between IoT device manufacturers, network operators, and security experts to stay ahead of evolving attack techniques and vulnerabilities.

Regular updates and patch management for IoT devices are consistently highlighted. The literature stresses the necessity of keeping devices and associated software up-to-date to address known vulnerabilities, reducing the risk of compromised devices becoming unwitting participants in DDoS attacks.

Incident response planning is a prominent focus in the literature, with researchers advocating for well-defined plans outlining steps to be taken in the event of a DDoS attack. These plans encompass communication strategies, containment procedures, and recovery efforts, ensuring a structured and efficient response to mitigate the impact of attacks.

CHAPTER-3

EXISTING SYSTEM

Blockchain technology (e.g., Bitcoin [10] and Ethereum [11]) is considered as a new technology to secure and store information in a decentralized manner without any trusted tier; it has proven its success and effectiveness in multiple application domains (e.g., Healthcare [12], financial field [13]) to achieve high level of security and transparency. One such application domain is the IoT [14] due to its decentralized structure and the resource-constraints of its devices. Using blockchain technology, which ensures trust between nodes in a trustless environment, can be an efficient approach to facilitate the future underlying infrastructure for IoT. Security and privacy for IoT have been an active research topic for decades and several DDoS collaboration mitigation schemes have been proposed. In the following, we present the most prominent schemes as well as their security issues.

EXISTING SYSTEM DISADVANTAGES:

1. Less Accuracy
2. Low Efficiency

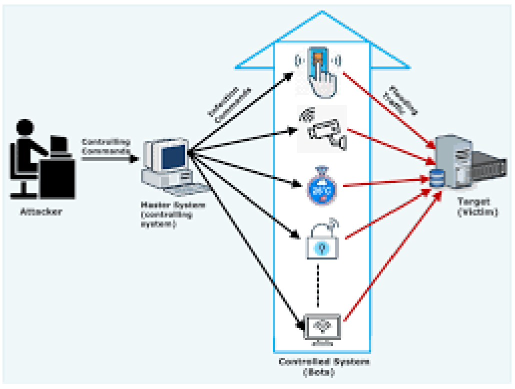
CHAPTER-4
PROPOSED SYSTEM

We implemented Co-IoT using both private (Ganache simulator [28]) and public blockchain (Ethereum official test network Ropsten). Once the collaboration contract is deployed, it can be self-executed without any human intervention. The process of deployment is elaborated using truffle framework [29] (see Fig. 4). First, we have coded the contract using the high-level language programming solidity [30]. Then, we compiled the contract into Ethereum Virtual Machine (EVM) byte code; once the collaboration contract gets compiled, it generates EVM byte code as well as Application Binary Interface (ABI). Afterwards, we deployed the collaboration contract to the blockchain. Initially, we have deployed the collaboration contract using Ganache, a private blockchain simulator to test Ethereum's smart contract in a fast way. Then, we have deployed the smart contract on Ethereum official test network Ropsten. Fig. 5 shows the smart contract lifecycle. Once deployed, the smart contract can be invoked using its address and the ABI definition. If needed, the contract can be deleted

PROPOSED SYSTEM ADVANTAGES:

1. High Accuracy
2. High Efficiency

SYSTEM ARCHITECTURE:



CHAPTER-5

HARDWARE & SOFTWARE REQUIREMENTS

HARD REQUIRMENTS :

• **System**	**:**	**i3 or above.**
• **Ram**	**:**	**4 GB.**
• **Hard Disk**	**:**	**40 GB**

SOFTWARE REQUIRMENTS :

• **Operating system**	**:**	**Windows8 or Above.**
• **Coding Language**	**:**	**python**

CHAPTER-6

SYSTEM STUDY FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified.

Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources.

This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER-7

SYSTEM DESIGN

UML DIAGRAMS:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

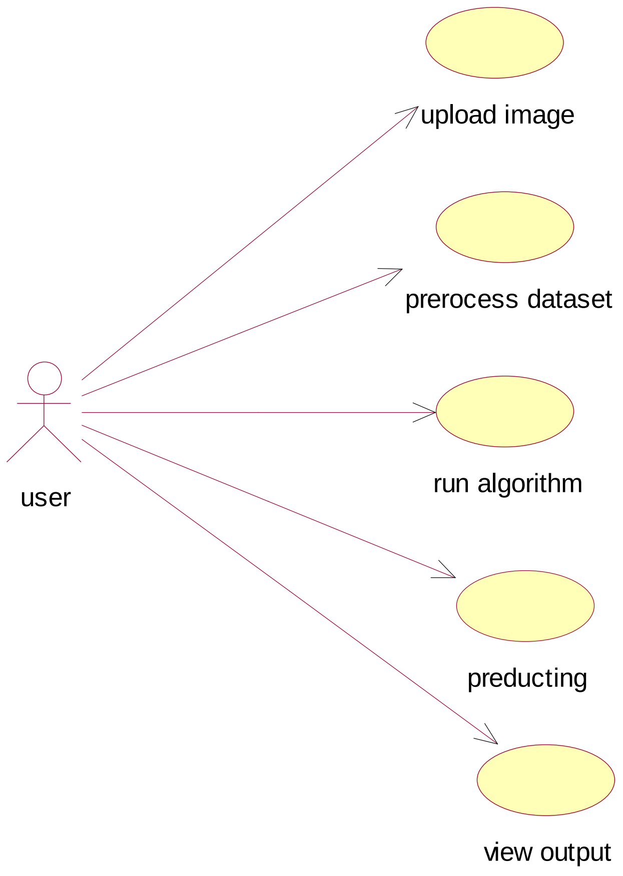
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

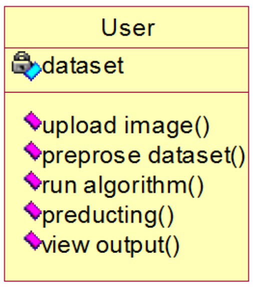
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



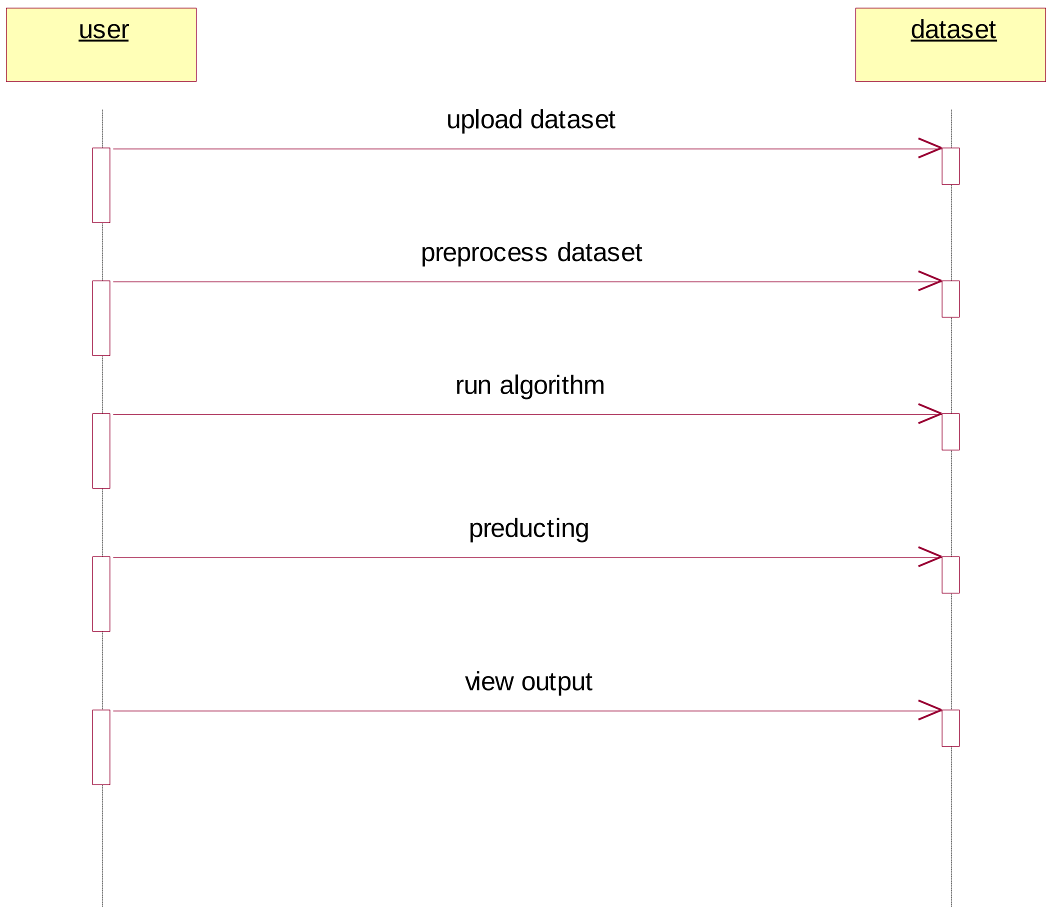
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



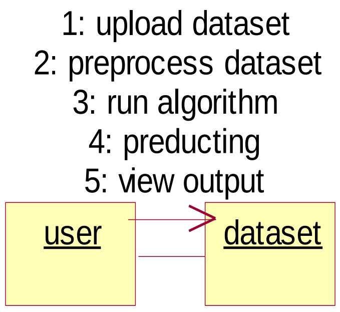
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



COLLABRATION DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



IMPLEMENTATION:

MODULES:

Upload image

Preprocess dataset

Run algorithm

Predicting

View output

SOFTWARE ENVIRONMENT

What is Python :

- Below are some facts about Python.
- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

Let’s see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

6. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

7. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

8. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

9. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

10. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages:

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages Of Python:

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

1.Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2.Weak in Mobile Computing and Browsers:

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**. The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don’t need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck.
While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC(Java Data Base Connectivity)** and **ODBC (Open Data Base Connectivity)**, Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python:

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica).

The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system.

In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI).

I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it.

"Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing.

I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked.

I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What Is Machine Learning ?

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning: -

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning: -

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Application of Machine Learning:

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How To Start Learning Machine Learning?

Arthur Samuel coined the term **“Machine Learning”** in 1959 and defined it as a **“Field of study that gives computers the capability to learn without being explicitly programmed”.** And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year. But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer.

Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available.

But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc.

Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label) –** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training –** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Type of Machine Learning

- **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behavior.
- Text Vs. Data Instead Of Unicode Vs. 8-bitrint
- Print is now a function

Purpose:-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

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 runtime 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. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors.

This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project: -

TensorFlow

TensorFlow is a free and open-source software library for dataflow 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 powerful 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 data-types 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. It had 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

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 runtime 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. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac:

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How To Install Python On Windows And Mac:

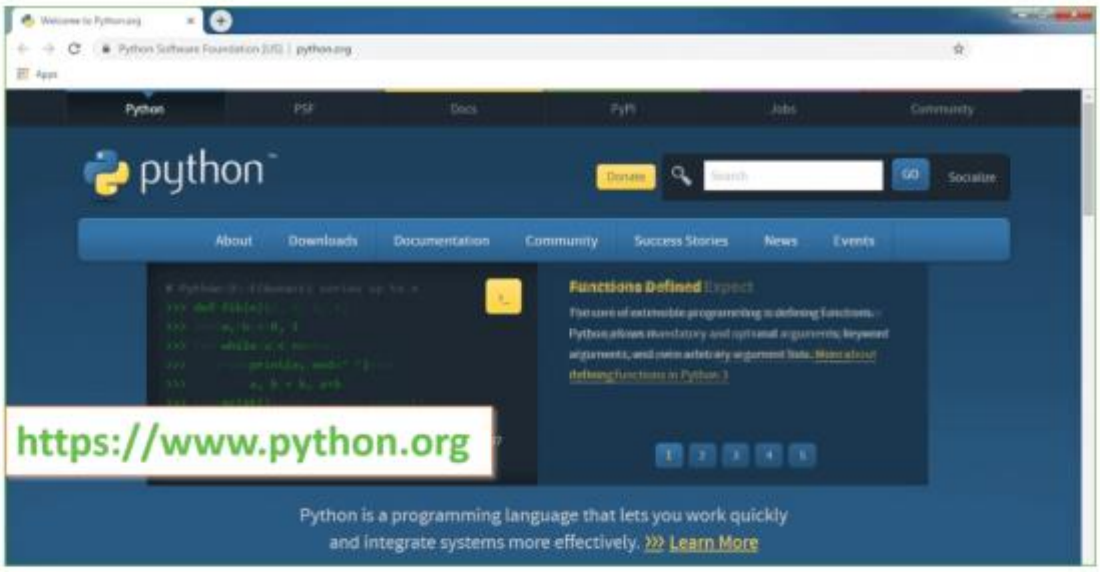
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

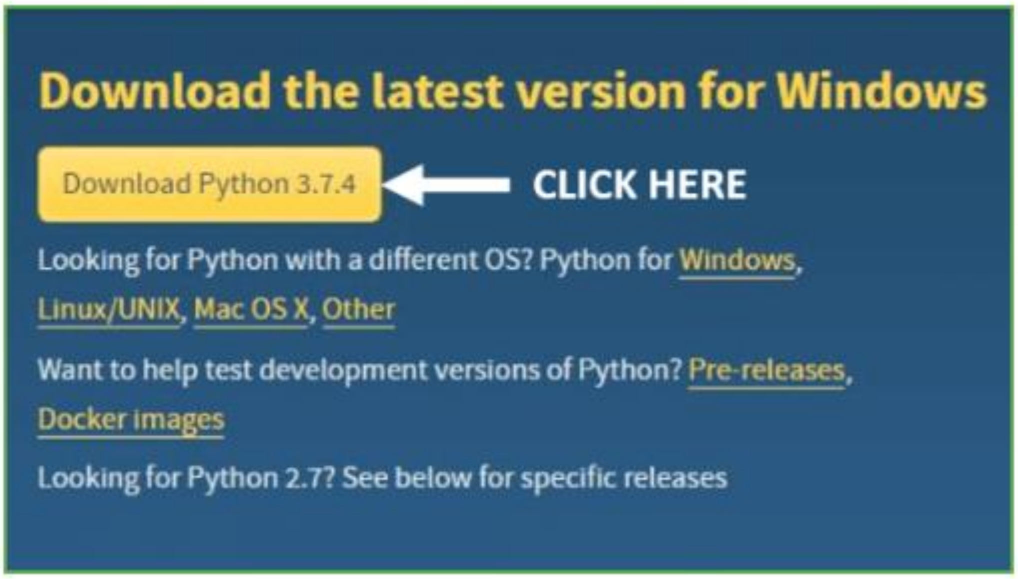
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Download the latest version for Windows

