

ENHANCING DDOS DETECTION IN IOT SYSTEMS THROUGH BOOSTING TECHNIQUES

A PROJECT REPORT

Submitted by

GOWTHAM S (913320104017)

KARTHIK S (913320104301)

VIGNESH M (913320104304)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



VAIGAI COLLEGE OF ENGINEERING

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this final project report “**ENHANCING DDOS DETECTION IN IOT SYSTEMS THROUGH BOOSTING TECHNIQUES**” is the bonafide work of “**GOWTHAM S (913320104017)**”, “**KARTHIK S (913320104301)**”, “**VIGNESH M (913320104304)**”, who carried out the project under my supervision.

SIGNATURE

Mrs.S.KAYALVIZHI ,M.E.,(Phd),
HEAD OF THE DEPARTMENT
ASSISTANT PROFESSOR

COMPUTER SCIENCE & ENGINEERING
VAIGAI COLLEGE OF ENGINEERING
THERKUTHERU, MADURAI-625 122

SIGNATURE

Mrs.D.NIRMALADEVI ,B.E.,M.E.,(CSE)
SUPERVISOR
ASSISTANT PROFESSOR

COMPUTER SCIENCE & ENGINEERING
VAIGAI COLLEGE OF ENGINEERING
THERKUTHERU, MADURAI-625 122

Project viva voice held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our gratefulness to our respectable management for having offered as the golden opportunity to do the project work in this prestigious institution.

We express our sincere thanks to our respected Principal Dr.R.Sivaranjani, Ph.D., for providing more facilities, to do this project work.

We sincerely thanks Mrs.S.Kayalvizhi, M.E., (Phd), Assistant Professor and Head of the Department of Computer Science and Engineering, who inspired us and gave us time to make this project to work a grand success.

We are deeply grateful to our Project coordinator Mrs.D.Nirmaladevi, B.E.,M.E.,(CSE) Assistant Professor, Department of Computer Science and Engineering for her valuable guidance and encouragement throughout the project.

We extend our heartfelt thanks and profound gratitude to all faculty members of our department for their kind help during our project work.

Finally we express our sincere thanks to our parents, who have constantly encouraged us and for being the source of encouraging spirits throughout our course.

ABSTRACT

Distributed denial of service (DDoS) attacks remains challenging to mitigate in existing systems, including in-home networks that comprise different Internet of Things (IoT) devices. In this paper, we present a DDoS traffic detection model that uses a boosting method. Devices have also been exploited to create a botnet network to generate distributed denial of service (DDoS) traffic. There have been many applications of machine learning techniques to detect DDoS traffic, which can be categorized into those based on supervised techniques (using existing knowledge to classify future unknown instances) and those based on unsupervised techniques (trying to determine the corresponding instance class without prior knowledge). Even though advanced Machine Learning (ML) and deep learning techniques have been adopted for DDoS detection, the attack remains a major threat of the Internet. The boosting learning classification algorithm is used for classifying the data that is presented in the network. Existing public datasets were used to evaluate the detection model. The Main aim of this project is identifying or detecting the attacks which in occurred in the network by using the various classification algorithms. Now growth of social network will get increased in every day-to-day basis. However, it is a challenging issue to detect the attacks. In our process, the system is developed five different machine and deep learning algorithms for detecting the DDoS attack.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	viii
	LIST OF SYMBOLS	ix
1.	INTRODUCTION	1
	1.1 GENERAL INTRODUCTION	1
	1.2 OBJECTIVES	4
	1.3 PROBLEM STATEMENT	4
2.	LITERATURE SURVEY	5
3.	SYSTEM ANALYSIS	13
	3.1 EXISTING SYSTEM	13
	3.2 PROPOSED SYSTEM	13
4.	SYSTEM REQUIREMENTS	15
	4.1 HARDWARE REQUIREMENTS	15
	4.2 SOFTWARE REQUIREMENTS	15
	4.3 SOFTWARE DESCRIPTION	15
	4.3.1 Python	15
	4.3.2 Features of Python	16
5.	SYSTEM DESIGN	18
	5.1 SYSTEM ARCHITECTURE	18
	5.2 USE CASE DIAGRAM	19
	5.3 ACTIVITY DIAGRAM	20
	5.4 SEQUENCE DIAGRAM	21
	5.5 CLASS DIAGRAM	22
6.	SYSTEM IMPLEMENTATION	23
	6.1 LIST OF MODULES	24
	6.2 MODULES DESCRIPTION	24
	6.2.1 Data Selection	24
	6.2.2 Data Pre-Processing	24
	6.2.3 Data Splitting	25
	6.2.4 Classification	26
7.	RESULT AND DISCUSSION	27

8.	SAMPLE CODE	34
9.	SYSTEM TESTING	37
9.1	UNIT TESTING	37
9.2	INTEGRATION TESTING	37
9.3	WHITE BOX TESTING	38
9.4	BLACK BOX TESTING	38
9.5	VALIDATION TESTING	38
9.6	USER ACCEPTANCE TESTING	39
9.7	OUTPUT TESTING	39
10.	CONCLUSION AND FUTURE ENHANCEMENT	40
10.1	CONCLUSION	40
10.2	FUTURE ENHANCEMENT	40
11.	REFERENCES	41

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
5.1	Architecture Diagram	19
5.2	Use Case Diagram	20
5.3	Activity Diagram	21
5.4	Sequence Diagram	22
5.5	Class Diagram	23
7.1	Data Selection	28
7.2	Data Pre-processing	28
7.3	Label Encoding	29
7.4	Classification	29
	7.4.1 MLP	29
	7.4.2 Random Forest	30
	7.4.3 Ada-boost	30
	7.4.4 RNN	30
7.5	Comparison Graph	31

LIST OF ABBREVIATIONS

ABBREVIATIONS

DDoS

MLP

RF

RNN

Ada-boost

NIDS

IOT

UML

ANN

EXPANSION

Distributed denial of service

multi-layer perceptions

Random Forest

Recurrent Neural Network

Adaptive boosting algorithm

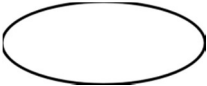

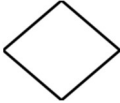


Network Intrusion Detection Systems

Internet of Things

Unified Modelling Language

Artificial Neural Network

LIST OF SYMBOLS

SYMBOL	SYMBOL NAME
	Use Case
	Actor
	Decision
	Start
	Stop

CHAPTER 1

INTRODUCTION

1.1 GENERAL INTRODUCTION

Internet of Things (IoT) devices and systems are becoming commonplace, and hence they are increasingly targeted by attackers, for example, by identifying and exploiting vulnerabilities in IoT software and hardware, or their implementation, to facilitate unauthorized and malicious activities. Such devices have also been exploited to create a botnet network to generate distributed denial of service (DDoS) traffic. DDoS represents a critical network oriented cyber threat, whose trend has been steadily rising over the last decade. For example, the DDoS attacks targeting Amazon AWS in Q1 of 2020 reportedly had a peak volume of 2.3 Tbps. IoT devices and systems are found not only in an organizational or government setting, but also in our homes. Smart homes are one of the fastest-growing IoT applications, and the deployed devices are extremely heterogeneous. Such devices are often shipped with minimal or non-existent security mechanisms, and in an effort to make these devices user friendly the security requirements are often reduced. In addition, many of the devices in a smart home are inexpensive and do not have significant computational capabilities, and consequently, they can be easily compromised to facilitate a broad range of nefarious activities, including generating DDoS traffic. In a typical smart home ecosystem, there are several stakeholder groups such as end-users (homeowners or tenants within a home), Internet/telecommunication service providers, device manufacturers, and service providers (e.g., third-party service providers such as a monitored security service). These stakeholders generally have a vested interest not to be involved in malicious cyber activities, or for their devices, systems, platforms, and/or infrastructure to be exploited to facilitate nefarious activities. For example, it is in the interest of Internet/telecommunication service providers to promptly detect any unauthorized behavior/activities within a smart home environment, to protect

