



**A RELIABLE & ROBUST DDOS ATTACK DETECTION IN IOT USING NEURAL NETWORK**

##### A PROJECT REPORT

###### ***Submitted by***

##### VARSHA.S [REGISTER NO: 211417104292]

**VALARMATHI.R [REGISTER NO: 211417104289]**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

IN

# COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

##### APRIL 2020

**BONAFIDE CERTIFICATE**

Certified that this project report **“A RELIABLE & ROBUST DDOS ATTACK DETECTION IN IOT USING NEURAL NETWORK”** is the bonafide work of “**VARSHA.S(211417104292), VALARMATHI.R(211417104289).”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.S.MURUGAVALLI,M.E.,Ph.D., Mrs. R. DEVI**

**HEAD OF THE DEPARTMENT ASSISTANT PROFESSOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE

NASARATHPETTAI, NASARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI,M.A.,Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHIKUMAR, M.E.,Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my project guide **Mrs. R. DEVI** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

**VARSHA.S**

**VALARMATHI.R**

**ABSTRACT**

Internet-of-things (IOT) plays a prominent role in the digital revolution . The rapid development of IoT leads to various emerging cybersecurity threats. This is because IoT devices are often limited in making them particularly vulnerable to adversaries Therefore, detecting and preventing attacks in IoT networks have to be noticed by the people in the industry. There are many attacks takes place out of them distributed denial-of-service(DDoS) attack is most challenging.

A distributed denial-of-service(DDoS) attack is a malicious attempt to disrupt normal flow of targeted server, by overwhelming the target or its surrounding with a flood of Internet traffic. Denial of service is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled. This could be sending a web server so many requests to serve a page that it crashes under the demand, or it could be a database being hit with a high volume of queries. The result is available internet bandwidth, CPU and RAM capacity becomes overwhelmed. This paper presents Distributed denial-of-service attack detection using Neural network. The main contributions of this project are Data Analysis, Dataset Preprocessing, Training the Model, Testing of Dataset. This method will produce better results compared to other techniques.

**TABLE OF CONTENTS**

| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
|  | **ABSTRACT** | iv |
|  | **LIST OF TABLES** | vii |
|  | **LIST OF FIGURES** | vii |
|  | **LIST OF SYMBOLS, ABBREVIATIONS** | viii |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Overview | 1 |
|  | 1.2 Problem Definition | 1 |
| **2.** | **LITERATURE SURVEY** | 2 |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 3.1 Existing System | 10 |
|  | 3.2 Proposed system | 10 |
|  | 3.3 Requirement Analysis and Specification | 12 |
|  | 3.4 Hardware Environment | 19 |
|  | 3.5 Software Environment | 20 |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1. ER diagram | 20 |
|  | 4.2 Data dictionary | 21 |
|  | 4.3 Data flow Diagram | 21 |
|  | 4.4 UML Diagrams | 21 |
| **5.** | **SYSTEM ARCHITECTURE** |  |
|  | 5.1 Architecture Overview | 24 |
|  | 5.2 Module Design Specification | 24 |
|  | 5.3 Program Design Language | 27 |
| **6.** | **SYSTEM IMPLEMENTATION** |  |
|  | 6.1 coding | 31 |
| **7.** | **SYSTEM TESTING** |  |
|  | 7.1 Unit Testing | 40 |
|  | 7.2 Integration Testing | 40 |
|  | 7.3 Performance Analysis | 41 |
| **8.** | **CONCLUSION** |  |
|  | 8.1 Conclusion and Future Enhancements | 42 |
|  | **APPENDICES** |  |
|  | A.1 Sample Screens | 42 |
|  | A.2 Publications | 46 |
|  | **REFERENCES** | 53 |

|  |  |
| --- | --- |
|  |  |

**LIST OF TABLES**

|  |  |
| --- | --- |
| 2.1 | Literature Survey on SDN-Assisted Slow HTTP DDoS Attack Defense Method…………………………………………………………………………. |
| 2.2 | Literature Survey on Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges……………………………….. |
| 2.3 | Literature Survey on Botnet in DDoS Attacks: Trends and Challenges………. |
| 2.4 | Literature Survey on DDoS Tools: Classification, Analysis and Comparison... |
| 4.1 | Data dictionary………………………………………………………………… |

**LIST OF FIGURES**

* + 1. Python diagram ……………………………………………………….............
    2. Package diagram ……………………………………………………………...
  1. ER diagram …………………………………………………………………..
  2. Data flow diagram ……………………………………………………………
     1. Use case diagram ……………………………………………………………..

4.3.2 Activity diagram ……………………………………………………………...

5.1 Architecture diagram …………………………………………………............

* + 1. Procedural diagram …………………………………………………………...
    2. Neural network diagram ……………………………………………………...
    3. Performance analysis diagram ………………………………………………..
  1. Anaconda Prompt ……………………………………………………………..
  2. Value per class ………………………………………………………………...
  3. Pie chart for normal & attack …………………………………………………
  4. IP address with diagram ………………………………………………………
  5. TCP acknowledgement ………………………………………………………..
  6. BRNN Model accuracy graph………………………………………….............
  7. BRNN Model loss graph ……………………………………………………..

9.8 Output …………………………………………………………………………

**LIST OF ABBREVIATIONS**

1. DDOS - Distributed Denial Of Service
2. HTTP - Hyper Text Transfer Protocol
3. DNN - Deep Neural Network
4. MDS - Misbehavior Detection System
5. PDM - Period based Defence Mechanism
6. QOS - Quality Of Service
7. IDE - Integrated Development Environment
8. EDA - Exploratory Data Analysis
9. ANN - Artificial Neural Network
10. RNN - Recurrent Neural Network

**1.INTRODUCTION**

**1.1 OVERVIEW**

A new attack, the Ad Hoc Flooding Attack, results in denial of service when used against all on-demand ad hoc networks routing protocols. In this attack, the attacker either broadcasts a lot of Route Request packets for node ID who is not in networks, or sends a lot of DATA packets to consume the bandwidth so as to congest in links.

**1.2 PROBLEM DEFINITION**

Application Layer Distributed Denial of Service (DDoS) attacks are very challenging to detect and mitigate. The various possible application layer attacks are HTTP flooding, XML attack, DNS attacks, etc. The most common and renowned application layer attack is HTTP flooding. The HTTP flooding detection and mitigation is an interesting research topic in computer networks.

There are various research solutions proposed by validating against HTTP flooding; using tools such as Golden Eye, LOIC, proprietary tools, etc. HTTP flooding attacks generated using any existing tools may not exhibit similar characteristics of the real time HTTP flooding attack.

Various methods were used to defend these attacks based on distributed schemes with certain difficulties to count the packets or duplicates sent by a node. This is due to lack of communication infrastructure. Two limits are used to mitigate packet flood and replica flood attacks, respectively. Violation of both the limits can be easily noticed by claim-carry-and- check. The inconsistency check against full claims is trivial. This is designed to work in a distributed system. Moreover, it allows tolerating a little amount of attackers for collision.

**2. LITERATURE SURVEY**

|  |  |
| --- | --- |
| Title | SDN-Assisted Slow HTTP DDoS Attack Defense Method |
| Authors | Kiwon Hong, Younjun Kim, Hyungoo Choi, and Jinwoo Park |
| Published Year | 2017 |
| Efficiency | * Defeat application-level DDoS attacks * Use cross-layer traffic analysis * Bound to a variety of transport protocols |
| Drawbacks | * Various DDoS has not been well addressed in this paper * Computational cost in training the model is high * Very calculation intensive while training the model |
| Description | A Slow HTTP Distributed Denial of Service (DDoS) attack causes a web server to be unavailable, but it is difficult to detect in a network because its traffic patterns are similar to those of legitimate clients. In this paper, we propose a network-based Slow HTTP DDoS attack defense method which is assisted by a Software-Defined Network (SDN) that can detect and mitigate Slow HTTP DDoS attacks in the network. Simulation results show that the proposed Slow HTTP DDoS attack defense method successfully protects web servers against Slow HTTP DDoS attacks.  The operation of the proposed SDN-assisted Slow DDoS attack defense method is as follows. When a web server receives an incomplete HTTP request in a situation where the number of open connections on the web server exceeds the predetermined threshold number of concurrent connections being processed, it is determined to be under attack, and it requests the SHDA to perform a DDoS attack check that includes the transmission of the received incomplete HTTP request. The SHDA then receives HTTP continuations of the received incomplete HTTP request on behalf of the web server through the flow rule update executed by the SDN controller, and on this basis it manages suspicious HTTP requests. Since, unlike HTTP requests from slow clients, incomplete HTTP requests from attackers will not be completed for a long period of time, it is possible for the SHDA to perform timeout-based DDoS attack detection on them and to isolate or identify requests from the suspected attackers among thE incomplete HTTP requests transmitted by the web server.  When an incomplete HTTP request is received from Client i, which is connected to the web server via a normal TCP 3-way handshake, the web server first sends a TCP ACK in response and compares the number of currently open HTTP connections Copen with the threshold value Cth . If Copen≥Cth, the web server suspects that it is currently under attack and initiates a Slow HTTP DDoS attack check request, in the following way. The web server first sends the incomplete HTTP request received from Client i to the SHDA. The SHDA initiates a timer Tc to perform timeout-based DDoS attack detection, which identifies an incomplete HTTP request that is not completed before Tc expires as a DDoS attack. Then, the SHDA requests the SDN controller to send a Flow-Mod message to Scw and Swc in order to establish the flow rule as shown in Table I, and transmits an Update Complete message indicating completion of the flow rule update to the web server. |