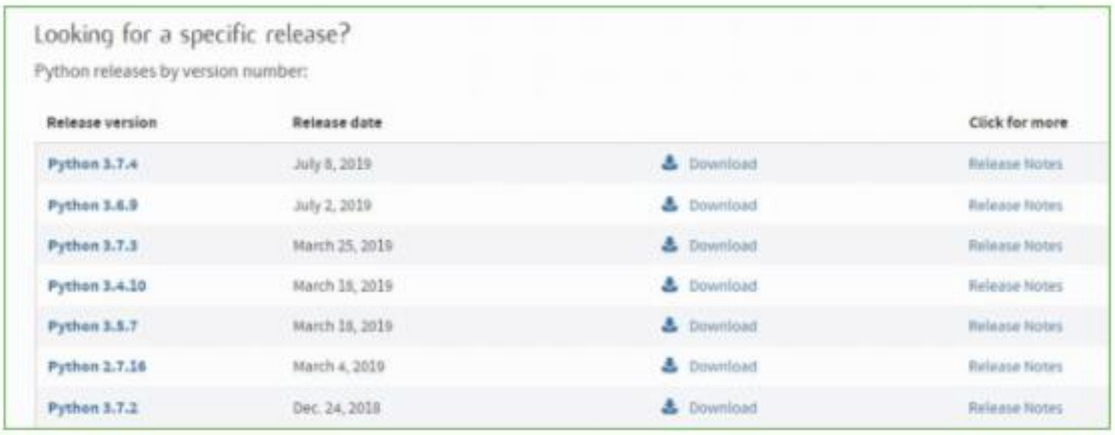
Download Python 3.7.4 ← CLICK HERE

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



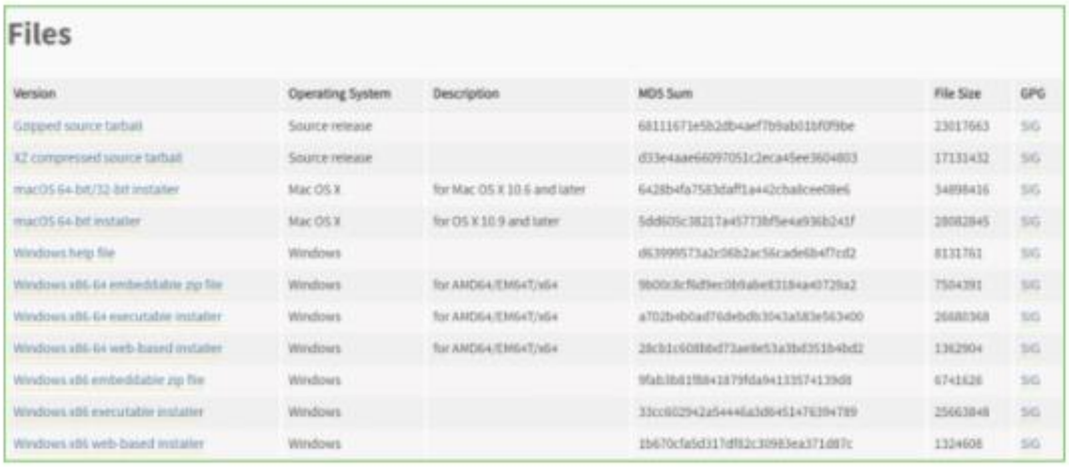
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cbalcee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

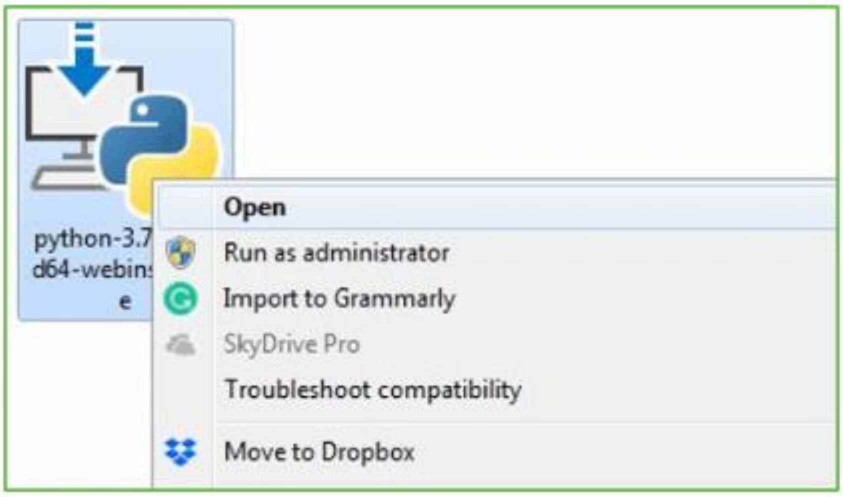
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

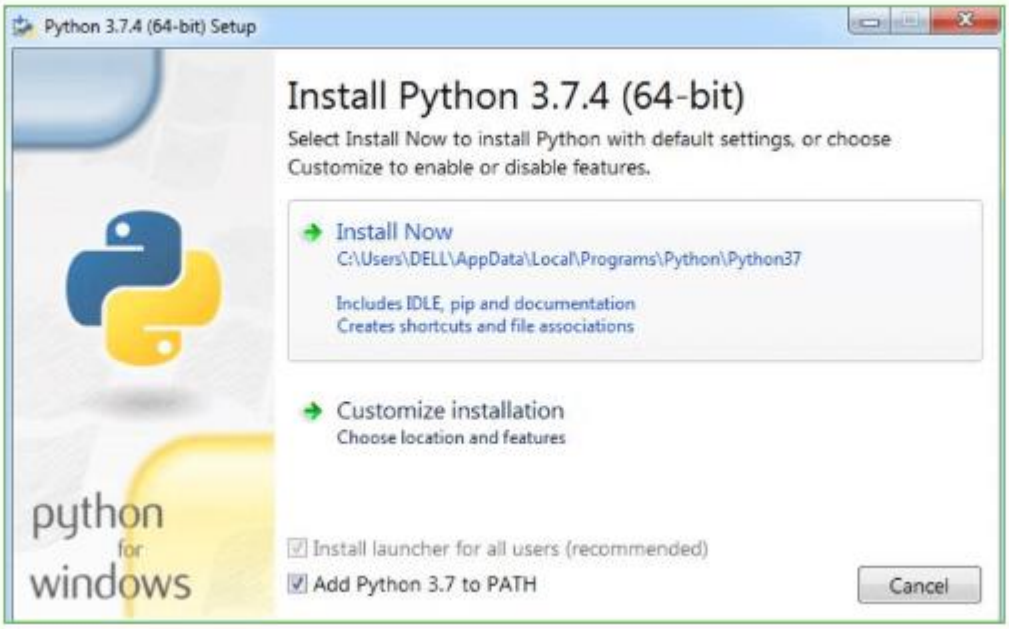
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

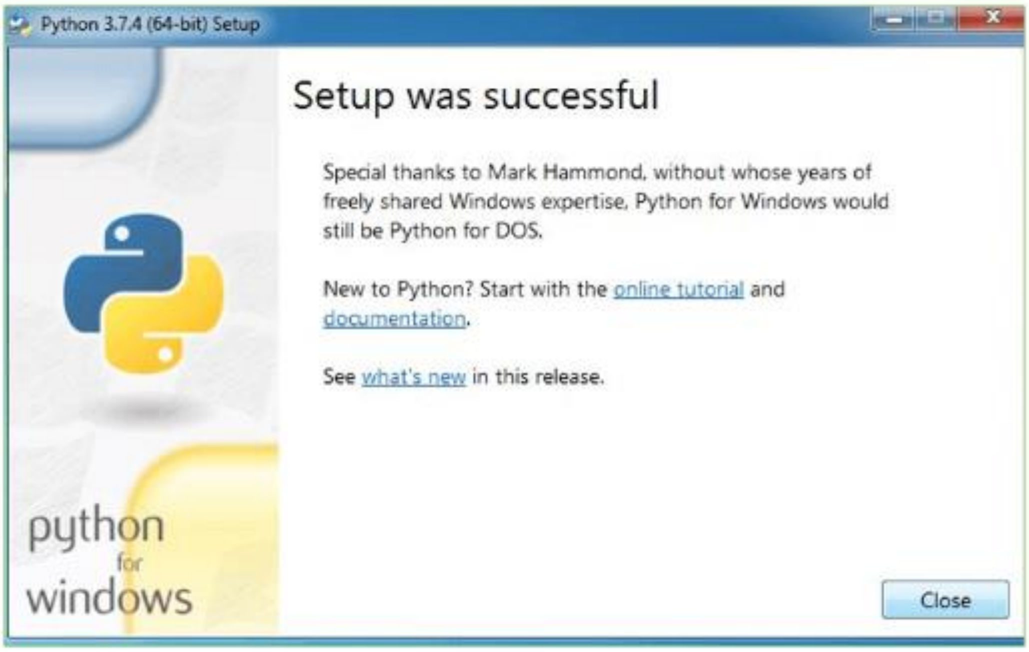
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

[✓] Install launcher for all users (recommended)
[✓] Add Python 3.7 to PATH

Cancel

Step 3: Click on Install NOW After the installation is successful. Click on Close.


Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

Close

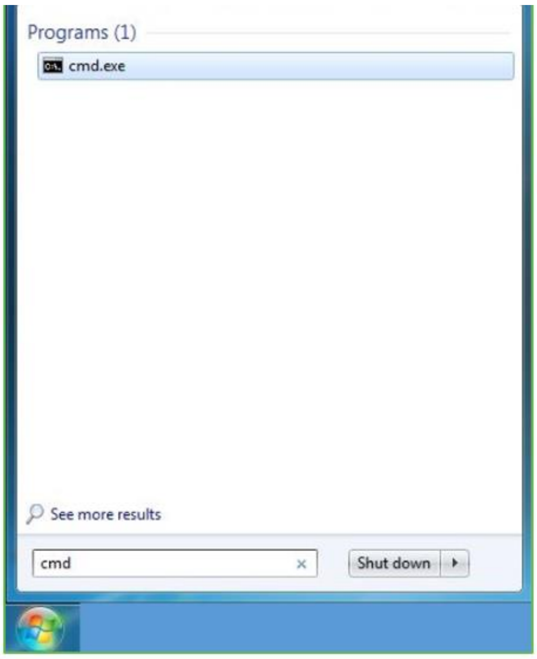
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

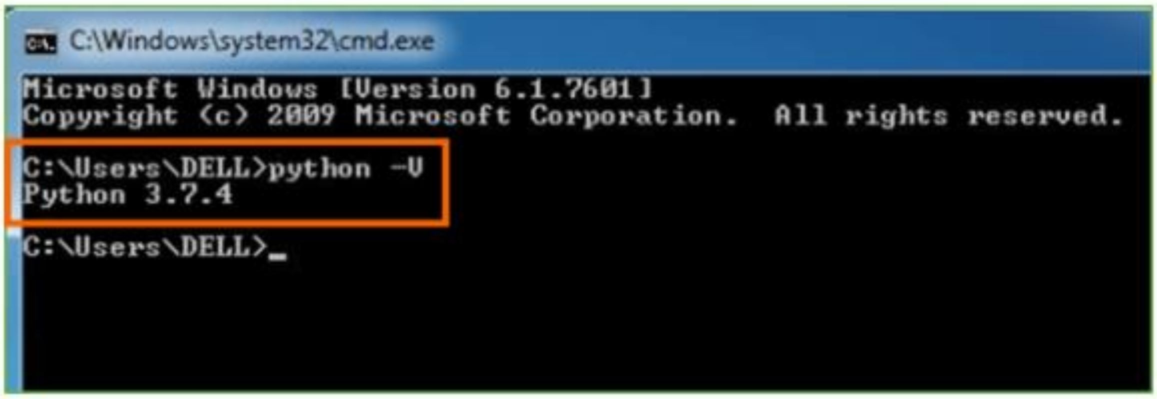
Step 1: Click on Start

Step 2: In the Windows Run Command, type "cmd".



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

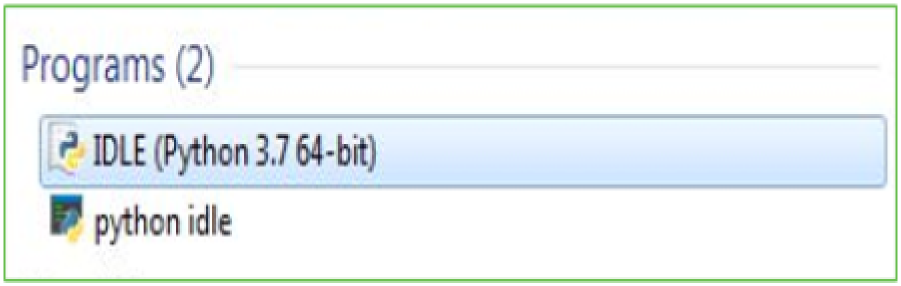
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle".