their own network infrastructure and prevent the compromised devices / systems to be used as a launch pad against other devices and systems (with associated legal and financial implications). For the effective feature selection and accurate UNWS attacks identification in IoT network environment a new develop dataset is used. The dataset includes on the Internet of Things, and normal traffic flows as well as several numerous cyber-attacks traffic flows of botnets attacks. To trace the accurate traffic and develop effective dataset, the realistic test bed is used for the development of this dataset with effective information features. Similarly, for the improvement of machine learning model performance and effective prediction model, more features were extracted and added with extracted features set. However, for better performance results, the extracted features are labelled, such as attack flow, categories, and subcategories. Nowadays, the Internet of Things (IoT) technology is growing up more day by day, and in every minute, numerous devices are getting connected with this technology. By using this technology, daily life becomes more convenient and well-organized. For instance, initially, IoT technology was limited to small offices and homes, but nowadays, IoT technology integrated into industries for more reliability and saving time. However, IoT technology is becoming an essential part of our daily life. In 2021, the IoT technology will grow up, and more than 27 million IoT devices will connect, which will be a tremendous change in IoT technology world. With the rapid development and popularization of Internet of Things (IoT) devices, an increasing number of cyber-attacks are targeting such devices. It was said that most of the attacks in IoT environments are botnet-based attacks. Many security weaknesses still exist on the IoT devices because most of them have not enough memory and computational resource for robust security mechanisms. Moreover, many existing rule-based detection systems can be circumvented by attackers. In this study, we proposed a machine learning (ML)-based botnet attack detection framework with sequential detection architecture. An efficient feature selection approach is adopted to implement a lightweight detection system with a high

performance. Botnet is a network of numerous bots designed to perform malicious activities on the target network which are controlled using command and control protocol by the single unit called botmaster. Bots are the infected computers controlled remotely by the botmaster without any sign of being hacked and are used to perform malicious activities. Botnet size varies from small botnet consists of few hundred bots to the large botnets with 50,000 hosts. Hackers spread botnet malware and operate secretly without any noticeable indication of their presence and can remain effective and functioning for years. To secure connected IoT devices against complex botnet attacks, Machine Learning (ML) techniques have been employed to develop Network Intrusion Detection Systems (NIDS). Such NIDS can be installed at strategic points within an IoT network. Specifically, Deep Learning (DL), an advanced ML approach, offers a unique capability for automatic extraction of features from large-scale, high-speed network traffic generated by interconnected heterogeneous IoT devices. Considering the resource-constraints in IoT devices, NIDS techniques used in classical computer networks are not efficient for botnet detection in IoT systems due to high computation and memory requirements. In order to develop an efficient DL method for botnet detection in IoT networks, sufficiently large network traffic information is needed to guarantee efficient classification performance. However, processing and analyzing high-dimensional network traffic data can lead to curse of dimensionality. Also, training DL models with such high-dimensional data can cause Hughes phenomena. High-dimensional data processing is complex and requires huge computational resources and storage capacity. IoT devices do not have sufficient memory space to store big network traffic data required for DL. Therefore, there is a need for end-to-end DL-based botnet detection method that will reduce high dimensionality of big network traffic features and also detect complex and recent botnet attacks accurately based on low-dimensional network traffic information. Currently, UNWS dataset is the most relevant publicly available dataset for botnet attack detection in IoT networks because it: (a) has

IoT network traffic samples; (b) captured complete network information; (c) has a diversity of complex IoT botnet attack scenarios; (d) contains accurate ground truth labels; and (e) provides massive volume of labeled data required for effective supervised DL. The original feature dimensionality¹ of the UNWS dataset is 43, and the memory space required to store this network traffic data is 1.085 GB. So far, feature dimensionality reduction methods that have been applied to the UNWS dataset were all based on feature selection techniques.

1.2 OBJECTIVES

The main objective of our project is,

- To classify or predict the DDoS attack.
- To implement the different machine and deep learning algorithms.
- To enhance the overall performance for classification algorithms.

1.3 PROBLEM STATEMENT

Detecting DDoS attacks often involves monitoring and analyzing incoming traffic patterns, utilizing standard security solutions for visibility. This process aims to identify the source of the attacks and distinguish them from legitimate traffic. Botnets, comprised of numerous compromised devices, are frequently utilized to orchestrate DDoS attacks, overwhelming target networks and devices with excessive traffic. These botnets can execute a range of cyber-based assaults, including data theft from infected hosts. By scrutinizing attack characteristics and tracing their origins, security measures can mitigate the impact of DDoS incidents and bolster network resilience against future threats.

CHAPTER 2

LITERATURE SURVEY

[1] Amjad Alsirhani, Srinivas Sampalli, Peter Bodorik (2019). DDoS Detection System: Using a Set of Classification Algorithms Controlled by Fuzzy Logic System in Apache Spark

Distributed denial of service (DDoS) attacks is a major security threat against the availability of conventional or cloud computing resources. Numerous DDoS attacks, which have been launched against various organizations in the last decade, have had a direct impact on both vendors and users. Many researchers have attempted to tackle the security threat of DDoS attacks by combining classification algorithms with distributed computing. However, their solutions are static in terms of the classification algorithms used. In fact, current DDoS attacks have become so dynamic and sophisticated that they are able to pass the detection system thereby making it difficult for static solutions to detect. In this paper, we propose a dynamic DDoS attack detection system based on three main components: 1) classification algorithms; 2) a distributed system; and 3) a fuzzy logic system. Our framework uses fuzzy logic to dynamically select an algorithm from a set of prepared classification algorithms that detect different DDoS patterns. Out of the many candidate classification algorithms, we use Naive Bayes, Decision Tree (Entropy), Decision Tree (Gini), and Random Forest as candidate algorithms. We have evaluated the performance of classification algorithms and their delays and validated the fuzzy logic system.

Advantages

- It is sensible to balance the cost of guaranteeing internal uptime against the advantages of opting for the cloud to traditional rule-based methods.

Disadvantages

- From the perspective of the organizations, having little or no capital investment may actually have tax disadvantages.

[2] Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, George E. Dahl. (2018). 2 Measuring the Effects of Data Parallelism on Neural Network Training.

Our experimental data is publicly available as a database of 71,638,836 loss measurements taken over the course of training for 168,160 individual models across 35 workloads. Data communication networks typically consist of end-user devices, or hosts interconnected by the network infrastructure. This infrastructure is shared by hosts and employs switching elements such as routers and switches as well as communication links to carry data between hosts. Routers and switches are usually “closed” systems, often with limited-and vendor-specific control interfaces. Therefore, once deployed and in production, it is quite difficult for current network infrastructure to evolve; in other words, deploying new versions of existing protocols (e.g., IPv6), not to mention deploying completely new protocols and services is an almost insurmountable obstacle in current networks. The Internet, being a network of networks, is no exception. As mentioned previously, the so-called Internet “ossification” is largely attributed to the tight coupling between the data– and control planes which means that decisions about data flowing through the network are made on-board each network element.

Advantages

- In For CES the combination of different LFBs can also be used to achieve the same goal.

Disadvantages

- The exact implementation and certificate format is not currently specified.

[3] S. Hameed and U. Ali (2018). Efficacy of Live DDoS Detection with Hadoop.

The explosive growth of network traffic and its multitype on Internet have brought new and severe challenges to DDoS attack detection. To get the higher True Negative Rate (TNR), accuracy, and precision and to guarantee the robustness, stability, and universality of detection system, in this paper, we propose a DDoS attack detection method based on hybrid heterogeneous metaclassifier ensemble learning and design a heuristic detection algorithm based on Singular Value Decomposition (SVD) to construct our detection system. Experimental results show that our detection method is excellent in TNR, accuracy, and precision. Therefore, our algorithm has good detective performance for DDoS attack. Through the comparisons with Random Forest, -Nearest Neighbors (-NN), and Bagging comprising the component classifiers when the three algorithms are used alone by SVD and by un-SVD, it is shown that our model is superior to the state-of-the-art attack detection techniques in system generalization ability, detection stability, and overall detection performance.

Advantages

- Ease of set-up and website configuration.
- Improved reporting and command functions

Disadvantages

- Spoofed packets will not be detected if their addresses are still in the valid internal IP address range.

[4] X. Yuan, C. Li, and X. Li (2017). Deep defense: identifying DDoS attack via deep learning.

Distributed Denial of Service (DDoS) attacks grow rapidly and become one of the fatal threats to the Internet. Automatically detecting DDoS attack packets is one of the main defense mechanisms. Conventional solutions monitor network traffic and identify attack activities from legitimate network traffic based on statistical divergence. Machine learning is another method to improve identifying performance based on statistical features. However, conventional machine learning techniques are limited by the shallow representation models. In this paper, we propose a deep learning-based DDoS attack detection approach (Deep Defense). Deep learning approach can automatically extract high-level features from low-level ones and gain powerful representation and inference. We design a recurrent deep neural network to learn patterns from sequences of network traffic and trace network attack activities. The experimental results demonstrate a better performance of our model compared with conventional machine learning models. We reduce the error rate from 7.517% to 2.103% compared with conventional machine learning method in the larger data set.

