

SECURING DATA BY DETECTING MULTI CHANNEL ATTACKS USING DEEP LEARNING

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering Degree in Computer Science and Engineering
by

KARTHIKA .R(3711385)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI

SALAI, CHENNAI – 600119.

APRIL 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
Chennai - 600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **KARTHIKA .R(3711385)** who carried out the project entitled "**SECURING DATA BY DETECTING MULTI CHANNEL ATTACKS USING DEEP LEARNING**" under our supervision from November 2020 to April 2021.

Internal Guide
Dr.P.JEYANTHI,M.Tech.,Phd.,
Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **KARTHIKA .R(3711385)** hereby declare that the Project Report entitled“”**SECURING DATA BY DETECTING MULTI CHANNEL ATTACKS USING DEEP LEARNING**” done by me under the guidance of **Dr.R.Jeyanthi**, at Sathyabama Institute of Science and technology is submitted in partial fulfillment of the requirements of the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey our thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean, School of Computing, Dr. S. VIGNESHWARI M.E., Ph.D. and Dr. L. LAKSHMANAN M.E., Ph.D., Heads of the Department**, Department of Computer Science and Engineering for providing us the necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr.R.Jeyanthi,M.Tech.,Phd.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

ABSTRACT

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

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NUMBER
	ABSTRACT	
	LIST OF FIGURES	
1	INTRODUCTION	
	1.1 OVERVIEW	
	1.2 SCOPE OF THE PROJECT	
	1.3 DOMAIN OVERVIEW	
	1.3.1 DEEP LEARNING	
2	LITERATURE SURVEY	
3	METHODS AND TECHNIQUES	
	3.1 EXISTING SYSTEM	
	3.2 DRAWBACKS OF EXISTING SYSTEM	
	3.3 PROPOSED SYSTEM	
	3.4 ADVANTAGES OF PROPOSED SYSTEM	
	3.5 HARDWARE & SOFTWARE SPECIFICATION	
	3.5.1 HARDWARE SPECIFICATION	
	3.5.2 SOFTWARE SPECIFICATION	
	3.6 SOFTWARE DESCRIPTION	
	3.7 INTRODUCTION	
	3.8 ARCHITECTURE DIAGRAM	
	3.9 DATA PRE-PROCESSING	

4 RESULTS & DISCUSSIONS

5 Conclusion and Future Work

APPENDIX-I S A M P L E C O D E

APPENDIX-II SCREEN SHOTS

REFERENCES

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.1	DEEP LEARNING	3
1.3.1	NEURAL NETWORK	4
3.6	TENSORFLOW ARCHITECTURE	25
3.6	TENSOR FLOW STRUCTURE	27
3.6	KERAS LAYERS	28
4.2	ARCHIECTURE DIAGRAM	31

CHAPTER 1

INTRODUCTI

ON

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

1.1 OVERVIEW

The proposed system identifies the factors that augment the attack impact. We selected the second scenario because it stands for the straightest case for an attacker to exploit vulnerabilities in the current NDN design and its NFD implementation.

1.2 SCOPE OF THE PROJECT

The main contributions of this project therefore are:

- Data Analysis
- Dataset preprocessing
- Training the Model
- Testing of Dataset

1.2 DOMAIN OVERVIEW

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from

a lamppost. It

is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

1.3.1 DEEP LEARNING

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.



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.

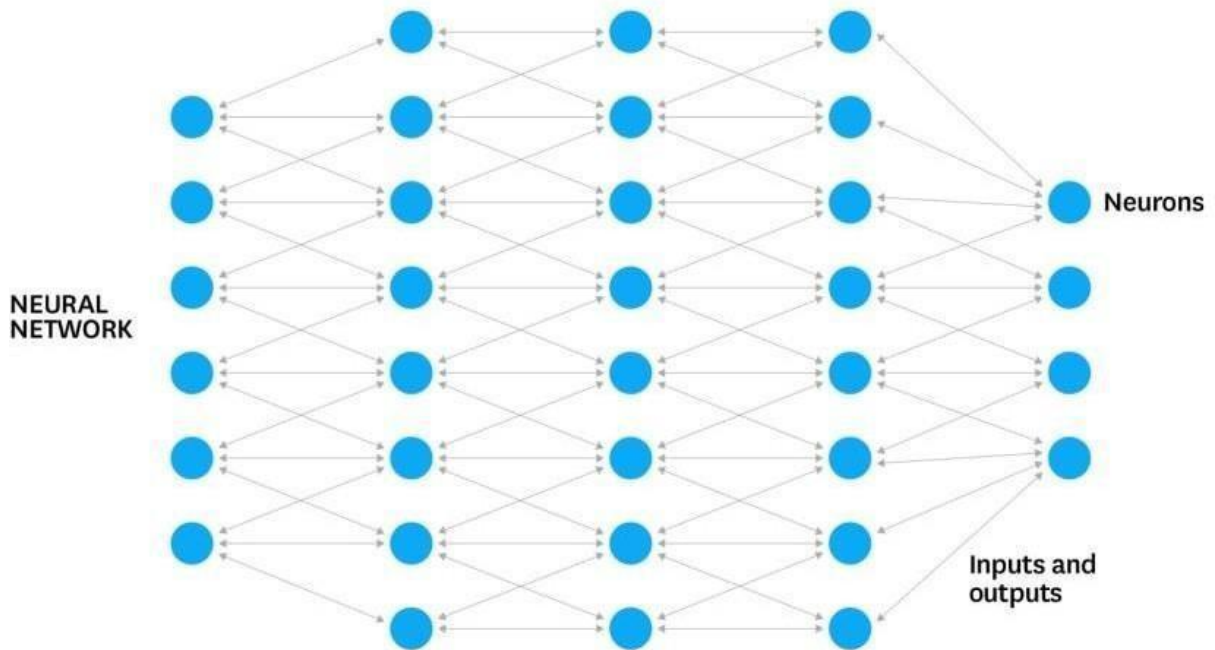
The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration

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).



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.

CHAPTER 2

LITERATURE

SURVEY

Literature Survey 1	
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 C_{open} with the threshold value C_{th}. If $C_{open} \geq C_{th}$, 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 T_c to perform timeout-based DDoS attack detection, which identifies an incomplete HTTP request that is not completed before T_c 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.
--	--

Literature Survey 2

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 jamming 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 jamming 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-a-service 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
--	---

Literature Survey 3	
Title	Botnet in DDoS Attacks: Trends and Challenges
Authors	Nazrul Hoque, Dhruva 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 jamming and request jamming. In a session jamming 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 jamming, 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.
--	--

Literature Survey 4	
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 Jamming, RST Jamming and ICMP Jamming. The methodologies are briefly explained in Table I. 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.
--	---

CHAPTER 3

Methods and Techniques

3.1 EXISITING SYSTEM

ICN is a networking paradigm which is based on data objects. The key concept in ICN is that it names each data object in the network, instead of using IP addresses for naming hosts and nodes. ICN also deploys in-network caching to enhance the delivery of popular data objects. Besides, a node in ICN does not have to connect to one specific server to get data. Alternately, this node sends a request with the data name and the network will return the corresponding object, either from a cache or from the original provider.

In NDN, communications are based on requests for hierarchical content names and are performed by Interest and Data packets. A user sends an Interest to request a content and receives a Data in return. An NDN router includes three main components. First is the Forwarding Information Base (FIB) which contains routing information for Interest. Secondly, the Content Store (CS) is essentially a local cache which stores recently requested Data to improve performances. Finally, the Pending Interest Table (PIT) contains entries for each forwarded Interest, and uses them as reverse-path routing information for Data. A PIT entry contains an NDN name and multiple incoming interfaces.

3.2 DRAWBACKS OF EXISITING 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

3.3 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.

3.4 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

3.5 HARDWARE & SOFTWARE SPECIFICATION

3.5.1 HARDWARE SPECIFICATION

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

3.5.2 SOFTWARE SPECIFICATION

Backend Technologies

- ✓ Python
- ✓ Numpy
- ✓ Sci-learn
- ✓ Eclipse

IDE Frontend

Technologies

- ✓ Web Technologies
- ✓ Bootstrap

hon to implement machine learning in the back-end.