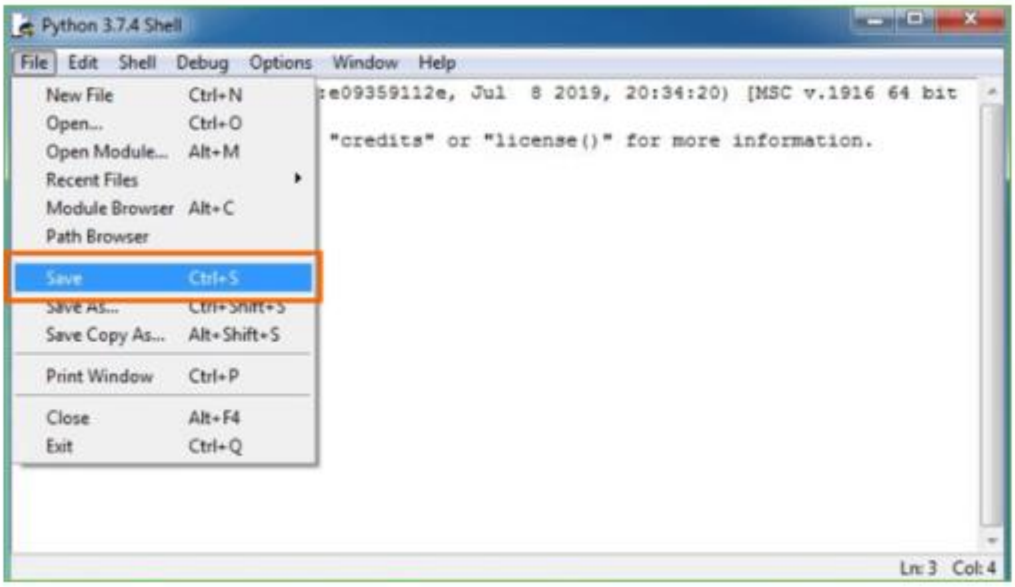
Programs (2)

IDLE (Python 3.7 64-bit)

python idle

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit

"credits" or "license()" for more information.
```

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

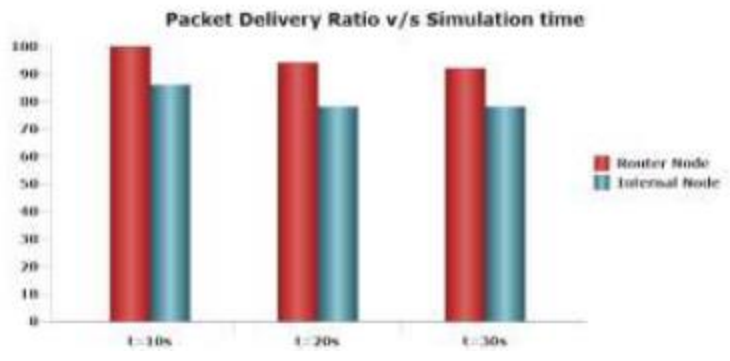
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

WORKING METHODOLOGY

The propose framework is simulated in Cooja and fig. 3 shows the simulation environment. Sky motes are used for border router and RPL is used for DODAG formation. UDP clients and servers are connected inside the internal network to represent different internal devices. Testing of the framework against flooding DDoS attacks has been performed. Packet Delivery Ratio, Response Time, True Positives and False Negatives are considered as the testing parameters based on comparison with the existing approaches. It helps us to compare the results with the existing approaches as similar information about those approaches is published.

In specific cases, we have got 100% delivery rate for border router and 86% for the internal node. However, we expect some more reduction in the delivery ratio if we go for real-time hardware implementation, which is planned as a future work. Fig. 4 and 5 shows results of our proposed solution in terms of efficiency in successful communication. We obtained a better packet delivery ratio to the internal nodes and failed communications are comparatively lesser in our work. Further, the complexity of our proposed algorithm is lesser compared to the existing approaches that we have mentioned in the related works. It helps to achieve the improvements in the communication parameters.



Mitigating Distributed Denial of Service (DDoS) attacks in an Internet of Things (IoT) network environment requires a comprehensive and adaptive working methodology. Here's a step-by-step guide on how to approach the mitigation of DDoS attacks in IoT networks:

1. Risk Assessment and Vulnerability Analysis:

Begin by conducting a thorough risk assessment to identify potential vulnerabilities in the IoT network. Assess the impact of DDoS attacks on critical assets and prioritize mitigation efforts based on the identified risks.

2. Define Security Policies:

Establish clear security policies that govern IoT device access, communication, and behavior. Define rules for authentication, authorization, and encryption to ensure that only legitimate devices can access the network.

3. Implement Network Segmentation:

Segment the IoT network to isolate critical devices and services. This helps contain the impact of a DDoS attack by preventing lateral movement and limiting the exposure of vulnerable devices.

4. Deploy Traffic Monitoring and Anomaly Detection Tools:

Implement traffic monitoring tools to continuously analyze network traffic. Utilize anomaly detection mechanisms that can identify unusual patterns or spikes in traffic, which may indicate a DDoS attack.

5. Device Profiling and Behavior Analysis:

Develop profiles for normal behavior of IoT devices. Use behavioral analysis techniques to detect deviations from these profiles, signaling potential security threats. This proactive approach allows for early detection of compromised devices participating in DDoS attacks.

6. Distributed Firewalls and Intrusion Prevention Systems (IPS):

Install distributed firewalls and IPS strategically within the IoT network. These devices can filter malicious traffic, blocking DDoS attacks at various entry points and preventing them from reaching critical infrastructure.

7. Rate Limiting and Traffic Shaping:

Implement rate limiting and traffic shaping mechanisms to control the volume of incoming requests. By setting thresholds for acceptable traffic, organizations can mitigate the impact of volumetric DDoS attacks.

8. Cloud-Based DDoS Protection Services:

Consider leveraging cloud-based DDoS protection services. These services can absorb and filter out DDoS traffic before it reaches the organization's network, providing an additional layer of defense.

9. Device Authentication and Access Controls:

Strengthen device authentication processes and enforce strict access controls. Only authenticated and authorized devices should be allowed to communicate with the network, reducing the likelihood of compromised devices participating in DDoS attacks.

10. Incident Response Planning:

Develop and regularly update an incident response plan specifically tailored for DDoS attacks. Clearly outline roles, responsibilities, communication protocols, and steps for containing and mitigating the impact of an ongoing attack.

11. Regular Security Audits and Penetration Testing:

Conduct regular security audits and penetration testing to identify and address new vulnerabilities. This proactive approach helps organizations stay ahead of potential threats and fine-tune their DDoS mitigation strategies.

12. Collaboration and Information Sharing:

Foster collaboration with industry peers, threat intelligence providers, and security communities. Share information about emerging DDoS threats and vulnerabilities to enhance collective resilience.

By following this working methodology, organizations can enhance their ability to detect, prevent, and mitigate DDoS attacks in IoT network environments. The key is to adopt a proactive and adaptive approach to stay resilient against evolving threat.

CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import 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
global filename
global X,Y
global dataset
global main
global text
global accuracy, precision, recall, fscor
global X_train, X_test, y_train, y_test
global classifier
global label_encoder, labels, columns, types, pca
main = tkinter.Tk()
main.title("Mitigating DDOS Attack In IOT Network Environment") #designing main screen
main.geometry("1300x1200")
def getLabel(name):
 label = -1
 for i in range(len(labels)):
 if name == labels[i]:
 label = i
 break
 return label
#fucntion to upload dataset
def upload Dataset():
 global filename, dataset, labels
 text.delete('1.0', END)
 filename = filedialog.askdirectory(initialdir=".")
 text.insert(END,filename+" loaded\n\n")
 df1 = pd.read_csv(filename+"/DrDOS_DNS.csv")
 df2 = pd.read_csv(filename+"/DrDOS_LDAP.csv")
 df3 = pd.read_csv(filename+"/DrDOS_MSSQL.csv")
 df4 = pd.read_csv(filename+"/DrDOS_NTP.csv")
 df5 = pd.read_csv(filename+"/DrDOS_NetBIOS.csv")
 df6 = pd.read_csv(filename+"/DrDOS_SNMP.csv")
 df7 = pd.read_csv(filename+"/DrDOS_SSDP.csv")
 df8 = pd.read_csv(filename+"/DrDOS_UDP.csv")
 df9 = pd.read_csv(filename+"/Syn.csv")
 df10 = pd.read_csv(filename+"/UDP_LAG.csv")
 dataset = [df1, df2, df3, df4, df5, df6, df7, df8, df9, df10]
 dataset = pd.concat(dataset)
 labels = np.unique(dataset['Label']).tolist()
 print(labels)
 text.insert(END,str(dataset.head()))
 text.update_idletasks()
 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()
def preprocessDataset():
 global dataset, label_encoder, X, Y, columns, types, pca
 global X_train, X_test, y_train, y_test
 text.delete('1.0', END)
 label_encoder = []
 columns = dataset.columns
 types = dataset.dtypes.values
 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])
 dataset.fillna(0, inplace = True)
 Y = dataset['Label'].ravel()
 temp = []
 for i in range(len(Y)):
 temp.append(getLabel(Y[i]))
 temp = np.asarray(temp)
 Y = temp
 dataset = dataset.values
 X = dataset[:,0:dataset.shape[1]-1]
 X = normalize(X)
 indices = np.arange(X.shape[0])
 np.random.shuffle(indices)
 X = X[indices]
 Y = Y[indices]
 print(np.unique(Y))
 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")
 pca = PCA(n_components = 50)
 X = pca.fit_transform(X)
 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
 text.insert(END,"Dataset Train and Test Split\n\n")
 text.insert(END,"80% dataset records used to train ML algorithms :
"+str(X_train.shape[0])+"\n")
 text.insert(END,"20% dataset records used to test ML algorithms :
"+str(X_test.shape[0])+"\n")
def calculateMetrics(algorithm, predict, y_test):
 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
 accuracy.append(a)
 precision.append(p)
 recall.append(r)
 fscore.append(f)
 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+" FScore : "+str(f)+"\n\n")
 text.update_idletasks()
 print(np.unique(predict))
 print(np.unique(y_test))
 conf_matrix = confusion_matrix(y_test, predict)
 #plt.figure(figsize =(6, 6))
 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()
def runNaiveBayes():
 global X, Y, X_train, X_test, y_train, y_test
 global accuracy, precision,recall, fscore
 accuracy = []
 precision = []
 recall = []
 fscore = []
 text.delete('1.0', END)
 if os.path.exists('model/nb.txt'):
 with open('model/nb.txt', 'rb') as file:
 nb = pickle.load(file)
 file.close()
 else:
 nb = GaussianNB()
 nb.fit(X_train, y_train)
 with open('model/nb.txt', 'wb') as file:
 pickle.dump(nb, file)
 file.close()
 predict = nb.predict(X_test)
 calculateMetrics("Naive Bayes", predict, y_test)
def runRandomForest():
 global classifier
 if os.path.exists('model/rf.txt'):
 with open('model/rf.txt', 'rb') as file:
 rf = pickle.load(file)
 file.close()
 else:
 rf = RandomForestClassifier()
 rf.fit(X_train, y_train)
 with open('model/rf.txt', 'wb') as file:
 pickle.dump(rf, file)
 file.close()
 predict = rf.predict(X_test)
 classifier = rf
 calculateMetrics("Random Forest", predict, y_test)
def runSVM():
 if os.path.exists('model/svm.txt'):
 with open('model/svm.txt', 'rb') as file:
 svm_cls = pickle.load(file)
 file.close()
 else:
 svm_cls = svm.SVC()
 svm_cls.fit(X_train, y_train)
 with open('model/svm.txt', 'wb') as file:
 pickle.dump(svm_cls, file)
 file.close()
 predict = svm_cls.predict(X_test)
 calculateMetrics("SVM", predict, y_test)
def runXGBoost():
 if os.path.exists('model/xgb.txt'):
 with open('model/xgb.txt', 'rb') as file:
 xgb_cls = pickle.load(file)
 file.close()
 else:
 xgb_cls = XGBClassifier()
 xgb_cls.fit(X_train, y_train)
 with open('model/xgb.txt', 'wb') as file:
 pickle.dump(xgb_cls, file)
 file.close()
 predict = xgb_cls.predict(X_test)
 calculateMetrics("XGBoost", predict, y_test)
def runAdaBoost():
 if os.path.exists('model/adb.txt'):
 with open('model/adb.txt', 'rb') as file:
 adb_cls = pickle.load(file)
 file.close()
 else:
 adb_cls = AdaBoostClassifier()
 adb_cls.fit(X_train, y_train)
 with open('model/adb.txt', 'wb') as file:
 pickle.dump(adb_cls, file)
 file.close()
 predict = adb_cls.predict(X_test)
 calculateMetrics("AdaBoost", predict, y_test)
```