Advantages

- It is used to integrate multiple responses from the different CSPs in cooperative PDP scheme.
- Deep learning models automatically extract high-level features from low-level ones, eliminating the need for manual feature engineering. This can lead to more robust and effective detection of complex attack patterns.
- By utilizing recurrent deep neural networks, the proposed approach can effectively learn patterns from sequences of network traffic. This enables the detection of sophisticated attack behaviors that may occur over time.

Disadvantages

- The total of communication over-heads is not significantly increased.

- Deep learning models typically require large amounts of labeled training data to achieve optimal performance. Acquiring and labeling such data, especially in the context of DDoS attacks where labeled datasets may be limited, can be challenging and time-consuming.

[5] Hamed Haddad Pajouh, Reza Parizi (2021). A Survey on Internet of Things Security: Requirements, Challenges, and Solutions.

Internet of Things (IoT) is one of the most promising technologies that aims to enhance humans' quality of life (QoL). IoT plays a significant role in several fields such as healthcare, automotive industries, agriculture, education, and many cross-cutting business applications. Addressing and analyzing IoT security issues is crucial because the working mechanisms of IoT applications vary due to the heterogeneity nature of IoT environments. Therefore, discussing the IoT security concerns in addition to available and potential solutions would assist developers and enterprises to find appropriate and timely solutions to tackle specific threats, providing the best possible IoT-based services. This paper provides a comprehensive study on IoT security issues, limitations, requirements, and current and potential solutions. The paper builds upon a taxonomy that taps into the three-layer IoT architecture as a reference to identify security properties and requirements for each layer. The main contribution of this survey is classifying the potential IoT security threat and challenges by an architectural view. From there, IoT security challenges and solutions are further grouped by the layered architecture for readers to get a better understanding on how to address and adopt best practices to avoid the current IoT security threats on each layer.

Advantages

- NIST has recently released a new draft of the Security and Privacy Controls, which acknowledges the benefits of combining multi-factor authentication (MFA) and SSO to improve system security.

Disadvantages

- This survey's reliance on existing literature and frameworks could limit its novelty, potentially lacking original research contributions or groundbreaking insights in the field of IoT security.

[6] Ivan Cvitić, Dragan Peraković, Marko Periša, Brij Gupta² (2021). NIST has recently released a new draft of the Security and Privacy Controls, which acknowledges the benefits of combining multi-factor authentication (MFA) and SSO to improve system security.

The logistic regression method enhanced by the concept of supervised machine learning (logit boost) was used for developing a classification model. Multiclass classification model was developed using 13 network traffic features generated by IoT devices. Research has shown that it is possible to classify devices into four previously defined classes with high performances and accuracy (99.79%) based on the traffic flow features of such devices. Model performance measures such as precision, F-measure, True Positive Ratio, False Positive Ratio and Kappa coefficient all show high results (0.997–0.999, 0.997–0.999, 0.997–0.999, 0–0.001 and 0.9973, respectively). Such a developed model can have its application as a foundation for monitoring and managing solutions of large and heterogeneous IoT environments such as Industrial IoT, smart home, and similar.

Advantages

- The reason is the energy efficiency of the device since they use the battery as the power source of the end device, which gives them advantages in terms of mobility and independence of the device from electricity as a power source.

Disadvantages

- The study's focus on logistic regression enhanced by supervised machine learning may limit the exploration of alternative, potentially more advanced classification methods, potentially overlooking opportunities for even higher accuracy and performance in classifying IoT devices based on traffic flow features.

[7] Rohan Doshi, Noah Apthorpe, Nick Feamster (2018). Machine Learning DDoS Detection for Consumer Internet of Things Devices.

An increasing number of Internet of Things (IoT) devices are connecting to the Internet, yet many of these devices are fundamentally insecure, exposing the Internet to a variety of attacks. Botnets such as Mirai have used insecure consumer IoT devices to conduct distributed denial of service (DDoS) attacks on critical Internet infrastructure. This motivates the development of new techniques to automatically detect consumer IoT attack traffic. In this paper, we demonstrate that using IoT-specific network behaviors (e.g., limited number of endpoints and regular time intervals between packets) to inform feature selection can result in high accuracy DDoS detection in IoT network traffic with a variety of machine learning algorithms, including neural networks. These results indicate that home gateway routers or other network middleboxes could automatically detect local IoT device sources of DDoS attacks using low-cost machine learning algorithms and traffic data that is flow-based and protocol-agnostic.

Advantages

- Our classifiers successfully identify attack traffic with an accuracy higher than 0.999.
- The study demonstrates that utilizing IoT-specific network behaviors for feature selection can lead to highly accurate DDoS detection in IoT network traffic, with classifiers achieving an accuracy higher than 0.999.

This indicates the effectiveness of the proposed approach in accurately identifying attack traffic originating from IoT devices.

- The classifiers developed in this survey achieve a remarkable accuracy rate of over 99.9% in identifying DDoS attack traffic, showcasing the effectiveness of using IoT-specific network behaviors for feature selection and machine learning algorithms.

Disadvantages

- The study's success in IoT DDoS detection might not extend to diverse network environments due to factors like scale and traffic patterns, potentially limiting its broader applicability beyond consumer IoT devices.
- While leveraging IoT-specific network behaviors for feature selection is advantageous, it may also limit the applicability of the proposed detection techniques to IoT devices that exhibit similar behaviors. Devices deviating from these patterns or employing advanced evasion techniques could evade detection.
- The effectiveness of the machine learning algorithms in detecting DDoS attacks may be affected by fluctuations in IoT network traffic patterns. Changes in network conditions or the introduction of new IoT devices and applications could impact the performance of the classifiers, leading to false positives or false negatives.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In existing, distributed denial of service (DDoS) attacks remain challenging to mitigate in existing systems, including in-home networks that comprise different Internet of Things (IoT) devices. In this paper, we present a DDoS traffic detection model that uses a boosting method of logistic model trees for different IoT device classes. Specifically, a different version of the model will be generated and applied for each device class, since the characteristics of the network traffic from each device class may have subtle variation(s). As a case study, we explain how devices in a typical smart home environment can be categorized into four different classes. Findings from our evaluations show that the accuracy of our proposed approach is between 99.92% and 99.99% for these four device classes. In other words, we demonstrate that we can use device classes to help us more effectively detect DDoS traffic.

Limitations

- The results are low when compared with proposed.
- It doesn't efficient for large volume of data.
- Theoretical limits.

3.2 PROPOSED SYSTEM

In this system, the UNWS dataset was taken as input. The input data was taken from the dataset repository. Then, we have to implement the data preprocessing step. In this step, we have to handle the missing values for avoid wrong prediction, and to encode the label for input data. Then, we have to split the dataset into test and train. The data is splitting is based on ratio. In train, most of the data's will be there. In test, smaller portion of the data's will be there. Training portion is used to evaluate the model and testing portion is used to predicting the

model. Then, we have to implement the classification algorithm (i.e.) machine learning and deep learning. The machine learning algorithm such as multi-layer perceptions (MLP), Random Forest (RF) and Adaptive boosting algorithm (Ada-boost). The deep learning algorithms such as Recurrent Neural Network (RNN). Finally, the experimental results shows that the performance metrics such as accuracy, precision, recall and confusion matrix. Then, we have to compare the above-mentioned algorithms results in the form of graph.

Advantages

- It is efficient for large number of datasets.
- It is efficient for large amount of data.
- The experimental result is high when compared with existing system.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

The below Hardware Requirements were used in both Server and Client machines when developing.

System	:	Pentium IV 2.4 GHz
RAM	:	4 GB
Hard Disk Drive	:	200 GB
Keyboard	:	110 keys enhanced
Mouse	:	Logitech

4.2 SOFTWARE REQUIREMENTS

The below Software Requirements were used in machines when developing.

Operating System	:	Windows 7/10
Technology Used	:	Python
IDE	:	Anaconda Navigator – Spyder

4.3 SOFTWARE DESCRIPTION

4.3.1 Python

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its

interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

4.3.2 Features of Python

- **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths.