Table 2.1 Literature Survey on SDN-Assisted Slow HTTP DDoS Attack Defense Method

|  |  |
| --- | --- |
| Title | Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges |
| Authors | Qiao Yan, F. Richard Yu Qingxiang Gong, and Jianqiang Li |
| Published Year | 2015 |
| Efficiency | * It is cost effective, by allowing reuse of information extracted during detection * It makes no compromise of QoS * Reduces the consumption of hardware resources |
| Drawbacks | * Several important issues and research challenges which are open and yet to be addressed * It is not appropriate when the dataset available is large * Lead to the generation of bad candidate proposals |
| Description | In this paper, we discuss the new trends and characteristics of DDoS attacks in cloud computing, and provide a comprehensive survey of defense mechanisms against DDoS attacks using SDN. In addition, we review the studies about launching DDoS attacks on SDN, as well as the methods against DDoS attacks in SDN. To the best of our knowledge, the contradictory relationship between SDN and DDoS attacks has not been well addressed in previous works. This work can help to understand how to make full use of SDN’s advantages to defeat DDoS attacks in cloud computing environments and how to prevent SDN itself from becoming a victim of DDoS attacks, which are important for the smooth evolution of SDN-based cloud without the distraction of DDoS attacks.  **Network/transport-level DDoS flooding attacks:** These attacks have been mostly launched using TCP, UDP, ICMP and DNS protocol packets and focus on disrupting legitimate user’s connectivity by exhausting victim network’s bandwidth . Application-level DDoS  **Flooding attacks:** These attacks focus on disrupting legitimate users’ services by exhausting the server resources (e.g., Sockets, CPU, memory, disk/database bandwidth, and I/O bandwidth).It remains fairly complex infecting a sufficient number of machines in a short time frame in traditional networks. But on-demand self- service capabilities of cloud that let legitimate businesses quickly add or subtract computing power could be used to instantly create a powerful botnet [40]. With cloud computing development, malware-as-a-service operations have started to take off since 2006. Malware-as-aservice is used for spamming and launching denial-of-service attacks. Because of competition among suppliers, prices of malware-as-a-service have been falling rapidly. Today, one can buy a 10, 000-computer botnet for $1, 000. Broad Network Access and Rapid Elasticity Leading to More Immense, Flexible and Sophisticated .  **DDoS Attacks:** With cloud computing’s capabilities of broad network access and rapid elasticity, attackers can not only launch immense DDoS attacks, but also produce more flexible and more sophisticated DDoS attacks using heterogeneous thin or thick client platforms. |

Table 2.2 Literature Survey on Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges

|  |  |
| --- | --- |
| Title | Botnet in DDoS Attacks: Trends and Challenges |
| Authors | Nazrul Hoque, Dhruba K Bhattacharyya and Jugal K Kalita |
| Published Year | 2015 |
| Efficiency | * Integrates multiple traceback mechanism with customization support * Effectively block Slow HTTP DDoS attacks, allowing a web server to sustain its normal operation * Supports distributed architecture |
| Drawbacks | * Does not provide a generic solution * Still elusive to find a prevention mechanism that can protect network resources from unknown attacks * Leads to a biased outcome of predictions in terms of misclassification error and accuracy rates. |
| Description | Botnets pose a major threat to network security as they are widely used for many Internet crimes such as DDoS attacks, identity theft, email spamming and click fraud. Botnet based DDoS attacks are catastrophic to the victim network as they can exhaust both network bandwidth and resources of the victim machine. This paper presents a comprehensive overview of DDoS attacks, their causes, types with a taxonomy and technical details of various attack launching tools. A detailed discussion of several botnet architectures, tools developed using botnet architectures and pros and cons analysis are also included. Furthermore, a list of important issues and research challenges is also reported in the paper.  **Degree of automation:** This class of attacks can be manual as well as semi-automatic, depending upon the degree of automation used when an attack is launched. When a manual DDoS attack is focused on the application layer, after identifying the vulnerable hosts to compromise, the attacker executes the application with command and control processes. On the other hand, in a semi-automatic DDoS attack, after the identification of the victim machines, the attacker manually specifies the attack rate, attack type and attack duration. The attack generation is still automatic using the compromised hosts. Although, an automatic application layer DDoS attack is not commonly found, generation of attack traffic is possible with the help of malicious code that executes automatically.  **Exploited vulnerability:** Some application layer DDoS attacks are launched by exploiting weaknesses of protocols such as HTTP, FTP and TELNET against a server or a host. Some common examples of this class of DDoS attacks are session flooding and request flooding. In a session flooding attack, the attacker aims to cause malfunction in the victim server by sending sessions at a higher rate than for normal users, whereas in request flooding, the attacker sends a large number of request packets to the victim machine at a rate higher than for normal users to devastate the machine.  **Attack network**: In an application layer DDoS attack, an attacker can also launch the attack by creating or hiring a network of victim machines. Such attack networks are usually of three types, viz., agent handler networks, IRC networks and P2P networks. Unlike agent handler and IRC architectures, in a P2P network, each node can communicate with any of the nodes in the network. The intention of the attacker is to trick a large number of client computers running P2P software into requesting a file from the intended target of the DDoS attack to overwhelm the target site with traffic   |

Table 2.3 Literature Survey on Botnet in DDoS Attacks: Trends and Challenges

|  |  |
| --- | --- |
| Title | DDoS Tools: Classification, Analysis and Comparison |
| Authors | Bharti Nagpal, Pratima Sharma, Naresh Chauhan, Angel Panesar |
| Published Year | 2015 |
| Efficiency | * Detecting either low-rate or high-rate DDoS attacks * can leverage network-wide knowledge of its own network to detect DDoS attacks through techniques such as traffic pattern analysis, or machine learning * Achieved competitive performance on various datasets |
| Drawbacks | * Performance is also highly influenced by multiple user parameters. * Lack of unbiased evaluation frameworks, including benchmark datasets * Rely on the observation of abnormal traffic patterns found in network |
| Description | In the last few years, it is recognised that DDoS attack tools and techniques are emerging as effective, refined, and complex to indicate the actual attackers. Due to the seriousness of the problem many detection and prevention methods have been recommended to deal with these types of attacks. This paper aims to provide a better understanding of the existing tools, methods and attack mechanism. In this paper, we commenced a detailed study of various DDoS tools. This paper can be useful for researchers and readers to provide the better understanding of DDoS tools in present times.  It is an attack in which multiple compromised computers are used to flood the victim servers, with a large number of packets and block them so that resources are inaccessible by the authorized users. Most of the time, the owners of the compromised hosts does not realize that they are being used by attackers. In some examples, attackers only flood the web servers in order to damage the utility, on the contrary of taking it down fully. Therefore, now a day DDoS attacks are the major concern for securing the systems present in the cyberspace world. The DDoS attack consists of four main segments – an attacker, controller, zombies and a victim and is taken place in multiple steps. The attacker compromises the multiple hosts for launching a DDoS attack to the target machine. The attacker uses the single source machine for attacking the target and also handles all the compromised machines by using remote authentication in order to command them to send the multiple requests at the same time so that it depletes the bandwidth and resources of the target machine. In this process handler uses multiple agents or daemons for sending the multiple request at the particular instant of time. In the way attackers overwhelm the victim hosts or routers, making them incompetent of providing utilities.In a direct attack, the victim is overwhelmed by large number of packets sent directly by the attacker. Types of attack packets may be TCP, ICMP, UDP, or a mixture of them. Various methods used to implement the attack include SYN Flooding, RST Flooding and ICMP Flooding.Another aspect is IP traceback. IP traceback is the process in which identification of the original sender of the packet across the Internet is done, without depend upon the source detail in the packet. IP traceback is possible in direct attacks but not when the DDoS attack is being performed. This can be done after the attack has been performed. |

Table 2.4 Literature Survey on DDoS Tools: Classification, Analysis and Comparison

**3.SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

The existing system misbehavior detection system (MDS) can be deployed to detect and prevent internal attacks. In this paper, we propose a machine learning and reputation based MDS to enhance the detection accuracy as well as to ensure the reliability of both vehicles and messages. Existing system MDS is trained using datasets generated through extensive simulation based on the realistic vehicular network environment. To improve the accuracy of the detection, we have employed the DempsterShafer (DS) theory-based collaborative misbehavior detection system. In the existing system scheme, the reputation score of each vehicle is used as a belief value for Dempster-Shafer based feedback combination. In addition, existing systema beta distribution based reputation update and revocation scheme.