3.6 SOFTWARE

DESCRIPTION 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 DataFrame 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 DataFrame. DataFrames 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 Notebook

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' DataFrames 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 DataFrames.

1. Series: a pandas Series is a one dimensional data structure ("a one dimensional ndarray") that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame 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, DataFrames 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 DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- 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 DataFrame.
- 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 mathematical 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.

Matplotlib

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 visualization, 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, errorcharts, scatterplots, etc, with just a few lines of code."

TensorFlow is the second machine learning framework that Google created and used to design, build, and train deep learning models. You can use the TensorFlow library do to numerical computations, which in itself doesn't seem all too special, but these computations are done with data flow graphs. In these graphs, nodes represent mathematical operations, while the edges represent the data, which usually are multidimensional data arrays or tensors, that are communicated between these edges.

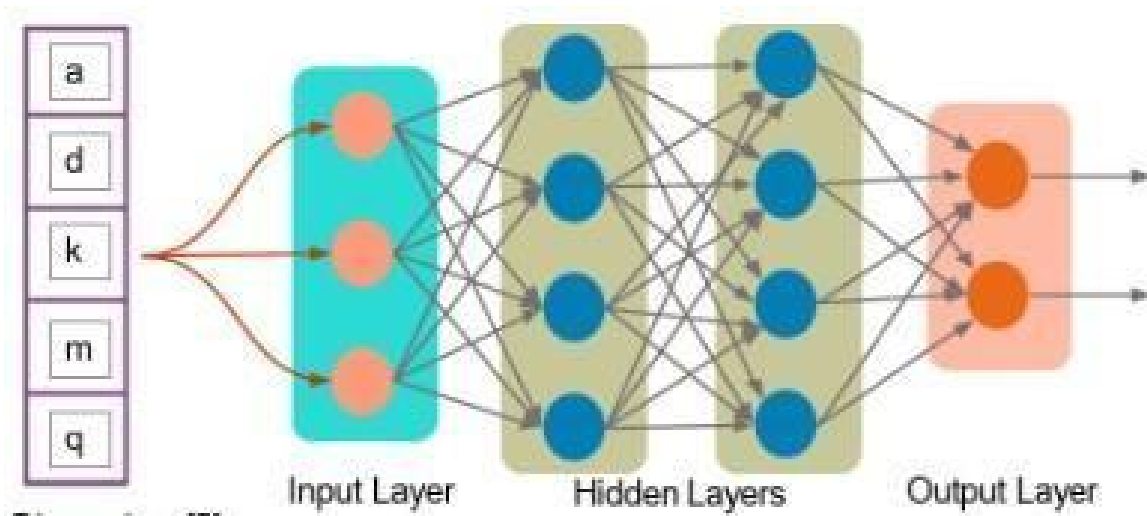
TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources.

To understand tensors well, it's good to have some working knowledge of linear algebra and vector calculus. You already read in the introduction that tensors are implemented in TensorFlow as multidimensional data arrays, but some more introduction is maybe needed in order to completely grasp tensors and their use in machine learning.

Plane Vectors

Before you go into plane vectors, it's a good idea to shortly revise the concept of "vectors"; Vectors are special types of matrices, which are rectangular arrays of numbers. Because vectors are ordered collections of numbers, they are often seen as column matrices: they have just one column and a certain number of rows. In other terms, you could also consider vectors as scalar magnitudes that have been given a direction.

A tensor is a mathematical object represented as arrays of higher dimensions. These arrays of data with different sizes and ranks get fed as input to the neural network. These are the tensors.



Tensorflow architecture works in three parts:

1. Preprocessing the data
2. Build the model
3. Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

Tensor flow's name is directly derived from its core framework: Tensor. In Tensorflow, all the computations involve tensors. A tensor is a vector or matrix of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) shape. The shape of the data is the dimensionality of the matrix or array.

A tensor can be originated from the input data or the result of a computation. In TensorFlow, all the

operations are conducted inside a graph. The graph is a set of computation that takes place successively. Each operation is called an op node and are connected to each other.

The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes is the tensor, i.e., a way to populate the operation with data.

Graphs

TensorFlow makes use of a graph framework. The graph gathers and describes all the series computations done during the training. The graph has lots of advantages:

It was done to run on multiple CPUs or GPUs and even mobile operating system

The portability of the graph allows to preserve the computations for immediate or later use. The graph can be saved to be executed in the future.

All the computations in the graph are done by connecting tensors together

A tensor has a node and an edge. The node carries the mathematical operation and produces an endpoints outputs. The edges the edges explain the input/output relationships between nodes.

Generally, training the model is where a large part of the computation goes. Also, the process of training is repeated multiple times to solve any issues that may arise. This process leads to the consumption of more power, and therefore, you need a distributed computing. If you need to process large amounts of data, TensorFlow makes it easy by running the code in a distributed manner.

GPUs, or graphical processing units, have become very popular. Nvidia is one of the leaders in this space. It is good at performing mathematical computations, such as matrix multiplication, and plays a significant role in deep learning. TensorFlow also has integration with C++ and Python API, making development much faster.

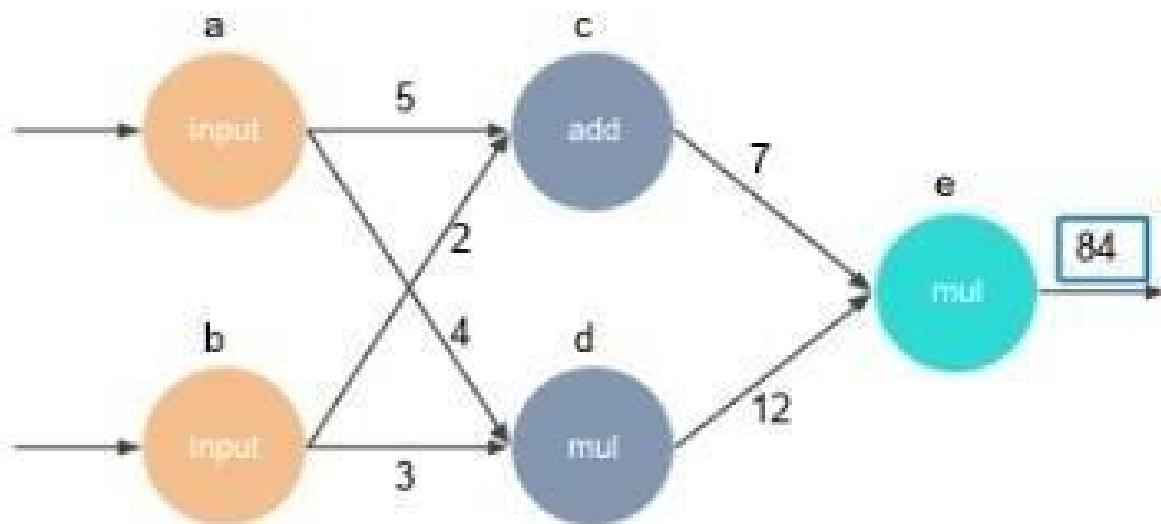
Everything in TensorFlow is based on designing a computational graph. The graph has a network of nodes, with each node operating addition, multiplication, or evaluating some multivariate equation.

The code is written to build the graph, create a session, and execute that graph. A graph has nodes

that represent mathematical operations, and an edge represents tensors. In TensorFlow, a computation is explained using a data flow graph.

Everything is an operation. Not only adding two variables but creating a variable is also an operation. Every time you assign a variable, it becomes a node. You can perform mathematical operations, such as addition and multiplication on that node.

You start by building up these nodes and executing them in a graphical format. That is how the TensorFlow program is structured.



Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano. Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras in Python doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

One of the most widely used concepts today is Deep Learning. It is very vital that you learn Keras metrics and implement them actively. Deep Learning originates from Machine Learning and eventually contributes to the achievement of Artificial Intelligence. With a neural network, inputs can easily be supplied to it and processed to obtain insights. The processing is done by making use of hidden layers with weights, which are continuously monitored and tweaked when training the model. These weights are used to find patterns in data to arrive at a prediction. With neural networks, users need not specify

what pattern to hunt for because neural networks learn this aspect on their own and work with it

Backend is a term in Keras that performs all low-level computation such as tensor products, convolutions and many other things with the help of other libraries such as Tensorflow or Theano. So, the "backend engine" will perform the computation and development of the models. Tensorflow is the default "backend engine" but we can change it in the configuration.

