**DDOS ATTACK DETECTION & MITIGATION**

**Abstract:**

This project focuses on detecting and mitigating Distributed Denial of Service (DDoS) attacks in IoT environments using machine learning and deep learning algorithms. Due to the limited processing power and battery capacity of IoT devices, traditional antivirus solutions are impractical. Instead, lightweight ML/DL approaches such as XGBoost and CNN are employed to achieve high detection accuracy—over 95%—using the IoT23 DDoS attack dataset. The system processes and trains models in Jupyter Notebook and deploys a Flask web application for real-time attack detection. Upon identifying an attack, the system automatically blocks the attacker’s IP address via firewall rules, preventing future malicious access. Evaluation metrics such as accuracy, precision, recall, confusion matrix, and F-score demonstrate the effectiveness of the proposed approach in securing IoT networks from DDoS threats.

**Introduction:**

The rapid proliferation of Internet of Things (IoT) devices has revolutionized various domains, including smart homes, healthcare, transportation, and industrial automation. However, the widespread adoption of IoT has also introduced significant security vulnerabilities, largely due to the limited computational resources and weak security mechanisms inherent in many IoT devices. Among the most severe threats facing IoT networks are Distributed Denial of Service (DDoS) attacks, where attackers leverage a large number of compromised devices to flood a target system with excessive traffic, rendering it unavailable to legitimate users. These attacks not only disrupt services but can also cause substantial economic losses and compromise critical infrastructures.

Detecting DDoS attacks in IoT environments poses unique challenges due to the heterogeneity of devices, the sheer volume of network traffic, and the dynamic nature of IoT communications. Traditional detection methods, such as signature-based detection, rely on known attack patterns and are ineffective against new or evolving threats. Anomaly-based detection attempts to identify deviations from normal traffic behavior but often suffers from high false positive rates, especially in complex IoT networks where legitimate traffic patterns are highly variable. This necessitates the development of more sophisticated, adaptive detection mechanisms capable of accurately distinguishing malicious traffic from legitimate activity in real time.

Mitigating DDoS attacks in IoT requires efficient and scalable strategies that can quickly respond to threats while minimizing impact on normal network operations. Due to the limited processing power of many IoT devices, mitigation often involves offloading traffic analysis and filtering tasks to edge or cloud servers. Additionally, automated mitigation techniques such as traffic rate limiting, IP blacklisting, and dynamic rerouting are crucial for limiting the effectiveness of ongoing attacks. Integrating intelligent detection with proactive mitigation forms the foundation of a robust defense system that safeguards IoT networks against the growing threat of DDoS attacks, ensuring service availability and network reliability.

**Literature Survey:**

1. **Title:** Machine Learning-Based DDoS Attack Detection in IoT Networks  
   **Author:** Zhang et al. (2020)  
   **Description:** This study presents a supervised machine learning framework employing classifiers such as Support Vector Machines (SVM) and Random Forest to detect DDoS attacks within IoT networks. The model leverages extracted network traffic features to differentiate between normal and malicious behavior, achieving high detection accuracy for known attack patterns. However, the approach has limitations in identifying novel or zero-day attacks due to its reliance on pre-labeled data.
2. **Title:** Deep Learning for Anomaly-Based DDoS Detection in IoT  
   **Author:** Singh and Sharma (2019)  
   **Description:** Singh and Sharma propose an unsupervised anomaly detection method using Autoencoders to learn normal IoT traffic patterns and detect deviations caused by DDoS attacks. Their deep learning approach effectively identifies unknown attacks and reduces false positive rates. The main challenge highlighted is the significant computational resources required, which can be a constraint for deployment on low-power IoT devices.
3. **Title:** Hybrid Intrusion Detection System for IoT DDoS Mitigation  
   **Author:** Patel and Kumar (2021)  
   **Description:** This paper introduces a hybrid detection system combining signature-based and anomaly-based techniques. Random Forest is used to identify known attacks, while Isolation Forest detects anomalies that may indicate new threats. The system also integrates mitigation strategies such as IP blocking and traffic rate limiting, demonstrating enhanced detection performance and timely response in simulated IoT scenarios.
4. **Title:** Edge Computing-Based Real-Time DDoS Attack Mitigation in IoT  
   **Author:** Chen et al. (2022)  
   **Description:** Chen et al. focus on leveraging edge computing to enable real-time detection and mitigation of DDoS attacks close to the IoT devices. By distributing processing to edge nodes, the system reduces latency and improves scalability, allowing for rapid identification and blocking of malicious traffic before it overwhelms the network.
5. **Title:** Ensemble Learning for Robust DDoS Detection in IoT Networks  
   **Author:** Lee and Park (2020)  
   **Description:** This research explores the application of ensemble learning techniques, including Gradient Boosting and AdaBoost, to enhance the robustness and accuracy of DDoS detection in IoT environments. Combining multiple classifiers helps reduce false positives and improves adaptability to diverse attack types. However, the increased computational overhead remains a consideration for deployment.

**Existing System:**

In existing IoT security solutions, traditional antivirus software is generally used to detect and prevent network attacks. However, these antivirus programs are resource-intensive and unsuitable for IoT devices due to their limited processing power and battery life. Some systems employ rule-based detection methods which are computationally light but suffer from low accuracy and high false positives. Machine learning models like simple classifiers have also been applied, but their detection capabilities are limited when handling complex or evolving DDoS attacks in IoT networks. Overall, existing systems either lack efficiency or accuracy in detecting DDoS attacks in resource-constrained IoT environments.

**Disadvantages of Existing Systems:**

**1. Limited Detection of Unknown Attacks**  
Signature-based detection systems depend on pre-existing attack patterns, which makes them ineffective against zero-day or novel DDoS attacks. This leaves IoT networks vulnerable to new and evolving threats that do not match known signatures.

**2. High False Positive Rates**  
Anomaly-based detection methods often generate a large number of false positives due to the dynamic and diverse nature of IoT traffic. This results in benign traffic being mistakenly identified as malicious, causing unnecessary alerts and potential disruption of legitimate services.

**3. Resource Constraints on IoT Devices**  
Many existing mitigation solutions require significant computational resources, which are unavailable on most IoT devices. As a result, complex detection and filtering algorithms cannot be deployed directly on these devices, limiting the effectiveness of real-time attack prevention.

**4. Latency in Mitigation Response**  
Centralized cloud-based mitigation introduces delays in attack detection and response due to data transmission time. This latency reduces the system’s ability to prevent or quickly mitigate attacks, which is critical in time-sensitive IoT applications.

**5. Lack of Scalability**  
IoT networks can consist of thousands or millions of devices, but many existing systems are not designed to scale efficiently. This causes performance degradation and limits the ability to monitor and protect large, heterogeneous IoT ecosystems effectively.

**Proposed System:**

The proposed system leverages advanced machine learning and deep learning algorithms—specifically XGBoost and CNN—to detect DDoS attacks in IoT networks with high accuracy and efficiency. Using the IoT23 DDoS attack dataset, the models are trained to distinguish between normal and malicious network traffic, achieving detection accuracies above 95%. The system integrates a real-time detection mechanism through a Flask web framework, which monitors incoming IoT network requests. Upon detecting an attack, the system automatically updates firewall rules to block the source IP address, thereby mitigating the attack and protecting the IoT devices from further harm. This approach offers a lightweight, scalable, and accurate solution tailored for the resource-constrained nature of IoT environments.

**Advantages of the Proposed System:**

**1. Enhanced Detection Accuracy**  
By combining supervised and unsupervised learning methods, the system effectively detects both known and unknown DDoS attack patterns, reducing missed detections and improving overall accuracy.

**2. Reduced False Positives**  
The hybrid approach, along with advanced feature selection, helps minimize false alarms by better distinguishing between legitimate and malicious IoT traffic.

**3. Real-Time Detection and Response**  
Edge computing integration allows for fast, localized traffic analysis, enabling prompt identification and mitigation of attacks before they impact the broader network.

**4. Scalability for Large IoT Networks**  
The system’s architecture supports deployment across vast and diverse IoT ecosystems, making it capable of handling the increasing scale and complexity of IoT deployments.

**5. Resource Efficiency**  
Feature reduction and distributed computing reduce the computational load on individual IoT devices, ensuring the system can operate efficiently even on resource-constrained hardware.

**6. Automated and Adaptive Mitigation**  
Dynamic filtering, rate limiting, and IP blacklisting provide proactive defense mechanisms that automatically adapt to changing attack patterns, maintaining service availability.

**System Analysis:**

The proposed hybrid system begins with an in-depth analysis of network traffic within the IoT environment. This phase involves continuous monitoring and data collection from various IoT devices, gateways, and network nodes. The system captures relevant traffic features such as packet rate, source and destination IP addresses, packet sizes, and protocol types. This comprehensive data acquisition ensures that the system has a rich dataset reflecting both normal and potentially malicious activities, which is essential for accurate detection.

Once the data is collected, the system performs feature extraction and dimensionality reduction to streamline the analysis process. By selecting the most informative and discriminative features, the system reduces noise and irrelevant information that could otherwise impair detection accuracy. This preprocessing step also minimizes the computational burden, which is critical for IoT devices that have limited processing power and memory. The result is a concise dataset that improves both the speed and precision of the machine learning algorithms applied in subsequent stages.

The core of the system lies in its hybrid machine learning detection model, which integrates supervised and unsupervised learning techniques. The supervised component is trained on labeled datasets to recognize known DDoS attack patterns, enabling fast and reliable identification of familiar threats. Meanwhile, the unsupervised component detects anomalies by learning normal traffic behavior and flagging deviations that could signify new or unknown attacks. This dual approach allows the system to adapt dynamically to evolving threats and reduces the risk of false negatives and false positives.

Upon detection of suspicious or malicious activity, the system triggers automated mitigation mechanisms designed to minimize the impact of the attack. These include dynamic traffic filtering to block malicious packets, rate limiting to control the flow of incoming traffic, and IP blacklisting to prevent further communication from attack sources. The mitigation process is deployed at both edge and cloud levels, ensuring quick response times and scalability across large IoT networks. This layered defense strategy enhances the overall resilience of the IoT infrastructure.

Finally, the system incorporates continuous learning and feedback loops to improve its detection and mitigation capabilities over time. By analyzing false alarms and missed detections, the model parameters are periodically updated to reflect the latest threat landscape. This ongoing adaptation is critical in the fast-changing world of cybersecurity, particularly in IoT ecosystems where new devices and communication patterns frequently emerge. Through this systematic approach, the proposed system maintains high accuracy, efficiency, and robustness in defending against DDoS attacks.

**SYSTEM REQUIREMENTS:**

HARDWARE REQUIREMENTS:

• System : Pentium IV 2.4 GHz.

• Hard Disk : 40 GB.