**DRAWBACKS OF EXISTING SYSTEM**

* It achieves low-latency in delivery with less energy consumption.
* The drawback is not improve detection performance.
* Neural network must be trained in advance.
* Parties need to be involved.
* Cannot overcome inter-packet dependency.

**3.2 PROPOSED SYSTEM**

The proposed system presents a period-based defense mechanism (PDM) scheme is based on the periods and uses a blacklist to efficiently prevent the data flooding attack, by checking the data packet floods at the end of each period in order to enhance the throughput of burst traffic. Therefore, it can guarantee the Quality of Service (QoS) of burst traffic. As a result of which many data packets are forwarded at a high rate for the whole duration.

Flood attacks are launched by malicious or selfish nodes. Malicious nodes, which can be the nodes purposely setup by the opponent or subverted by the opponent via mobile phone worms begin attacks to congest the network and misuse the resources of other nodes. Selfish nodes may also develop flood attacks to increase their communication throughput. In DTNs, a single packet usually can only be carried to the destination with chances smaller than 1 due to the opportunistic connectivity. If a selfish node floods many replicas of its own packet, it can increase the probability of its packet being delivered, since the delivery of any replica means successful delivery of the packet. With packet flood attacks, selfish nodes can also boost their throughput, although in a subtler manner.

In the proposed Single-copy routing after sending a packet out, a node deletes its own copy of the packet. Thus, each packet only has one copy in the network. In the proposed Multicopy routing to the source node of a packet sprays a certain number of replicas of the packet to other nodes and each copy is separately routed using the single-copy strategy. The maximum number of copies that each packet can have is set.

In the proposed, Propagation routing (when a node locates it appropriate (according to the routing algorithm) to send a packet to another encountered node, it replicates that packet to the encountered node and keeps its own fake. There is no preset limit over the number of copies a packet can have. In Propagation, a node duplicates a packet to another encountered node if the latter has more regular contacts with the destination of the packet.

**ADVANTAGES OF PROPOSED SYSTEM**

* Reduce false alarm probability
* Cost reduction for its users
* Higher accuracy
* Opaqueness since routers need not be involved.
* Successfully minimize the amount of traffic.

**3.3 REQUIREMENT ANALYSIS AND SPECIFICATION**

**SOFTWARE DESCRIPTION**

ABOUT PYTHON

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.

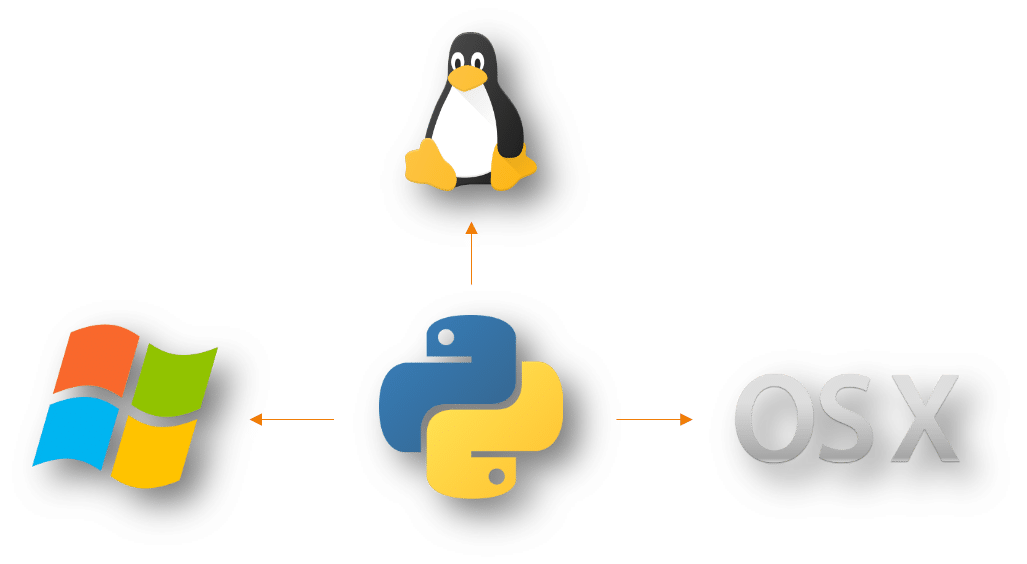


Fig 1.3.1. Python diagram

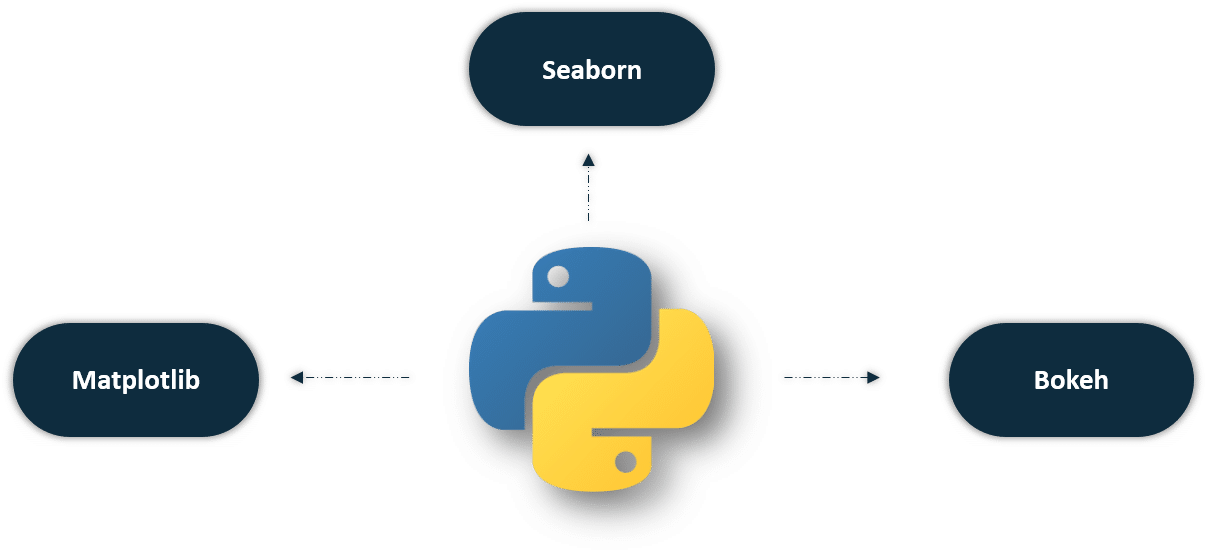
Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualization.

Fig 1.3.2. Package diagram

In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Advantages of Python

* Universal Language Construct
* Support both High Level and Low Level Programming
* Language Interoperability
* Fastest Development life cycle therefore more productive coding environment
* Less memory used because a single container hold multiple data types and each type doesn’t require its own function
* Learning Ease and open source development
* Speed and user-friendly data structure
* Extensive and extensible libraries.
* Simple & support IoT
* And many more

The important motives are

* Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
* Interactive mode
* Large and comprehensive standard libraries
* Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a difficulty when compared to C++ Java, Perl, and so forth.

ABOUT PANDAS

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The Data Frame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the Data Frame. Data Frames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas’ Data Frames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and Data Frames.

1. Series: a pandas Series is a one dimensional data structure (“a one dimensional array”) that can store values — and for every value it holds a unique index, too.
2. Data Frame: a pandas Data Frame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame’s rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, Data Frames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas Data Frame consists of three main components: the data, the index, and the columns.