The simple API structure of Keras is designed for both new developers and

experts. The Keras interface is very user friendly and is pretty optimized for

general use cases. In Keras, you can write custom blocks to extend it.

Keras is the second most popular deep learning framework after TensorFlow.

Tensorflow also provides Keras implementation using its `tf.keras` module. You can access all the functionalities of Keras in TensorFlow using `tf.keras`.

Keras Layers



Input nodes. As it so happens, each input node is associated with a numerical value, which can be any real number. Remember that real numbers make up the full spectrum of numbers: they can be positive or negative, whole or decimal numbers.

Connections. Similarly, each connection that departs from the input node has a weight associated with

it and this can also be any real number.

Next, all the values of the input nodes and weights of the connections are brought together: they are used as inputs.

3.6 INTRODUCTION

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.

3.8 ARCHITECTURE DIAGRAM

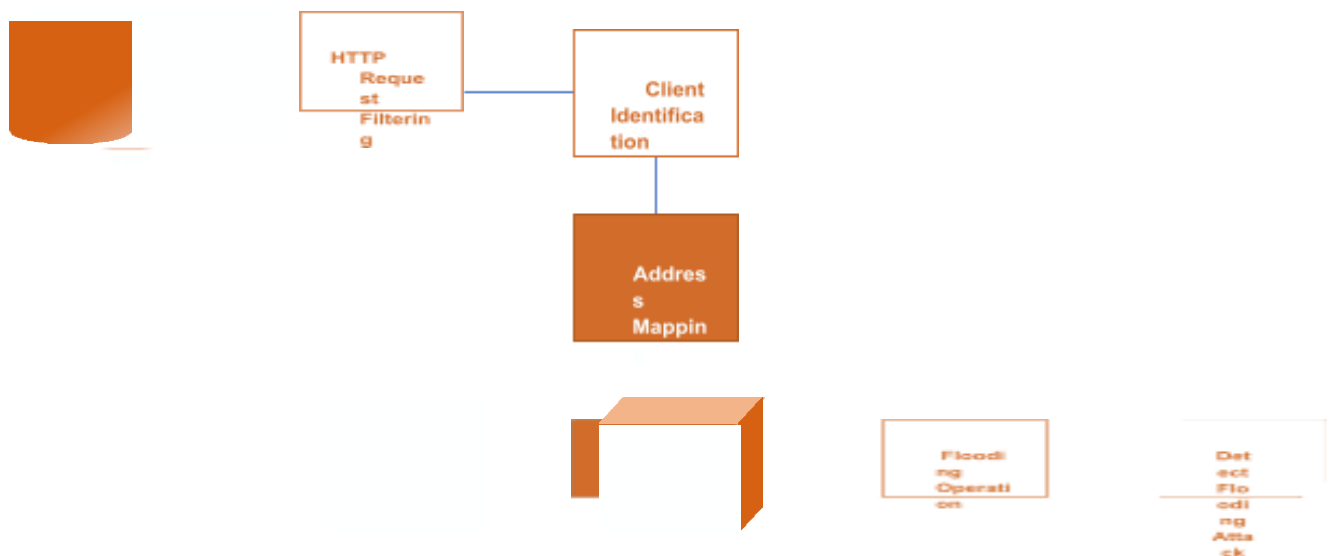
An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase

helping partners understand the architecture, discuss changes, and communicate intentions clearly.



3.9 Data Processing

Preparing the data is one of the most time-consuming parts of any data analysis/data mining project. This chapter outlines concepts and steps necessary to prepare a data set prior to any data analysis or data mining exercise. How the data is collected and prepared is critical to the confidence with which decisions can be made. The data needs to be pulled together into a table. This may involve integration of the data from multiple sources. Once the data is in a tabular format it should be fully characterized. The data should also be cleaned by resolving any ambiguities, errors, and removing redundant and problematic data. Certain columns of data can be removed if it is obvious that they would not be useful in any analysis. For a number of reasons, new columns of data may need to be calculated.

A survey or poll can be useful for gathering data to answer specific questions. An interview using a set

of predefined questions is usually conducted either over the phone, in person or over the Internet. They are often used to elicit information on people's opinions, preferences and behavior. For example, a poll may be used to understand how a population of eligible voters will cast their vote in an upcoming election. The specific questions to be answered along with the target population should be clearly defined prior to any survey. Any bias in the survey should be eliminated. To achieve this, a true random sample of the target population should be taken. Bias can be introduced in situations where only those responding to the questionnaire are included in the survey since this group may not represent an unbiased random sample. The questionnaire should contain no leading questions, that is, questions that favor a particular response. It is also important that no bias relating to the time the survey was conducted, is introduced. The sample of the population used in the survey should be large enough to answer the questions with confidence. This will be described in more detail within the chapter on statistics.

The variable's scale indicates the accuracy at which the data has been measured. This classification has implications as to the type of analysis that can be performed on the variable.

Nominal: Scale describing a variable with a limited number of different values. This scale is made up of the list of possible values that the variable may take. It is not possible to determine whether one value is larger than another. For example, a variable Industry would be nominal where it takes values such as financial, engineering, retail, and so on. The order of these values has no meaning.

Ordinal: This scale describes a variable whose values are ordered; however, the difference between the values does not describe the magnitude of the actual difference. For example, a scale where the only values are low, medium, and high tells us that high is larger than medium, and medium is larger than low. However, it is impossible to determine the magnitude of the difference between the three values.

Interval: Scales that describe values where the interval between the values has meaning. For example, when looking at three data points measured on the Fahrenheit scale, 5 F, 10 F, 15 F, the differences between the values from 5 to 10 and from 10 to 15 are both 5 and a difference of 5 F in both cases has the same meaning. Since the Fahrenheit scale does not have a lowest value at zero, a doubling of a value does not imply a doubling of the actual measurement. For example, 10 F is not twice as hot as 5 F. Interval scales do not have a natural zero.

It is also useful to think about how the variables will be used in any subsequent analysis. Example roles in data analysis and data mining include

Labels: Variables that describe individual observations in the data.

Descriptors: These variables are almost always collected to describe an observation. Since they are often present, these variables are used as the input or descriptors to be used in both creating a predictive model and generating predictions from these models. They are also described as predictors or X variables.

Response: These variables (usually one variable) are predicted from a predictive model (using the descriptor variables as input). These variables will be used to guide the creation of the predictive model. They will also be predicted, based on the input descriptor variables that are presented to the model. They are also referred to as Y variables.

For variables with an ordered scale (ordinal, interval, or ratio), it is useful to look at the frequency distribution. The frequency distribution is based on counts of values or ranges of values (in the case of interval or ratio scales).

Since the data available for analysis may not have been originally collected with this project's goal in mind, it is important to spend time cleaning the data. It is also beneficial to understand the accuracy with which the data was collected as well as correcting any errors. For variables measured on a nominal or ordinal scale (where there are a fixed number of possible values), it is useful to inspect all possible values to uncover mistakes and/or inconsistencies. Any assumptions made concerning possible values that the variable can take should be tested.

It can be more challenging to clean variables measured on an interval or ratio scale since they can take any possible value within a range. However, it is useful to consider outliers in the data. Outliers are a single or a small number of data values that are not similar to the rest of the data set. There are many reasons for outliers. An outlier may be an error in the measurement. A series of outlier data points could be a result of measurements made using a different calibration. An outlier may also be a genuine data point. Histograms, scatterplots, box plots and z-scores can be useful in identifying outliers

It is important to consider applying certain mathematical transformations to the data since many data

analysis/data mining programs will have difficulty making sense of the data in its raw form. Some common transformations that should be considered include normalization, value mapping, discretization, and aggregation. When a new variable is generated, the transformation procedure used should be retained. The inverse transformation should then be applied to the variable prior to presenting any analysis results that include this variable.