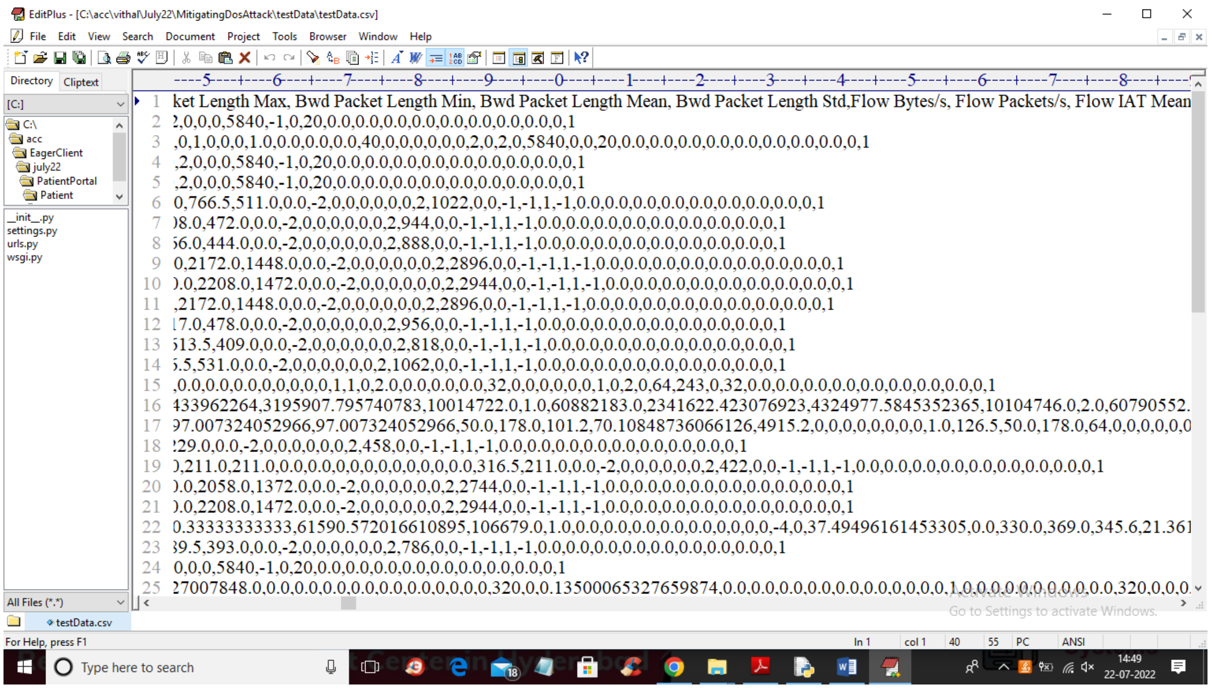
```
def runKNN():
 if os.path.exists('model/knn.txt'):
 with open('model/knn.txt', 'rb') as file:
 knn_cls = pickle.load(file)
 file.close()
 else:
 knn_cls = KNeighborsClassifier(n_neighbors = 2)
 knn_cls.fit(X_train, y_train)
 with open('model/knn.txt', 'wb') as file:
 pickle.dump(knn_cls, file)
 file.close()
 predict = knn_cls.predict(X_test)
 calculateMetrics("KNN", predict, y_test)
def predict():
 global label_encoder, labels, columns, types, pca
 text.delete('1.0', END)
 filename = filedialog.askopenfilename(initialdir="testData")
 testData = pd.read_csv(filename)
 count = 0
 for i in range(len(types)-1):
 name = types[i]
 if name == 'object':
 print(columns[i])
 if columns[i] == 'Flow Bytes/s':
 testData[columns[i]] =
pd.Series(label_encoder[count].fit_transform(testData[columns[i]].astype(str)))
 else:
 testData[columns[i]] =
pd.Series(label_encoder[count].transform(testData[columns[i]].astype(str)))
 count = count + 1
 testData.fillna(0, inplace = True)
 testData = testData.values
 testData = normalize(testData)
 testData = pca.transform(testData)
 predict = classifier.predict(testData)
 print(predict)
 for i in range(len(predict)):
 text.insert(END,"Test DATA : "+str(testData[i])+" ===> PREDICTED AS
"+labels[predict[i]]+"\n\n")

def graph():
 output = "<html><body><table align=center border=1><tr><th>Algorithm
Name</th><th>Accuracy</th><th>Precision</th><th>Recall</th>"
 output+="<th>FSCORE</th></tr>"
 output+="<tr><td>Naive Bayes
Algorithm</td><td>"+str(accuracy[0])+"</td><td>"+str(precision[0])+"</td><td>"+str(r
ecall[0])+"</td><td>"+str(fscore[0])+"</td></tr>"
output+="<tr><td>Random Forest
Algorithm</td><td>"+str(accuracy[1])+"</td><td>"+str(precision[1])+"</td><td>"+str(r
ecall[1])+"</td><td>"+str(fscore[1])+"</td></tr>"
 output+="<tr><td>SVM
Algorithm</td><td>"+str(accuracy[2])+"</td><td>"+str(precision[2])+"</td><td>"+str(r
ecall[2])+"</td><td>"+str(fscore[2])+"</td></tr>"
 output+="<tr><td>XGBoost
Algorithm</td><td>"+str(accuracy[3])+"</td><td>"+str(precision[3])+"</td><td>"+str(r
ecall[3])+"</td><td>"+str(fscore[3])+"</td></tr>"
 output+="<tr><td>AdaBoostBoost
Algorithm</td><td>"+str(accuracy[4])+"</td><td>"+str(precision[4])+"</td><td>"+str(r
ecall[4])+"</td><td>"+str(fscore[4])+"</td></tr>"
 output+="<tr><td>KNN
Algorithm</td><td>"+str(accuracy[5])+"</td><td>"+str(precision[5])+"</td><td>"+str(r
ecall[5])+"</td><td>"+str(fscore[5])+"</td></tr>"
```

```
 output+="</table></body></html>"
 f = open("table.html", "w")
 f.write(output)
 f.close()
 webbrowser.open("table.html",new=2)
  df = pd.DataFrame([['Naive Bayes','Precision',precision[0]],['Naive
Bayes','Recall',recall[0]],['Naive Bayes','F1 Score',fscore[0]],['Naive
Bayes','Accuracy',accuracy[0]],
 ['Random Forest','Precision',precision[1]],['Random
Forest','Recall',recall[1]],['Random Forest','F1 Score',fscore[1]],['Random
Forest','Accuracy',accuracy[1]],
 ['SVM','Precision',precision[2]],['SVM','Recall',recall[2]],['SVM','F1
Score',fscore[2]],['SVM','Accuracy',accuracy[2]],
 ['XGBoost','Precision',precision[3]],['XGBoost','Recall',recall[3]],['XGBoost','F1
Score',fscore[3]],['XGBoost','Accuracy',accuracy[3]],
 ['AdaBoost','Precision',precision[4]],['AdaBoost','Recall',recall[4]],['AdaBoost','F1
Score',fscore[4]],['AdaBoost','Accuracy',accuracy[4]],
 ['KNN','Precision',precision[5]],['KNN','Recall',recall[5]],['KNN','F1
Score',fscore[5]],['KNN','Accuracy',accuracy[5]],
 ],columns=['Algorithms','Performance Output','Value'])
 df.pivot("Algorithms", "Performance Output", "Value").plot(kind='bar')
 plt.show()
font = ('times', 16, 'bold')
title = Label(main, text='Mitigating DDOS Attack In IOT Network Environment')
title.config(bg='greenyellow', fg='dodger blue')
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="Upload DDOS Dataset", command=uploadDataset)
uploadButton.place(x=50,y=550)
uploadButton.config(font=font1)
preprocessButton = Button(main, text="Preprocess Dataset", command=preprocessDataset)
preprocessButton.place(x=330,y=550)
preprocessButton.config(font=font1)
nbButton = Button(main, text="Run Naive Bayes Algorithm", command=runNaiveBayes)
nbButton.place(x=630,y=550)
nbButton.config(font=font1)
rfButton = Button(main, text="Run Random Forest Algorithm", command=runRandomForest)
rfButton.place(x=920,y=550)
rfButton.config(font=font1)
svmButton = Button(main, text="Run SVM Algorithm", command=runSVM)
svmButton.place(x=50,y=600)
svmButton.config(font=font1)
xgButton = Button(main, text="Run XGBoost Algorithm", command=runXGBoost)
xgButton.place(x=330,y=600)
xgButton.config(font=font1)
adaboostButton = Button(main, text="Run AdaBoost Algorithm", command=runAdaBoost)
adaboostButton.place(x=630,y=600)
adaboostButton.config(font=font1)
knnButton = Button(main, text="Run KNN Algorithm", command=runKNN)
knnButton.place(x=920,y=600)
knnButton.config(font=font1)
graphButton = Button(main, text="Comparison Graph", command=graph)
graphButton.place(x=50,y=650)
graphButton.config(font=font1)
```

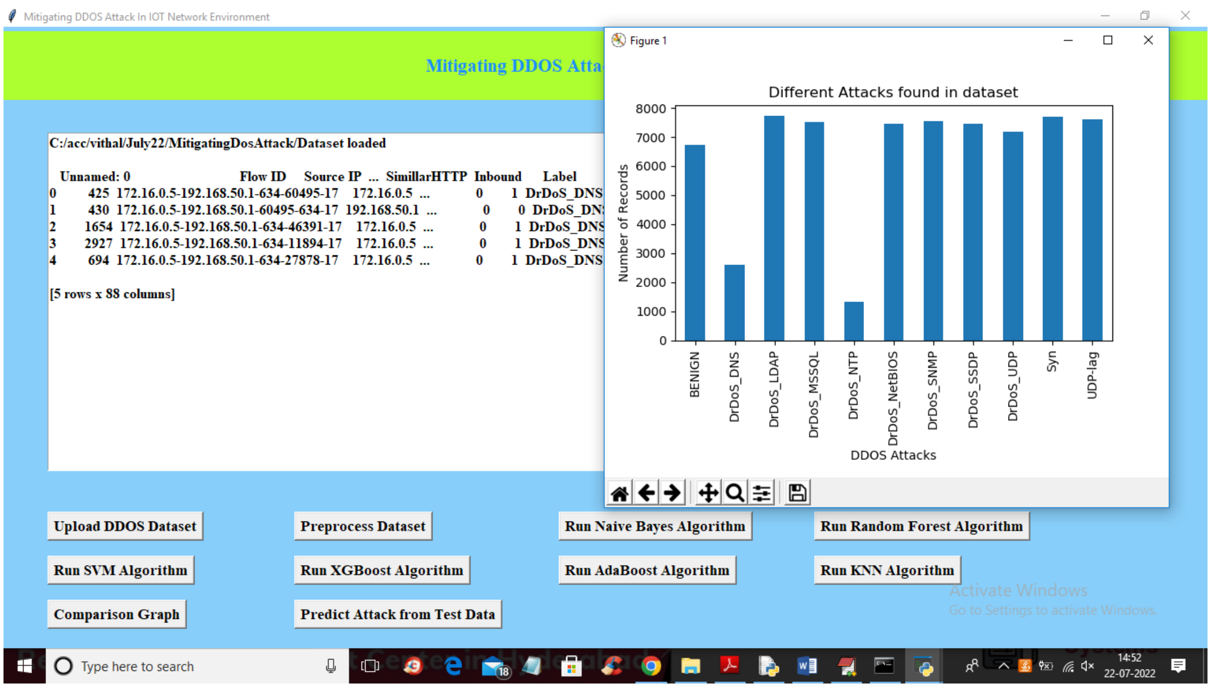
```
predictButton = Button(main, text="Predict Attack from Test Data", command=predict)
predictButton.place(x=330,y=650)
predictButton.config(font=font1)
main.config(bg='LightSkyBlue')
main.mainloop()
```

RESULT

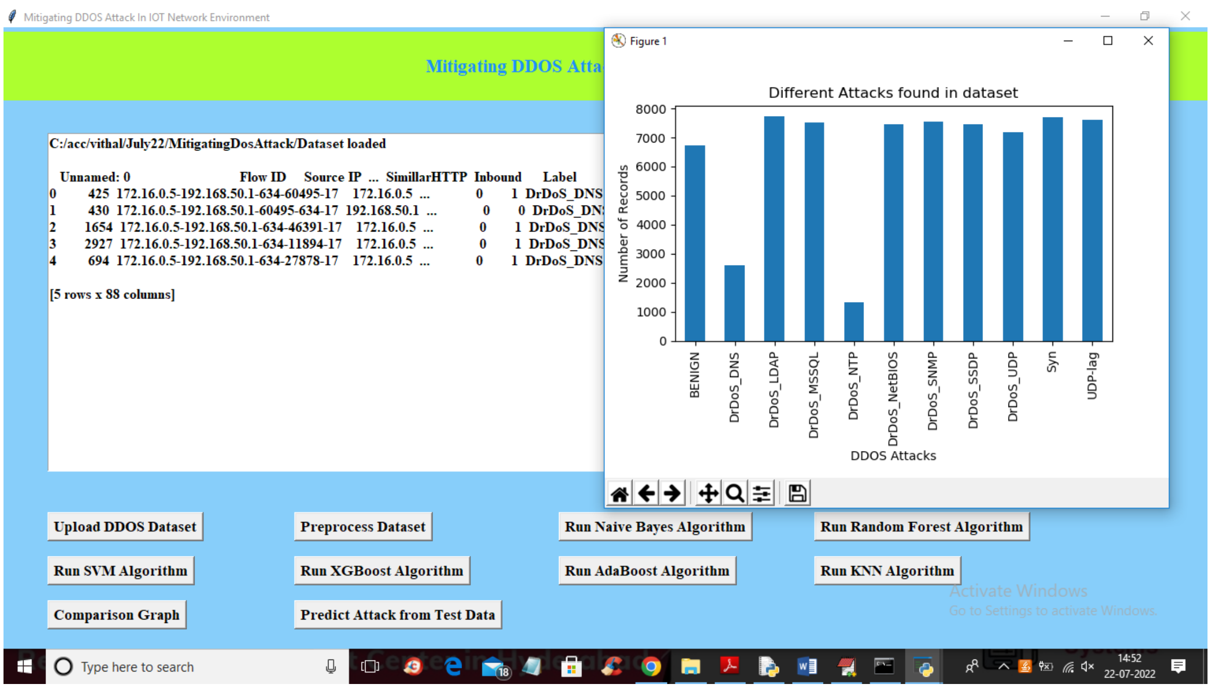


EditPlus - [C:\acc\vithal\July22\MitigatingDosAttack\testData\testData.csv]