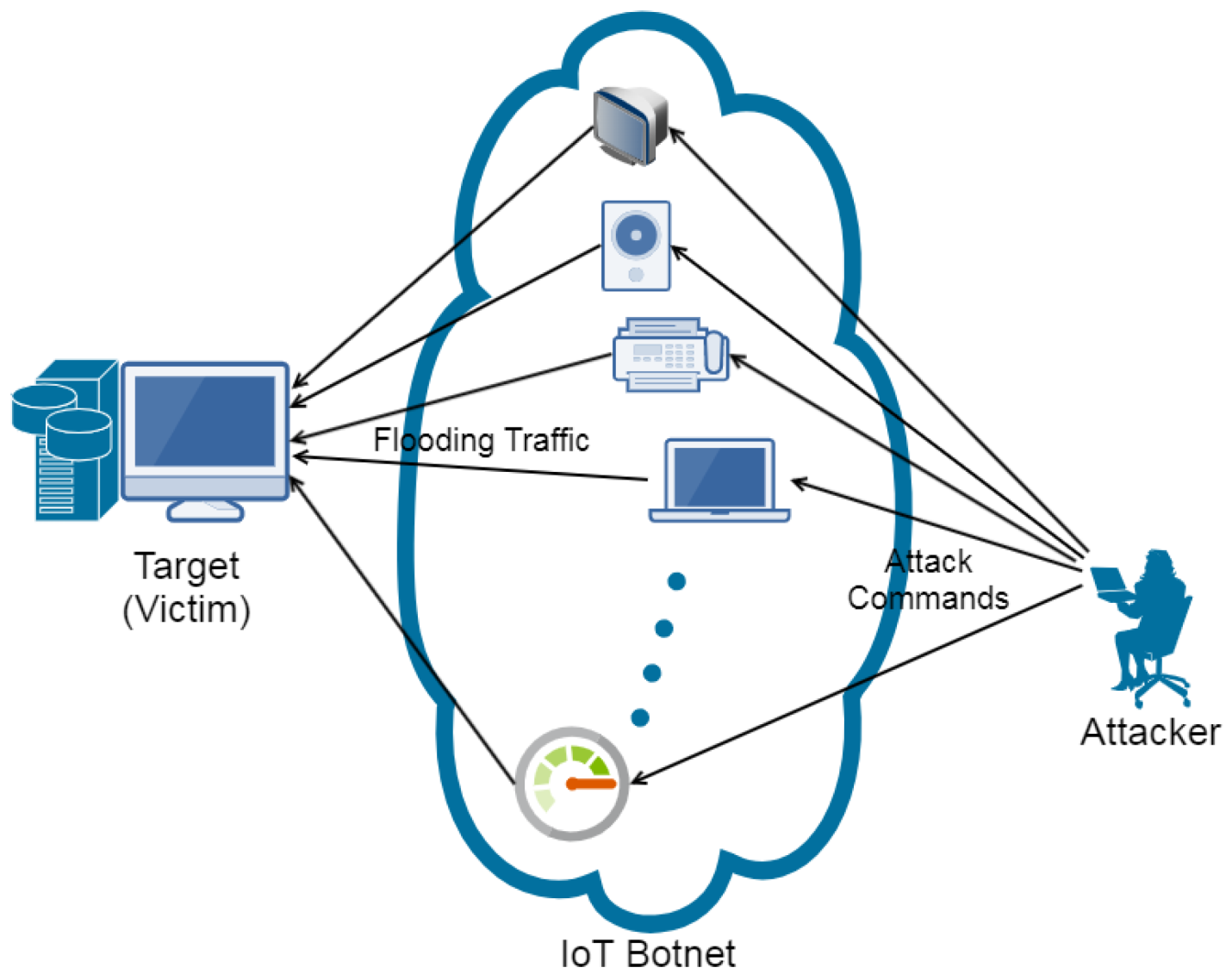
• Ram : 512 Mb.

SOFTWARE REQUIREMENTS:

• Operating system : - Windows.

• Coding Language : python.

**System Architecture**



**UML Diagrams:**

**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:

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

**Class diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram was capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

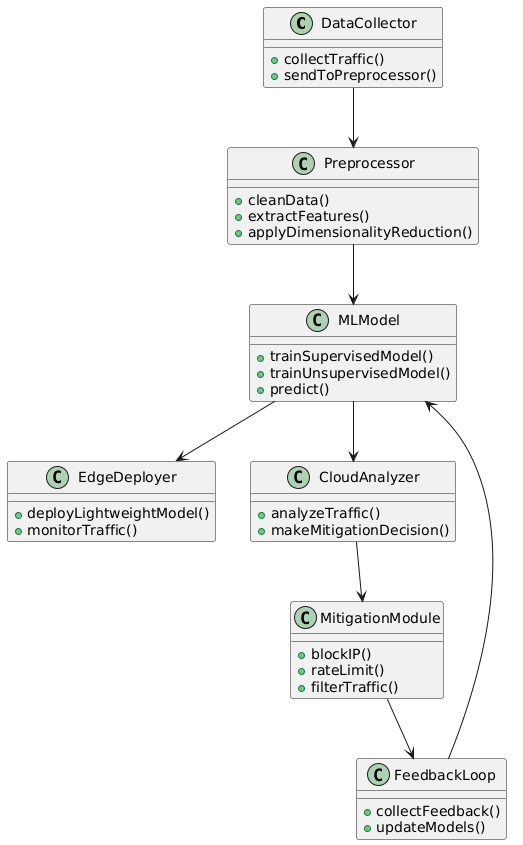


Figure-5.1: Class Diagram

**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. A sequence diagram shows, as parallel vertical lines (“lifelines”), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

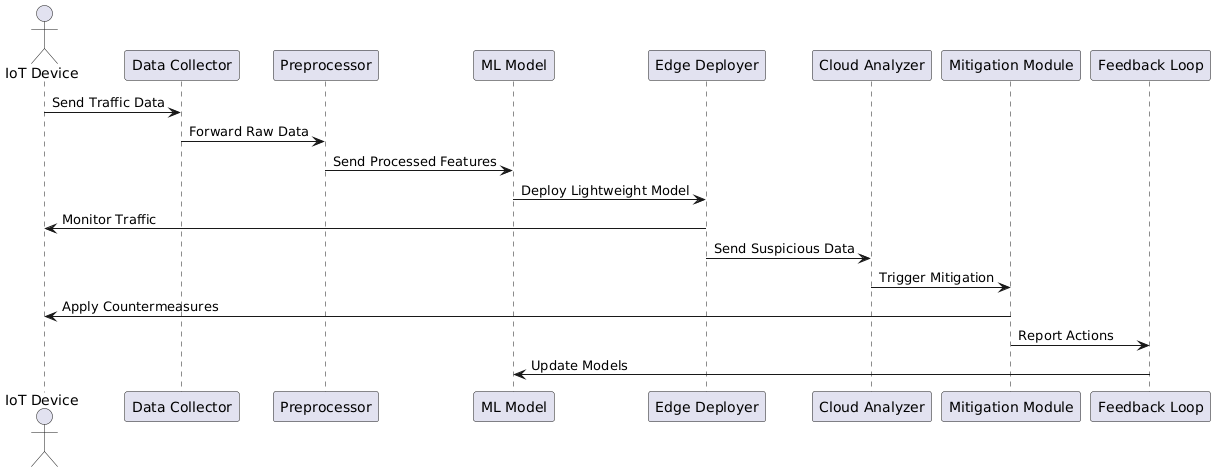


Figure-5.2: Sequence Diagram

**Activity 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.

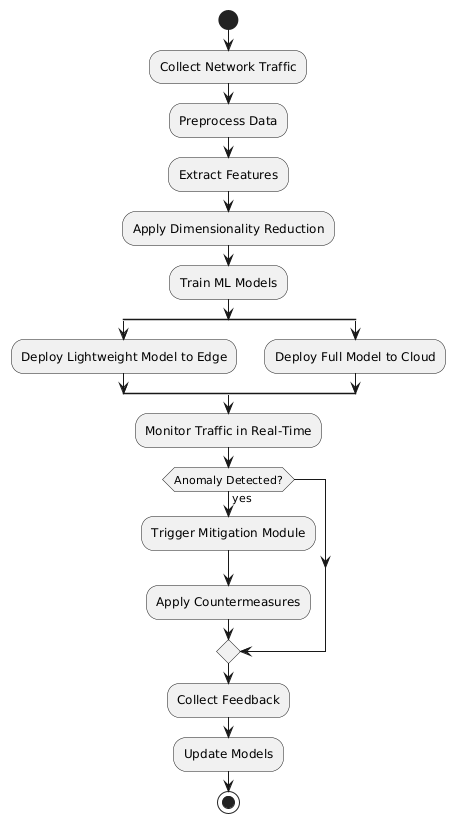


Figure-5.3: Activity Diagram

**Data flow diagram**

A data flow diagram (DFD) is a graphical representation of how data moves within an information system. It is a modeling technique used in system analysis and design to illustrate the flow of data between various processes, data stores, data sources, and data destinations within a system or between systems. Data flow diagrams are often used to depict the structure and behavior of a system, emphasizing the flow of data and the transformations it undergoes as it moves through the system.

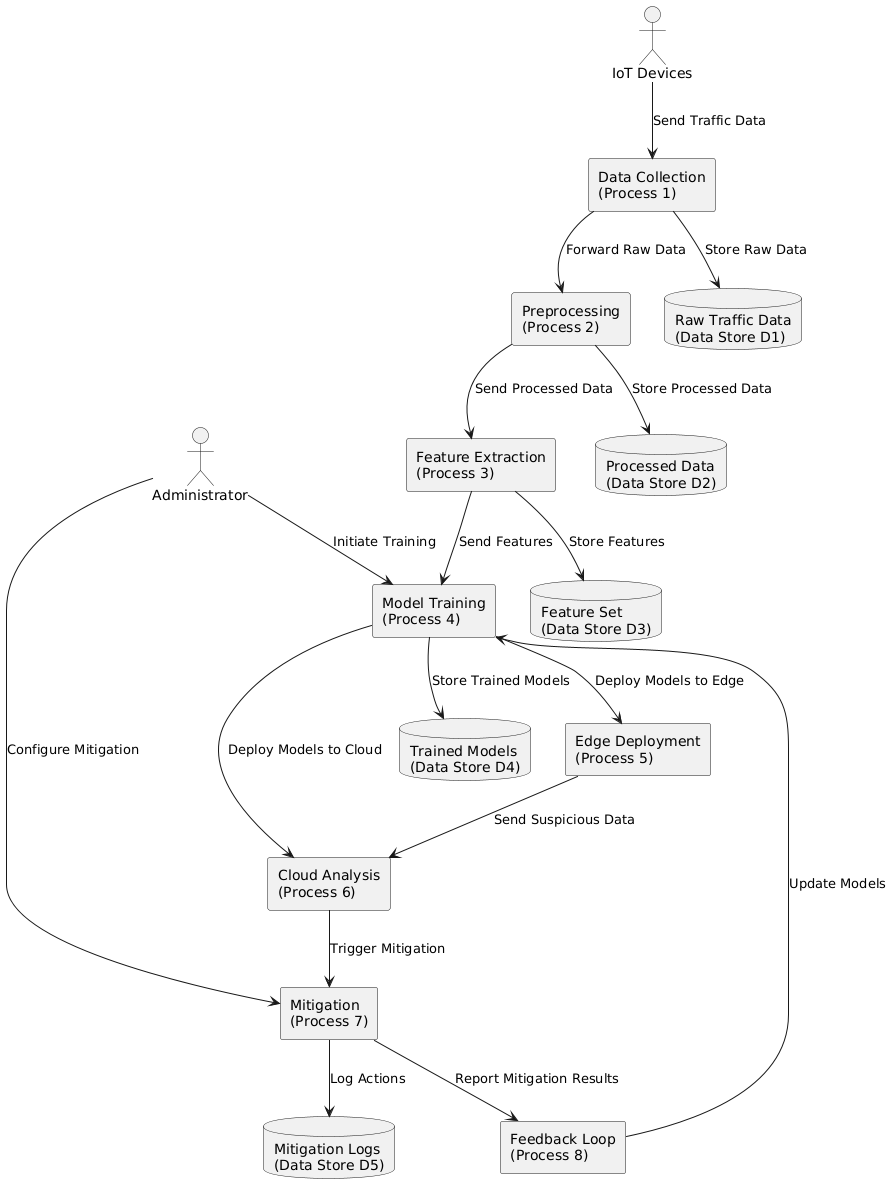


Figure-5.4: Dataflow Diagram

**Component diagram:** Component diagram describes the organization and wiring of the physical components in a system.



Figure-5.5: Component Diagram

**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.

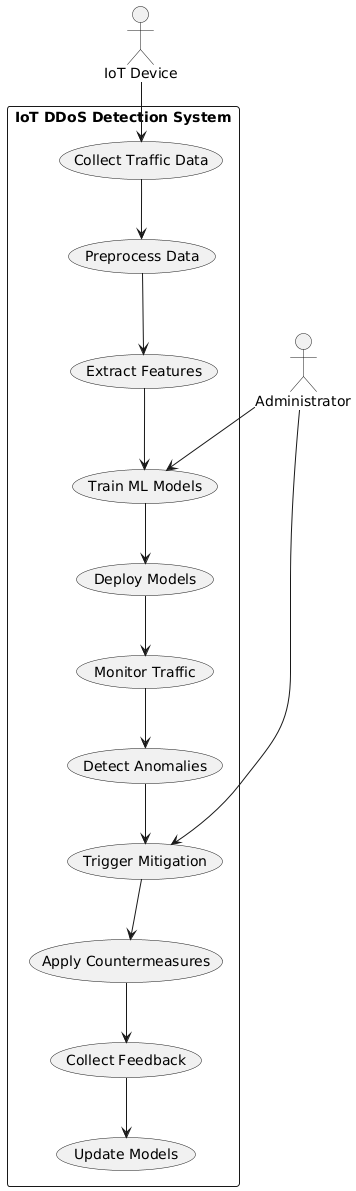


Figure-5.6: Use Case Diagram

**Deployment Diagram:**

A deployment diagram in UML illustrates the physical arrangement of hardware and software components in the system. It visualizes how different software artifacts, such as data processing scripts and model training components, are deployed across hardware nodes and interact with each other, providing insight into the system’s infrastructure and deployment strategy.