To use variables that have been assigned as ordinal and described using text values within certain numerical analysis methods, it will be necessary to convert the variable's values into numbers. For example, a variable with low, medium, and high values may have low values replaced by 0, medium values replaced by 1, and high values replaced by 2. However, this conversion should be approached with care and with as much subject matter expertise as possible to assign the appropriate score to each value.

By converting continuous data into discrete values, it would appear that we are losing information. However, this conversion is desirable in a number of situations. Firstly, where a value is defined on an interval or ratio scale but when knowledge about how the data was collected suggests the accuracy of the data does not warrant these scales, a variable may be a candidate for discretization. This is often referred to as data smoothing. It may be more desirable to convert the data into more broadly defined categories that reflect the true variation in the data. Secondly, certain techniques can only process categorical data and hence converting continuous data into discrete values makes the variable accessible to these methods.

The most common way of looking at data is through a table, where the raw data is displayed in familiar rows of observations and columns of variables. It is essential for reviewing the raw data; however, the table can be overwhelming with more than a handful of observations or variables. Sorting the table based on one or more variables is useful for organizing the data. It is virtually impossible to identify any trends or relationships looking at the raw data alone.

Contingency tables (also referred to as two-way cross-classification tables) provide insight into the relationship between two variables. The variables must be categorical (dichotomous or discrete), or transformed to a categorical variable. A variable is often dichotomous; however, a contingency table can represent variables with more than two values.

Tables allow us to look at individual observations or summaries, whereas graphs present the data visually replacing numbers with graphical elements. Tables are important when the actual data values

are important to show. Graphs enable us to visually identify trends, ranges, frequency distributions, relationships, outliers and make comparisons. There are many ways of visualizing information in the form of a graph. This section will describe some of the common graphs used in exploratory data analysis and data mining: frequency polygrams, histograms, scatterplots, and box plots. In addition, looking at multiple graphs simultaneously and viewing common subsets can offer new insights into the whole data set.

Histograms present very similar information to frequency polygrams, that is, the frequency distribution of a particular variable. The length of the bar is proportional to the size of the group.

Histograms provide a clear way of viewing the frequency distribution for a single variable. The central values, the shape, the range of values as well as any outliers can be identified.

Scatterplots can be used to identify whether any relationship exists between two continuous variables based on the ratio or interval scales. The two variables are plotted on the x- and y-axes. Each point displayed on the scatterplot is a single observation. The position of the point is determined by the value of the two variables. The scatterplot in Figure 4.6 presents many thousands of observations on a single chart. Scatterplots allow you to see the type of relationship that may exist between the two variables.

Parameters are numbers that characterize a population, whereas statistics are numbers that summarize the data collected from a sample of the population. For example, a market researcher asks a portion or a sample of consumers of a particular product, about their preferences, and uses this information to make general statements about all consumers. The entire population, which is of interest, must be defined (i.e. all consumers of the product). Care must be taken in selecting the sample since it must be an unbiased, random sample from the entire population. Using this carefully selected sample, it is possible to make confident statements about the population in any exploratory data analysis or data mining project.

Finding hidden relationships: Grouping methods organize observations in different ways. Looking at the data from these different angles will allow us to find relationships that are not obvious from a summary alone. For example, a data set of retail transactions is grouped and these groups are used to find nontrivial associations.

Segmentation: Techniques for grouping data may lead to divisions that simplify the data for

analysis. Type of variables: Certain grouping methods will only accept categorical data, whereas others only accept continuous data. Other techniques handle all types of data. Understanding these limitations will allow you to select the appropriate method. Alternatively, you could decide to restrict the variables used in the method or perform a transformation on the data.

Data set size limit: There are methods that only work with data sets less than a certain size. Others work best with data sets over a certain size. Understanding the limitations placed on the number of observations and/or number of variables helps in the selection of particular methods. In situations where the data set is too large to process, one solution would be to segment the data prior to grouping.

Data Analysis

The arithmetic mean – colloquially referred to as the average – is calculated differently depending on the nature of the data. In empirical research, data most frequently appears in a raw data table that includes all the individual trait values.

To “balance” the scale, the ripe old turtle would have to be alone on the right side, while all the other animals are on the left side. We find that the mean value is a poor measure to describe the average age in this case because only one other animal is older than three. To reduce or eliminate the outlier effect, practitioners frequently resort to a trimmed mean. This technique “trims” the smallest and largest 5 % of values before calculating the mean, thus partly eliminating outliers. In our example, the 5 % trim covers both the youngest and oldest observation (the terrarium has 20 animals), thereby eliminating the turtle’s age from the calculation. This results in an average age of 2 years, a more realistic description of the age distribution. We should remember, however, that this technique eliminates 10 % of the observations, and this can cause problems, especially with small samples.

A measure seldom required in economics is the so-called harmonic mean. Because of the rarity of this measure, researchers tend to forget it, and instead use the arithmetic mean. However, sometimes the arithmetic mean produces false results. The harmonic mean is the appropriate method for averaging ratios consisting of numerators and denominators.

As the mean is sometimes not “representative” of a distribution, an alternative is required to identify

the central tendency.

In addition to the median, there are several other important measures of central tendency that are based on the quantization of an ordered dataset. These parameters are called quantiles. When quantiles are distributed over 100 equally sized intervals, they are referred to as percentiles. Their calculation requires an ordinal or cardinal scale and can be defined in a manner analogous to the median. In an ordered dataset, the p percentile is the value at which no less than p per cent of the observations are smaller or equal in value and no less than $(1-p)$ per cent of the observations are larger or equal in value.

In practical applications, one particular important group of quantiles is known as the quartiles. It is based on an ordered dataset divided into four equally sized parts. These are called the first quartile (the lower quartile or 25th percentile), the second quartile (the median or 50th percentile), and the third quartile (the upper quartile or 75th percentile).

If the boxplot is symmetrical – i.e. with the median in the centre of the box and whiskers of similar length – the distribution is symmetrical. When the value spread is large, the distribution is flat and lacks a clear-cut modal value. Such a distribution results, for instance, when plotting ages at a party with guests from various generations. If the value spread is small – i.e. with a compact box and whiskers – the distribution is narrow. This type of distribution results when plotting ages at a party with guests from a single generation. Boxplots can also express asymmetrical datasets. If the median is shifted to the left and the left whisker is short, then the middle 50% falls within a narrow range of relatively low values. The remaining 50 % of observations are mostly higher and distributed over a large range.

The boxplot provides an indication of the value spread around the median. The field of statistics has developed parameters to describe this spread, or dispersion, using a single measure. In the last section we encountered our first dispersion parameter: the interquartile range, i.e. the difference between the upper and lower quartile.

If the data are classed, the range results from the difference between the upper limit of the largest class of values and the lower limit of the smallest class of values. Yet we can immediately see why range is problematic for measuring dispersion. No other parameter relies so much on external distribution values for calculation, making range highly susceptible to outliers. If, for instance, 99 values are gathered close together and a single value appears as an outlier, the resulting range

predicts a high dispersion level. But this belies the fact that 99 % of the values lie very close together.

One alternative parameter is the median absolute deviation. Using the median as a measure of central tendency, this parameter is calculated by adding the absolute deviations of each observation and dividing the sum by the number of observations.

An accurate measure of dispersion must indicate average deviation from the mean. The first step is to calculate the deviation of every observation. Our intuition tells us to proceed as with the arithmetic mean – that is, by adding the values of the deviations and dividing them by the total number of deviations

Here, however, we must recall a basic notion about the mean. In an earlier section we likened the mean to a balance scale: the sum of deviations on the left side equals the sum of deviations on the right. Adding together the negative and positive deviations from the mean always yields a value of 0. To prevent the substitution of positive with negative values, we can add the absolute deviation amounts and divide these by the total number of observations

Yet statistics always make use of another approach: squaring both positive and negative deviations, thus making all values positive. The squared values are then added and divided by the total number of observations. The resulting dispersion parameter is called empirical variance, or population variance, and represents one of the most important dispersion parameters in empirical research

The boxplot not only provides information about central tendency and dispersion, but also describes the symmetry of the distribution. Recall for a moment that the student party produced a distribution that was right-skewed (peak on the left), and the retirement-home birthday party produced a distribution that was left-skewed (peak on the right). Skewness is a measure of distribution asymmetry. A simple parameter from Yule & Pearson uses the difference between median and mean in asymmetric distributions.