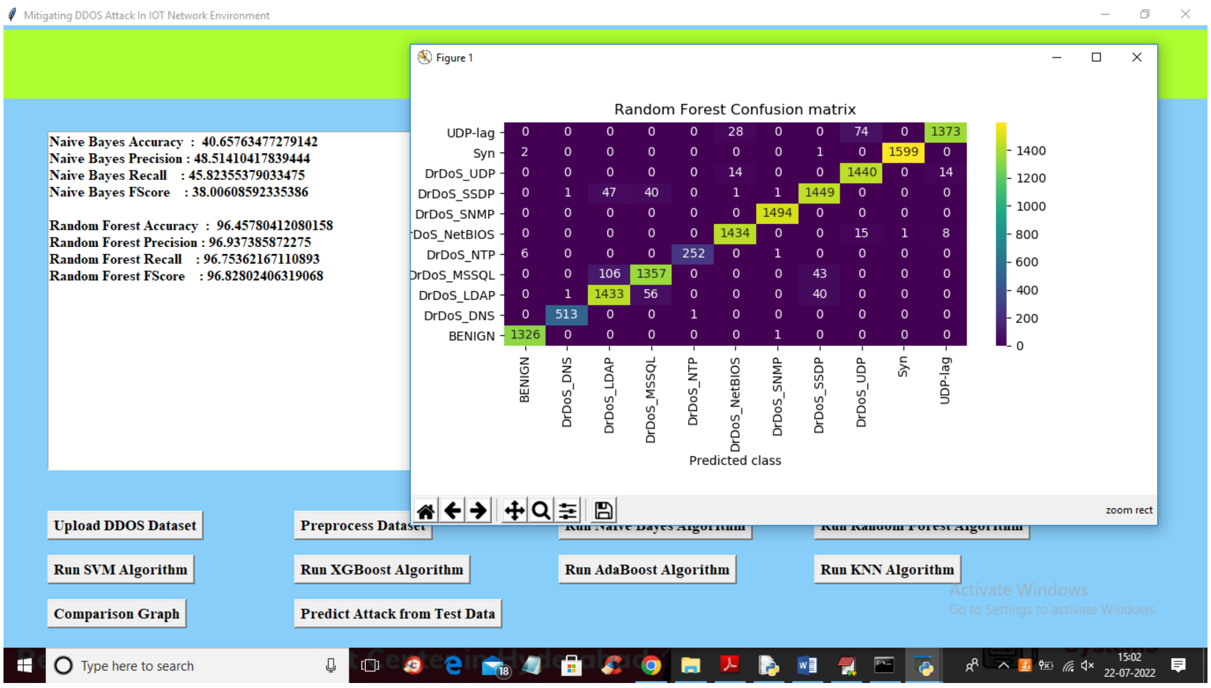
```
ket Length Max, Bwd Packet Length Min, Bwd Packet Length Mean, Bwd Packet Length Std,Flow Bytes/s, Flow Packets/s, Flow IAT Mean
2,0,0,0,5840,-1,0,20,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
,0,1,0,0,0,1.0,0.0,0.0,0.0,0.0,40,0,0,0,0,0,0,2,0,2,0,5840,0,0,20,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
,2,0,0,0,5840,-1,0,20,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
,2,0,0,0,5840,-1,0,20,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
0,766.5,511.0,0.0,-2,0,0,0,0,0,0,2,1022,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
)8.0,472.0,0.0,-2,0,0,0,0,0,0,2,944,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
56.0,444.0,0.0,-2,0,0,0,0,0,0,2,888,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
0,2172.0,1448.0,0.0,-2,0,0,0,0,0,0,2,2896,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
).0,2208.0,1472.0,0.0,-2,0,0,0,0,0,0,2,2944,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
,2172.0,1448.0,0.0,-2,0,0,0,0,0,0,2,2896,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
17.0,478.0,0.0,-2,0,0,0,0,0,0,2,956,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
513.5,409.0,0.0,-2,0,0,0,0,0,0,2,818,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
5.5,531.0,0.0,-2,0,0,0,0,0,0,2,1062,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
,0.0,0.0,0.0,0,0,0,0,0,0,1,1,0,2.0,0.0,0.0,0.0,0.0,32,0,0,0,0,0,0,1,0,2,0,64,243,0,32,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
433962264,3195907.795740783,10014722.0,1.0,60882183.0,2341622.423076923,4324977.5845352365,10104746.0,2.0,60790552.
97.007324052966,97.007324052966,50.0,178.0,101.2,70.10848736066126,4915.2,0,0,0,0,0,0,0,0,1.0,126.5,50.0,178.0,64,0,0,0,0,0,0
229.0,0.0,-2,0,0,0,0,0,0,2,458,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
),211.0,211.0,0.0,0.0,0,0,0,0,0,0,0,0,0,0,0,316.5,211.0,0.0,-2,0,0,0,0,0,0,2,422,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
).0,2058.0,1372.0,0.0,-2,0,0,0,0,0,0,2,2744,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
).0,2208.0,1472.0,0.0,-2,0,0,0,0,0,0,2,2944,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
0.3333333333,61590.572016610895,106679.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0,0,0,0,-4,0,37.49496161453305,0.0,330.0,369.0,345.6,21.361
39.5,393.0,0.0,-2,0,0,0,0,0,0,2,786,0,0,-1,-1,1,-1,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
0,0,0,5840,-1,0,20,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1
27007848.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,0,0,320,0,0.13500065327659874,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1,0,0,0,0,0,0,0,0,0,0,320,0,0,0,
```



```
C:/acc/vithal/July22/MitigatingDosAttack/Dataset loaded

 Unnamed: 0                        Flow ID    Source IP  ...  SimillarHTTP  Inbound     Label
0       425  172.16.0.5-192.168.50.1-634-60495-17   172.16.0.5  ...         0        1  DrDoS_DNS
1       430  172.16.0.5-192.168.50.1-60495-634-17  192.168.50.1  ...         0        0  DrDoS_DN
2      1654  172.16.0.5-192.168.50.1-634-46391-17   172.16.0.5  ...         0        1  DrDoS_DNS
3      2927  172.16.0.5-192.168.50.1-634-11894-17   172.16.0.5  ...         0        1  DrDoS_DNS
4       694  172.16.0.5-192.168.50.1-634-27878-17   172.16.0.5  ...         0        1  DrDoS_DNS

[5 rows x 88 columns]
```



```
C:/acc/vithal/July22/MitigatingDosAttack/Dataset loaded

 Unnamed: 0                        Flow ID    Source IP  ...  SimillarHTTP  Inbound     Label
0       425  172.16.0.5-192.168.50.1-634-60495-17   172.16.0.5  ...         0        1  DrDoS_DNS
1       430  172.16.0.5-192.168.50.1-60495-634-17  192.168.50.1  ...         0        0  DrDoS_DN
2      1654  172.16.0.5-192.168.50.1-634-46391-17   172.16.0.5  ...         0        1  DrDoS_DNS
3      2927  172.16.0.5-192.168.50.1-634-11894-17   172.16.0.5  ...         0        1  DrDoS_DNS
4       694  172.16.0.5-192.168.50.1-634-27878-17   172.16.0.5  ...         0        1  DrDoS_DNS

[5 rows x 88 columns]
```

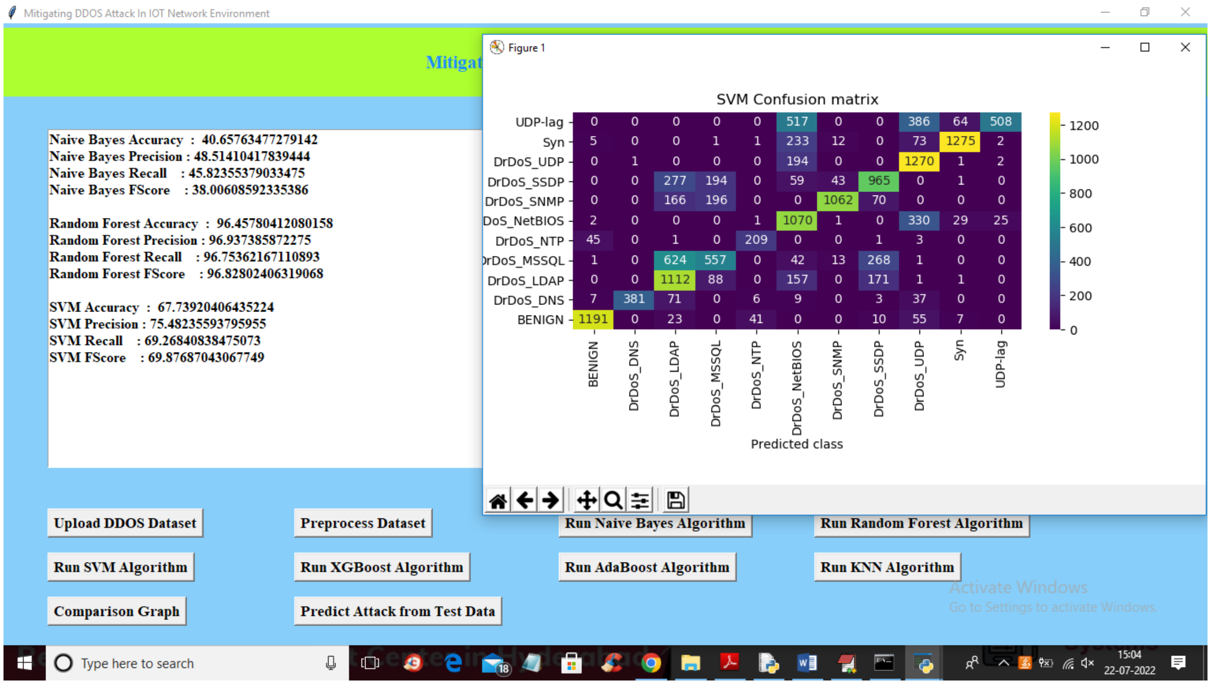


```
Naive Bayes Accuracy  :  40.65763477279142
Naive Bayes Precision : 48.51410417839444
Naive Bayes Recall    : 45.82355379033475
Naive Bayes FScore    : 38.00608592335386

Random Forest Accuracy  :  96.45780412080158
Random Forest Precision : 96.937385872275
Random Forest Recall    : 96.75362167110893
Random Forest FScore    : 96.82802406319068
```

Random Forest Confusion matrix

	BENIGN	DrDoS_DNS	DrDoS_LDAP	DrDoS_MSSQL	DrDoS_NTP	DrDoS_NetBIOS	DrDoS_SNMP	DrDoS_SSDP	DrDoS_UDP	Syn	UDP-lag
UDP-lag	0	0	0	0	0	28	0	0	74	0	1373
Syn	2	0	0	0	0	0	0	1	0	1599	0
DrDoS_UDP	0	0	0	0	0	14	0	0	1440	0	14
DrDoS_SSDP	0	1	47	40	0	1	1	1449	0	0	0
DrDoS_SNMP	0	0	0	0	0	0	1494	0	0	0	0
DoS_NetBIOS	0	0	0	0	0	1434	0	0	15	1	8
DrDoS_NTP	6	0	0	0	252	0	1	0	0	0	0
rDoS_MSSQL	0	0	106	1357	0	0	0	43	0	0	0
DrDoS_LDAP	0	1	1433	56	0	0	0	40	0	0	0
DrDoS_DNS	0	513	0	0	1	0	0	0	0	0	0
BENIGN	1326	0	0	0	0	0	1	0	0	0	0

Predicted class

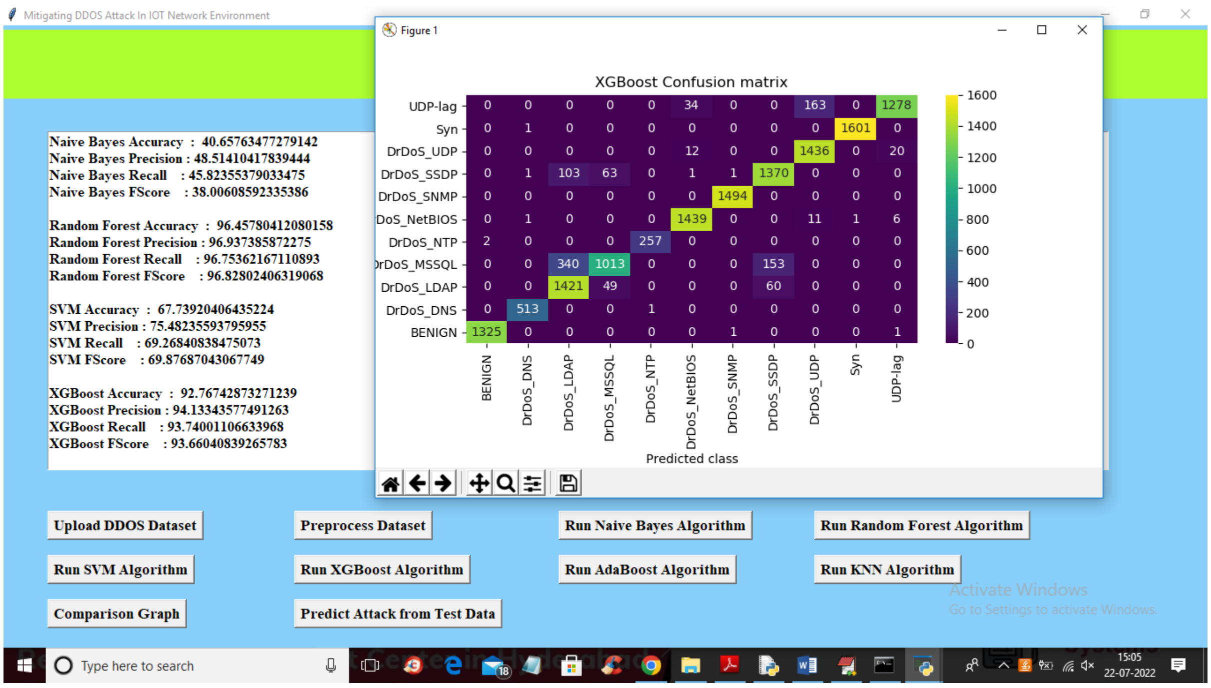


```
Naive Bayes Accuracy  :  40.65763477279142
Naive Bayes Precision : 48.51410417839444
Naive Bayes Recall    : 45.82355379033475
Naive Bayes FScore    : 38.00608592335386

Random Forest Accuracy  :  96.45780412080158
Random Forest Precision : 96.937385872275
Random Forest Recall    : 96.75362167110893
Random Forest FScore    : 96.82802406319068

SVM Accuracy  :  67.73920406435224
SVM Precision : 75.48235593795955
SVM Recall    : 69.26840838475073
SVM FScore    : 69.87687043067749
```

SVM Confusion matrix

	BENIGN	DrDoS_DNS	DrDoS_LDAP	DrDoS_MSSQL	DrDoS_NTP	DrDoS_NetBIOS	DrDoS_SNMP	DrDoS_SSDP	DrDoS_UDP	Syn	UDP-lag
UDP-lag	0	0	0	0	0	517	0	0	386	64	508
Syn	5	0	0	1	1	233	12	0	73	1275	2
DrDoS_UDP	0	1	0	0	0	194	0	0	1270	1	2
DrDoS_SSDP	0	0	277	194	0	59	43	965	0	1	0
DrDoS_SNMP	0	0	166	196	0	0	1062	70	0	0	0
DoS_NetBIOS	2	0	0	0	1	1070	1	0	330	29	25
DrDoS_NTP	45	0	1	0	209	0	0	1	3	0	0
rDoS_MSSQL	1	0	624	557	0	42	13	268	1	0	0
DrDoS_LDAP	0	0	1112	88	0	157	0	171	1	1	0
DrDoS_DNS	7	381	71	0	6	9	0	3	37	0	0
BENIGN	1191	0	23	0	41	0	0	10	55	7	0