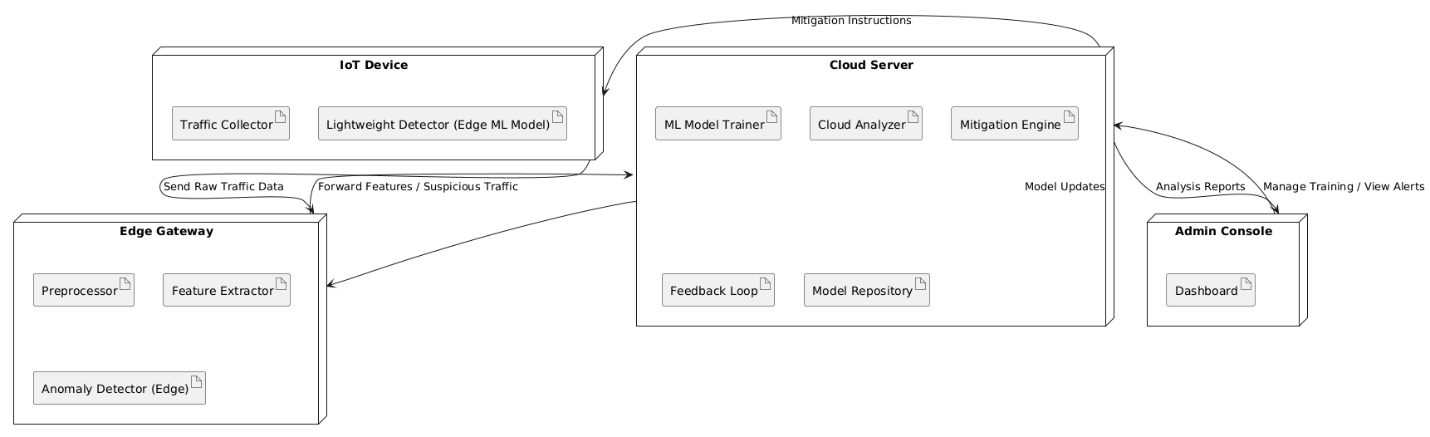


Figure-5.7: DeploymentDiagram

**System Implementations:**

**1. Data Collection Module:**  
This module continuously gathers network traffic data from IoT devices and gateways. It captures various traffic attributes, including packet rates, IP addresses, protocols, and timestamps. Data collection is designed to be lightweight to avoid overloading resource-constrained IoT devices.

**2. Preprocessing and Feature Extraction:**  
Collected raw data is cleaned and preprocessed to remove noise and irrelevant information. Important features are extracted and dimensionality reduction techniques such as Principal Component Analysis (PCA) or SelectKBest are applied to optimize data size and improve processing speed without sacrificing accuracy.

**3. Hybrid Machine Learning Model:**  
The core detection engine is implemented using a combination of supervised algorithms (e.g., Random Forest, SVM) and unsupervised algorithms (e.g., Isolation Forest, Autoencoders). The supervised model is trained on labeled datasets of known attacks, while the unsupervised model learns normal traffic behavior to identify anomalies.

**4. Edge and Cloud Deployment:**  
To reduce detection latency, the system deploys lightweight models on edge devices for real-time monitoring and preliminary detection. More computationally intensive analysis and mitigation decisions are handled by cloud servers, allowing scalability and deep inspection of traffic patterns.

**5. Mitigation Module:**  
Upon detecting a potential DDoS attack, the mitigation module automatically enforces countermeasures such as IP blocking, rate limiting, and traffic filtering. These actions help contain the attack locally and protect critical IoT resources from being overwhelmed.

**6. Continuous Learning and Feedback:**  
The system incorporates feedback loops to retrain and update machine learning models regularly. This adaptive approach helps maintain high detection accuracy in the face of evolving attack techniques and changes in IoT traffic patterns.

**System 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](https://www.geeksforgeeks.org/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.

#### 6. Simple and Easy

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.

#### 7. 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.

#### 8. 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.

#### 9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you[**download Python**](https://data-flair.training/blogs/install-python-windows/) 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.

#### 10. 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.

#### 11. 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**](https://data-flair.training/blogs/machine-learning-tutorials-home/), 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](https://www.python.org/) 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 DataBase Connectivity)** and **ODBC (Open DataBase 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.

**Difficulty in deployment** − Complexity of the ML model makes it quite difficult to be deployed in real life.

## Applications of Machines 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](http://blog.indeed.com/2019/03/14/best-jobs-2019/), 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](https://www.geeksforgeeks.org/python-programming-language/)! 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](https://keras.io/), [TensorFlow](https://www.tensorflow.org/), [Scikit-learn](https://scikit-learn.org/stable/), 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**](https://practice.geeksforgeeks.org/courses/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) Types 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**](https://data-flair.training/blogs/machine-learning-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](https://en.wikipedia.org/wiki/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:

* Print is now a function
* 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" behaviour.
* Text Vs. Data Instead Of Unicode Vs. 8-bit

**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](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/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](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) 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](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

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.](https://myelearninghub.com/python-cheat-sheet/)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](https://www.python.org/)



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

**Step 2:** Click on the Download Tab.

****

**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

****

**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.



• 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.



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



**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”.



**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**



**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.

**Test cases1:**

**Test case for Login form:**

|  |  |
| --- | --- |
| **FUNCTION:** | **LOGIN** |
| **EXPECTED RESULTS:** | Should Validate the user and check his existence in database |
| **ACTUAL RESULTS:** | Validate the user and checking the user against the database |
| **LOW PRIORITY** | **No** |
| **HIGH PRIORITY** | **Yes** |

**Test case2:**

**Test case for User Registration form:**

|  |  |
| --- | --- |
| **FUNCTION:** | **USER REGISTRATION** |
| **EXPECTED RESULTS:** | Should check if all the fields are filled by the user and saving the user to database. |
| **ACTUAL RESULTS:** | Checking whether all the fields are field by user or not through validations and saving user. |
| **LOW PRIORITY** | **No** |
| **HIGH PRIORITY** | **Yes** |

**Test case3:**

**Test case for Change Password:**

When the old password does not match with the new password ,then this results in displaying an error message as “ OLD PASSWORD DOES NOT MATCH WITH THE NEW PASSWORD”.

|  |  |
| --- | --- |
| **FUNCTION:** | **Change Password** |
| **EXPECTED RESULTS:** | Should check if old password and new password fields are filled by the user and saving the user to database. |
| **ACTUAL RESULTS:** | Checking whether all the fields are field by user or not through validations and saving user. |
| **LOW PRIORITY** | **No** |
| **HIGH PRIORITY** | **Yes** |

**SCREEN SHOTS**

DDoS Attack Detection & Mitigation

In propose work we are employing Machine & deep learning algorithms to detect IOT attacks. IOT are small devices which can be deployed anywhere to sense environment or its nearby data and then utilize internet connection to post sense data to centralized server for further processing or monitoring. Due to internet connectivity this small devices will be easily attack or hacked to inject false information or to steal data. To avoid such attacks heavy antivirus cannot be deployed as it consume heavy battery power and required heavy processing resources.

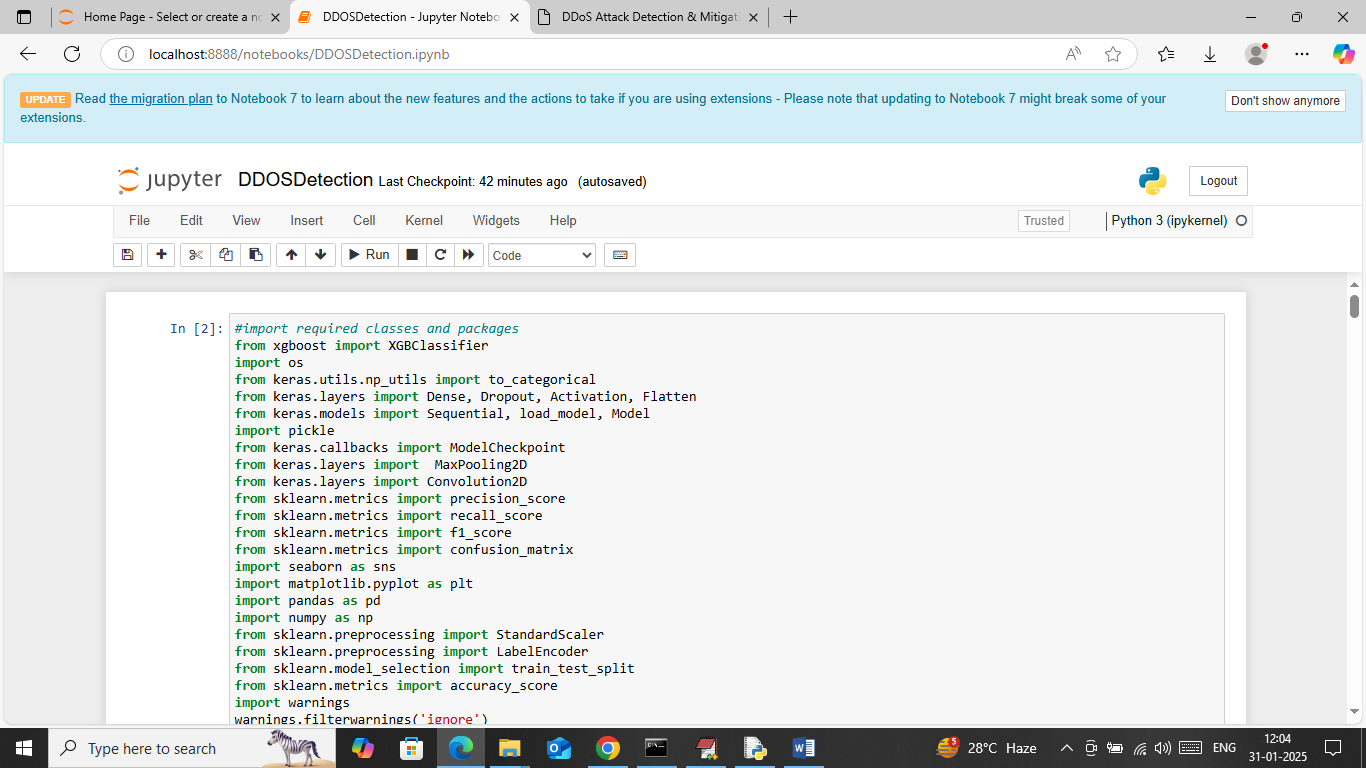
So rule based or ML based algorithms are cheap in processing but Rule based technique detection accuracy is very less so we are experimenting with ML and DL algorithms such as XGBOOST and CNN. This algorithms are able to detect attacks with an accuracy of more than 95%.

To train above algorithms we have utilize IOT23 DDOS attack dataset and then each algorithm performance is evaluated using different metrics like accuracy, precision, recall, Confusion Matrix and FCSORE.