Firstly, the Data Frame can contain data that is:

* a Pandas Data Frame
* a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a Data Frame.
* a NumPy ndarray, which can be a record or structured
* a two-dimensional ndarray
* dictionaries of one-dimensional ndarray’s, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

* There are tools available for loading data of different file formats into in-memory data objects.
* It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
* It is very simple to perform pivoting and reshaping of data sets in Pandas.
* It also provides indexing, label-based slicing, and sub-setting of large data sets.
* We can easily insert and delete columns from a data structure.
* Data aggregation and transformations can be done using group by.
* High-performance merging and joining of data can be done using Pandas.
* It also provides time series functionality.
* Inserting and deleting columns in data structures.
* Merging and joining data sets.
* Reshaping and pivoting data sets.
* Aligning data and dealing with missing data.
* Manipulating data using integrated indexing for DataFrame objects.
* Performing split-apply-combine on data sets using the group by engine.
* Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
* Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
* Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
* Reading from files with CSV, XLSX, TXT, among other formats.
* Arranging data in an order ascending or descending.
* Filtering data around a condition.
* Analyzing time series.
* Iterating over a data set.

NUMPY

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can’t miss when you’re learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there’s a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It’s a combination of a memory address, a data type, a shape, and strides:

* The data pointer indicates the memory address of the first byte in the array,
* The data type or dtype pointer describes the kind of elements that are contained within the array,
* The shape indicates the shape of the array, and
* The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We’ll dive into all of the possible types of multidimensional arrays later on, but for now, we’ll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it’s just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we’re able to extract an element from a matrix.

We can create a NumPy array using the numpy.array function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we’ll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathemtatical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

MATHPLOTLIB

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data vistualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc, with just a few lines of code."

**3.4 HARDWARE ENVIRONMENT**

Hardware Minimum Requirement

Disk Space 32 GB or more,10 GB or more for Foundation Edition

Processor 1.4 GHz 64 bit

Memory 512 MB

Display (800 × 600) Capable video adapter and monitor

**3.5 SOFTWARE ENVIRONMENT**

BACKEND TECHNOLOGIES

* Python
* Numpy
* Sci-learn
* Jupyter notebook

FRONTEND TECHNOLOGIES

* Web Technologies
* Bootstrap

**4.SYSTEM DESIGN**

**4.1 ER DIAGRAM**

The process of defining the architecture , product design , modules , interfaces and data for a system to satisfy specified requirements . The basic study of system design is the understanding of component parts and their subsequent interactions.

Diagram

Description automatically generated

Fig 4.1. Er diagram

**4.2 DATA DICTIONARY**

|  |  |
| --- | --- |
| Login | A Process by which an individual gains access to a web or app. |
| User | The person who uses the web or app in good way. |
| Attacker | The Person who access the web or app without authentication. |
| Packet number | A number assigned for packet. |
| Data Packet | A data Packet is a unit of data made into a single package that travels along network path. |
| Administrator | The Person who admins the web or app. |
| Dataset | The Collection of data |
| Trained set | The dataset on learning process |
| Testset | The dataset which is ready for test. |

Table 4.1 Data dictionary

**4.3 DATAFLOW DIAGRAM**

**Diagram

Description automatically generated**

Fig 4.2. Data flow diagram

**4.4 UML DIAGRAM**

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

4.4.1 USECASE DIAGRAM

Use case diagram is the simplest representation of a user’s interaction with the system that shows the relationship between the user and different use cases in which the user is involved.

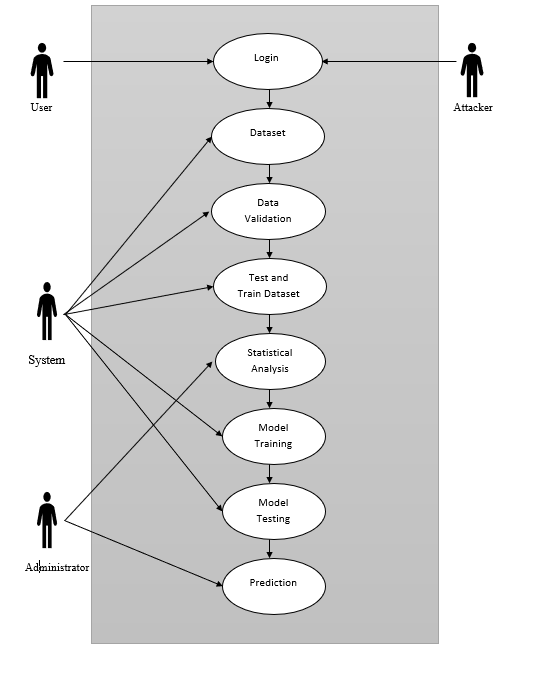


Fig 4.3.1 Use case diagram

4.4.2ACTIVITYDIAGRAM

Activity diagrams are graphical representation of workflows of stepwise activities and actions with support for choice , iteration and concurrency .Activity diagram is similar to a flowchart or a data flow diagram .They also describe the steps in use case diagram .Activities modeled can be sequential and concurrent. They include swimlanes , branching ,parallel flow , control nodes , expansion nodes and object nodes. The Advantage of activity diagram is demonstrating the logic of an algorithm and also simplifies any process by clarifying complicated use cases.

**Diagram

Description automatically generated**

Fig 4.3.2. Activity diagram

**5.SYSTEM ARCHITECTURE**

**5.1 ARCHITECTURE OVERVIEW**

**Diagram

Description automatically generated**

Fig 5.1.Architecture diagram

Graphicalrepresentationofpredictingtheddosattack. Here the flow chart shows how thehttprequestsarefilteredandformated for the analyisationof floodingoperationandatlastdetecting the attackers activity .

**5.2 MODULE DESIGN SPECIFICATION**

MODULE 1:EXPLORATORY DATA EVALUATION

Exploratory Data Analysis (EDA) is the first step in your data analysis process. Here, you make sense of the data you have and then figure out what questions you want to ask and how to frame them, as well as how best to manipulate your available data sources to get the answers you need. You do this by taking a broad look at patterns, trends, outliers, unexpected results and so on in your existing data, using visual and quantitative methods to get a sense of the story this tells.

Exploratory Data Analysis is valuable to data science projects since it allows to get closer to the certainty that the future results will be valid, correctly interpreted, and applicable to the desired business contexts. Such level of certainty can be achieved only after raw data is validated and checked for anomalies, ensuring that the data set was collected without errors. EDA also helps to find insights that were not evident or worth investigating to business stakeholders and data scientists but can be very informative about a particular business.

EDA is performed in order to define and refine the selection of feature variables that will be used for machine learning. Once data scientists become familiar with the data set, they often have to return to feature engineering step, since the initial features may turn out not to be serving their intended purpose. Once the EDA stage is complete, data scientists get a firm feature set they need for supervised and unsupervised machine learning.

MODULE 2:PRE-PROCESSING

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common idea to handle the problem is to take a mean of all the values of the same column and have it to replace the missing data.

The library that we are going to use for the task is called Scikit Learn preprocessing. It contains a class called Imputer which will help us take care of the missing data.

Sometimes our data is in qualitative form, that is we have texts as our data. We can find categories in text form. Now it gets complicated for machines to understand texts and process them, rather than numbers, since the models are based on mathematical equations and calculations. Therefore, we have to encode the categorical data.

Now we need to split our dataset into two sets — a Training set and a Test set. We will train our machine learning models on our training set, i.e. our machine learning models will try to understand any correlations in our training set and then we will test the models on our test set to check how accurately it can predict. A general rule of the thumb is to allocate 80% of the dataset to training set and the remaining 20% to test set. For this task, we will import test\_train\_split from model\_selection library of scikit

MODULE 3:FEATURE ENGINEERING

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here. For basic guidance, you can refer to the following table for defining correlation co-efficient.

Pearson’s Correlation: It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1.

LDA: Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.

ANOVA: ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.

Chi-Square: It is a is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution

MODULE 4:PREDICTION

Once training is complete, it’s time to see if the model is any good, using Evaluation. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

A good rule of thumb I use for a training-evaluation split somewhere on the order of 80/20 or 70/30. Much of this depends on the size of the original source dataset. If you have a lot of data, perhaps you don’t need as big of a fraction for the evaluation dataset.

Once you’ve done evaluation, it’s possible that you want to see if you can further improve your training in any way. We can do this by tuning our parameters. There were a few parameters we implicitly assumed when we did our training, and now is a good time to go back and test those assumptions and try other values.

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn’t split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively. Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can’t ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or die. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

* 1. **PROGRAM DESIGN LANGUAGE**

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

* Deep learning algorithms are constructed with connected layers. The first layer is called the Input Layer
* The last layer is called the Output Layer
* All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.

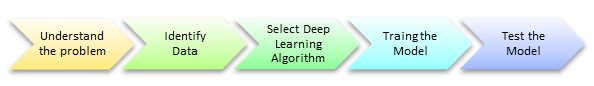


Fig 7.1.1. Procedural diagram

To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind.

A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

1. The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.
2. The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration Classification of Neural Networks

1. Shallow neural network: The Shallow neural network has only one hidden layer between the input and output.
2. Deep neural network: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers.