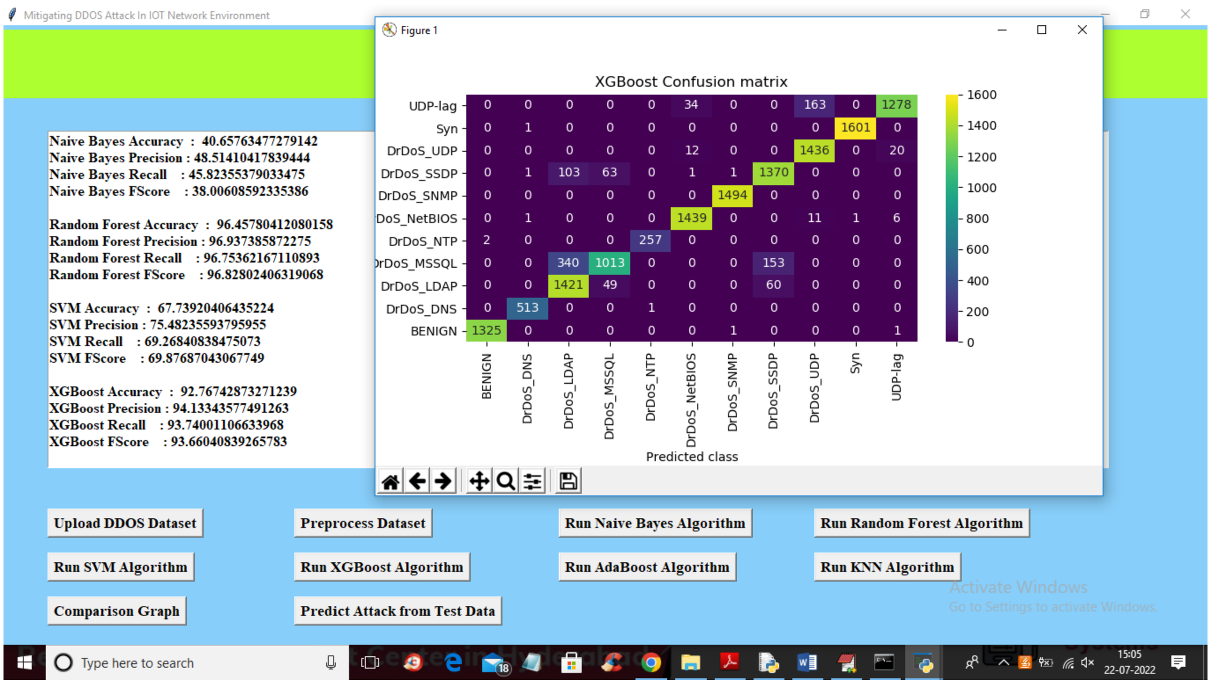
Predicted class



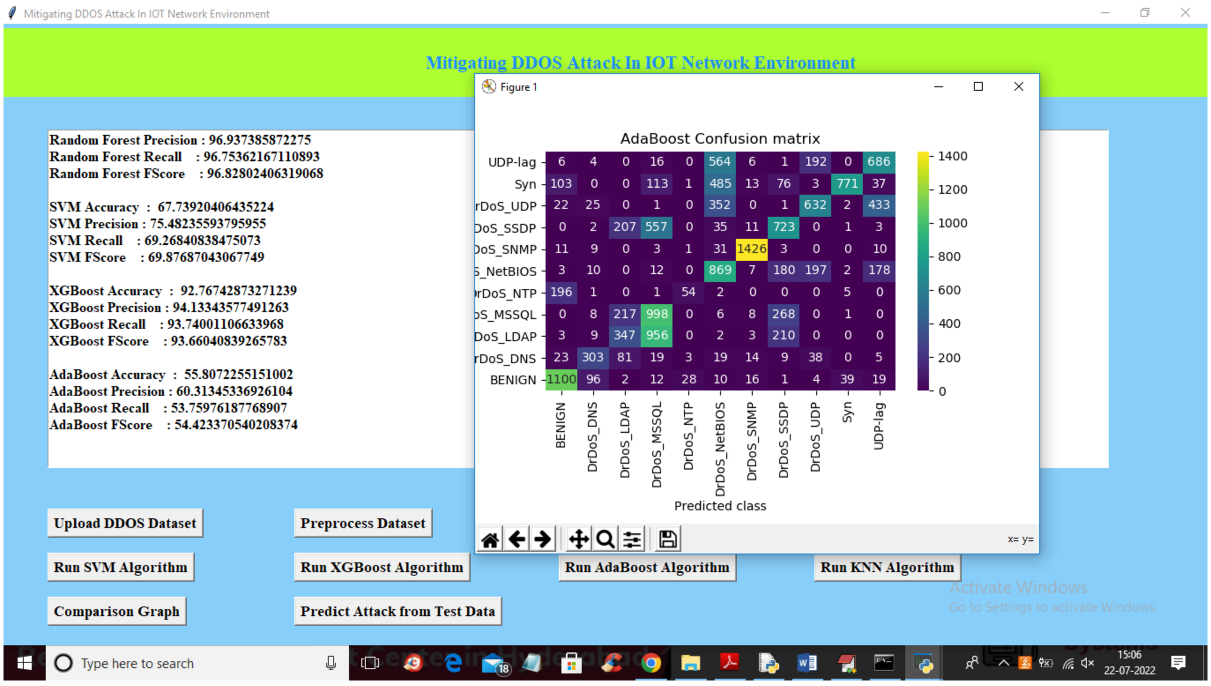
```
Naive Bayes Accuracy  :  40.65763477279142
Naive Bayes Precision : 48.51410417839444
Naive Bayes Recall    : 45.82355379033475
Naive Bayes FScore    : 38.00608592335386

Random Forest Accuracy  :  96.45780412080158
Random Forest Precision : 96.937385872275
Random Forest Recall    : 96.75362167110893
Random Forest FScore    : 96.82802406319068

SVM Accuracy  :  67.73920406435224
SVM Precision : 75.48235593795955
SVM Recall    : 69.26840838475073
SVM FScore    : 69.87687043067749

XGBoost Accuracy  :  92.76742873271239
XGBoost Precision : 94.13343577491263
XGBoost Recall    : 93.74001106633968
XGBoost FScore    : 93.66040839265783
```

XGBoost Confusion matrix

	BENIGN	DrDoS_DNS	DrDoS_LDAP	DrDoS_MSSQL	DrDoS_NTP	DrDoS_NetBIOS	DrDoS_SNMP	DrDoS_SSDP	DrDoS_UDP	Syn	UDP-lag
UDP-lag	0	0	0	0	0	34	0	0	163	0	1278
Syn	0	1	0	0	0	0	0	0	0	1601	0
DrDoS_UDP	0	0	0	0	0	12	0	0	1436	0	20
DrDoS_SSDP	0	1	103	63	0	1	1	1370	0	0	0
DrDoS_SNMP	0	0	0	0	0	0	1494	0	0	0	0
DoS_NetBIOS	0	1	0	0	0	1439	0	0	11	1	6
DrDoS_NTP	2	0	0	0	257	0	0	0	0	0	0
rDoS_MSSQL	0	0	340	1013	0	0	0	153	0	0	0
DrDoS_LDAP	0	0	1421	49	0	0	0	60	0	0	0
DrDoS_DNS	0	513	0	0	1	0	0	0	0	0	0
BENIGN	1325	0	0	0	0	0	1	0	0	0	1



```
Naive Bayes Accuracy  :  40.65763477279142
Naive Bayes Precision : 48.51410417839444
Naive Bayes Recall    : 45.82355379033475
Naive Bayes FScore    : 38.00608592335386

Random Forest Accuracy  :  96.45780412080158
Random Forest Precision : 96.937385872275
Random Forest Recall    : 96.75362167110893
Random Forest FScore    : 96.82802406319068

SVM Accuracy  :  67.73920406435224
SVM Precision : 75.48235593795955
SVM Recall    : 69.26840838475073
SVM FScore    : 69.87687043067749

XGBoost Accuracy  :  92.76742873271239
XGBoost Precision : 94.13343577491263
XGBoost Recall    : 93.74001106633968
XGBoost FScore    : 93.66040839265783
```

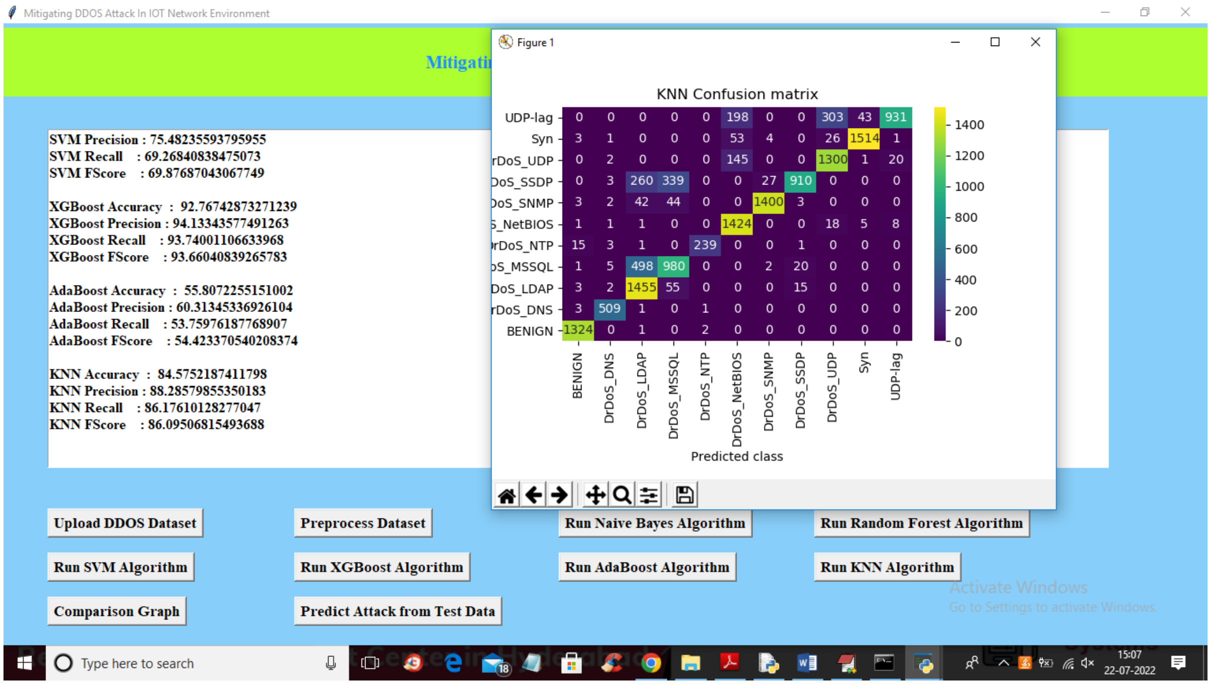


```
Random Forest Precision : 96.937385872275
Random Forest Recall    : 96.75362167110893
Random Forest FScore    : 96.82802406319068

SVM Accuracy  :  67.73920406435224
SVM Precision : 75.48235593795955
SVM Recall    : 69.26840838475073
SVM FScore    : 69.87687043067749

XGBoost Accuracy  :  92.76742873271239
XGBoost Precision : 94.13343577491263
XGBoost Recall    : 93.74001106633968
XGBoost FScore    : 93.66040839265783

AdaBoost Accuracy  :  55.8072255151002
AdaBoost Precision : 60.31345336926104
AdaBoost Recall    : 53.75976187768907
AdaBoost FScore    : 54.423370540208374
```

AdaBoost Confusion matrix

	BENIGN	DrDoS_DNS	DrDoS_LDAP	DrDoS_MSSQL	DrDoS_NTP	DrDoS_NetBIOS	DrDoS_SNMP	DrDoS_SSDP	DrDoS_UDP	Syn	UDP-lag
UDP-lag	6	4	0	16	0	564	6	1	192	0	686
Syn	103	0	0	113	1	485	13	76	3	771	37
rDoS_UDP	22	25	0	1	0	352	0	1	632	2	433
DoS_SSDP	0	2	207	557	0	35	11	723	0	1	3
DoS_SNMP	11	9	0	3	1	31	1426	3	0	0	10
S_NetBIOS	3	10	0	12	0	869	7	180	197	2	178
rDoS_NTP	196	1	0	1	54	2	0	0	0	5	0
oS_MSSQL	0	8	217	998	0	6	8	268	0	1	0
DoS_LDAP	3	9	347	956	0	2	3	210	0	0	0
rDoS_DNS	23	303	81	19	3	19	14	9	38	0	5
BENIGN	1100	96	2	12	28	10	16	1	4	39	19

Predicted class

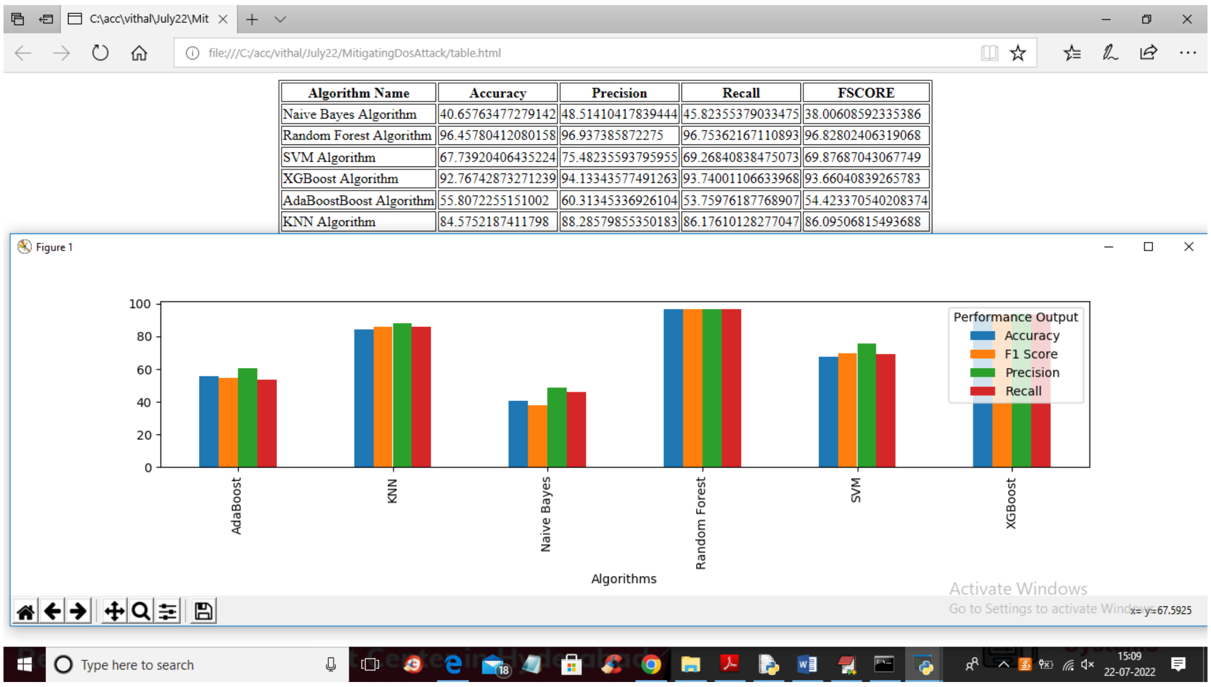


```
SVM Precision : 75.48235593795955
SVM Recall    : 69.26840838475073
SVM FScore    : 69.87687043067749

XGBoost Accuracy  :  92.76742873271239
XGBoost Precision : 94.13343577491263
XGBoost Recall    : 93.74001106633968
XGBoost FScore    : 93.66040839265783

AdaBoost Accuracy  :  55.8072255151002
AdaBoost Precision : 60.31345336926104
AdaBoost Recall    : 53.75976187768907
AdaBoost FScore    : 54.423370540208374

KNN Accuracy  :  84.5752187411798
KNN Precision : 88.28579855350183
KNN Recall    : 86.17610128277047
KNN FScore    : 86.09506815493688
```