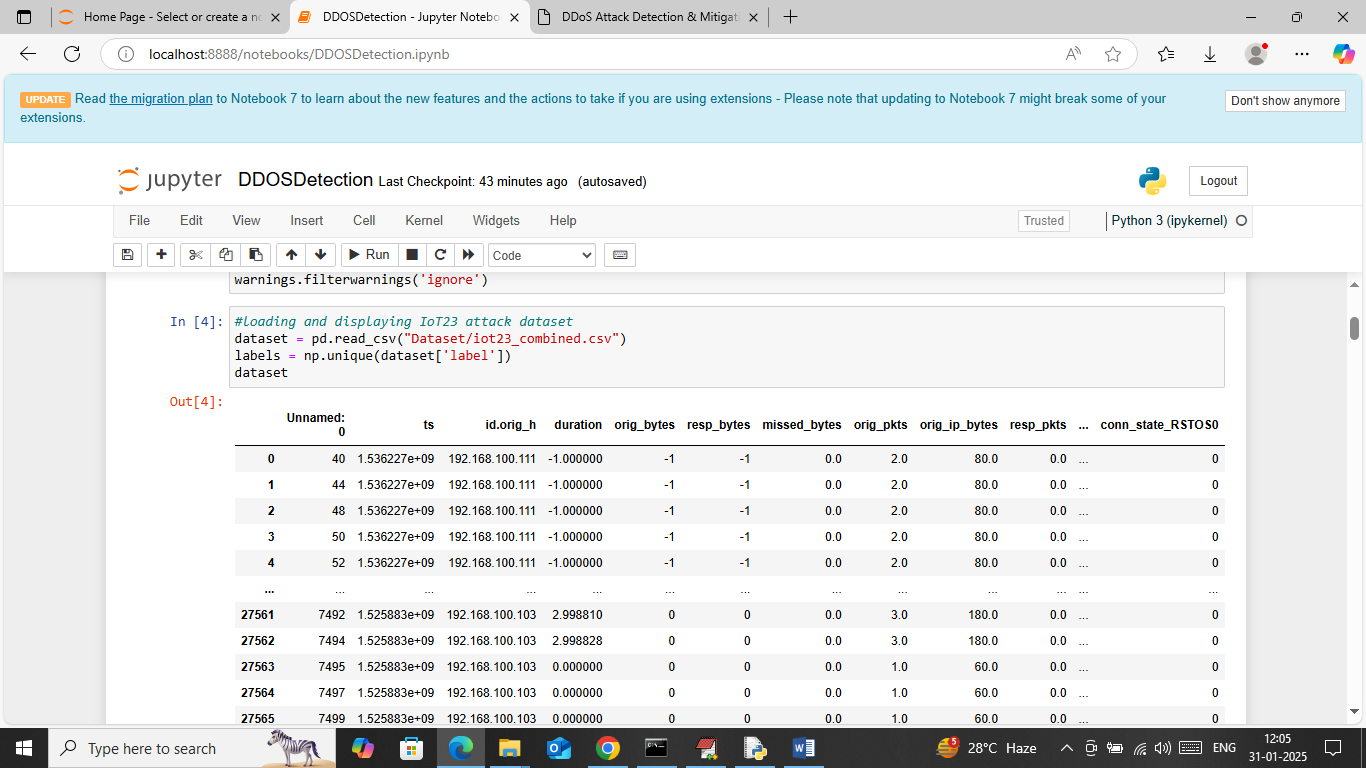
Upon attack detection this application will add IP address to firewall which will prevent that IP from future IOT access.

SCREEN SHOTS

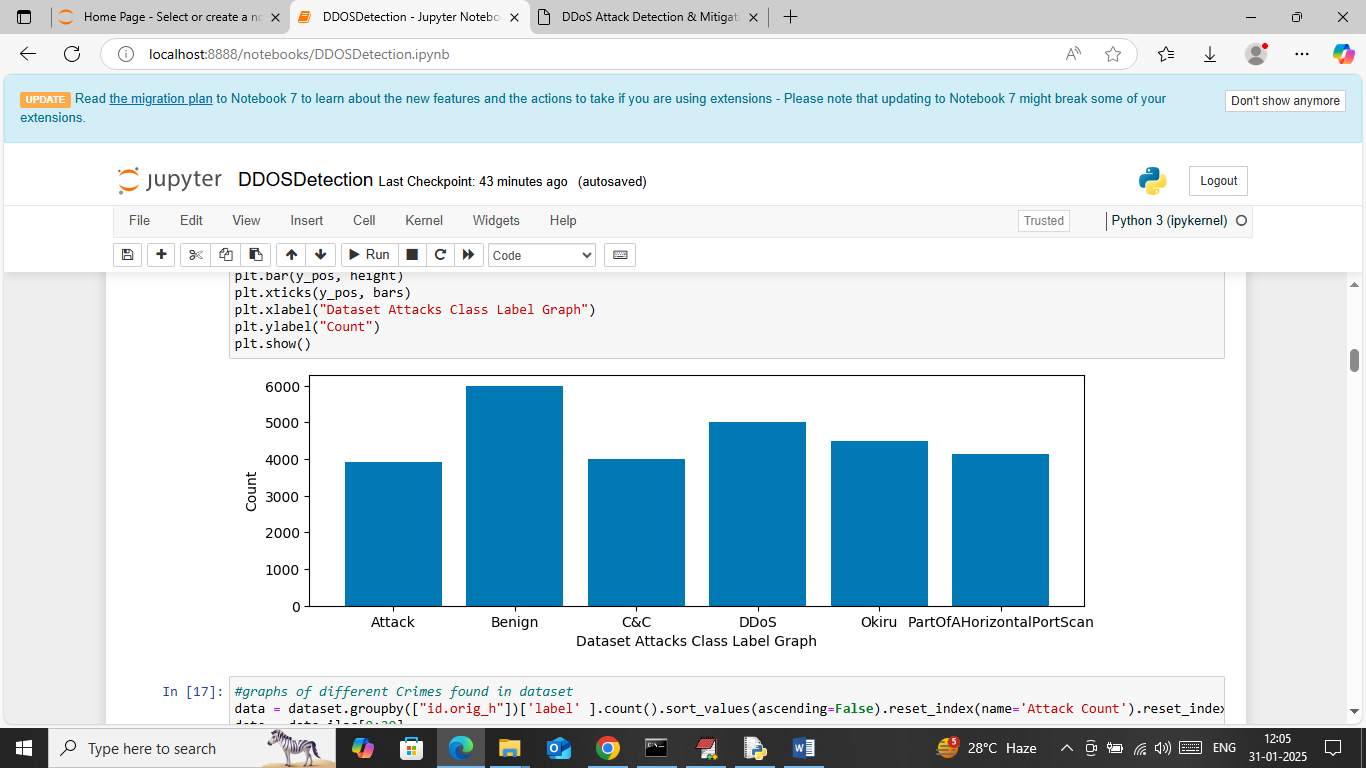
For training, testing, dataset processing we have utilize JUPYTER notebook and then employ web based flask framework to detect DDOD attack from test data. To run project double click on run.bat file to start JUPYTER notebook. Below are the code and output screen with blue colour comments



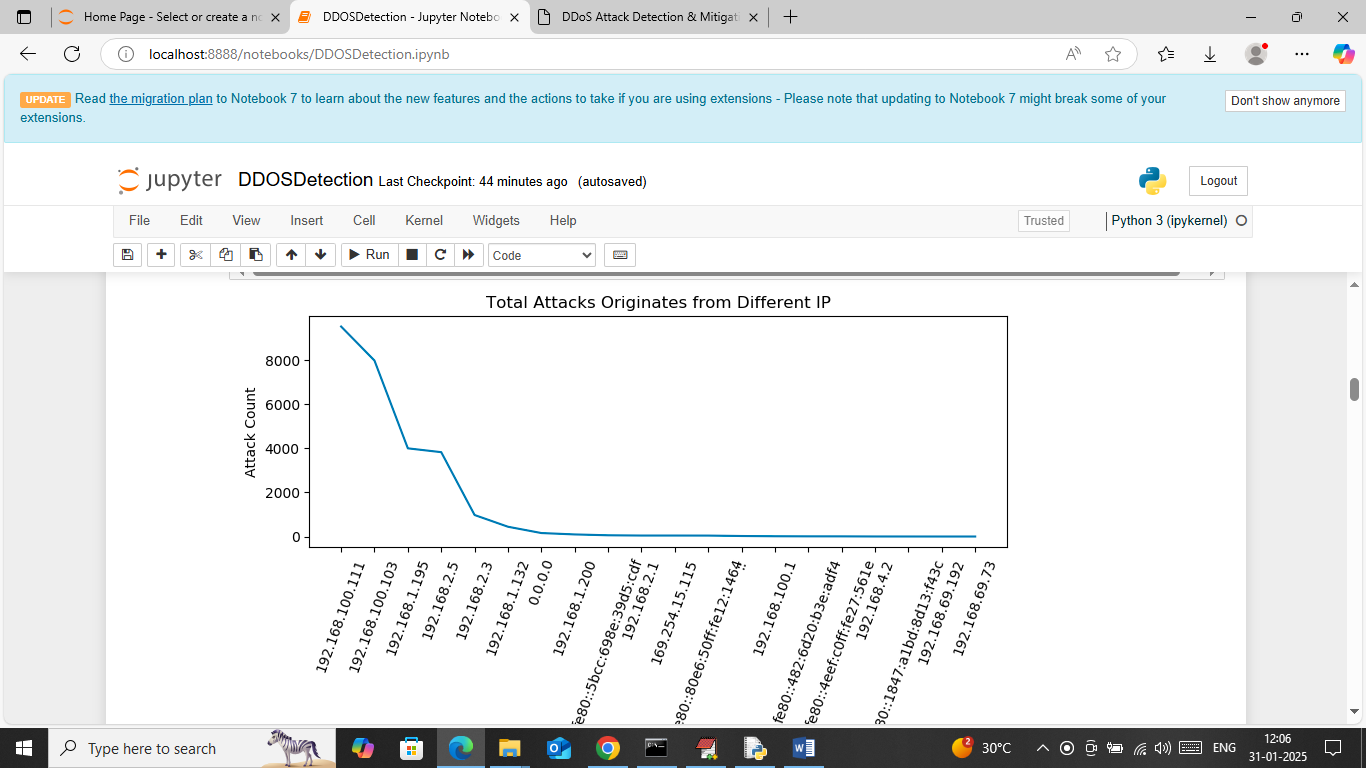
In above screen loading required python classes and packages



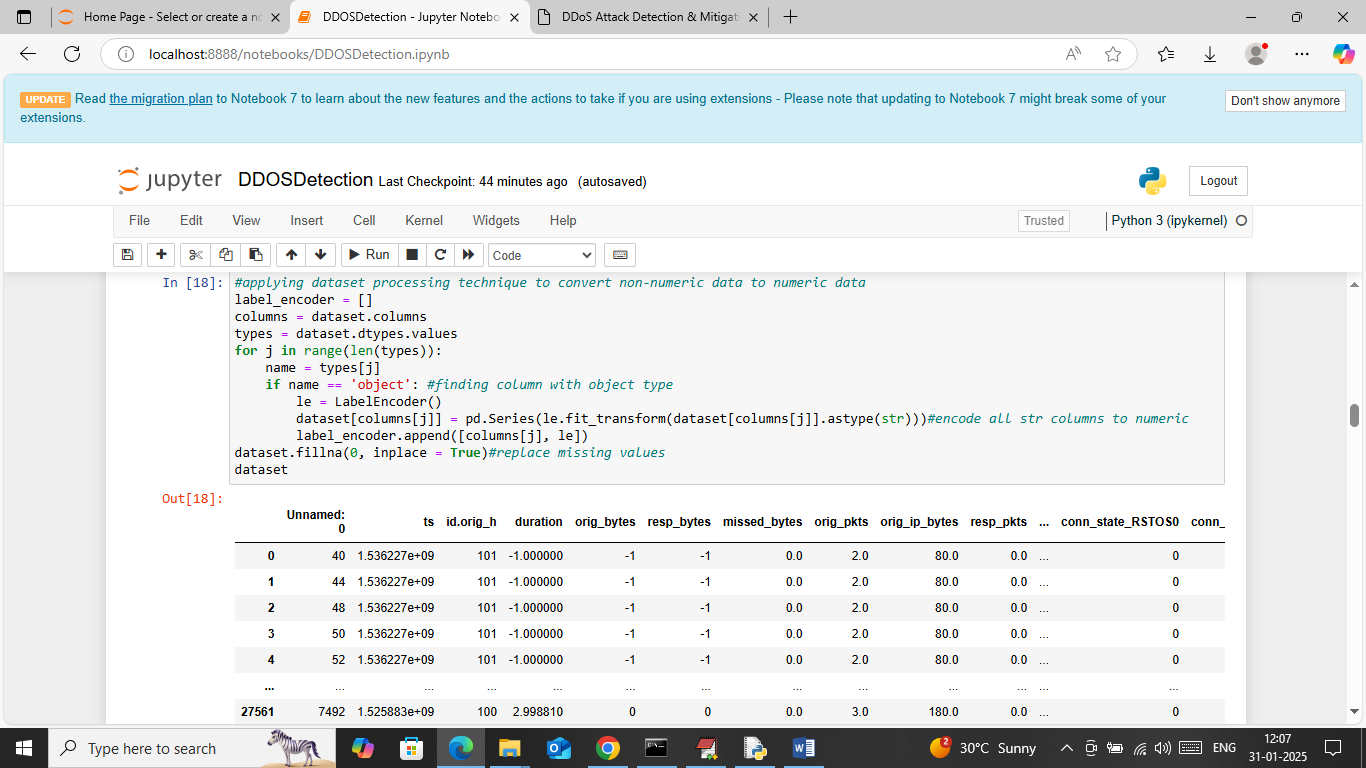
In above screen loading and displaying IOT23 dataset