The computational models in Deep Learning are loosely inspired by the human brain. The multiple layers of training are called Artificial Neural Networks (ANN).

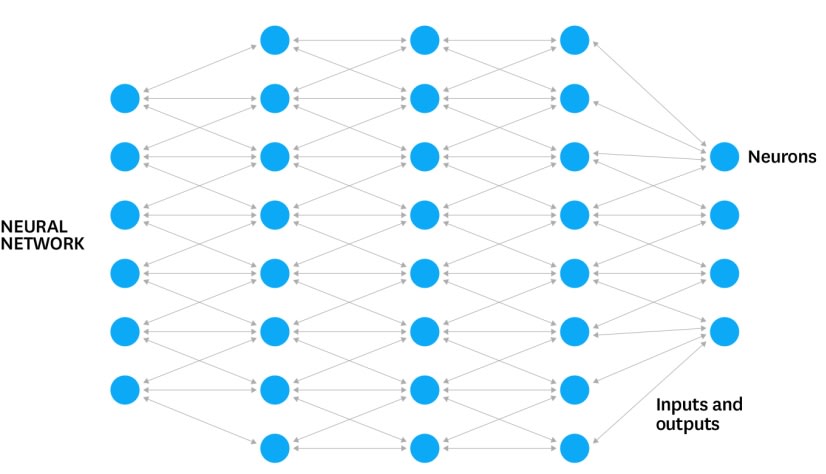


Fig 7.1.2. Neural network diagram

Neuron

Artificial Neural Networks contain layers of neurons. A neuron is a computational unit that calculates a piece of information based on weighted input parameters. Inputs accepted by the neuron are separately weighted.

Inputs are summed and passed through a non-linear function to produce output. Each layer of neurons detects some additional information, such as edges of things in a picture or tumors in a human body. Multiple layers of neurons can be used to detect additional information about input parameters.

Nodes

Artificial Neural Network is an interconnected group of nodes akin to the vast network of layers of neurons in a brain. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

Inputs

Inputs are passed into the first layer. Individual neurons receive the inputs, with

each of them receiving a specific value. After this, an output is produced based on these values.

Outputs

The outputs from the first layer are then passed into the second layer to be processed. This continues until the final output is produced. The assumption is that the correct output is predefined.

Each time data is passed through the network, the end result is compared with the correct one, and tweaks are made to their values until the network creates the correct final output each time.

Some of the commonly used neural networks are as follows:

1. Artificial Neural Network (ANN)
2. Convolutional Neural Network (CNN)
3. Recurrent Neural Network (RNN)
4. Deep Neural Network (DNN)
5. Deep Belief Network (DBN)

Artificial neural networks are one of the main tools used in machine learning. As the “neural” part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

Generally, the working of a human brain by making the right connections is the idea behind ANNs. That was limited to use of silicon and wires as living neurons and dendrites.

Here, neurons, part of human brain. That was composed of 86 billion nerve cells. Also, connected to other thousands of cells by Axons. Although, there are various inputs from sensory organs. That was accepted by dendrites. As a result, it creates electric impulses. That is used to travel through the Artificial neural network. Thus, to handle the different issues, neuron send a message to another neuron. As a result, we can say that ANNs are composed of multiple nodes. That imitate biological neurons of the human brain. Although, we connect these neurons by links. Also, they interact with each other. Although, nodes are used to take input data. Further, perform simple operations on the data. As a result, these operations are passed to other neurons. Also, output at each node is called its activation or node value.

**6.SYSTEM IMPLEMENTATION**

**CODING:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns; sns.set()

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.layers import Dense, LSTM, Bidirectional

from tensorflow.keras.utils import plot\_model

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras import utils

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.metrics import confusion\_matrix

number\_of\_samples = 50000

import plotly.express as px

import matplotlib.pyplot as plt

data\_attack = pd.read\_csv('dataset\_attack.csv', nrows = number\_of\_samples)

data\_normal= pd.read\_csv('dataset\_normal.csv', nrows = number\_of\_samples)

data\_normal.columns=[ 'frame.len', 'frame.protocols', 'ip.hdr\_len',

'ip.len', 'ip.flags.rb', 'ip.flags.df', 'p.flags.mf', 'ip.frag\_offset',

'ip.ttl', 'ip.proto', 'ip.src', 'ip.dst', 'tcp.srcport', 'tcp.dstport',

'tcp.len', 'tcp.ack', 'tcp.flags.res', 'tcp.flags.ns', 'tcp.flags.cwr',

'tcp.flags.ecn', 'tcp.flags.urg', 'tcp.flags.ack', 'tcp.flags.push',

'tcp.flags.reset', 'tcp.flags.syn', 'tcp.flags.fin', 'tcp.window\_size',

'tcp.time\_delta','class']

data\_attack.columns=[ 'frame.len', 'frame.protocols', 'ip.hdr\_len',

'ip.len', 'ip.flags.rb', 'ip.flags.df', 'p.flags.mf', 'ip.frag\_offset',

'ip.ttl', 'ip.proto', 'ip.src', 'ip.dst', 'tcp.srcport', 'tcp.dstport',

'tcp.len', 'tcp.ack', 'tcp.flags.res', 'tcp.flags.ns', 'tcp.flags.cwr',

'tcp.flags.ecn', 'tcp.flags.urg', 'tcp.flags.ack', 'tcp.flags.push',

'tcp.flags.reset', 'tcp.flags.syn', 'tcp.flags.fin', 'tcp.window\_size',

'tcp.time\_delta','class']

df = pd.concat([data\_attack, data\_normal])

df.head()

df.info()

print('number of classes:', df['class'].nunique())

print('')

label\_counts = df['class'].value\_counts()

plt.figure(figsize=(18,6));

sns.barplot(y=label\_counts.index, x=label\_counts.values, color='Grey');

plt.title('values per class');

display(label\_counts)

binary\_class = []

for label in df['class']:

if label !='normal':

binary\_class.append('Attack')

else:

binary\_class.append('normal')

binary\_class = pd.Series(binary\_class)

plt.figure()

binary\_class.value\_counts().plot(kind='pie', label='traffic proportions', autopct='%.2f%%' );

numeric\_columns = []

categorical\_columns = []

for column in df.columns:

if df[column].dtype != 'object':

numeric\_columns.append(column)

else:

categorical\_columns.append(column)

categorical\_columns = categorical\_columns[:-1]

labels=df['class'].unique()

for column in numeric\_columns:

plt.figure(figsize=(18,7))

sns.boxenplot(x='class', y=df[column], data=df);

plt.title(column);

fig = px.treemap(df, path=['ip.src','ip.dst'], title='Source IP => Destination IP',

width=1000, height=800)

fig.show()

def bar\_graph(feature):

df[feature].value\_counts().plot(kind="bar")

bar\_graph('ip.proto')

plt.figure(figsize=(15,3))

bar\_graph('tcp.ack')

bar\_graph('ip.flags.rb')

bar\_graph('ip.flags.rb')

bar\_graph('ip.flags.df')

bar\_graph('p.flags.mf')

df = df[[col for col in df if df[col].nunique() > 1]]# keep columns where there are more than 1 unique values

corr = df.corr()

plt.figure(figsize=(15,12))

sns.heatmap(corr)

plt.show()

data\_normal=data\_normal.drop(['ip.src', 'ip.dst','frame.protocols'],axis=1)

data\_attack=data\_attack.drop(['ip.src', 'ip.dst','frame.protocols'],axis=1)

features=[ 'frame.len', 'ip.hdr\_len',

'ip.len', 'ip.flags.rb', 'ip.flags.df', 'p.flags.mf', 'ip.frag\_offset',

'ip.ttl', 'ip.proto', 'tcp.srcport', 'tcp.dstport',

'tcp.len', 'tcp.ack', 'tcp.flags.res', 'tcp.flags.ns', 'tcp.flags.cwr',

'tcp.flags.ecn', 'tcp.flags.urg', 'tcp.flags.ack', 'tcp.flags.push',

'tcp.flags.reset', 'tcp.flags.syn', 'tcp.flags.fin', 'tcp.window\_size',

'tcp.time\_delta']

X\_normal= data\_normal[features].values

X\_attack= data\_attack[features].values

Y\_normal= data\_normal['class']

Y\_attack= data\_attack['class']

X=np.concatenate((X\_normal,X\_attack))

Y=np.concatenate((Y\_normal,Y\_attack))

scalar = StandardScaler(copy=True, with\_mean=True, with\_std=True)

scalar.fit(X)

X = scalar.transform(X)

for i in range(0,len(Y)):

if Y[i] =="attack":