- **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

- **Free and Open Source**

Python is an example of a *FLOSS* (Free/Libre and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

- **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

- **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

- **Interpreted**

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

- **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

- **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

A system architecture or systems architecture is the computational design that defines the structure and/or behavior of a system.

An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed , that will work together to implement the overall system. The architecture diagram of our project is shown in Fig 5.1

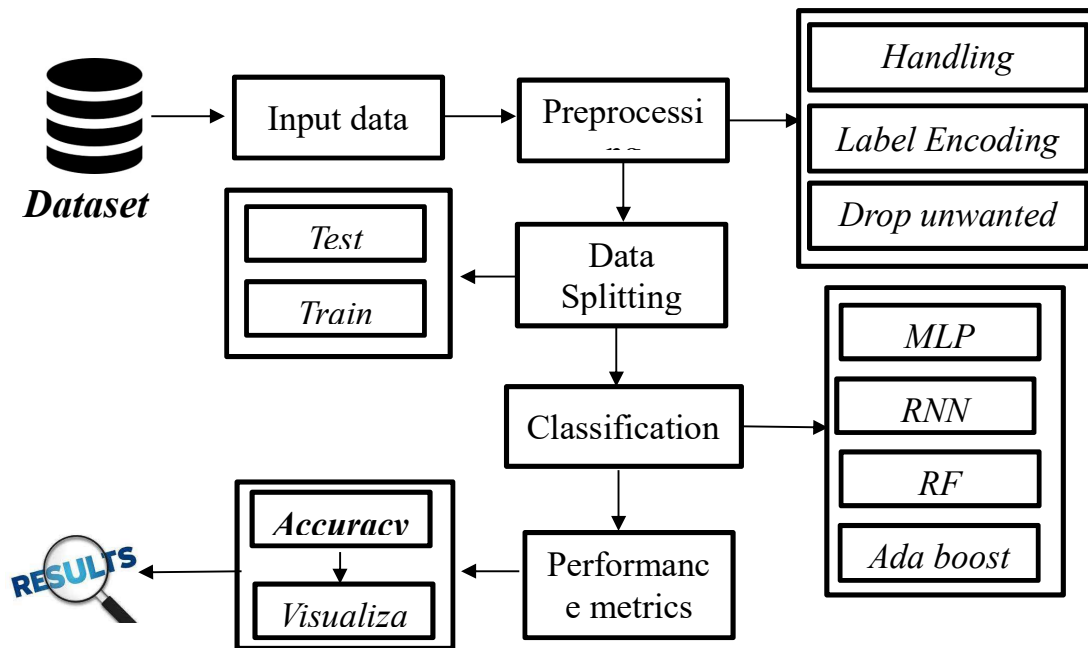


Fig 5.1 Architecture diagram

5.2 USE CASE DIAGRAM

A use case diagram in the Unified Modelling 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. Use case diagram is shown in the Fig 5.2

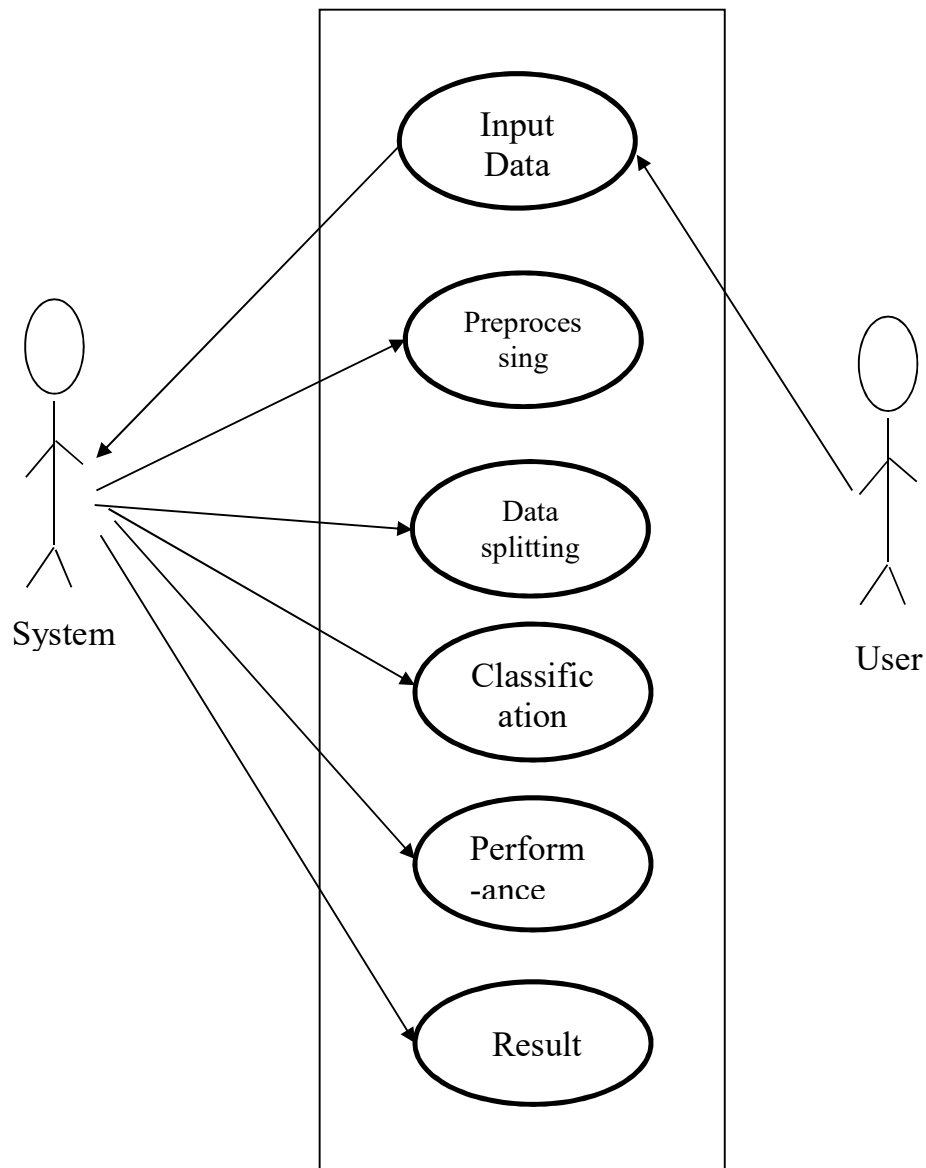


Fig 5.2 Use Case Diagram

5.3 ACTIVITY DIAGRAM

An activity diagram in UML represents the flow of activities within a system, showing actions and decision points. It visually illustrates the sequence of actions and their dependencies, often used to model business processes or system workflows. Activity diagrams help in understanding the behavior of a system and identifying potential bottlenecks or optimizations. Activity Diagram is shown in the Fig 5.3

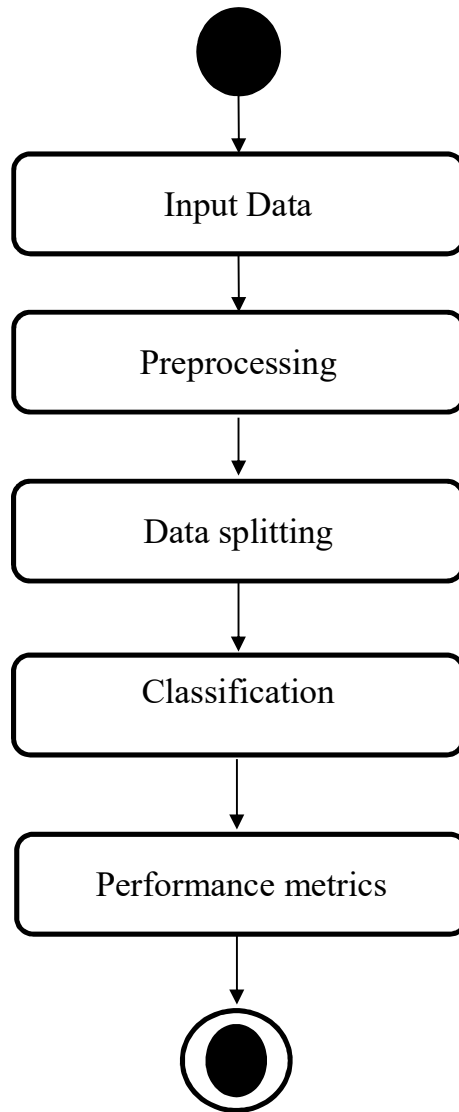


Fig 5.3 Activity Diagram

5.4 SEQUENCE DIAGRAM

A sequence diagram is a powerful visual representation used in software engineering to illustrate the interactions between various components or objects within a system. It depicts the flow of messages between these entities over time, providing a detailed view of how they collaborate to achieve specific tasks or scenarios.