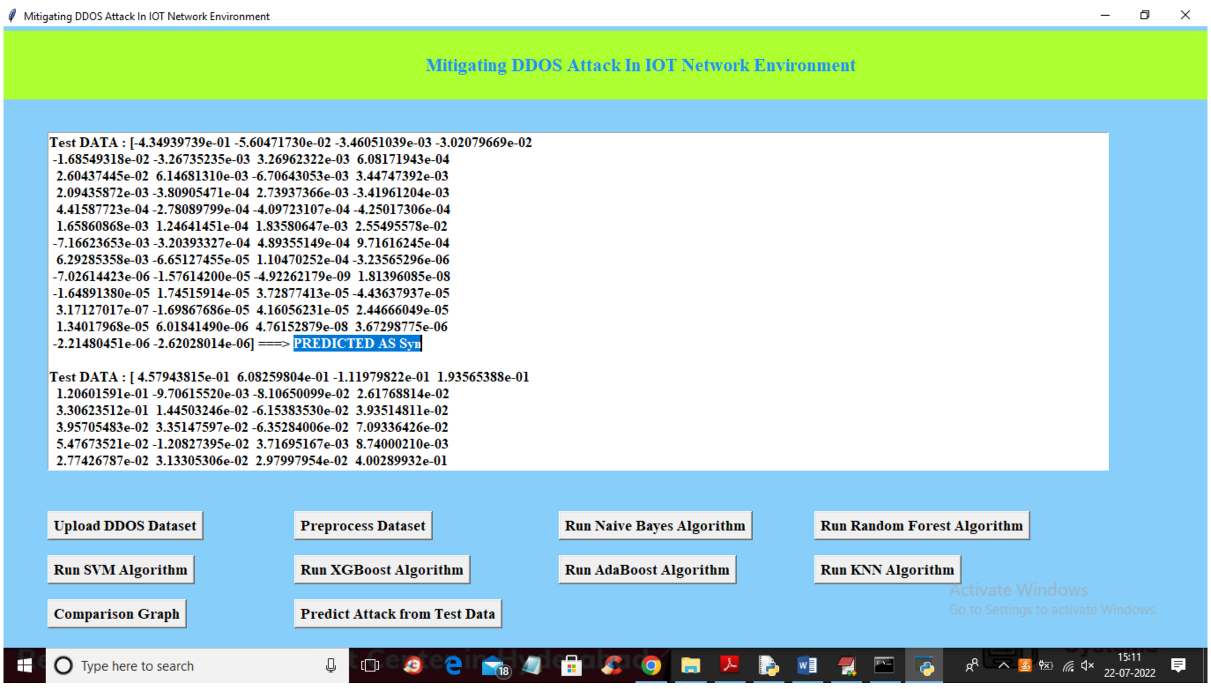
KNN Confusion matrix

	BENIGN	DrDoS_DNS	DrDoS_LDAP	DrDoS_MSSQL	DrDoS_NTP	DrDoS_NetBIOS	DrDoS_SNMP	DrDoS_SSDP	DrDoS_UDP	Syn	UDP-lag
UDP-lag	0	0	0	0	0	198	0	0	303	43	931
Syn	3	1	0	0	0	53	4	0	26	1514	1
rDoS_UDP	0	2	0	0	0	145	0	0	1300	1	20
DoS_SSDP	0	3	260	339	0	0	27	910	0	0	0
DoS_SNMP	3	2	42	44	0	0	1400	3	0	0	0
S_NetBIOS	1	1	1	0	0	1424	0	0	18	5	8
rDoS_NTP	15	3	1	0	239	0	0	1	0	0	0
oS_MSSQL	1	5	498	980	0	0	2	20	0	0	0
DoS_LDAP	3	2	1455	55	0	0	0	15	0	0	0
rDoS_DNS	3	509	1	0	1	0	0	0	0	0	0
BENIGN	1324	0	1	0	2	0	0	0	0	0	0

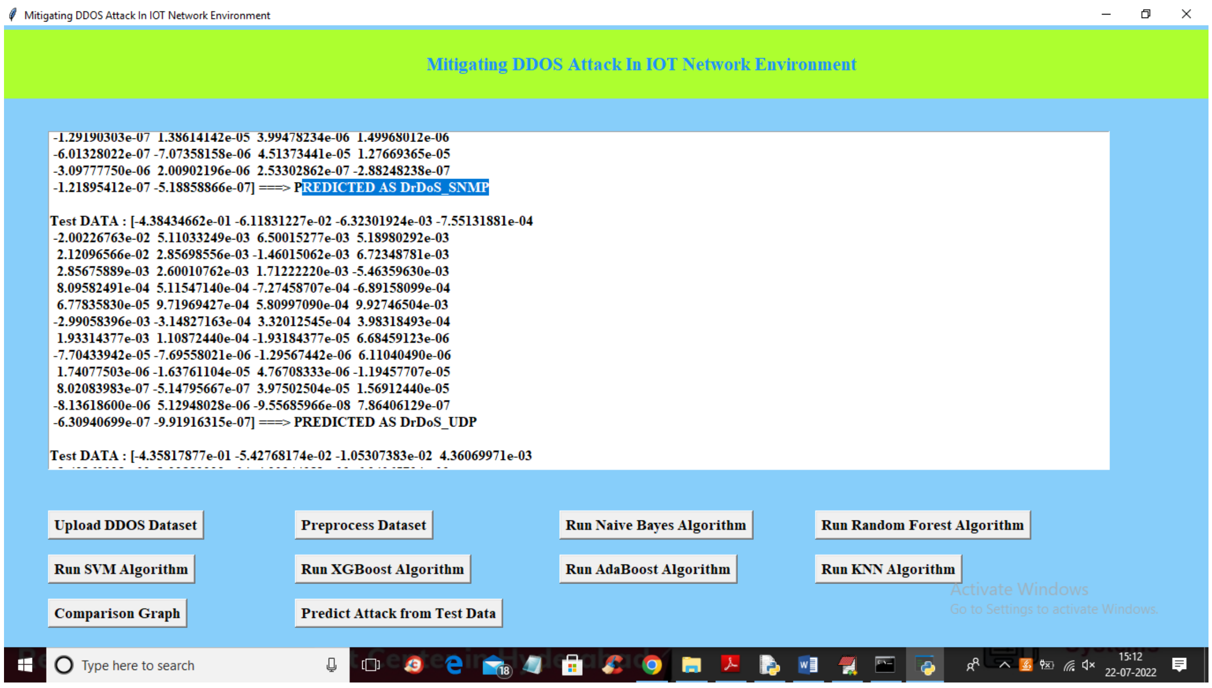
Predicted class



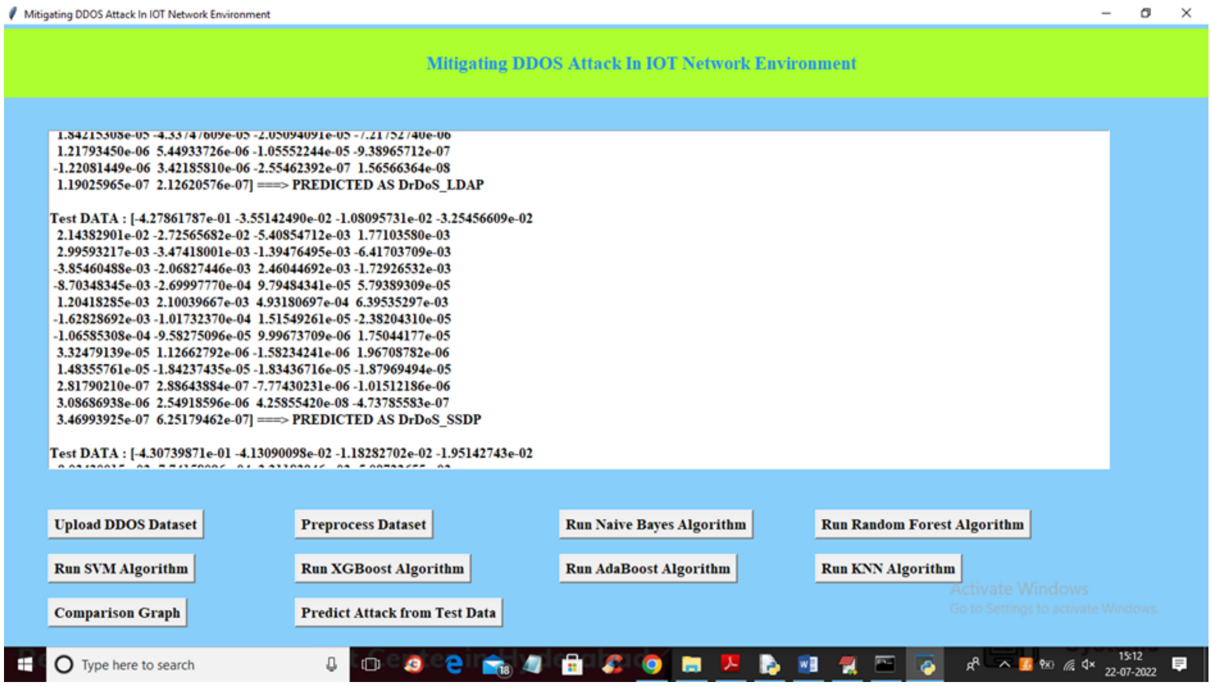
Algorithm Name	Accuracy	Precision	Recall	FSCORE
Naive Bayes Algorithm	40.65763477279142	48.51410417839444	45.82355379033475	38.00608592335386
Random Forest Algorithm	96.45780412080158	96.937385872275	96.75362167110893	96.82802406319068
SVM Algorithm	67.73920406435224	75.48235593795955	69.26840838475073	69.87687043067749
XGBoost Algorithm	92.76742873271239	94.13343577491263	93.74001106633968	93.66040839265783
AdaBoostBoost Algorithm	55.8072255151002	60.31345336926104	53.75976187768907	54.423370540208374
KNN Algorithm	84.5752187411798	88.28579855350183	86.17610128277047	86.09506815493688


Mitigating DDOS Attack In IOT Network Environment
Mitigating 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
Type here to search

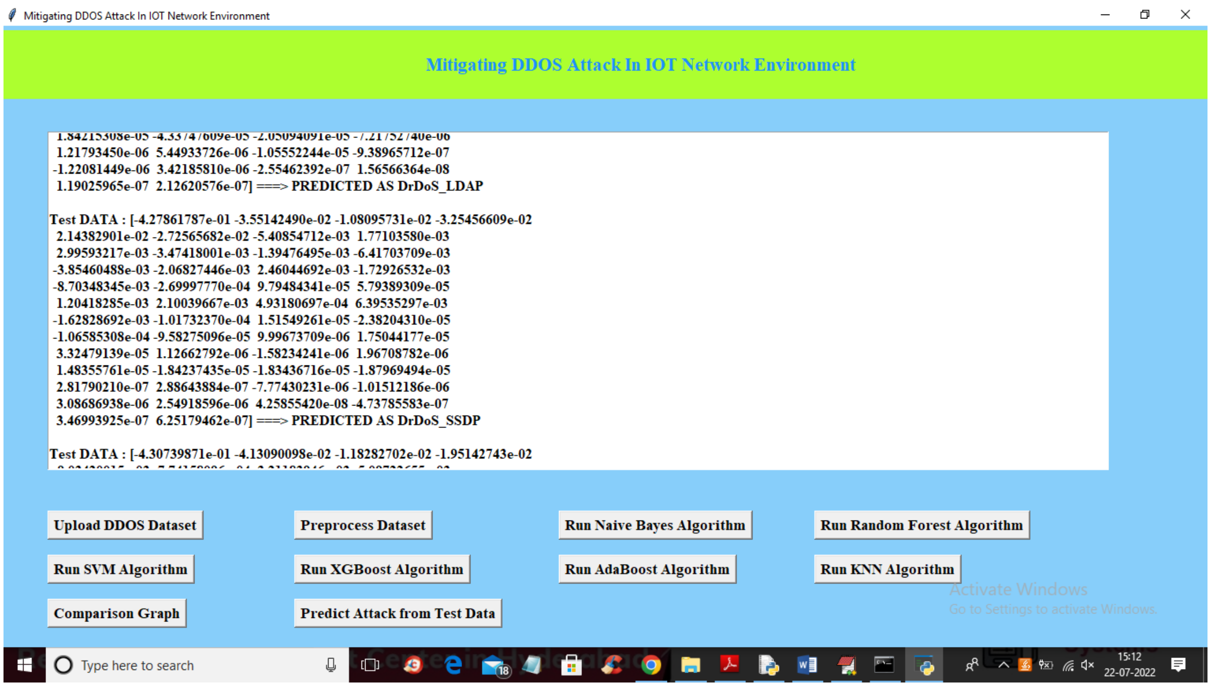
```
Test DATA : [-4.34939739e-01 -5.60471730e-02 -3.46051039e-03 -3.02079669e-02
 -1.68549318e-02 -3.26735235e-03  3.26962322e-03  6.08171943e-04
  2.60437445e-02  6.14681310e-03 -6.70643053e-03  3.44747392e-03
  2.09435872e-03 -3.80905471e-04  2.73937366e-03 -3.41961204e-03
  4.41587723e-04 -2.78089799e-04 -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 -4.92262179e-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
```


Mitigating DDOS Attack In IOT Network Environment
Mitigating 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
Type here to search
15:12
22-07-2022