Y[i]=0

else:

Y[i]=1

features = len(X[0])

samples = X.shape[0]

train\_len = 25

input\_len = samples - train\_len

I = np.zeros((samples - train\_len, train\_len, features))

for i in range(input\_len):

temp = np.zeros((train\_len, features))

for j in range(i, i + train\_len - 1):

temp[j-i] = X[j]

I[i] = temp

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(I, Y[25:100000], test\_size = 0.2)

def create\_model():

model = Sequential()

model.add(Bidirectional(LSTM(64, activation='tanh', kernel\_regularizer='l2')))

model.add(Dense(128, activation = 'relu', kernel\_regularizer='l2'))

model.add(Dense(1, activation = 'sigmoid', kernel\_regularizer='l2'))

model.compile(loss = 'binary\_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

return model

model = create\_model()

history = model.fit(X\_train, Y\_train, epochs = 3,validation\_split=0.2, verbose = 1)

plt.plot(history.history['acc'])

plt.plot(history.history['val\_acc'])

plt.title('BRNN Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='lower right')

plt.savefig('Model Accuracy.png')

plt.show()

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('BRNN Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.savefig('Model Loss.png')

plt.show()

predict = model.predict(X\_test, verbose=1)

tp = 0

tn = 0

fp = 0

fn = 0

predictn = predict.flatten().round()

predictn = predictn.tolist()

Y\_testn = Y\_test.tolist()

for i in range(len(Y\_testn)):

if predictn[i]==1 and Y\_testn[i]==1:

tp+=1

elif predictn[i]==0 and Y\_testn[i]==0:

tn+=1

elif predictn[i]==0 and Y\_testn[i]==1:

fp+=1

elif predictn[i]==1 and Y\_testn[i]==0:

fn+=1

to\_heat\_map =[[tn,fp],[fn,tp]]

to\_heat\_map = pd.DataFrame(to\_heat\_map, index = ["Attack","Normal"],columns = ["Attack","Normal"])

ax = sns.heatmap(to\_heat\_map,annot=True, fmt="d")

figure = ax.get\_figure()

figure.savefig('confusion\_matrix\_BRNN.png', dpi=400)

model.save('ddos\_model.h5')

scores = model.evaluate(X\_test, Y\_test, verbose=0)

print("%s: %.2f%%" % (model.metrics\_names[1], scores[1]\*100))

**7.SYSTEM TESTING**

**7.1 UNIT TESTING**

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast.

**7.2 INTEGRATION TESTING**

Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. It is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Integration testing is a key aspect of software testing.

**7.3 PERFORMANCE ANALYSIS**

The technique of comparing the performance of a specific situation in contrast to the aim and yet executed. The first step of analyzing raw data is validated and checked for anomalies , ensuring that the data set was collected without errors and then pre-processing step splits the dataset into trained set and test set , after training the model ,it’s completely ready for evaluation . Once evaluation done , It’s possible you can further improve the training any way and the prediction of attackers and non-attackers counts will be well aimed than the existing model.

There were a few parameters we implicitly assumed when we did our training and now is the good time to go back and test those assumptions and try other values.

Chart

Description automatically generated with low confidence

Fig 9.1. Performance analysis diagram

**8. CONCLUSION AND FUTURE ENHANCEMENTS**

Reduce the attacks by employing rate limitations and probablistically detect the number of packets and the Long-short term memory algorithm is used to detect the approximate counting of packets which are violating the rate limits . This works are implemented in distributed manner.They easily reduce the throughput of burst traffic by comparing with the simple threshold .This is achieved by using proposed scheme and more over it is better than old scheme.

**8.1 FUTURE ENHANCEMENTS**

In future, the model can be implemented on other attacks like Application Layer Attack, Protocol Attack, Volumetric Attack with large dataset using ANN.