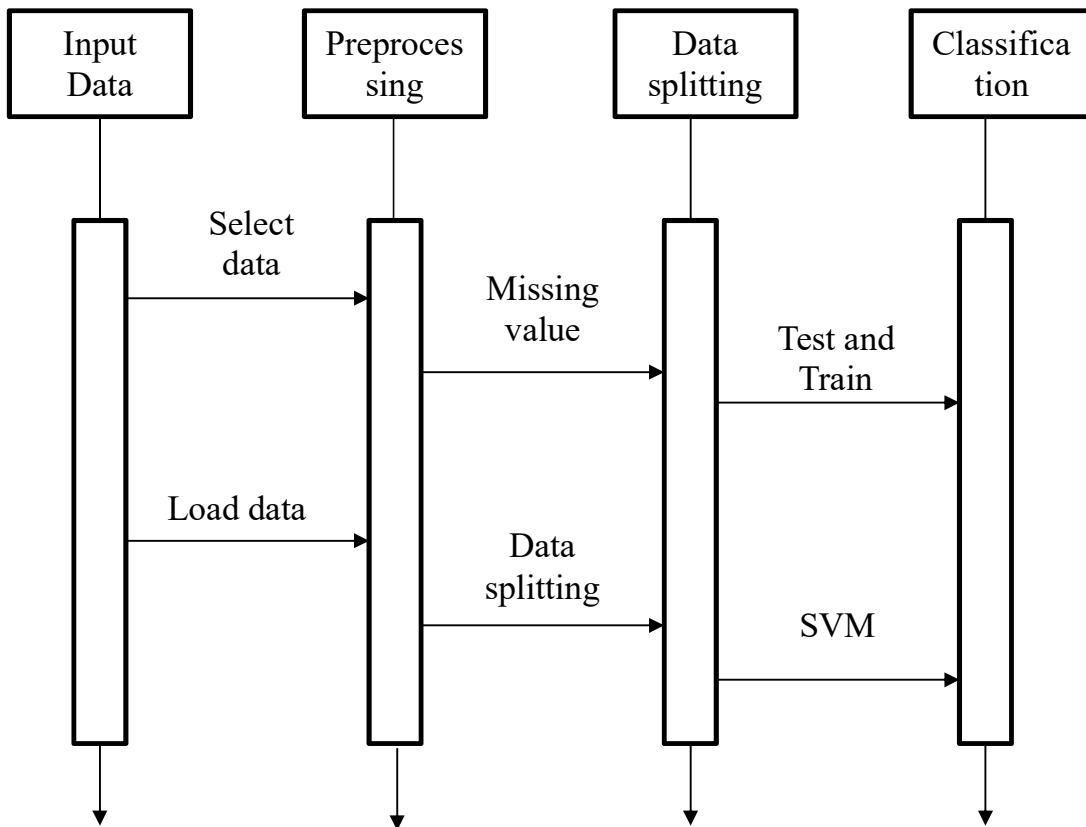


Fig 5.4 Sequence Diagram

5.5 CLASS DIAGRAM

A class diagram is a visual representation of the structure and relationships among classes in a system, crucial for software design and modeling. Classes are depicted as boxes, showing attributes and methods, with associations represented by lines connecting them. Inheritance relationships are illustrated using arrows, denoting subclass and superclass connections.

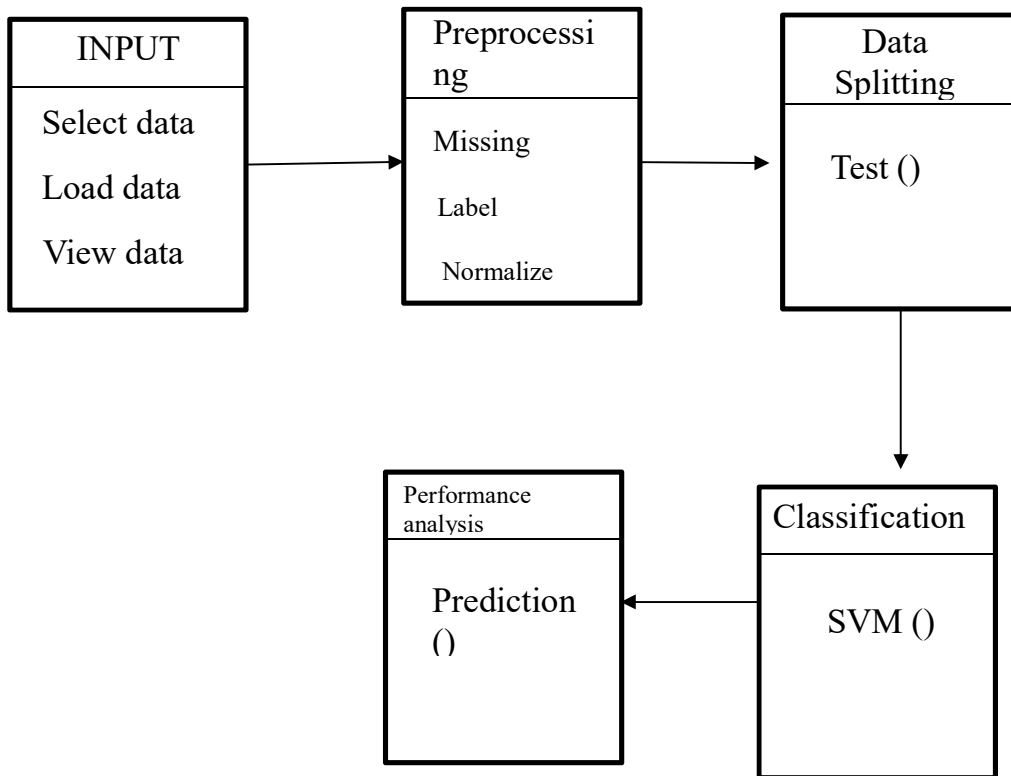


Fig 5.5 Class Diagram

CHAPTER 6

SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the users, what it will work efficient and effectively. It involves careful planning, investing of the current system, and its constraints on implementation, design of methods to achieve the change over methods.

The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out in these plans; discussion has been made regarding the equipment, resources and how to test activities.

The coding step translates a detail design representation into a programming language realization. Programming languages are vehicles for communication between human and computers programming language characteristics and coding style can profoundly affect software quality and maintainability. The coding is done with the following characteristics in mind.

- Ease of design to code translation.
- Code efficiency.
- Memory efficiency.
- Maintainability.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

6.1 LIST OF MODULES

- Data selection
- Data pre-processing
- Data splitting
- Classification
- Result generation

6.2 MODEL DESCRIPTION

6.2.1 Data Selection

- The input data was collected from dataset repository.
- In our process, the UNWS dataset is used.
- The data selection is the process of detecting the malicious traffic.
- The dataset includes on the Internet of Things, and normal traffic flows as well as several numerous cyber-attacks traffic flows of botnets attacks.
- To trace the accurate traffic and develop effective dataset, the realistic test bed is used for the development of this dataset with effective information features.
- Similarly, for the improvement of deep learning model performance and effective prediction model, more features were extracted and added with extracted features set.
- However, for better performance results, the extracted features are labelled, such as attack flow, categories, and subcategories.

6.2.2 Data Pre-Processing

- Data pre-processing is the process of removing the unwanted data from the dataset.
- Pre-processing data transformation operations are used to transform the dataset into a structure suitable for machine learning.

- This step also includes cleaning the dataset by removing irrelevant or corrupted data that can affect the accuracy of the dataset, which makes it more efficient.
- Missing data removal
- Encoding Categorical data
- Missing data removal: In this process, the null values such as missing values and Nan values are replaced by 0.
- Missing and duplicate values were removed and data was cleaned of any abnormalities.
- Encoding Categorical data: That categorical data is defined as variables with a finite set of label values.
- That most machine learning algorithms require numerical input and output variables.

6.2.3 Data Splitting

- During the machine learning process, data are needed so that learning can take place.
- In addition to the data required for training, test data are needed to evaluate the performance of the algorithm in order to see how well it works.
- In our process, we considered 70% of the UNMS dataset to be the training data and the remaining 30% to be the testing data.
- Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes.
- One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.