A common form of representing the association of two nominally scaled variables is the contingency table or crosstab. The bivariate contingency table takes the univariate frequency table one step further: it records the frequency of value pairs. Alongside absolute frequencies and the frequencies expressed relative to the total number of respondents we can also identify conditional relative frequencies.

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

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.

Irrelevant or partially relevant features can negatively impact model performance.

Feature selection and Data cleaning should be the first and most important step of your model designing.

Feature selection is the process of reducing the number of input variables when developing a predictive model.

It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.

Module 4 : Prediction

The layers act very much like the biological neurons: the outputs of one layer serve as the inputs for the next layer.

Among the layers, you can distinguish an input layer, hidden layers and an output layer. Multi-layer perceptrons are often fully connected. This means that there's a connection from each perceptron in a

certain layer to each perceptron in the next layer. Even though the connectedness is no requirement, this is typically the case.

Note that while the perceptron could only represent linear separations between classes, the multi-layer perceptron overcomes that limitation and can also represent more complex decision boundaries.

The first layer of a network reads the training data. So, we need to specify the size of images/training data that we are feeding the network. So, the `input_shape` parameter is used to specify the shape of input data.

This will help you build neural networks using Sequential Model. Let's come to the most important part. Once you have specified the architecture of the network, you need to specify the method for back-propagation by choosing an optimizer(like `rmsprop` or `adagrad`) and specify the loss(like `categorical_crossentropy`). We use `compile` function to do that in Keras. For example, in this line below we are asking the network to use the '`rmsprop`' optimizer to change weights in such a way that the loss '`binary_crossentropy`' is minimized at each iteration.

Once you are done with training using Keras, you can save your network weights in HDF5 binary data format. In order to use this, you must have the `h5py` package installed, which we did during installation. The HDF5 format is great to store huge amount of numerical data and manipulate this data from `numpy`. For example, it's easily possible to slice multi-terabyte datasets stored on disk as if they were real `numpy` arrays. You can also store multiple datasets in a single file, iterate over them or check out the `.shape` and `.dtype` attributes.

Sequential models are good for simpler networks and problems, but to build real-world complex networks you need to understand functional API. In most of the popular neural networks, we have a mini-network(a few layers arranged in a certain way like Inception module in GoogLeNet, fire module in squeezeNet) and this mini-network is repeated multiple times. Functional API allows you to call models like a function as we do for a layer. So, you can build a small model which can be repeated multiple times to build the complete network with even fewer lines of code. By calling a model on a tensor, you can reuse the model as well as weights.

The optimizer ensures to control and maintain the learning rate. A commonly used optimizer is the Adam optimizer. Again, just like Dense, it works in most cases, and it works well to adjust the learning rate throughout the training process.

The learning rate is the measure of how fast the correct weights are calculated for the model. Smaller the learning rate, the more accurate the weights will be. The downside here is that it might take more time to compute the weights.

When it comes to the loss function, MSE is a very widely used loss function. MSE stands for Mean Squared Error, and it is calculated by taking the average between predicted values and actual values and later squaring this result. If the loss function is closer to zero, it means that the model is working well.

Model training will use the `fit()` function and takes in five parameters for the process. The parameters include the training data, the target data, validation split, the number of epochs, and callbacks.

The validation split will simply split the data randomly as training and testing. Validation loss is seen during training as MSE on the validation set. If the validation split is set as 0.3, it means that 30 percent of the training data fed to the model will be kept aside to test the model performance later, and hence the model does not see this data at all.

The number of epochs denotes how many times the model will run through the data in an iteration. Until a certain point, more epochs will relate to model improvement directly, and further, it will not improve anymore. To check this and to stop the model, we make use of early stopping. It helps the model stop the training process if it reaches its culmination point before the number of epochs ends. `Patience = 3` means, it will check for improvements in 3 epochs. If there are no improvements for 3 epochs straight, the model will stop training.

Performing predictions on data is easily done by making use of the `predict()` function. With more nodes and layers, the capacity goes up. More capacity means more accuracy in learning to a certain limit. With this, presenting the model with more data will make the model large. Larger the model, more computation power is needed.

The model compilation is pretty straightforward as well. Categorical cross-entropy is used as the loss function as it works really well and is probably the most common choice to perform classification. The lower the score, the better the model performance

CHAPTER 4

RESULTS & DISCUSSIONS

Choosing the right metric is crucial while evaluating machine learning (ML) models. Various metrics are proposed to evaluate ML models in different applications, and I thought it may be helpful to provide a summary of popular metrics in a here, for better understanding of each metric and the applications they can be used for.

It is also worth mentioning that metric is different from loss function. Loss functions are functions that show a measure of the model performance and are used to train a machine learning model (using some kind of optimization), and are usually differentiable in model's parameters. On the other hand, metrics are used to monitor and measure the performance of a model (during training, and test), and do not need to be differentiable. However if for some tasks the performance metric is differentiable, it can be used both as a loss function (perhaps with some regularizations added to it), and a metric, such as MSE.

One of the key concept in classification performance is confusion matrix (AKA error matrix), which is a tabular visualization of the model predictions versus the ground-truth labels. Each row of confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class.

There are 4 important terms :

True Positives : The cases in which we predicted YES and the actual output was also

YES. True Negatives : The cases in which we predicted NO and the actual output was

NO. False Positives : The cases in which we predicted YES and the actual output was

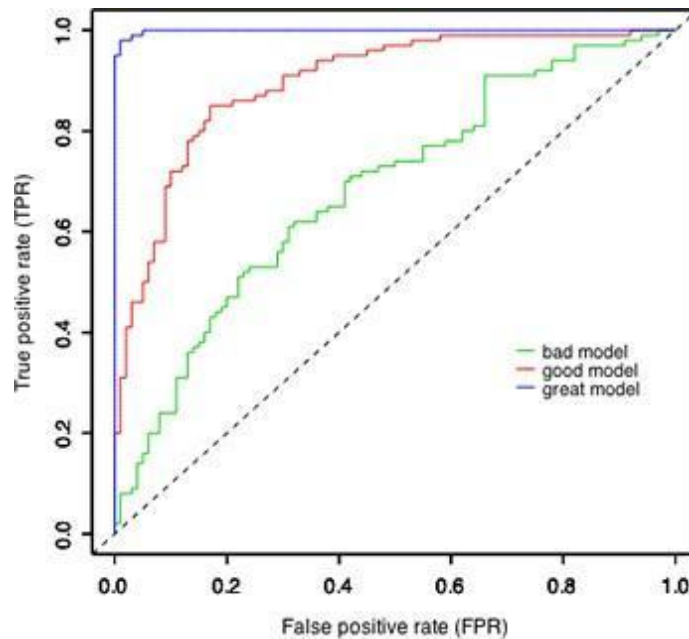
NO. False Negatives : The cases in which we predicted NO and the actual output was

YES.

	Actual = Yes	Actual = No
Predicted = Yes	TP	FP
Predicted = No	FN	TN

Classification accuracy is perhaps the simplest metrics one can imagine, and is defined as the number of correct predictions divided by the total number of predictions, multiplied by 100.

There are many cases in which classification accuracy is not a good indicator of your model performance. One of these scenarios is when your class distribution is imbalanced (one class is more frequent than others). In this case, even if you predict all samples as the most frequent class you would get a high accuracy rate, which does not make sense at all (because your model is not learning anything, and is just predicting everything as the top class). For example in our cat vs non-cat classification above, if the model predicts all samples as non-cat, it would result in a $1000/1100 = 90.9\%$.