**APPENDICES**

**A.1 SAMPLE SCREENS**

Text

Description automatically generated

Fig 9.2. Anaconda Prompt

Graphical user interface, background pattern

Description automatically generated

Fig 9.3.Value per class

Chart, pie chart

Description automatically generated

Fig 9.4. Pie chart for normal and attack

Chart, treemap chart

Description automatically generated

Fig 9.5. IP address with diagram

A picture containing text

Description automatically generated

Fig 9.6.TCP Acknowledgment

Chart, line chart

Description automatically generated

Fig 9.7.BRNN Model accuracy

Chart, line chart

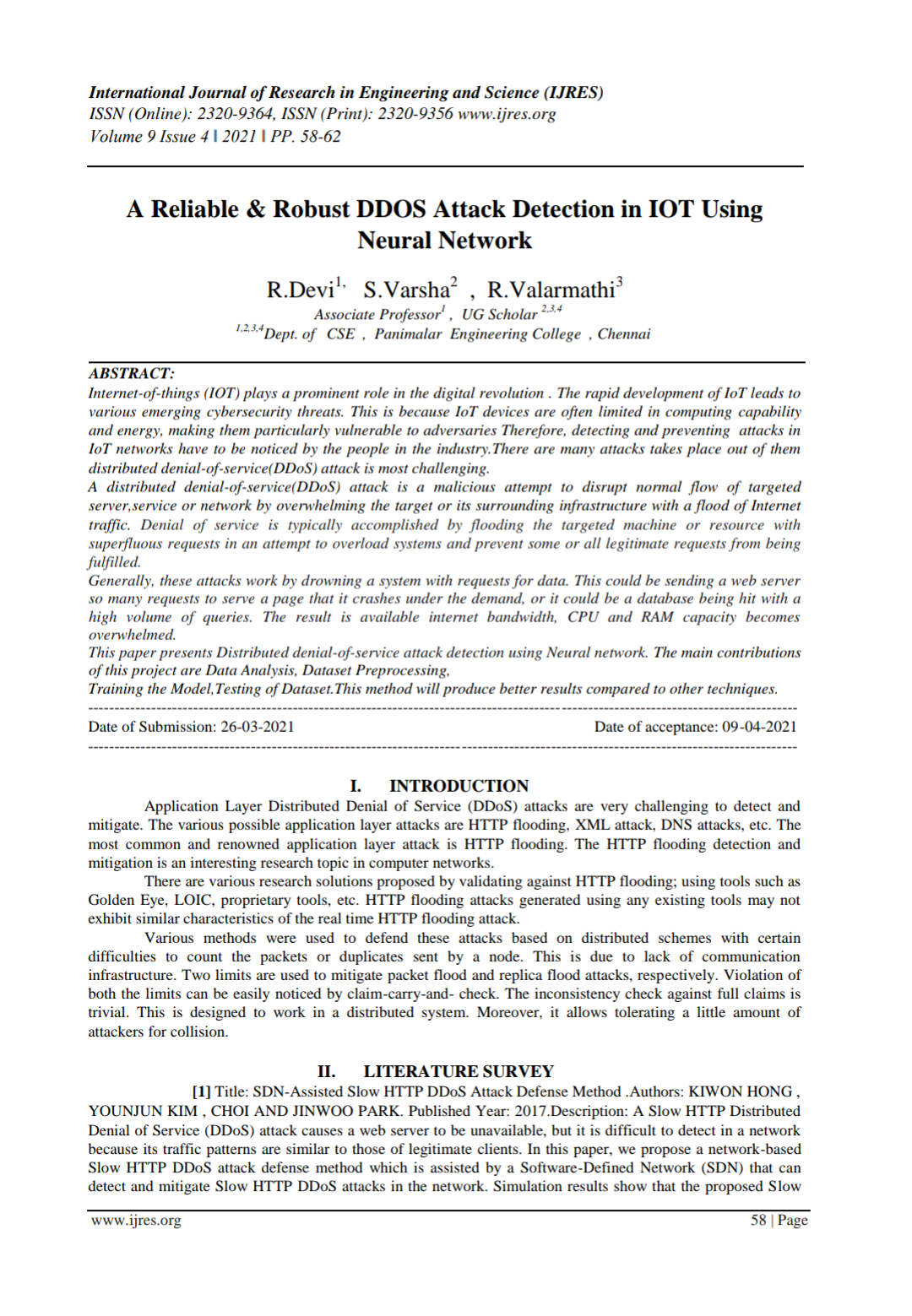
Description automatically generated

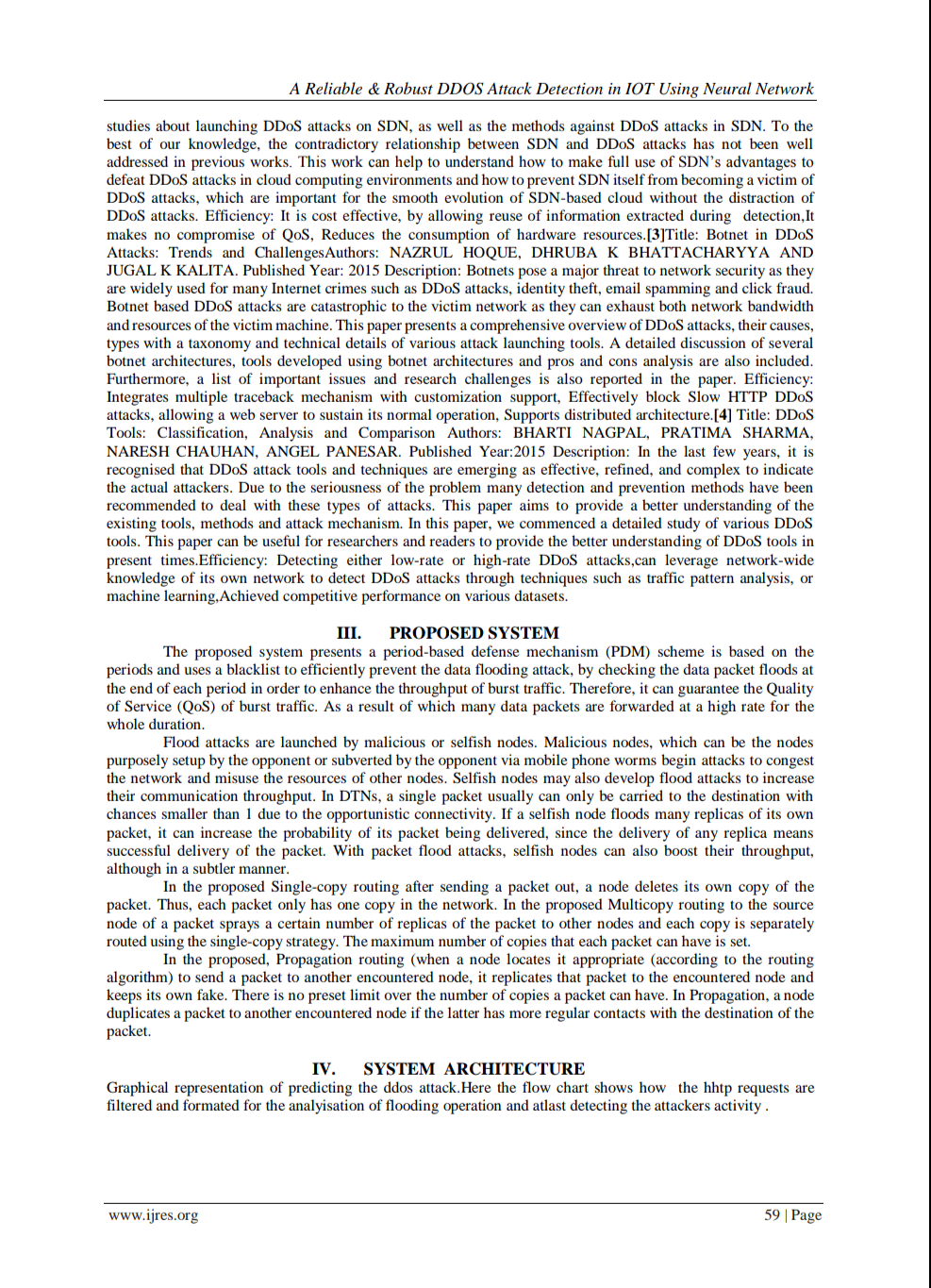
Fig 9.8.BRNN Model loss

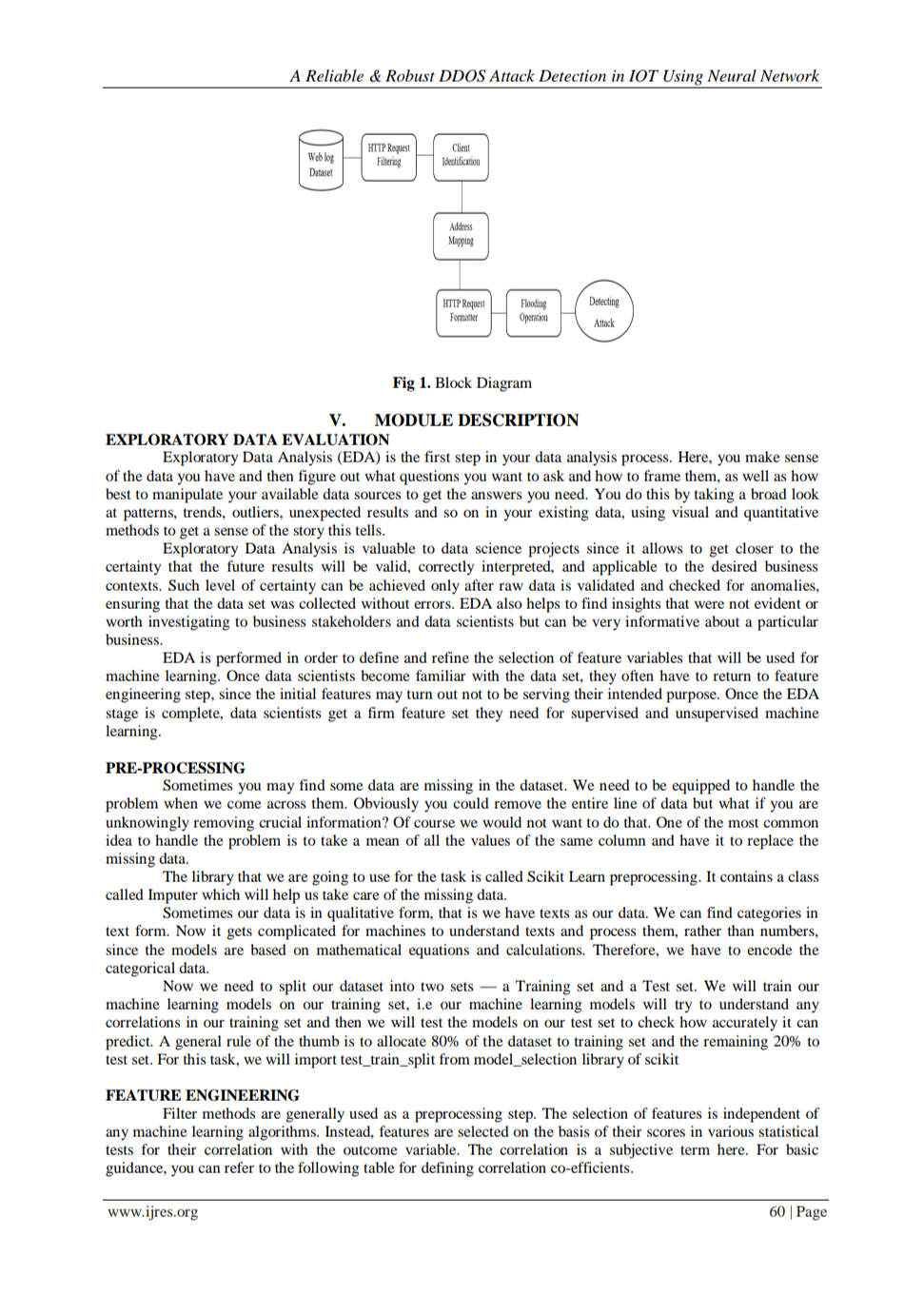
Chart

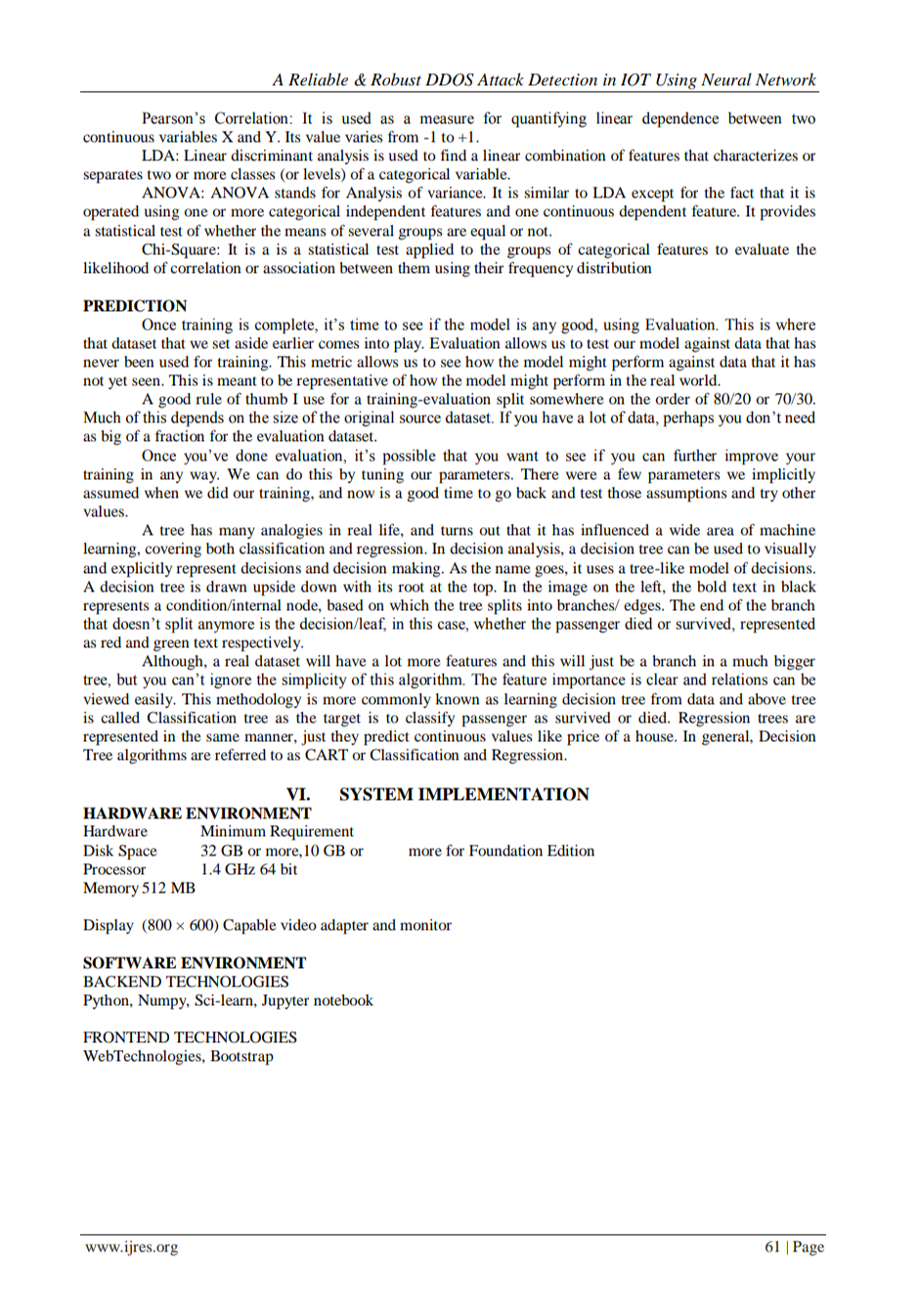
Description automatically generated with low confidence

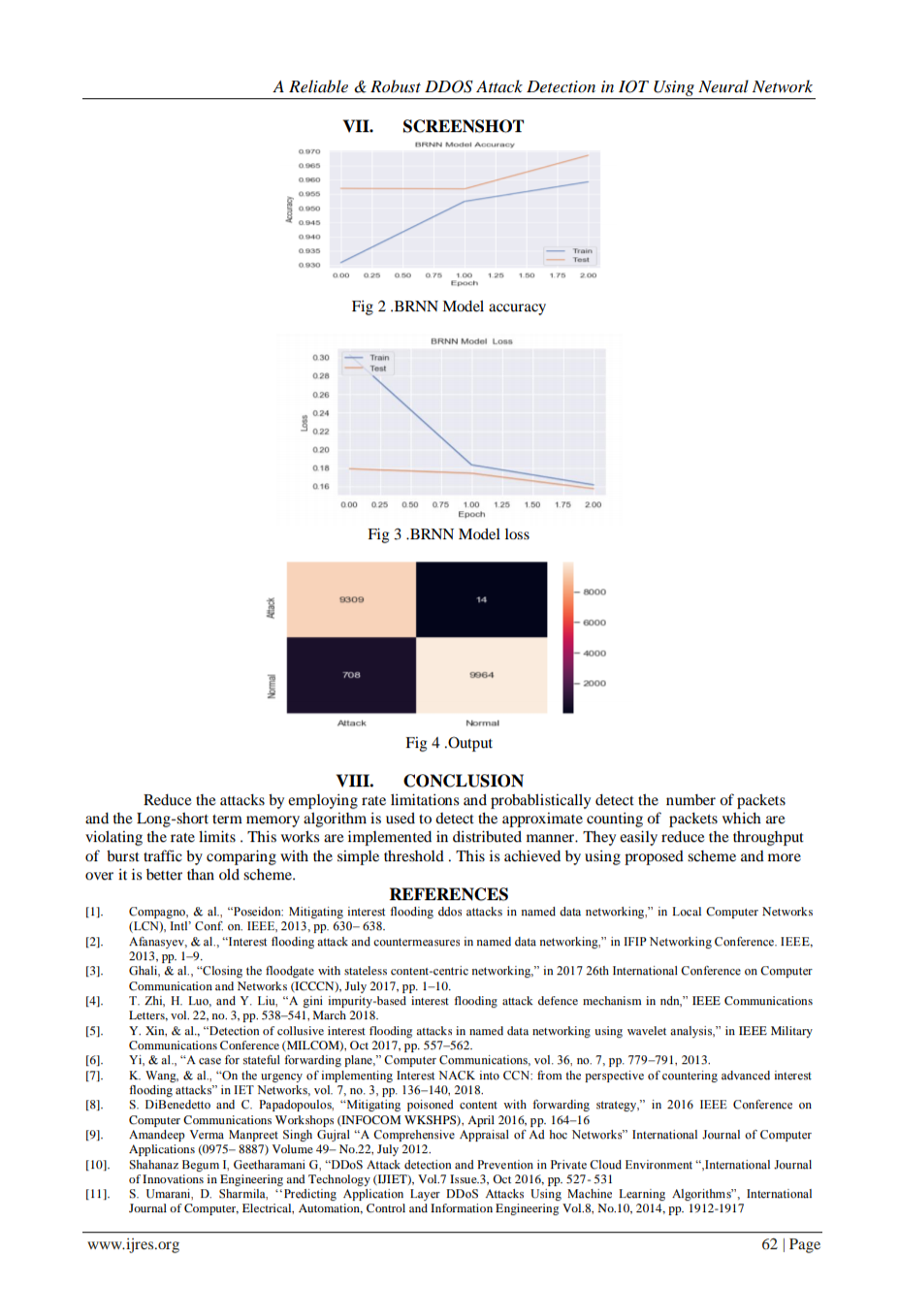
Fig 9.9.Output

**A.2 PUBLICATIONS**

****

****

****

****

****

****

****

****

**REFERENCES**

* Compagno, & al., “Poseidon: Mitigating interest flooding ddos attacks in named data networking,” in Local Computer Networks (LCN), Intl’ Conf. on. IEEE, 2013, pp. 630– 638.
* Afanasyev, & al., “Interest flooding attack and countermeasures in named data networking,” in IFIP Networking Conference. IEEE, 2013, pp. 1–9.
* Ghali, & al., “Closing the floodgate with stateless content-centric networking,” in 2017 26th International Conference on Computer Communication and Networks (ICCCN), July 2017, pp. 1–10.