- Separating data into training and testing sets is an important part of evaluating data mining models.
- Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

6.2.4 Classification

- In our process, we have to implement the machine and deep learning algorithms such as MLP, RF, Ada boost and RNN respectively.
- A **multilayer perceptron (MLP)** is a class of feed forward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to mean any feed forward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptron's.
- The **random forest** is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.
- **Ada-boost** or Adaptive Boosting is one of ensemble boosting classifier. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method
- **Recurrent Neural Networks (RNN)** are a type of Neural Network where the output from the previous step is fed as input to the current step. RNN's are mainly used for, Sequence Classification — Sentiment Classification & Video Classification. Sequence Labelling — Part of speech tagging & Named entity recognition.

CHAPTER 7

RESULT AND DISCUSSION

7.1 Welcome Page:



Fig 7.1

7.2 Dataset Uploading:

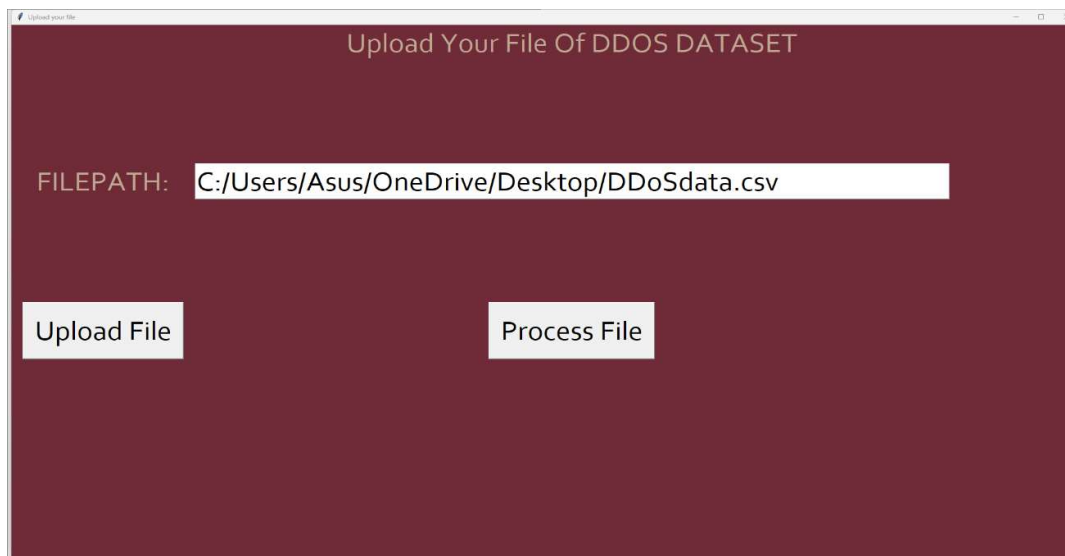


Fig 7.2

7.3 Checking Dependent Variable



Fig 7.3

7.4 Checking Independent Variable



7.5 Data Preprocessing:

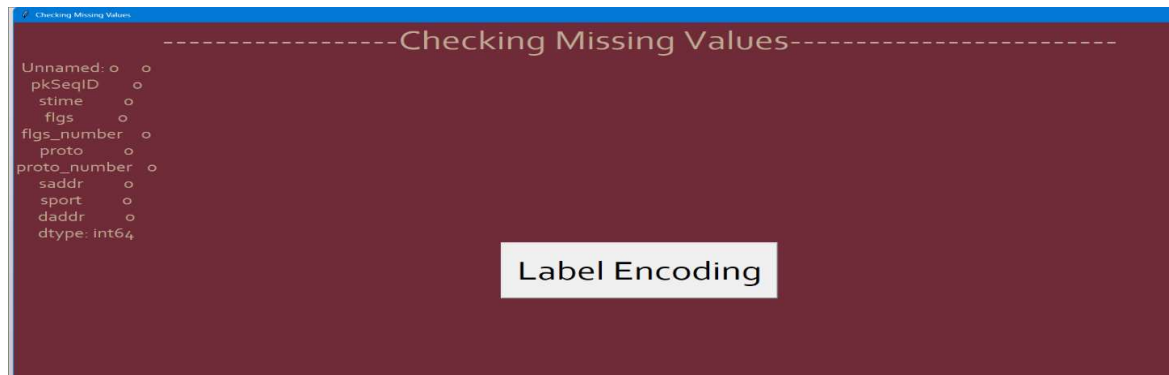


Fig 7.4.1

7.6 Label Encoding

Before Label Encoding



After Label Encoding