Therefore we need to look at class specific performance metrics too. Precision is one of such metrics, which is defined as:

$$\text{Precision} = \frac{\text{True_Positive}}{\text{True_Positive} + \text{False_Positive}}$$

Recall is another important metric, which is defined as the fraction of samples from a class which are correctly predicted by the model. More formally:

$$\text{Recall} = \frac{\text{True_Positive}}{\text{True_Positive} + \text{False_Negative}}$$

Depending on application, you may want to give higher priority to recall or precision. But there are many applications in which both recall and precision are important. Therefore, it is natural to think of a way to combine these two into a single metric. One popular metric which combines precision and recall is called F1-score, which is the harmonic mean of precision and recall defined as:

$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

Sensitivity and specificity are two other popular metrics mostly used in medical and biology related fields, and are defined as:

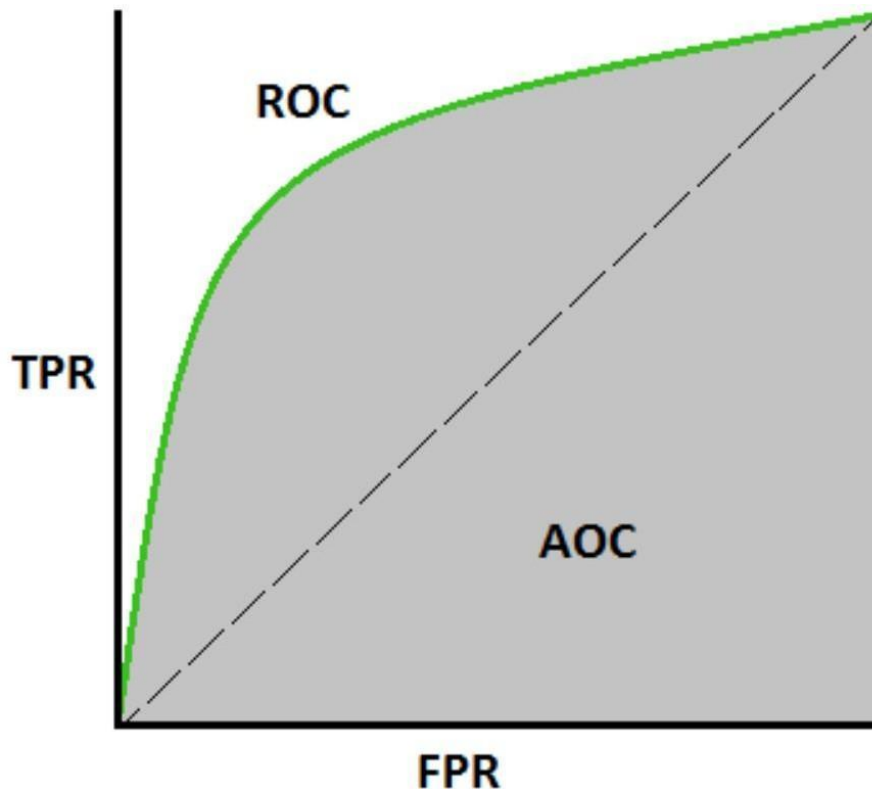
$$\text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

Specificity= True Negative Rate= $TN/(TN+FP)$

The receiver operating characteristic curve is plot which shows the performance of a binary classifier as function of its cut-off threshold. It essentially shows the true positive rate (TPR) against the false positive rate (FPR) for various threshold values.

The area under the curve (AUC), is an aggregated measure of performance of a binary classifier on all possible threshold values (and therefore it is threshold invariant).

AUC calculates the area under the ROC curve, and therefore it is between 0 and 1. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative



On high-level, the higher the AUC of a model the better it is. But sometimes threshold independent measure is not what you want, e.g. you may care about your model recall and require that to be higher than 99% (while it has a reasonable precision or FPR). In that case, you may want to tune your model threshold such that it meets your minimum requirement on those metrics (and you may not care even

if you model AUC is not too high).

Table 3

Measures for multi-class classification based on a generalization of the measures of Table 1 for many classes C_i : tp_i are true positive for C_i , and fp_i – false positive, fn_i – false negative, and tn_i – true negative counts respectively. μ and M indices represent micro- and macro-averaging.

Measure	Formula	Evaluation focus
Average Accuracy	$\frac{\sum_{i=1}^I \frac{tp_i + tn_i}{tp_i + fp_i + tn_i}}{I}$	The average per-class effectiveness of a classifier
Error Rate	$\frac{\sum_{i=1}^I \frac{fp_i + fn_i}{tp_i + fp_i + tn_i}}{I}$	The average per-class classification error
Precision $_{\mu}$	$\frac{\sum_{i=1}^I tp_i}{\sum_{i=1}^I (tp_i + fp_i)}$	Agreement of the data class labels with those of a classifiers if calculated from sums of per-text decisions
Recall $_{\mu}$	$\frac{\sum_{i=1}^I tp_i}{\sum_{i=1}^I (tp_i + fn_i)}$	Effectiveness of a classifier to identify class labels if calculated from sums of per-text decisions
Fscore $_{\mu}$	$\frac{(\beta^2 + 1) Precision_{\mu} Recall_{\mu}}{\beta^2 Precision_{\mu} + Recall_{\mu}}$	Relations between data's positive labels and those given by a classifier based on sums of per-text decisions
Precision $_M$	$\frac{\sum_{i=1}^I \frac{tp_i}{tp_i + fp_i}}{I}$	An average per-class agreement of the data class labels with those of a classifiers
Recall $_M$	$\frac{\sum_{i=1}^I \frac{tp_i}{tp_i + fn_i}}{I}$	An average per-class effectiveness of a classifier to identify class labels
Fscore $_M$	$\frac{(\beta^2 + 1) Precision_M Recall_M}{\beta^2 Precision_M + Recall_M}$	Relations between data's positive labels and those given by a classifier based on a per-class average

Logarithmic Loss or Log Loss, works by penalising the false classifications. It works well for multi-class classification. When working with Log Loss, the classifier must assign probability to each class for all the samples. Suppose, there are N samples belonging to M classes.

CHAPTER 5

Conclusion and Future Work

Reduce the flood attacks by employing rate limitation and probabilistically detect the number of attackers by developing the claim-carry-check. The learning automata algorithm is used to detect the approximate counting number of packets which are violating the rate limits. This works are implemented in a distributed manner. Also it can tolerate a little amount of attackers to collude. They easily reduce the throughput of burst traffic by comparing with the simple threshold. Hence, the main aim to enhance the throughput of burst traffic under the data flooding attack. This is achieved by using proposed scheme and it also gives guarantee the QOS compared to old scheme.

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

SAMPLE CODE

```
import numpy as
np import numpy
as np import
cython as cy
import pandas as
pd
from joblib import dump, load
from sklearn.metrics import accuracy_score, f1_score,
precision_score, recall_score from sklearn.linear_model import Perceptron
from sklearn.neural_network import
MLPClassifier from sklearn.linear_model import
LogisticRegression from sklearn.neural_network
import MLPClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import
AdaBoostClassifier import matplotlib.pyplot as
plt
from sklearn import svm, datasets
from sklearn.model_selection import
train_test_split from sklearn.metrics import
confusion_matrix
from sklearn.utils.multiclass import
unique_labels from sklearn.metrics import
roc_curve, auc
from sklearn.model_selection import
train_test_split from sklearn.preprocessing
```

```
import label_binarize from sklearn.multiclass
import OneVsRestClassifier from scipy import
interp
from itertools import
cycle import seaborn
as sns
from sklearn.datasets import
make_classification from sklearn.neighbors
import KNeighborsClassifier
```

```

from sklearn.ensemble import
RandomForestClassifier from
sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import
roc_auc_score import sklearn.metrics
as metrics
df = pd.read_csv('KDDTrain+.txt',
header=None) df.head()

df.shape
(125973,
43)
#Loading testing set into dataframe
qp = pd.read_csv('KDDTest+.txt',
header=None) qp.head()

df.columns = ['duration', 'protocol_type',
'service', 'flag', 'src_bytes', 'dst_bytes', 'land',
'wrong_fragment', 'urgent', 'hot',
'num_failed_logins', 'logged_in',
'num_compromised', 'root_shell',
'su_attempted', 'num_root',
'num_file_creations', 'num_shells',
'num_access_files', 'num_outbound_cmds',
'is_host_login', 'is_guest_login', 'count',
'srv_count', 'serror_rate',
'srv_serror_rate', 'error_rate', 'srv_error_rate', 'same_srv_rate',
'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
'dst_host_srv_count', 'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
'dst_host_same_src_port_rate',