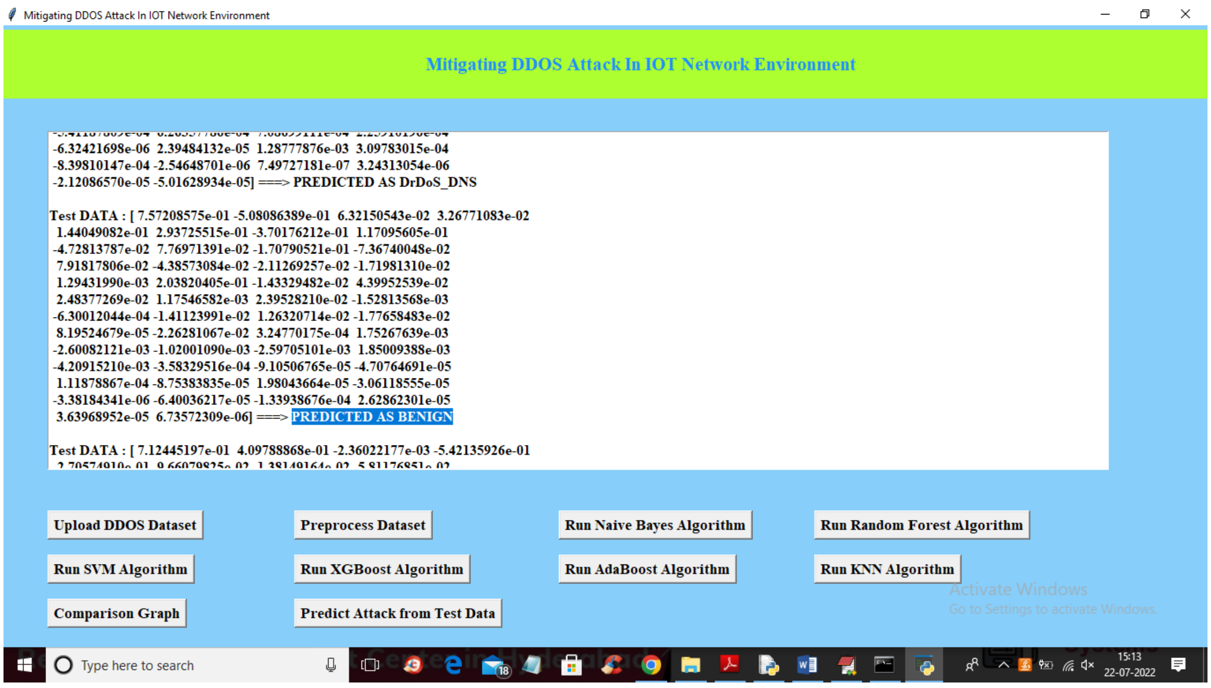
```
-1.29190303e-07  1.38614142e-05  3.99478234e-06  1.49968012e-06
-6.01328022e-07 -7.07358158e-06  4.51373441e-05  1.27669365e-05
-3.09777750e-06  2.00902196e-06  2.53302862e-07 -2.88248238e-07
-1.21895412e-07 -5.18858866e-07] ===> PREDICTED AS DrDoS_SNMP

Test DATA : [-4.38434662e-01 -6.11831227e-02 -6.32301924e-03 -7.55131881e-04
 -2.00226763e-02  5.11033249e-03  6.50015277e-03  5.18980292e-03
  2.12096566e-02  2.85698556e-03 -1.46015062e-03  6.72348781e-03
  2.85675889e-03  2.60010762e-03  1.71222220e-03 -5.46359630e-03
  8.09582491e-04  5.11547140e-04 -7.27458707e-04 -6.89158099e-04
  6.77835830e-05  9.71969427e-04  5.80997090e-04  9.92746504e-03
 -2.99058396e-03 -3.14827163e-04  3.32012545e-04  3.98318493e-04
  1.93314377e-03  1.10872440e-04 -1.93184377e-05  6.68459123e-06
 -7.70433942e-05 -7.69558021e-06 -1.29567442e-06  6.11040490e-06
  1.74077503e-06 -1.63761104e-05  4.76708333e-06 -1.19457707e-05
  8.02083983e-07 -5.14795667e-07  3.97502504e-05  1.56912440e-05
 -8.13618600e-06  5.12948028e-06 -9.55685966e-08  7.86406129e-07
 -6.30940699e-07 -9.91916315e-07] ===> PREDICTED AS DrDoS_UDP

Test DATA : [-4.35817877e-01 -5.42768174e-02 -1.05307383e-02  4.36069971e-03
```


Mitigating DDOS Attack In IOT Network Environment
Mitigating 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
Type here to search
15:12
22-07-2022

```
  1.84215308e-05 -4.33747609e-05 -2.05094091e-05 -7.21752740e-06
  1.21793450e-06  5.44933726e-06 -1.05552244e-05 -9.38965712e-07
 -1.22081449e-06  3.42185810e-06 -2.55462392e-07  1.56566364e-08
  1.19025965e-07  2.12620576e-07] ===> PREDICTED AS DrDoS_LDAP

Test DATA : [-4.27861787e-01 -3.55142490e-02 -1.08095731e-02 -3.25456609e-02
  2.14382901e-02 -2.72565682e-02 -5.40854712e-03  1.77103580e-03
  2.99593217e-03 -3.47418001e-03 -1.39476495e-03 -6.41703709e-03
 -3.85460488e-03 -2.06827446e-03  2.46044692e-03 -1.72926532e-03
 -8.70348345e-03 -2.69997770e-04  9.79484341e-05  5.79389309e-05
  1.20418285e-03  2.10039667e-03  4.93180697e-04  6.39535297e-03
 -1.62828692e-03 -1.01732370e-04  1.51549261e-05 -2.38204310e-05
 -1.06585308e-04 -9.58275096e-05  9.99673709e-06  1.75044177e-05
  3.32479139e-05  1.12662792e-06 -1.58234241e-06  1.96708782e-06
  1.48355761e-05 -1.84237435e-05 -1.83436716e-05 -1.87969494e-05
  2.81790210e-07  2.88643884e-07 -7.77430231e-06 -1.01512186e-06
  3.08686938e-06  2.54918596e-06  4.25855420e-08 -4.73785583e-07
  3.46993925e-07  6.25179462e-07] ===> PREDICTED AS DrDoS_SSDP

Test DATA : [-4.30739871e-01 -4.13090098e-02 -1.18282702e-02 -1.95142743e-02
```


Mitigating DDOS Attack In IOT Network Environment
Mitigating 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
Type here to search
15:12
22-07-2022

```
 1.84215308e-05 -4.33747609e-05 -2.05094091e-05 -7.21752740e-06
 1.21793450e-06  5.44933726e-06 -1.05552244e-05 -9.38965712e-07
-1.22081449e-06  3.42185810e-06 -2.55462392e-07  1.56566364e-08
 1.19025965e-07  2.12620576e-07] ===> PREDICTED AS DrDoS_LDAP

Test DATA : [-4.27861787e-01 -3.55142490e-02 -1.08095731e-02 -3.25456609e-02
 2.14382901e-02 -2.72565682e-02 -5.40854712e-03  1.77103580e-03
 2.99593217e-03 -3.47418001e-03 -1.39476495e-03 -6.41703709e-03
-3.85460488e-03 -2.06827446e-03  2.46044692e-03 -1.72926532e-03
-8.70348345e-03 -2.69997770e-04  9.79484341e-05  5.79389309e-05
 1.20418285e-03  2.10039667e-03  4.93180697e-04  6.39535297e-03
-1.62828692e-03 -1.01732370e-04  1.51549261e-05 -2.38204310e-05
-1.06585308e-04 -9.58275096e-05  9.99673709e-06  1.75044177e-05
 3.32479139e-05  1.12662792e-06 -1.58234241e-06  1.96708782e-06
 1.48355761e-05 -1.84237435e-05 -1.83436716e-05 -1.87969494e-05
 2.81790210e-07  2.88643884e-07 -7.77430231e-06 -1.01512186e-06
 3.08686938e-06  2.54918596e-06  4.25855420e-08 -4.73785583e-07
 3.46993925e-07  6.25179462e-07] ===> PREDICTED AS DrDoS_SSDP

Test DATA : [-4.30739871e-01 -4.13090098e-02 -1.18282702e-02 -1.95142743e-02
```


Mitigating DDOS Attack In IOT Network Environment
Mitigating 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
Type here to search
15:13
22-07-2022

```
-6.32421698e-06  2.39484132e-05  1.28777876e-03  3.09783015e-04
-8.39810147e-04 -2.54648701e-06  7.49727181e-07  3.24313054e-06
-2.12086570e-05 -5.01628934e-05] ===> PREDICTED AS DrDoS_DNS

Test DATA : [ 7.57208575e-01 -5.08086389e-01  6.32150543e-02  3.26771083e-02
 1.44049082e-01  2.93725515e-01 -3.70176212e-01  1.17095605e-01
-4.72813787e-02  7.76971391e-02 -1.70790521e-01 -7.36740048e-02
 7.91817806e-02 -4.38573084e-02 -2.11269257e-02 -1.71981310e-02
 1.29431990e-03  2.03820405e-01 -1.43329482e-02  4.39952539e-02
 2.48377269e-02  1.17546582e-03  2.39528210e-02 -1.52813568e-03
-6.30012044e-04 -1.41123991e-02  1.26320714e-02 -1.77658483e-02
 8.19524679e-05 -2.26281067e-02  3.24770175e-04  1.75267639e-03
-2.60082121e-03 -1.02001090e-03 -2.59705101e-03  1.85009388e-03
-4.20915210e-03 -3.58329516e-04 -9.10506765e-05 -4.70764691e-05
 1.11878867e-04 -8.75383835e-05  1.98043664e-05 -3.06118555e-05
-3.38184341e-06 -6.40036217e-05 -1.33938676e-04  2.62862301e-05
 3.63968952e-05  6.73572309e-06] ===> PREDICTED AS BENIGN

Test DATA : [ 7.12445197e-01  4.09788868e-01 -2.36022177e-03 -5.42135926e-01
```

CONCLUSION

In this paper, we describe a general framework for SD-IoT composed of an SD-IoT controller pool with controllers, SD-IoT switches integrated with the IoT gateway, and terminal IoT devices. Then, we propose an algorithm for detecting and mitigating DDoS attacks with the proposed SD-IoT framework. In the proposed algorithm, we obtain the threshold value of the cosine similarity of the vectors of the packet-in rate at the ports of the SD-IoT boundary switches; we use the threshold value to determine whether a DDoS attack has occurred, find the real DDoS attacker, and block the DDoS attack at the source. Finally, the simulation results show that the proposed algorithm can find the IoT device from which a DDoS attack is launched within a shorter time period, quickly handle and mitigate the DDoS attack, and ultimately improve the unveiled glaring vulnerabilities in IoT, in which the terminal devices have computational and memory requirement constraints. Future work will focus on how to proactively defend against DDoS attacks in SD-IoT. In addition, dynamic load-balancing algorithms in the con- troller pool will be designed and implemented, and more effi- cient algorithms for detecting and mitigating DDoS attacks based on the SD-IoT framework will be investigated.

Mitigating Distributed Denial of Service (DDoS) attacks in Internet of Things (IoT) network environments is a critical task to ensure the uninterrupted functionality and security of connected devices. In conclusion, implementing effective strategies to address the unique challenges posed by IoT devices is essential for maintaining a resilient and secure network. Several key considerations and conclusions can be drawn from efforts to mitigate DDoS attacks in IoT environments:

1.Proactive Defense Mechanisms:

Implementing proactive defense mechanisms is crucial. This includes intrusion detection systems, firewalls, and traffic filtering tools designed specifically for IoT devices. These measures can identify and block malicious traffic before it reaches its target, reducing the impact of potential DDoS attacks.

2.Traffic Monitoring and Anomaly Detection:

Regularly monitoring network traffic and employing anomaly detection techniques is essential for recognizing unusual patterns that may indicate a DDoS attack. By leveraging machine learning algorithms, network administrators can enhance their ability to identify and respond to abnormal activities quickly.

3.Device Authentication and Authorization:

Strengthening device authentication and authorization processes helps prevent unauthorized access to the IoT network. Only authenticated and authorized devices should be allowed to communicate, reducing the risk of compromised devices participating in DDoS attacks.

4.Bandwidth Scaling and Redundancy:

Investing in scalable bandwidth and redundancy measures can help absorb the impact of a DDoS attack. By distributing traffic across multiple servers and networks, organizations can ensure that their IoT infrastructure remains operational even during high-volume attacks.

5.Collaboration and Information Sharing:

Collaboration among different stakeholders, including IoT device manufacturers, network operators, and security experts, is crucial. Information sharing about emerging threats and vulnerabilities helps the community develop more effective countermeasures and safeguards against evolving DDoS techniques.

6.Regular Updates and Patch Management:

Keeping IoT devices and associated software up-to-date is essential. Regular updates and effective patch management help address known vulnerabilities, reducing the likelihood of devices becoming unwitting participants in DDoS attacks.

7.Incident Response Planning:

Having a well-defined incident response plan in place is essential. This plan should outline the steps to be taken in the event of a DDoS attack, including communication strategies, containment procedures, and recovery efforts.

8.Continuous Evaluation and Adaptation:

The threat landscape is continually evolving, and so should DDoS mitigation strategies. Continuous evaluation of existing defenses and adaptation to emerging threats is essential to stay ahead of potential attacks.

In conclusion, mitigating DDoS attacks in IoT network environments requires a multifaceted and dynamic approach. By combining technical solutions, collaboration, and a proactive mindset, organizations can significantly enhance their ability to protect IoT devices from the disruptive impact of DDoS attacks.

REFERENCES

[1] H. Ma, L. Liu, A. Zhou, and D. Zhao, ‘‘On networking of Internet of Things: Explorations and challenges,’’ IEEE Internet Things J., vol. 3, no. 4, pp. 441–452, Aug. 2016.

[2] P. G. Neumann, ‘‘Risks of automation: A cautionary total-system per- spective of our cyberfuture,’’ Commun. ACM, vol. 59, no. 10, pp. 26–30, Oct. 2016.

[3] X. Liu, S. Zhao, A. Liu, N. Xiong, and A. V. Vasilakos, ‘‘Knowledge- aware proactive nodes selection approach for energy management in Internet of Things,’’ Future Generat. Comput. Syst., Aug. 2017, doi: https://doi.org/10.1016/j.future.2017.07.022

[4] Y. Liu, A. Liu, S. Guo, Z. Li, Y.-J. Choi, and H. Sekiya, ‘‘Context-aware collect data with energy efficient in cyber– physical cloud systems,’’ Future Generat. Comput. Syst., Jun. 2017, doi: https://doi.org/10.1016/j.future.2017.05.029

[5] K. Sonar and H. Upadhyay, ‘‘A survey: DDoS attack on Internet of Things,’’ Int. J. Eng. Res. Develop., vol. 10, no. 11, pp. 58–63, Nov. 2014.

[6] U. Lindqvist and P. G. Neumann, ‘‘The future of the Internet of Things,’’ Commun. ACM, vol. 60, no. 2, pp. 26–30, Jan. 2017.

[7] R. Huo et al., ‘‘Software defined networking, caching, and computing for green wireless networks,’’ IEEE Commun. Mag., vol. 54, no. 11, pp. 185–193, Nov. 2016.

[8] J. Zhang, X. Zhang, M. A. Imran, B. Evans, Y. Zhang, and W. Wang, ‘‘Energy efficient hybrid satellite terrestrial 5G networks with software defined features,’’ J. Commun. Netw., vol. 19, no. 2, pp. 147–161, Apr. 2017.

[9] K. Wang, K. Yang, H.-H. Chen, and L. Zhang, ‘‘Computation diversity in emerging networking paradigms,’’ IEEE Wireless Commun., vol. 24, no. 1, pp. 88–94, Feb. 2017.

[10] L. Zhang, Q. Deng, Y. Su, and Y. Hu, ‘‘A box-covering-based rout- ing algorithm for largescale SDNs,’’ IEEE Access, vol. 5, no. 1, pp. 4048–4056, Mar. 2017.

[11] X. Xiong, L. Hou, K. Zheng, W. Xiang, M. S. Hossain, and S. M. M. Rahman, ‘‘SMDP-based radio resource allocation scheme in software-defined Internet of Things networks,’’ IEEE Sensors J., vol. 16, no. 20, pp. 7304–7314, Oct. 2016.

[12] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, ‘‘A software defined networking architecture for the Internet-of-Things,’’ in Proc. IEEE Netw. Oper. Manage. Symp. (NOMS).