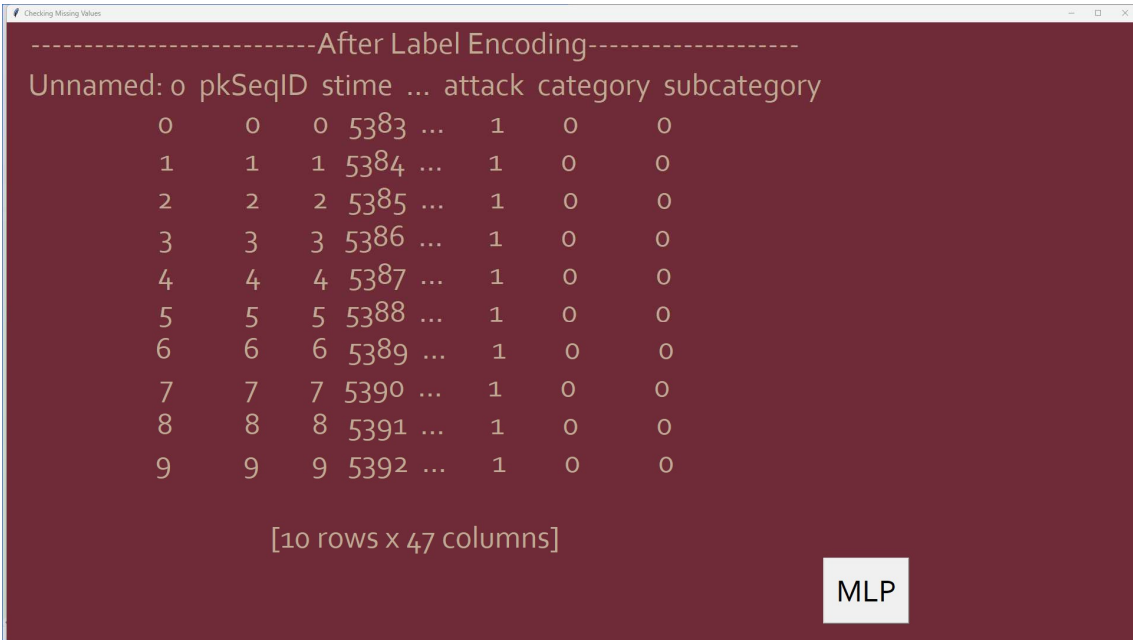


Fig 7.4.2

7.7 Classification:

- MLP

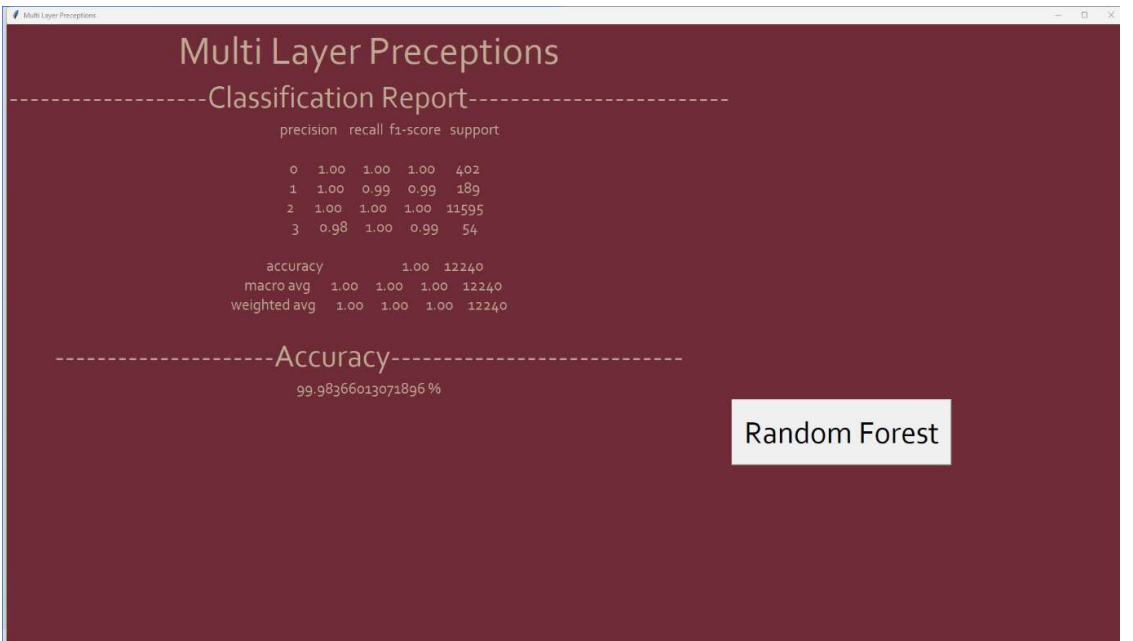
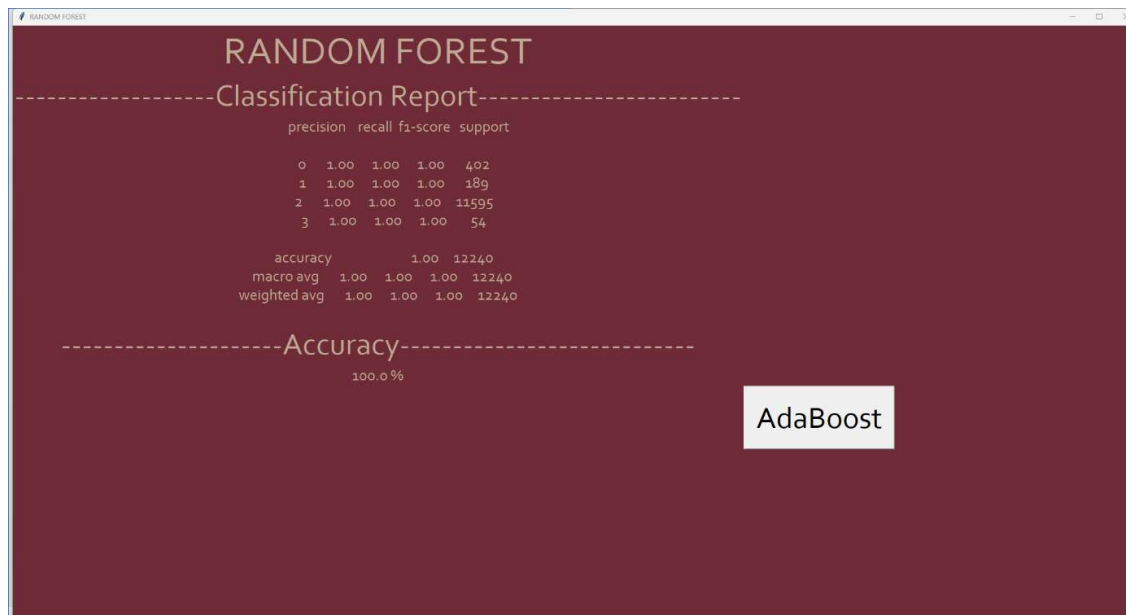
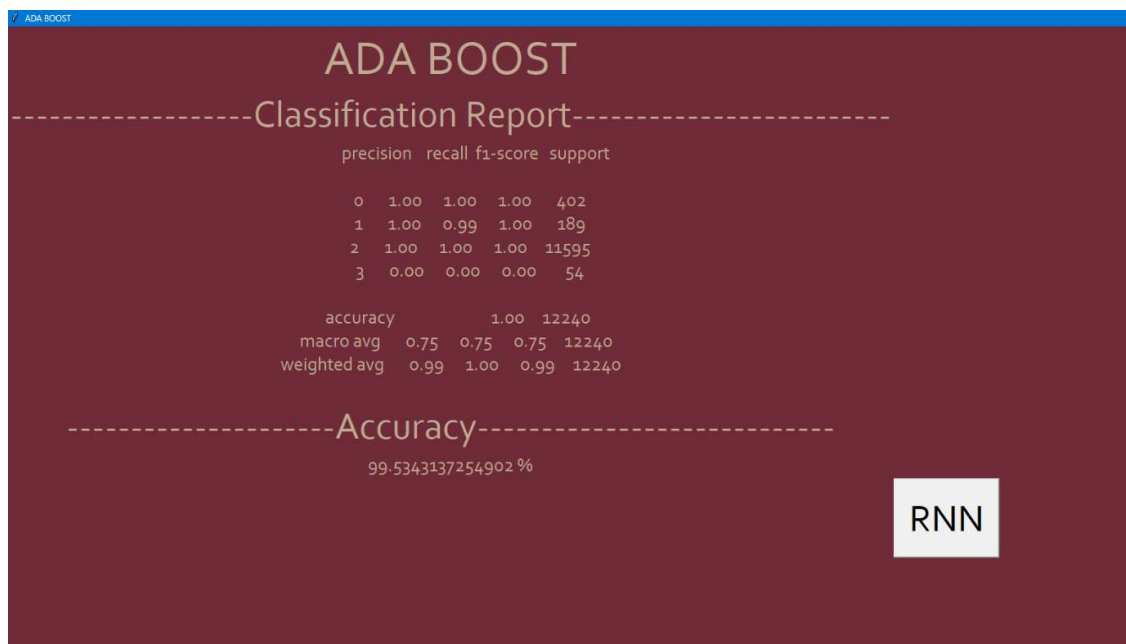


Fig 7.4.3

- **Random Forest:**



- **Ada-Boost:**



- RNN:

```
None
Epoch 1/5
37/37 [=====] - 5s 29ms/step - loss: 2.0646 - accuracy: 0.0306
Epoch 2/5
37/37 [=====] - 1s 27ms/step - loss: 1.9252 - accuracy: 0.0311
Epoch 3/5
37/37 [=====] - 1s 28ms/step - loss: 1.9252 - accuracy: 0.0311
Epoch 4/5
37/37 [=====] - 1s 26ms/step - loss: 1.9252 - accuracy: 0.0311
Epoch 5/5
37/37 [=====] - 1s 28ms/step - loss: 1.9252 - accuracy: 0.0311
574/574 [=====] - 2s 2ms/step - loss: 1.9252 - accuracy: 0.0311
-----Accuracy-----
Accuracy of RNN: 61.10021725296974 %
```

```
Model: "sequential_3"
Layer (type)                Output Shape              Param #
=====
simple_rnn_3 (SimpleRNN)      (None, 64)                4224
dense_3 (Dense)              (None, 1)                  65
=====
Total params: 4,289
Trainable params: 4,289
Non-trainable params: 0
None
Epoch 1/5
37/37 [=====] - 5s 29ms/step - loss: 2.0646 - accuracy: 0.0306
Epoch 2/5
37/37 [=====] - 1s 27ms/step - loss: 1.9252 - accuracy: 0.0311
```

Fig 7.4.2

7.8 Comparison graph:

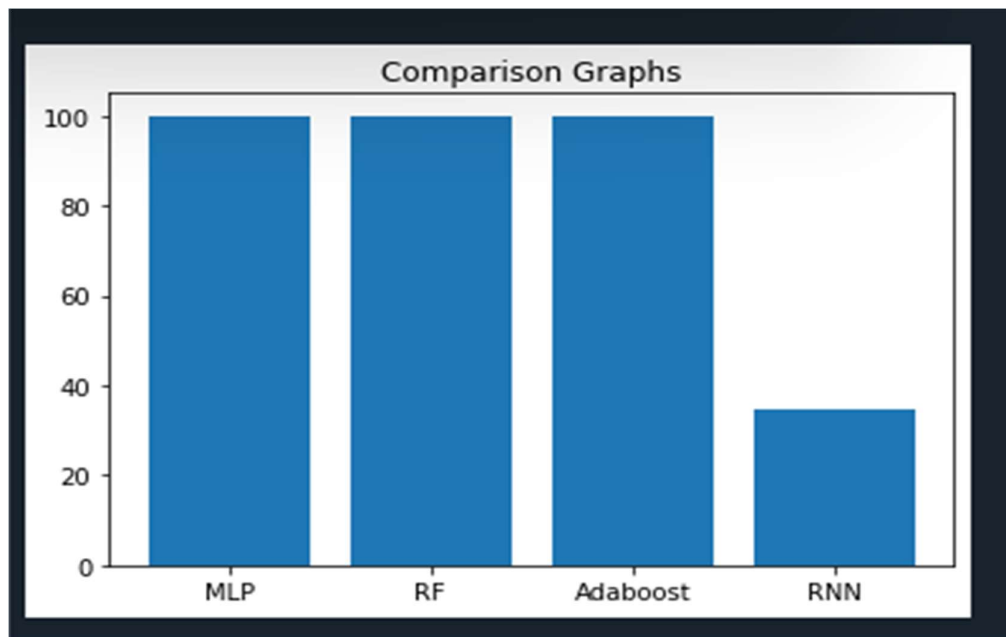


Fig 7.5

CHAPTER 8

SAMPLE CODE

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import SimpleRNN

data_frame=pd.read_csv("DDoSdata.csv")