```

'dst_host_srv_diff_host_rate',

'dst_host_serror_rate',

'dst_host_srv_serror_rate',

'dst_host_rerror_rate',

```

qp.columns = ['duration', 'protocol_type', 'service', 'flag',
'src_bytes', 'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
'num_access_files', 'num_outbound_cmds', 'is_host_login',
'is_guest_login', 'count', 'srv_count', 'serror_rate',
'srv_serror_rate', 'error_rate', 'srv_error_rate', 'same_srv_rate',
'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
'dst_host_srv_count', 'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
'dst_host_same_src_port_rate',
'dst_host_srv_diff_host_rate',
'dst_host_serror_rate', 'dst_host_srv_serror_rate',
'dst_host_error_rate', 'dst_host_srv_error_rate',
'subclass', 'difficulty_level'] qp.head()

```

```

'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
'dst_host_srv_count', 'dst_host_same_srv_rate',
'dst_host_diff_srv_rate',
'dst_host_same_src_port_rate',
'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
'dst_host_srv_serror_rate', 'dst_host_error_rate',
'dst_host_srv_error_rate', 'subclass',
'difficulty_level'], dtype='object')

```

```
testlst_names =
```

```
qp.columns
```

```
testlst_names
```

```

Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
'num_failed_logins', 'logged_in', 'num_compromised',

```

'root_shell', 'su_attempted', 'num_root', 'num_file_creations',
'num_shells', 'num_access_files', 'num_outbound_cmds',
'is_host_login',

```

    'is_guest_login', 'count', 'srv_count', 'error_rate',
    'srv_error_rate', 'error_rate', 'srv_error_rate',
    'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate',
    'dst_host_count', 'dst_host_srv_count',
    'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
    'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate',
    'dst_host_error_rate', 'dst_host_srv_error_rate',
    'dst_host_error_rate', 'dst_host_srv_error_rate', 'subclass',
    'difficulty_level'], dtype='object')

df =
df.drop('difficulty_level', 1)
df.shape
(125973, 42)

qp =
qp.drop('difficulty_level', 1)
qp.shape

classifier =
BaggingClassifier(base_esti
ma tor = PLA_clf)y_pred =
classifier.fit(X_train,
Y_train).predict(X_test)
plot_confusion_matrix(Y_test, y_pred, classes=class_names,title='Bagging with
PLA, without normalization')
plt.show()

classifier = AdaBoostClassifier(decision_clf)
y_pred = classifier.fit(X_train, Y_train).predict(X_test)
plot_confusion_matrix(Y_test, y_pred, classes=class_names,title='Adaboost ,
without normalization')
plt.show()

```

```
classifier = RandomForestClassifier()
```

```
y_pred = classifier.fit(X_train, Y_train).predict(X_test)
plot_confusion_matrix(Y_test, y_pred, classes=class_names, title='Random
Forest, without normalization')
plt.show()
```

SCREEN SHOTS

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	flag_REJ	flag_RSTO	flag_RS'
0	0	491	0	0		0	0	0	0	0	0 ...	0	0	
1	0	146	0	0		0	0	0	0	0	0 ...	0	0	
2	0	0	0	0		0	0	0	0	0	0 ...	0	0	
3	0	232	8153	0		0	0	0	0	1	0 ...	0	0	
4	0	199	420	0		0	0	0	0	1	0 ...	0	0	
...	...	...	...	...		...	...	...	...	...	...	...	...	...
125968	0	0	0	0		0	0	0	0	0	0 ...	0	0	
125969	8	105	145	0		0	0	0	0	0	0 ...	0	0	
125970	0	2231	384	0		0	0	0	0	1	0 ...	0	0	
125971	0	0	0	0		0	0	0	0	0	0 ...	0	0	
125972	0	151	0	0		0	0	0	0	1	0 ...	0	0	

125973 rows × 123 columns

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	flag_REJ	flag_RSTO	flag_RS'
0	0	0	0	0		0	0	0	0	0	0 ...	1	0	
1	0	0	0	0		0	0	0	0	0	0 ...	1	0	
2	2	12983	0	0		0	0	0	0	0	0 ...	0	0	
3	0	20	0	0		0	0	0	0	0	0 ...	0	0	
4	1	0	15	0		0	0	0	0	0	0 ...	0	1	
...	...	...	...	...		...	...	...	...	...	...	...	...	...
22539	0	794	333	0		0	0	0	0	1	0 ...	0	0	
22540	0	317	938	0		0	0	0	0	1	0 ...	0	0	
22541	0	54540	8314	0		0	0	2	0	1	1 ...	0	0	
22542	0	42	42	0		0	0	0	0	0	0 ...	0	0	
22543	0	0	0	0		0	0	0	0	0	0 ...	1	0	

22544 rows × 123 columns

Out[34]:

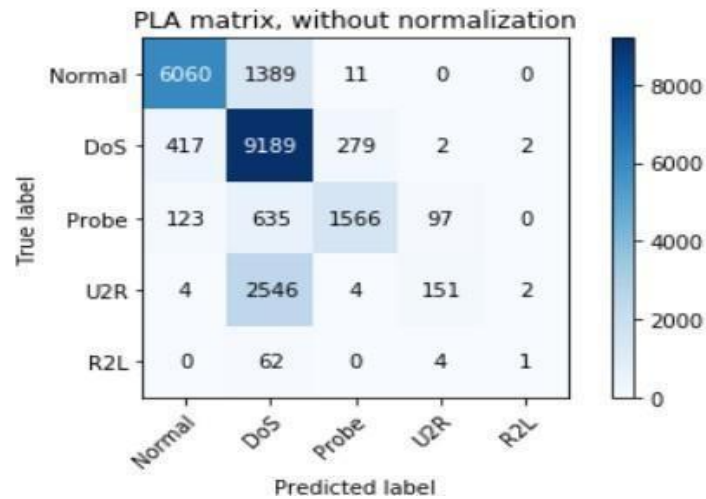
	0	1	2	3	4	5	6	7	8	9	...	33	34	35	36	37	38	39	40	41	42
0	0	tcp	private	REJ	0	0	0	0	0	0	...	0.04	0.06	0.00	0.00	0.0	0.0	1.00	1.00	neptune	21
1	0	tcp	private	REJ	0	0	0	0	0	0	...	0.00	0.06	0.00	0.00	0.0	0.0	1.00	1.00	neptune	21
2	2	tcp	ftp_data	SF	12983	0	0	0	0	0	...	0.61	0.04	0.61	0.02	0.0	0.0	0.00	0.00	normal	21
3	0	icmp	eco_i	SF	20	0	0	0	0	0	...	1.00	0.00	1.00	0.28	0.0	0.0	0.00	0.00	saint	15
4	1	tcp	telnet	RSTO	0	15	0	0	0	0	...	0.31	0.17	0.03	0.02	0.0	0.0	0.83	0.71	mscan	11

5 rows × 43 columns

		Mean Acc	Mean Precision	Mean Recall	Mean F1
0	PLA	0.752617	0.643781	0.491819	0.499653
1	Logistic Regression	0.749911	0.632140	0.487411	0.477869
2	NN	0.786906	0.742135	0.591325	0.622033
3	DTree	0.746584	0.573994	0.493553	0.497999
4	Voting	0.762908	0.646984	0.497790	0.494833
5	Bagging of PLA	0.746274	0.692901	0.501432	0.517405
6	AdaBoost	0.767033	0.689731	0.481173	0.500159
7	Random Forest	0.752040	0.884879	0.481155	0.492945