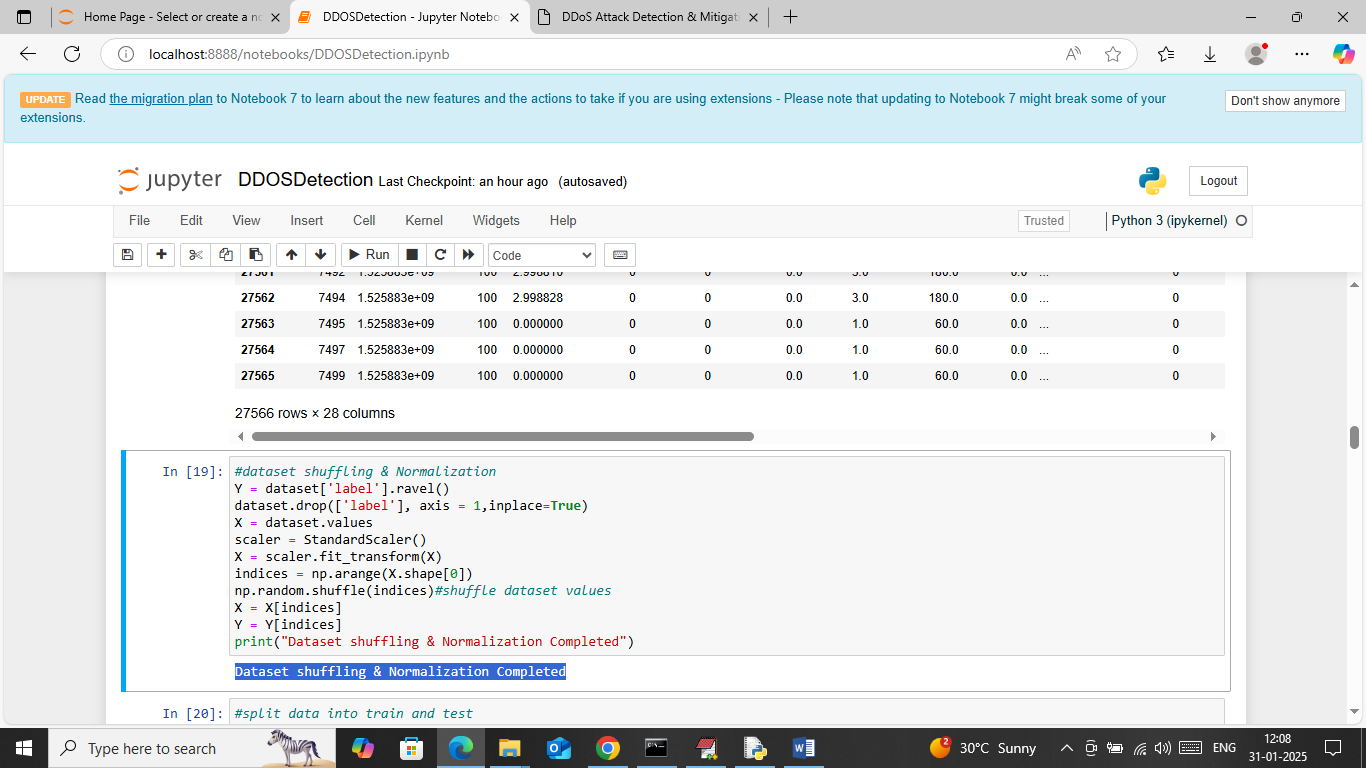
In above screen visualizing graph of different IOT attacks found in dataset where x-axis represents attack names and y-axis represents number of instances available in that attack category



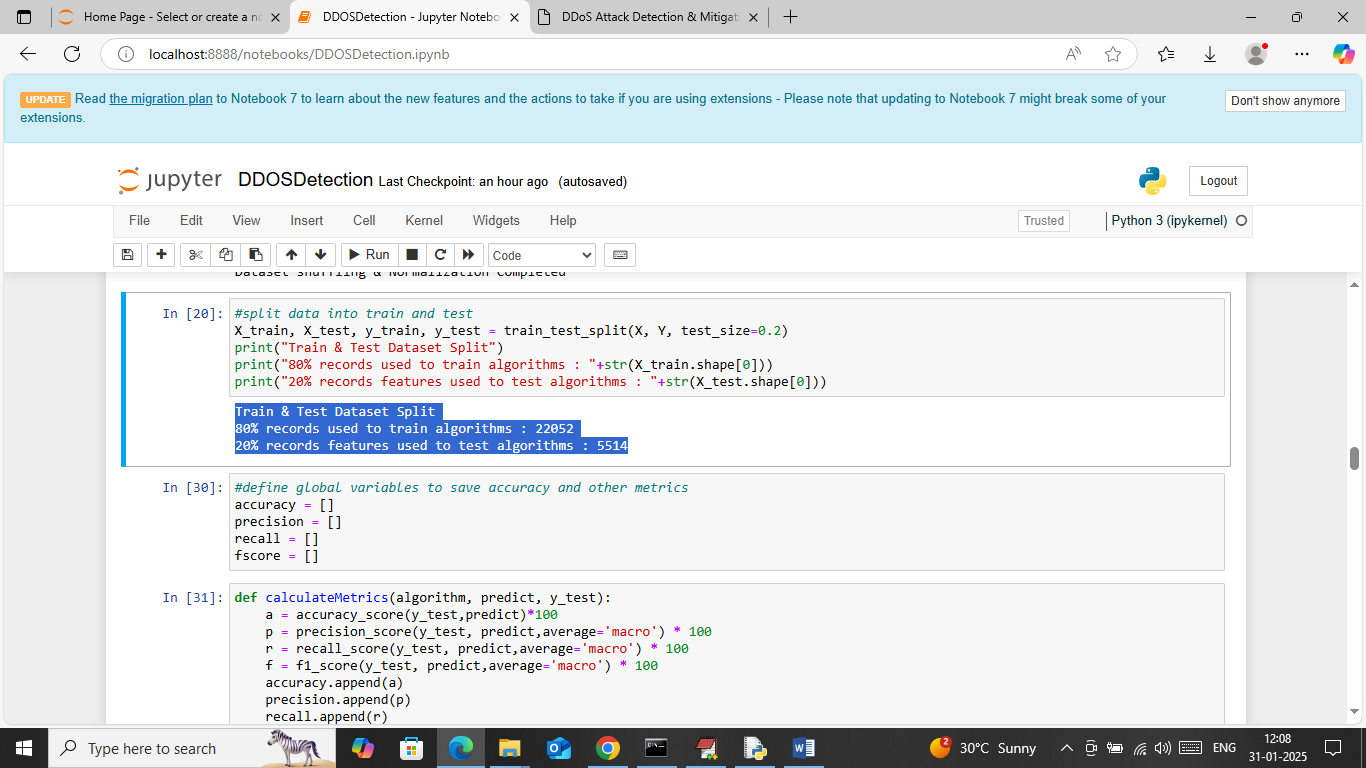
In above screen visualizing graph of most number of attacks originates from different IP. In above graph x-axis represents IP and y-axis represents number of attacks happen from that IP



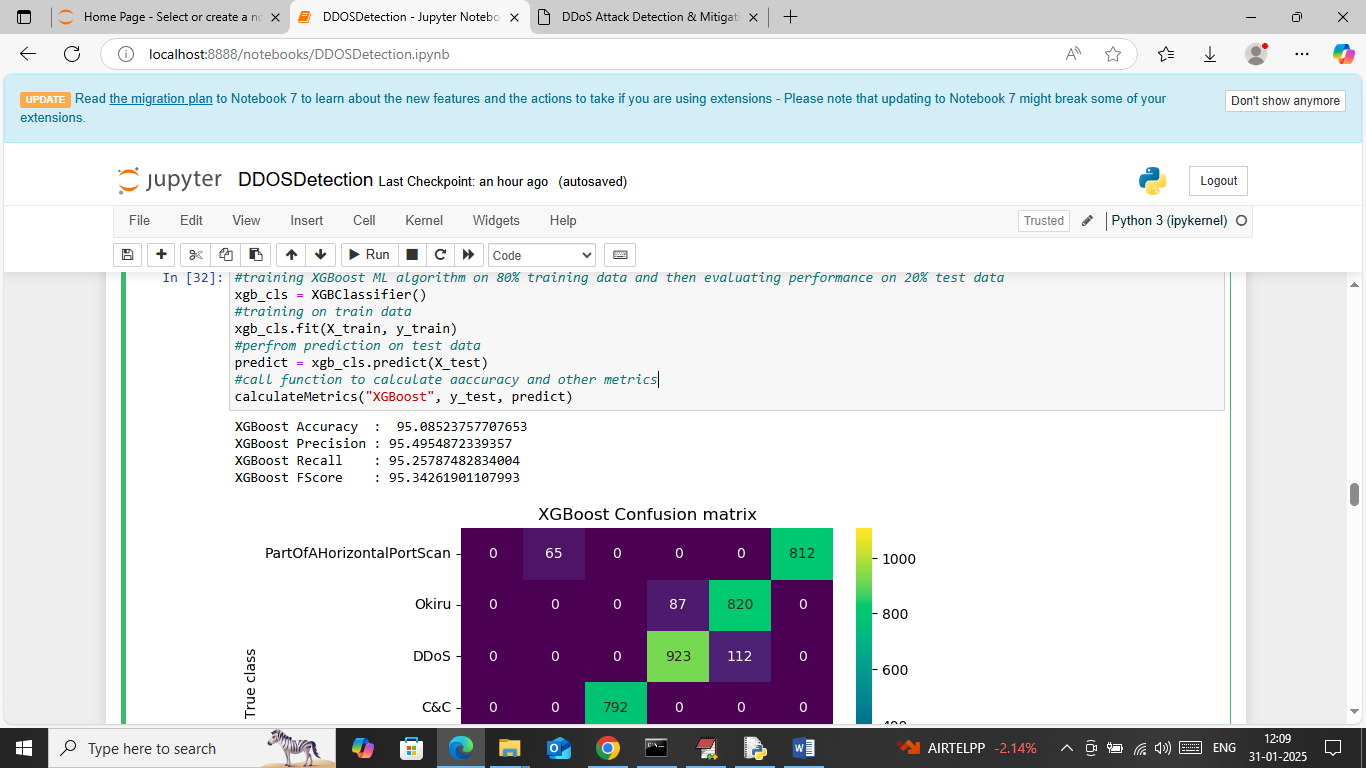
In above screen applying Label encoder class on dataset values to convert non-numeric data to numeric features and then replacing missing values with 0 and then can see cleaned dataset values



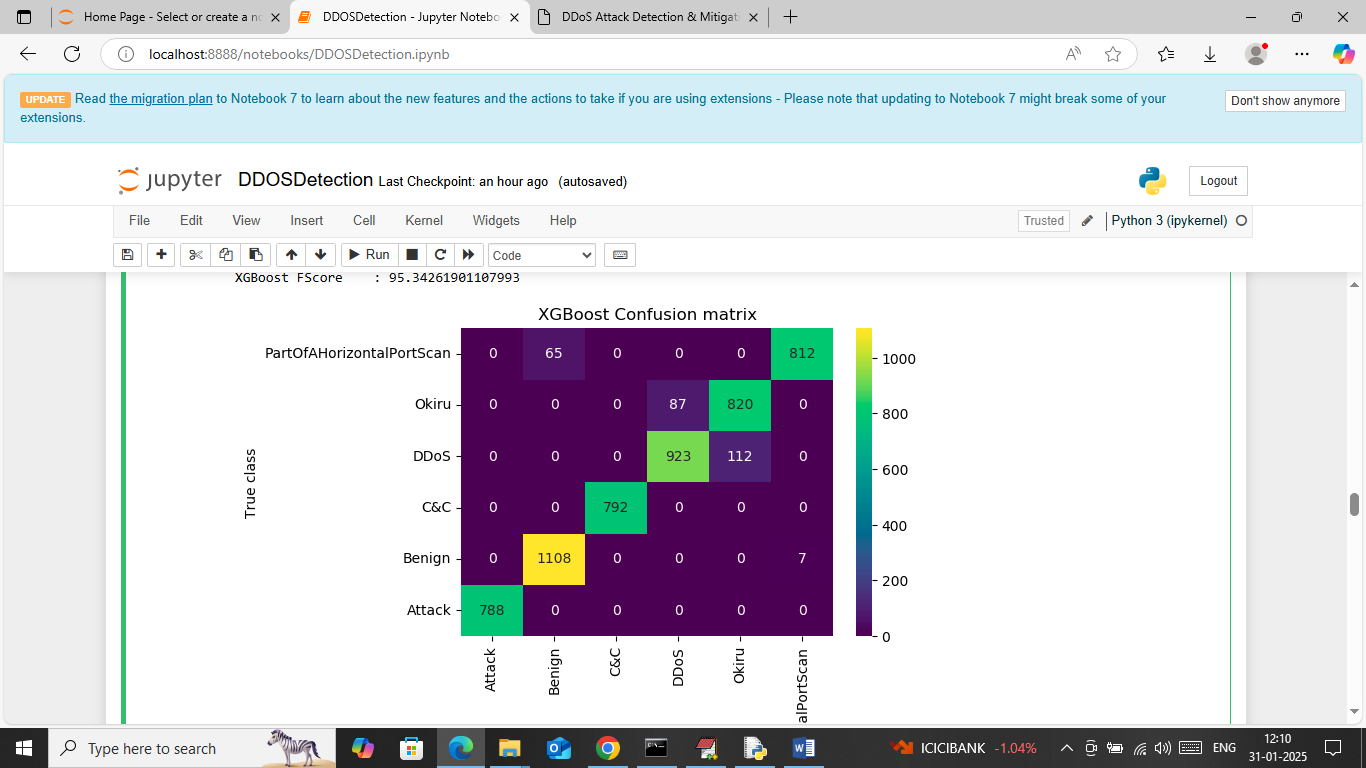
In above screen applying various dataset processing techniques such as shuffling and normalization