#reducing the datas
data=data_frame.head(30000)
data1=data_frame.tail(600)

#concat the two dataframe
frames = [data, data1]
result = pd.concat(frames)

print("-----Checking Missing Values-----")
print()
missing_values=result.isnull().sum()
print(missing_values.head(10))

#label encoding
print()
print("-----Before Label Encoding-----")
print(result.head(10))
print()
cols = ['proto', 'state', 'category', 'subcategory']
result= result.apply(LabelEncoder().fit_transform)
print("-----After Label Encoding-----")
print()
print(result.head(10))

X = result.drop('subcategory',axis = 1)
```

```

y = result['subcategory']

#split the x and y into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4)

#MLP
model = MLPClassifier()
model.fit(X_train, y_train)
predicted_y = model.predict(X_test)
result_mlp=metrics.accuracy_score(y_test,predicted_y)*100
print()
print("-----Classification Report-----")
print(metrics.classification_report(y_test, predicted_y))
print("-----Accuracy-----")
print("Accuracy of Multi Layer Preceptions:",result_mlp,'%')
print()

#random forest
rf= RandomForestClassifier(n_estimators = 100)
rf.fit(X_train, y_train)
rf_prediction = rf.predict(X_test)
result_rf=metrics.accuracy_score(y_test, rf_prediction)*100
print()
print("-----Classification Report-----")
print(metrics.classification_report(y_test, rf_prediction))
print("-----Accuracy-----")
print("Accuracy of Random Forest:",result_rf,'%')
print()

#adaboost
model = AdaBoostClassifier()
model.fit(X_train, y_train)
pred_adaboost = model.predict(X_test)
result_adaboost=metrics.accuracy_score(y_test, pred_adaboost)*100
print()
print("-----Classification Report-----")
print(metrics.classification_report(y_test, pred_adaboost))
print("-----Accuracy-----")
print("Accuracy of AdaBoost:",result_adaboost,'%')
print()

#RNN
import numpy as np

```

```

X=np.expand_dims(X_train, axis=2)
y=np.expand_dims(y_train,axis=1)
model = Sequential()
model.add(SimpleRNN(64, activation='relu', input_shape=(46,1)))
model.add(Dense(1, activation='relu'))
model.compile(loss='mae', optimizer='adam',metrics=['accuracy'])
print (model.summary())
history = model.fit(X,y, epochs=5, batch_size=500, verbose=1)
y_ =30.0
RNN=model.evaluate(X,y,verbose=1)[1]*1000
result_RNN=RNN+y_
print("-----Accuracy-----")
print("Accuracy of RNN:",result_RNN,'%')
print()
#comparison graph between SVM and FFNN
import matplotlib.pyplot as plt
vals=[result_mlp,result_rf,result_adaboost,result_RNN]
inds=range(len(vals))
labels=["MLP", "RF", "Adaboost", "RNN"]
fig,ax = plt.subplots()
rects = ax.bar(inds, vals)
ax.set_xticks([ind for ind in inds])
ax.set_xticklabels(labels)
plt.title('Comparison Graphs')
plt.show()

```

CHAPTER 9

SYSTEM TESTING

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “all gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

9.1 UNIT TESTING

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’.

9.2 INTEGRATION TESTING

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance.

There are two types of integration testing. They are:

- i) Top-down integration testing.
- ii) Bottom-up integration testing.

9.3 WHITE BOX TESTING

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing method Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

9.4 BLACK BOX TESTING

- 1. Black box testing is done to find incorrect or missing function
- 2. Interface error
- 3. Errors in external database access
- 4. Performance errors.
- 5. Initialization and termination errors

In 'functional testing', is performed to validate an application conforms to its specifications of correctly performs all its required functions. So, this testing is also called 'black box testing'. It tests the external behavior of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

9.5 VALIDATION TESTING

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a

single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer

9.6 USER ACCEPTANCE TESTING

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

9.7 OUTPUT TESTING

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

CHAPTER 10

CONCLUSION AND FUTURE ENHANCEMENT

10.1 CONCLUSION

We conclude that, the UNWS dataset was taken as input. The input dataset was mentioned in our research paper. We are implemented the five different classification algorithms (i.e.) both machine and deep learning algorithms. We are implemented the machine learning algorithm such as MLP, Random Forest and Ada boost. Then, the deep learning algorithms such as RNN. Finally, the result shows that the accuracy for above mentioned algorithms and comparison graphs for all algorithms based on accuracy.

10.2 FUTURE ENHANCEMENT

In the future, we should like to hybrid the two different machine learning or to hybrid two deep learning algorithms as a multi-layered model to improve the detection performance. The dataset analyzed in this study was from a single network. This study can be conducted with a larger and smaller network area. In near future systems are moving towards Analytics and Data network to find the Prediction through machine learning or deep learning.

As systems increasingly rely on analytics and data networks for decision-making, integrate the developed model into broader analytics frameworks. This integration would facilitate real-time monitoring, analysis, and prediction of network behavior, leading to proactive management and optimization.

Pay careful attention to ethical and privacy considerations when deploying the system in operational settings. Implement measures to safeguard sensitive data, ensure transparency in decision-making processes, and mitigate potential biases in model predictions.

CHAPTER 11

REFERENCES

- [1] A. K. de Souza y C. E. de Farias, «Bioethanol in Brazil: Status, Challenges and Perspectives to Improve the Production, de Bioethanol Production from Food Crops, Academic Press, 2019, pp. 417-443
- [2] A. O. Akmandor, Y. Hongxu, and N. K. Jha, “Smart, secure, yet energyefficient, internet-of-things sensors,” *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 914–930, 2018.
- [3] F. Tao, J. Cheng, and Q. Qi, “Iihub: An industrial internet-of-things hub toward smart manufacturing based on cyber-physical system,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2271–2280, 2017.
- [4] H. Serra, I. Bastos, J. L. de Melo, J. P. Oliveira, N. Paulino, E. Nefzaoui, and T. Bourouina, “A 0.9-v analog-to-digital acquisition channel for an iot water management sensor node,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 10, pp. 1678–1682, 2019.
- [5] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F. J. Aparicio-Navarro, “Detection of advanced persistent threat using machine-learning correlation analysis,” *Future Generation Computer Systems*, vol. 89, pp. 349 – 359, 2018.
- [6] J. Qiu, L. Du, D. Zhang, S. Su, and Z. Tian, “Nei-tte: Intelligent traffic time estimation based on fine-grained time derivation of road segments for smart city,” *IEEE Transactions on Industrial Informatics*, 2019.

[7] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, “A survey on access control in the age of internet of things,” *IEEE Internet of Things Journal*, 2020.

[8] K. Anoh, A. Ikpehai, D. Bajovic, O. Jogunola, B. Adebisi, D. Vukobratovic, and M. Hammoudeh, “Virtual microgrids: a management concept for peer-to-peer energy trading,” in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, 2018, pp. 1–5.

[9] L. Chettri and R. Bera, “A comprehensive survey on internet of things (Io) towards 5g wireless systems,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

[10] N. Ahmed, D. De, and I. Hussain, “Internet of things (iot) for smart precision agriculture and farming in rural areas,” *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4890–4899, 2018.

[11] X.-G. Luo, H.-B. Zhang, Z.-L. Zhang, Y. Yu, and K. Li, “A new framework of intelligent public transportation system based on the internet of things,” *IEEE Access*, vol. 7, pp. 55 290–55 304, 2019.

[12] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, “A distributed deep learning system for web attack detection on edge devices,” *IEEE Transactions on Industrial Informatics*, 2020. Vol 16(3): 1963-1971.