* T. Zhi, H. Luo, and Y. Liu, “A gini impurity-based interest flooding attack defence mechanism in ndn,” IEEE Communications Letters, vol. 22, no. 3, pp. 538–541, March 2018.
* Y. Xin, & al., “Detection of collusive interest flooding attacks in named data networking using wavelet analysis,” in IEEE Military Communications Conference (MILCOM), Oct 2017, pp. 557–562.
* Yi, & al., “A case for stateful forwarding plane,” Computer Communications, vol. 36, no. 7, pp. 779–791, 2013.
* K. Wang, & al., “On the urgency of implementing Interest NACK into CCN: from the perspective of countering advanced interest flooding attacks” in IET Networks, vol. 7, no. 3, pp. 136–140, 2018.
* S. DiBenedetto and C. Papadopoulos, “Mitigating poisoned content with forwarding strategy,” in 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), April 2016, pp. 164–16
* Amandeep Verma Manpreet Singh Gujral “A Comprehensive Appraisal of Ad hoc Networks” International Journal of Computer Applications (0975– 8887) Volume 49– No.22, July 2012.
* Shahanaz Begum I, Geetharamani G, “DDoS Attack detection and Prevention in Private Cloud Environment “,International Journal of Innovations in Engineering and Technology (IJIET), Vol.7 Issue.3, Oct 2016, pp. 527- 531
* S. Umarani, D. Sharmila, ‘‘Predicting Application Layer DDoS Attacks Using Machine Learning Algorithms”, International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol.8, No.10, 2014, pp. 1912-1917