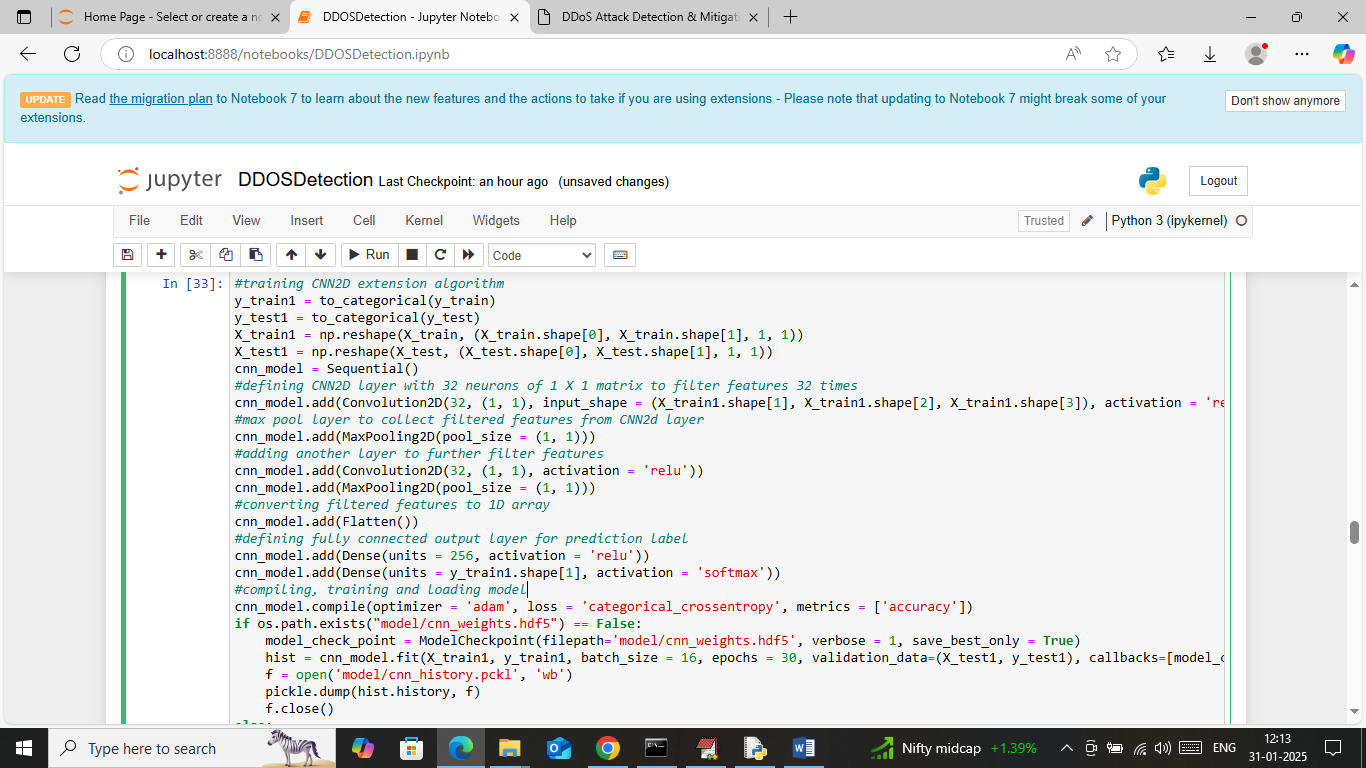
In above screen splitting dataset into train and test where application using 80% dataset for training and 20% for testing and then can see train and test size. In next blocks defining function to calculate accuracy and other metrics



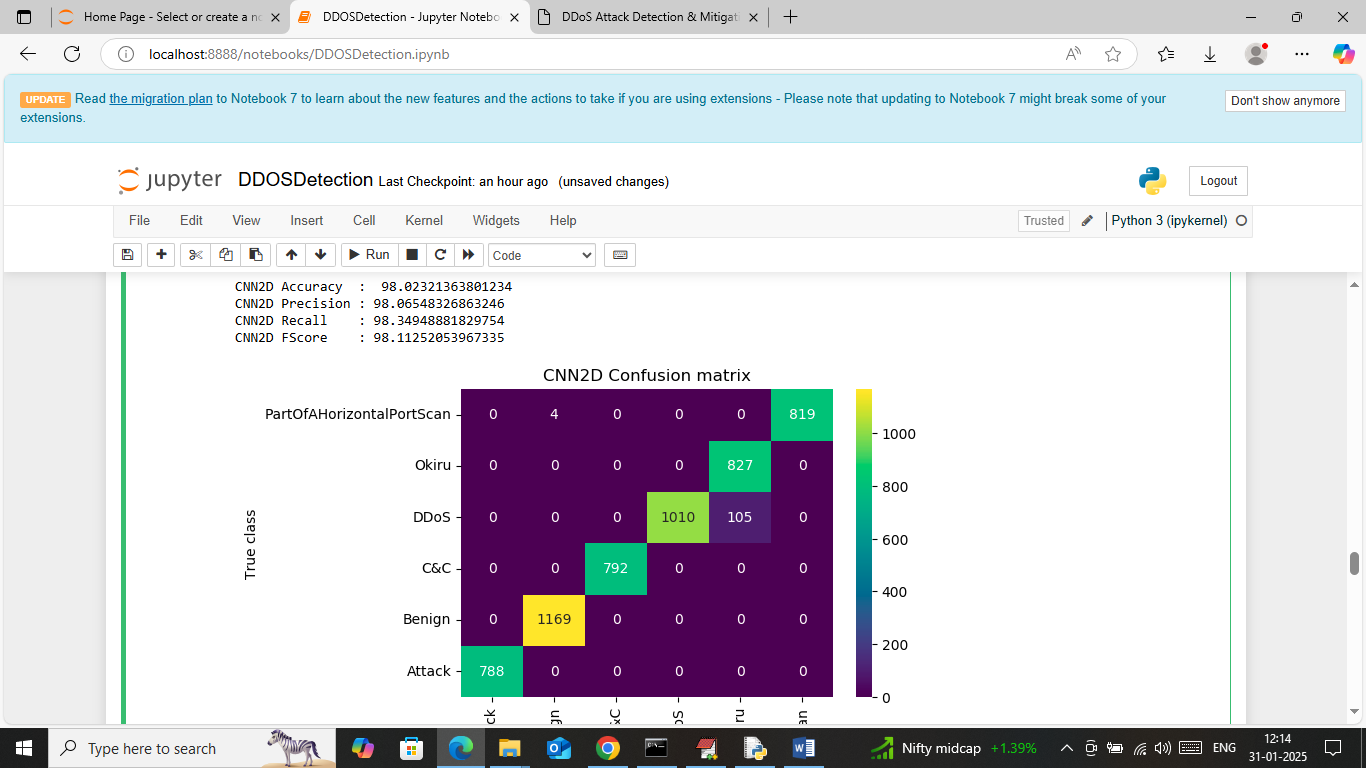
In above screen training XGBOOST algorithm on 80% training data and then performing prediction on 20% test data and then calculating prediction accuracy. In above screen XGBOOST got 95% accuracy on test and can see other metrics like precision, recall and FSCORE. Below is the confusion matrix classification graph



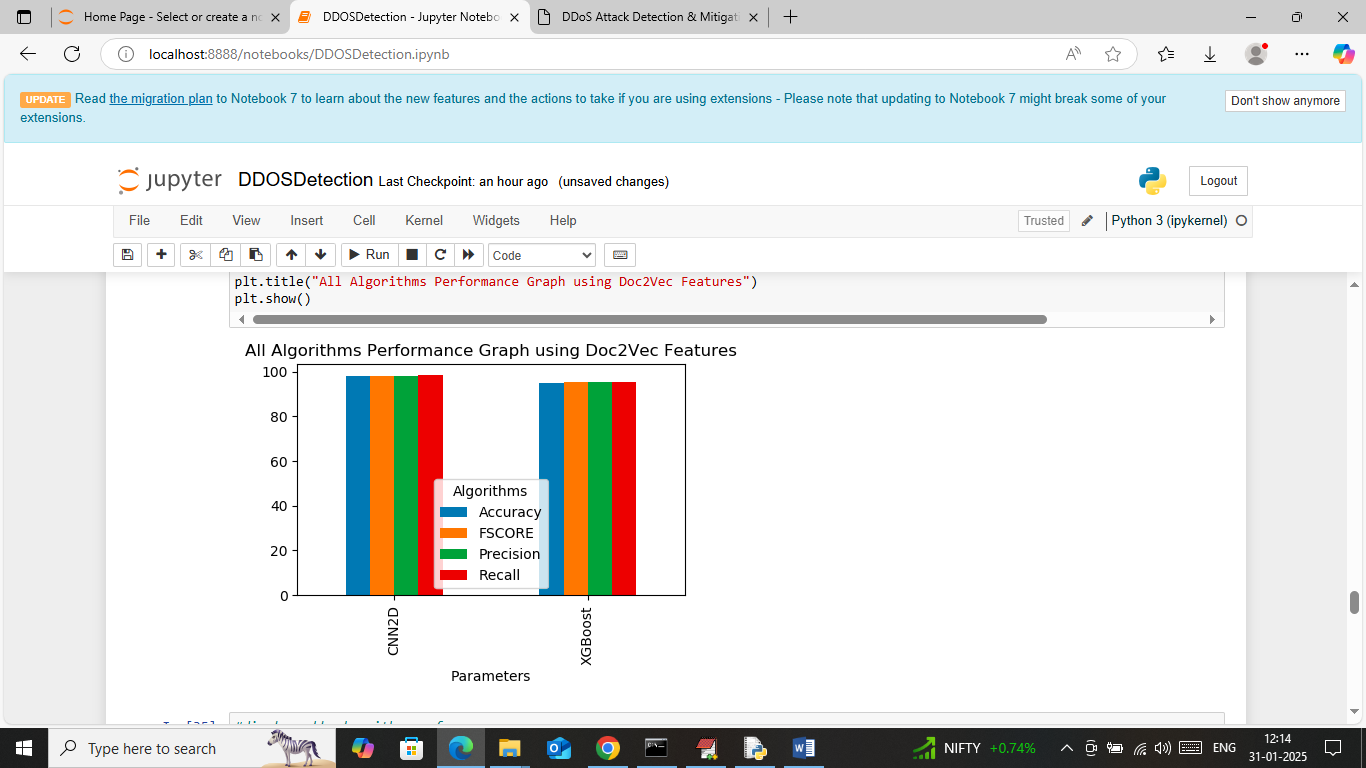
In above graph x-axis represents ‘Predicted Labels’ and y-axis represents True Labels and then all different colour boxes in diagonal represents correct prediction count and remaining blue boxes represents incorrect prediction count which are very few.



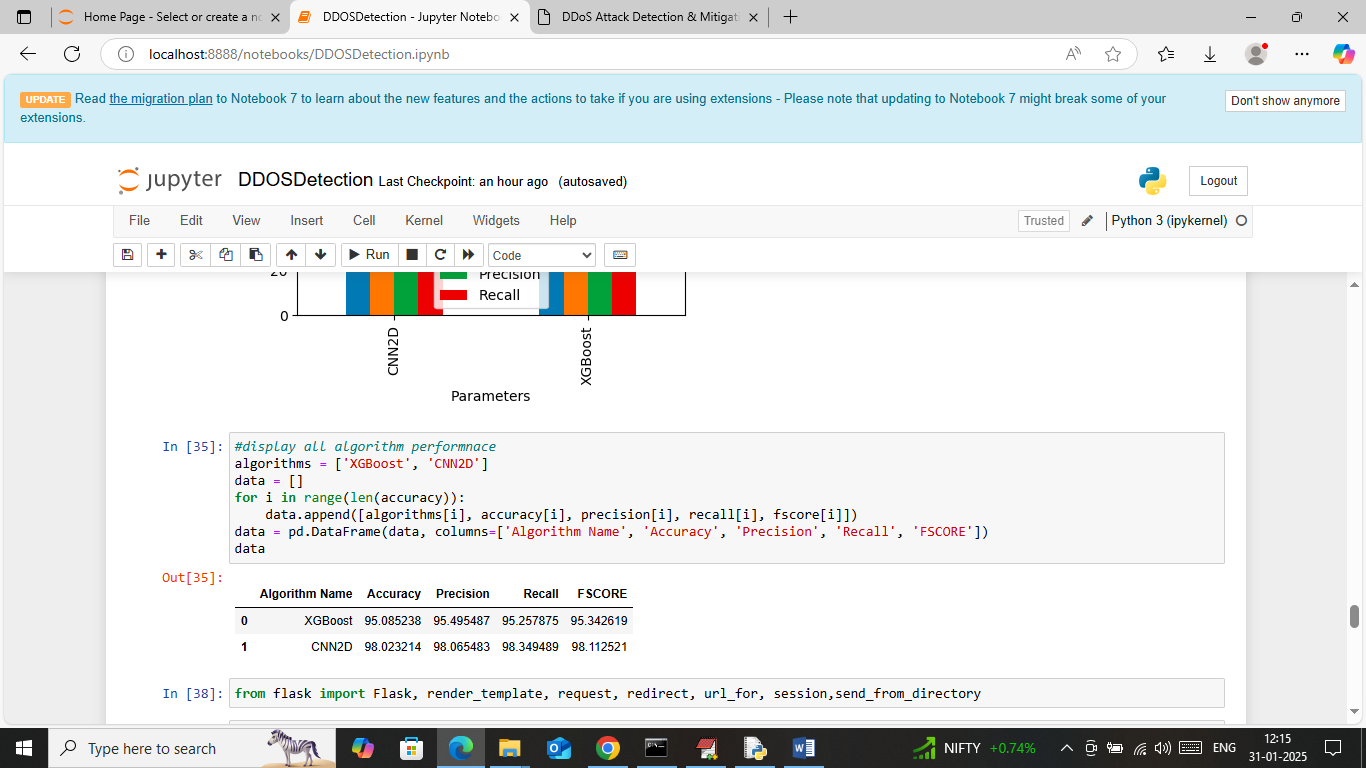
In above screen defining and training CNN2D algorithm and after executing this model will get below output



In above screen CNN got 98% accuracy and can see other metrics also

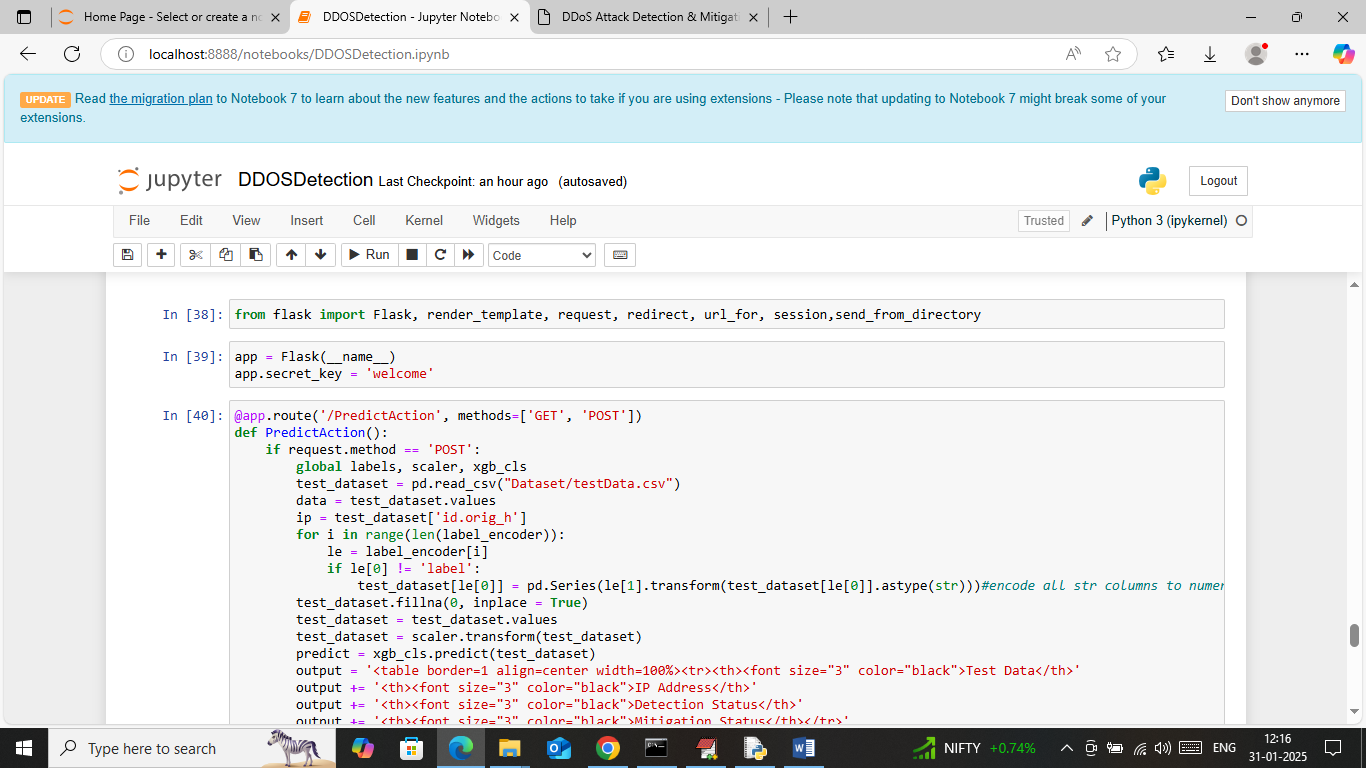


In above screen displaying comparison graph between both algorithms where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars.

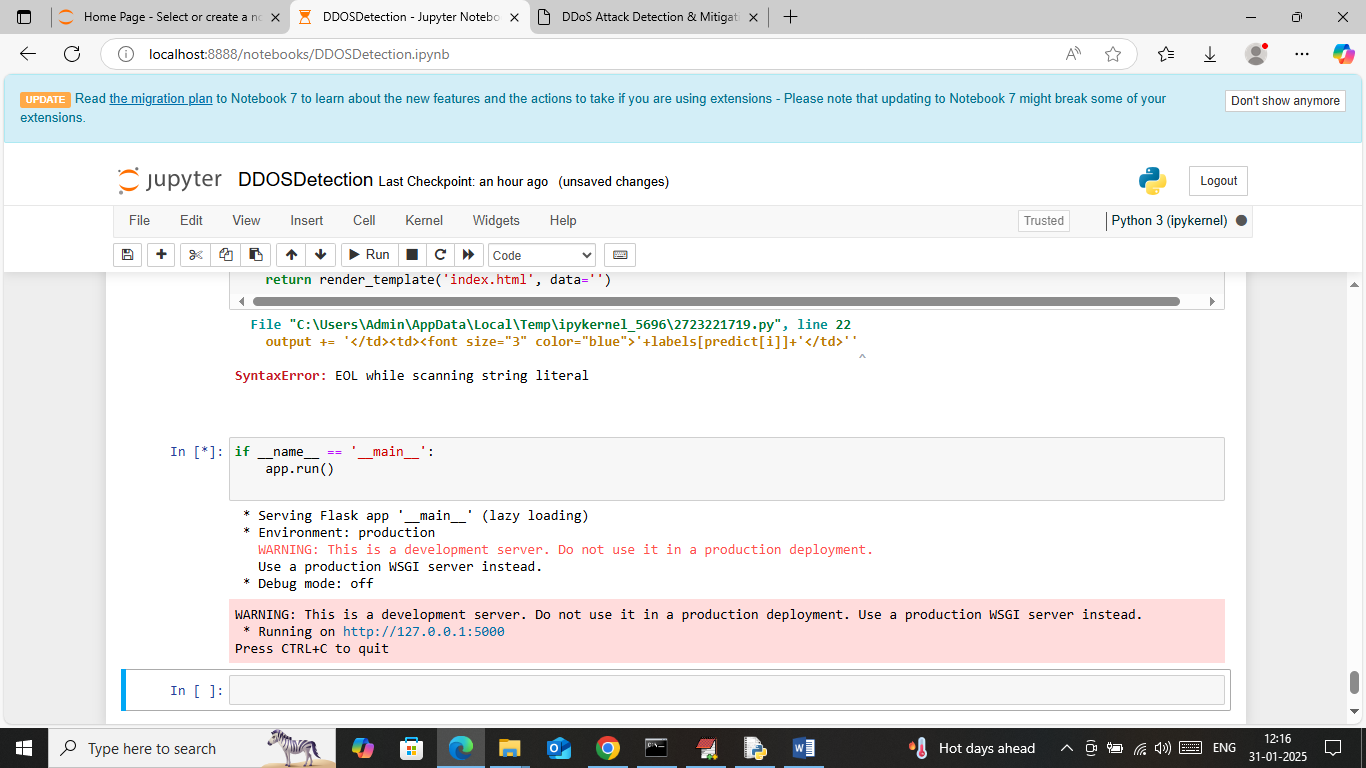


In above screen displaying both algorithm performance in tabular format. So in above screen we have done all data training and experiments execution.

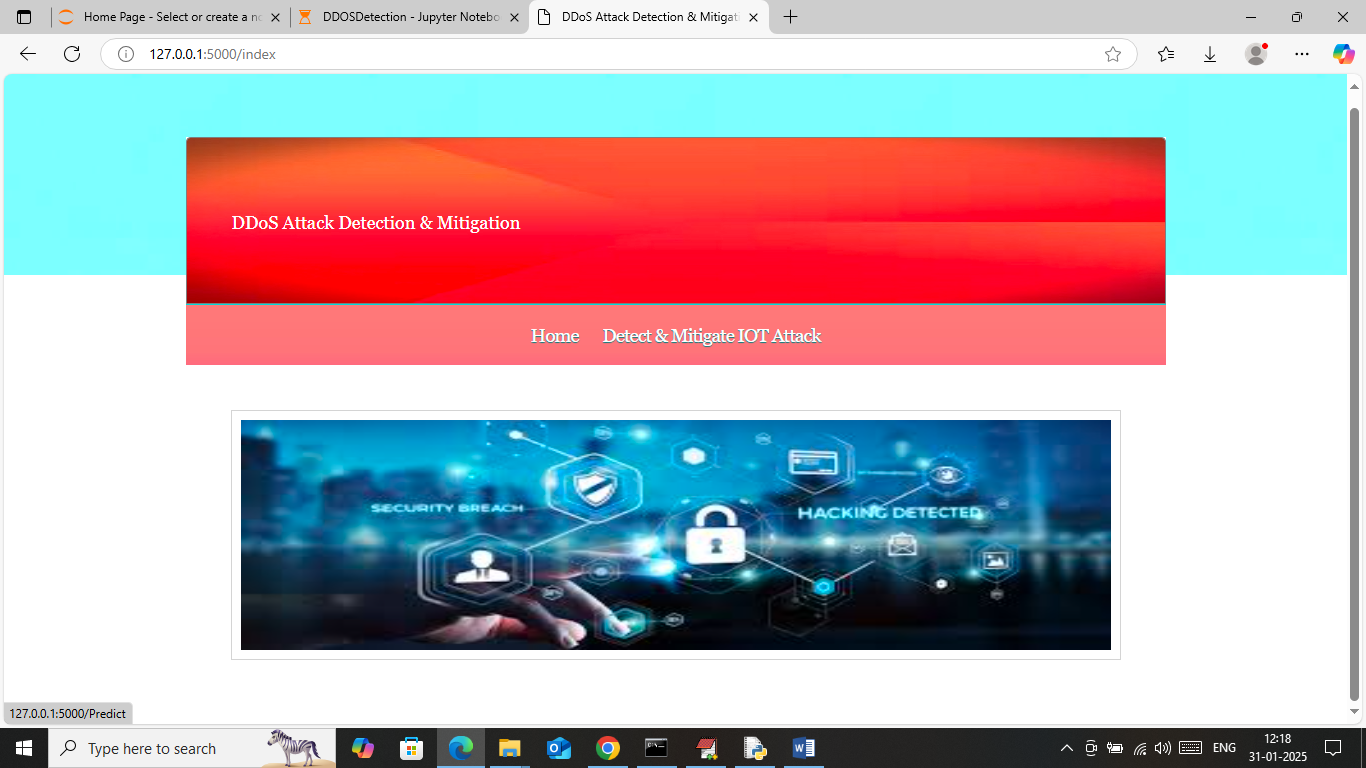
Now run all flask block code to mitigate attack from web page



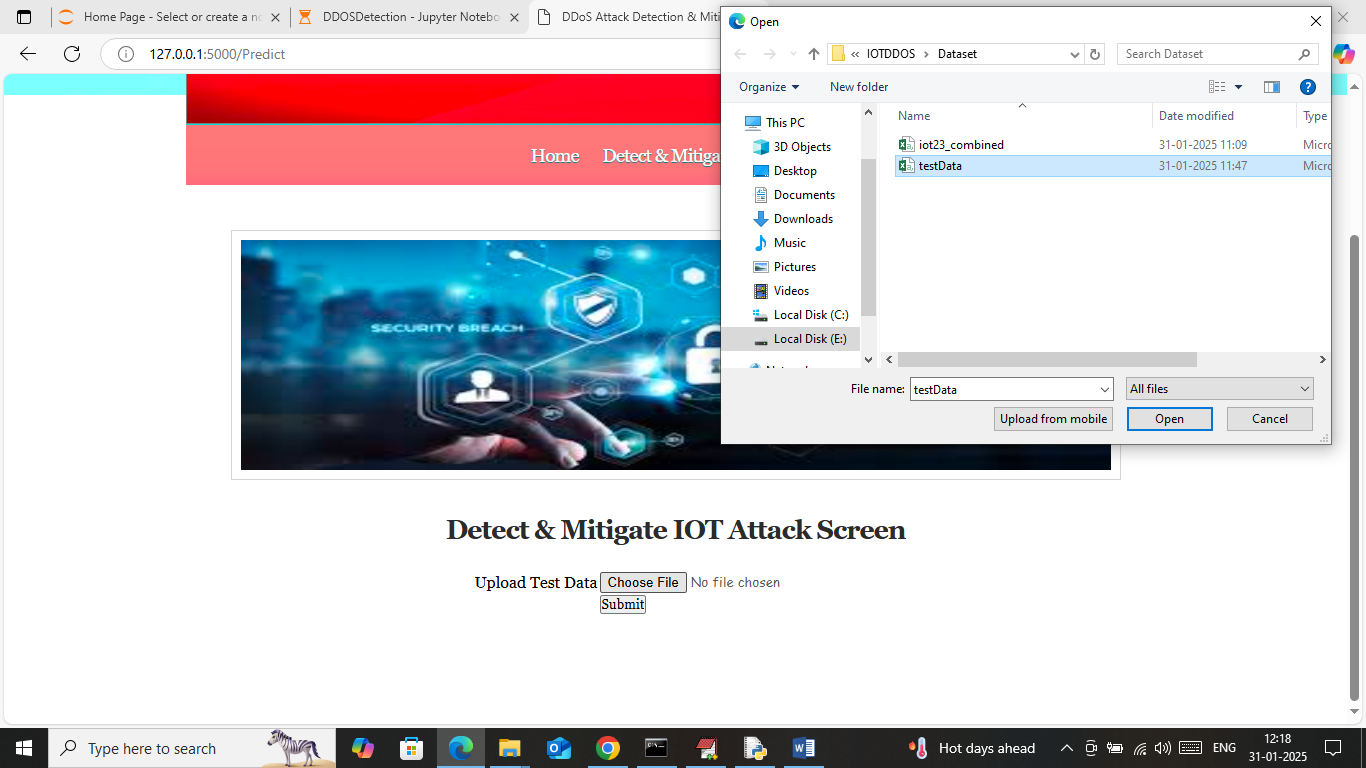
In above screen run all flask blocks to start flask server and then will get below page



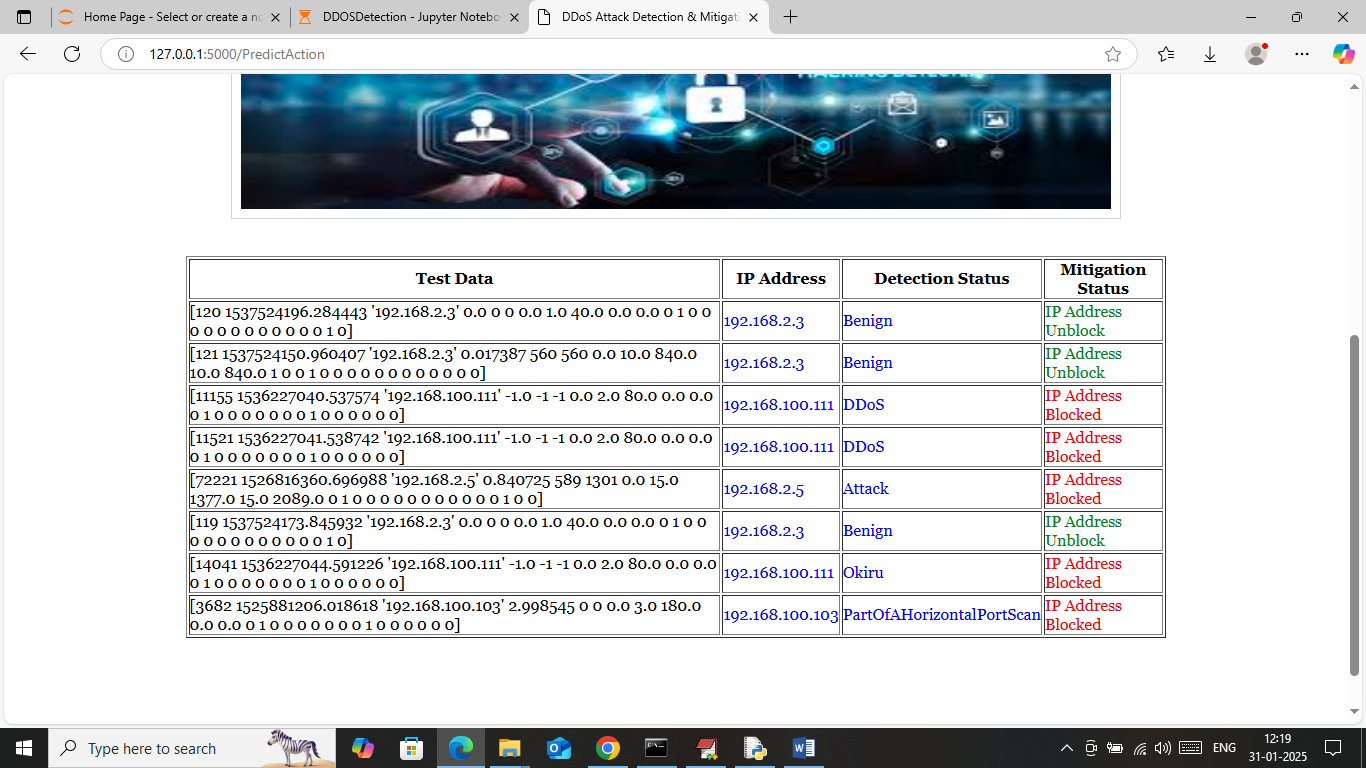
In above screen flask server started and now open browser and enter URL as <http://127.0.0.1:5000/index> and then press enter key to get below page



In above screen click on ‘Detect & Mitigate IOT Attack’ link to get below page



In above screen select and upload test data file which contains IOT network request data and then press button to get below page



In above screen in first column can see network request data and in second column can see SOURCE IP address which is sending request to IOT. In 3rd column can see predicted output as ‘Benign (normal) or attack’ and based on detected attack that IP will get blocked.

So by using above ML and DL algorithm you can detect and mitigate all possible IOT attacks.

**Conclusion:**

In conclusion, the increasing prevalence of Distributed Denial of Service (DDoS) attacks poses a significant threat to the reliability and security of IoT environments. Traditional detection systems are often inadequate due to their limitations in recognizing new and evolving attack patterns, high false positive rates, and their inability to operate efficiently within resource-constrained devices. To address these challenges, the proposed hybrid machine learning-based system offers a robust and adaptive solution by combining the strengths of both supervised and unsupervised learning techniques.

The system not only enhances detection accuracy but also reduces false alarms and ensures rapid, real-time response through edge computing integration. By incorporating intelligent traffic analysis, feature optimization, and automated mitigation strategies, the solution effectively minimizes the impact of DDoS attacks while maintaining the performance and availability of IoT services. Additionally, the adaptive learning mechanism ensures the system remains effective against new attack vectors, providing long-term reliability.

Overall, the hybrid approach represents a scalable, efficient, and intelligent method for securing modern IoT ecosystems against the growing threat of DDoS attacks. This solution lays a solid foundation for future enhancements, including integration with advanced threat intelligence, blockchain-based validation, or federated learning models for distributed security.

**Future Work:**

While the proposed hybrid machine learning model for DDoS attack detection in IoT environments shows promising results, several avenues remain open for future improvement and expansion. One potential direction is the incorporation of **deep learning models** such as Long Short-Term Memory (LSTM) networks or Convolutional Neural Networks (CNNs). These models can capture temporal and spatial patterns in network traffic more effectively, which may further enhance the accuracy and robustness of attack detection, especially for sophisticated and stealthy DDoS variants.

Another significant area for enhancement is the **integration of federated learning**. In federated learning, models are trained across multiple decentralized devices or servers without sharing raw data, which improves privacy and scalability. This approach would be particularly valuable in IoT ecosystems where data privacy and security are crucial. Implementing federated learning can help ensure that individual IoT nodes contribute to model training without exposing sensitive data, making the system more decentralized and secure.

Furthermore, **context-aware detection systems** could be developed to adapt dynamically based on the behavior and criticality of specific IoT devices. For example, an anomaly detected on a medical sensor may be treated with higher urgency compared to a smart light bulb. This kind of contextual intelligence can improve the prioritization and efficiency of mitigation efforts, ensuring that critical infrastructure remains protected even during widespread attacks.

Additionally, future work may explore **blockchain integration** for secure and tamper-proof logging of detected threats and mitigation actions. Blockchain can provide decentralized trust management and ensure the integrity of data exchanged between IoT nodes, making the system more transparent and resilient to manipulation by malicious actors.

**References**

1. **Doshi, R., Apthorpe, N., & Feamster, N.** (2018). Machine Learning DDoS Detection for Consumer Internet of Things Devices. In Proceedings of the IEEE Security and Privacy Workshops (SPW), pp. 29–35.  
   https://doi.org/10.1109/SPW.2018.00013
2. **Meidan, Y., Bohadana, M., Mathov, Y., et al.** (2018). N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Computing, 17(3), 12–22.  
   https://doi.org/10.1109/MPRV.2018.03367731
3. **Nguyen, T. T., & Armitage, G.** (2008). A Survey of Techniques for Internet Traffic Classification Using Machine Learning. IEEE Communications Surveys & Tutorials, 10(4), 56–76.  
   https://doi.org/10.1109/SURV.2008.080406
4. **Sivanathan, A., Sherratt, D., Gharakheili, H. H., et al.** (2018). Low-Cost Flow-Based IoT Device Fingerprinting and Attack Detection. In Proceedings of the 14th International Conference on Network and Service Management (CNSM).
5. **Tian, R., Jian, M., & Yang, Y.** (2020). A Survey on DDoS Attack Detection and Defense in SDN. Security and Communication Networks, Hindawi.  
   https://doi.org/10.1155/2020/5034793
6. **Kumar, P., Tripathi, R., & Raw, R. S.** (2021). A Hybrid Machine Learning Approach for Botnet Detection in IoT Networks. International Journal of Information Security and Privacy (IJISP), 15(1), 1–14.  
   https://doi.org/10.4018/IJISP.2021010101
7. **Stojmenovic, I., & Wen, S.** (2014). The Fog Computing Paradigm: Scenarios and Security Issues. 2014 Federated Conference on Computer Science and Information Systems, 1–8.
8. **Yin, C., Zhu, Y., Fei, J., & He, X.** (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access, 5, 21954–21961.  
   https://doi.org/10.1109/ACCESS.2017.2762418