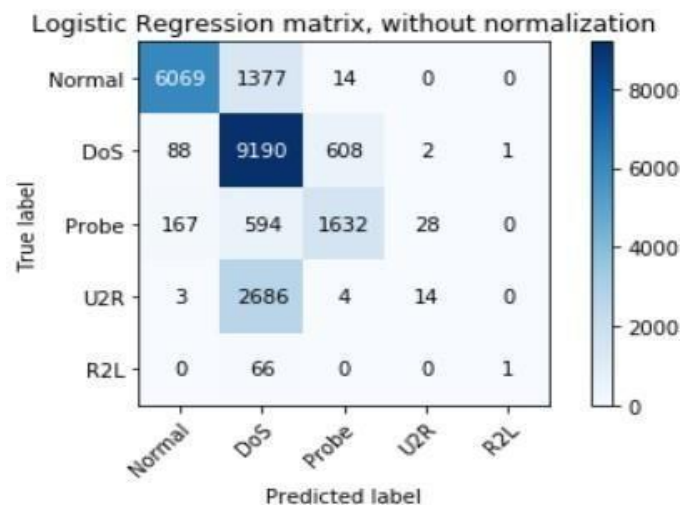
Confusion matrix, without normalization

```
[[6060 1389 11 0 0]
 [ 417 9189 279 2 2]
 [ 123 635 1566 97 0]
 [ 4 2546 4 151 2]
 [ 0 62 0 4 1]]
```



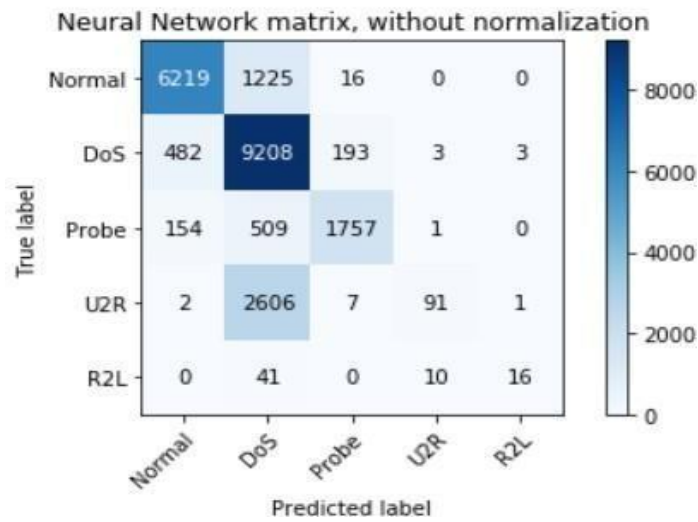
Confusion matrix, without normalization

```
[[6069 1377 14 0 0]
 [ 88 9190 608 2 1]
 [ 167 594 1632 28 0]
 [ 3 2686 4 14 0]
 [ 0 66 0 0 1]]
```



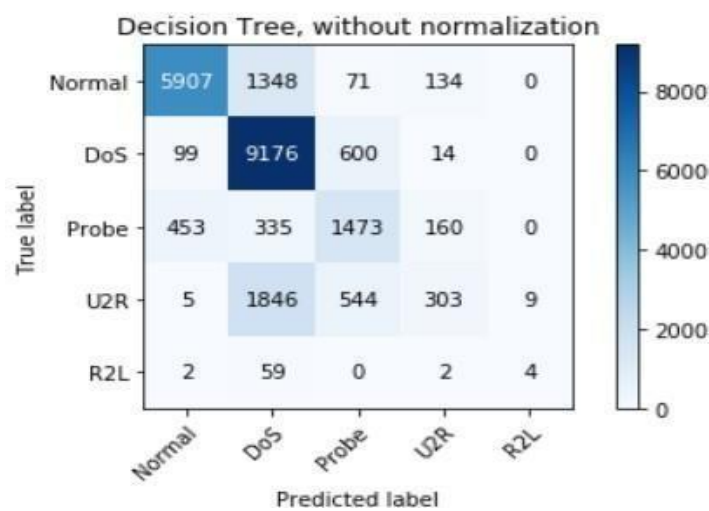
Confusion matrix, without normalization

```
[[ 6219 1225   16    0    0]
 [  482 9208  193    3    3]
 [  154  509 1757    1    0]
 [    2 2606    7   91    1]
 [    0   41    0   10   16]]
```



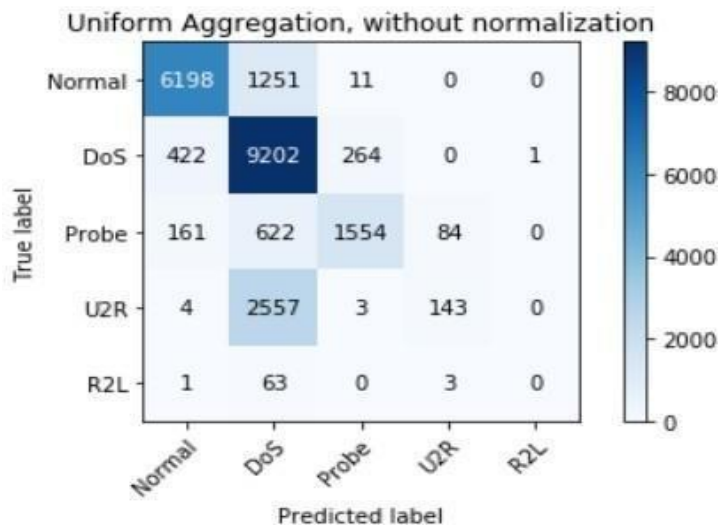
Confusion matrix, without normalization

```
[[ 5907 1348   71  134    0]
 [   99 9176  600   14    0]
 [  453  335 1473  160    0]
 [    5 1846  544  303    9]
 [    2   59    0    2    4]]
```



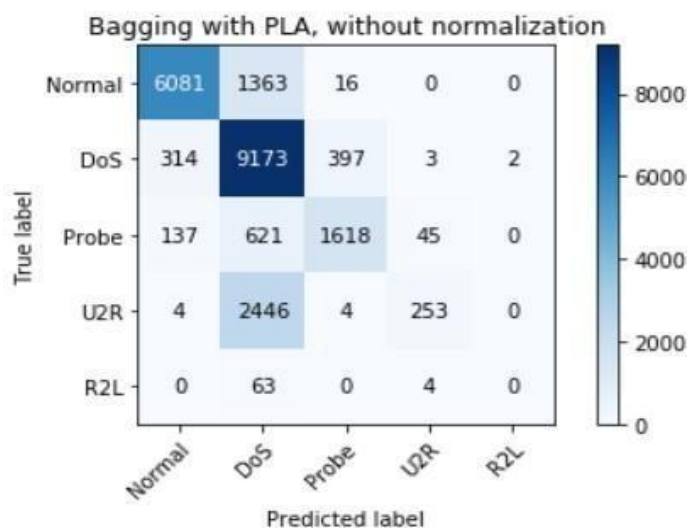
Confusion matrix, without normalization

```
[[6198 1251  11  0  0]
 [ 422 9202 264  0  1]
 [ 161  622 1554  84  0]
 [  4 2557  3 143  0]
 [  1  63  0  3  0]]
```



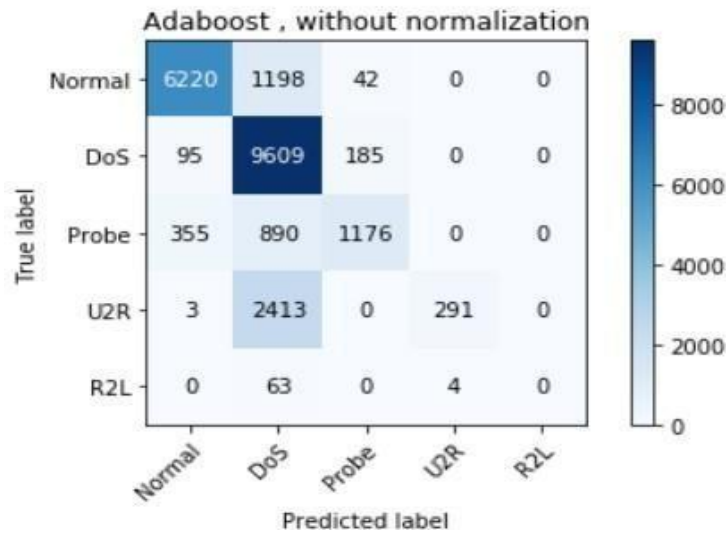
Confusion matrix, without normalization

```
[[6081 1363  16  0  0]
 [ 314 9173 397  3  2]
 [ 137  621 1618  45  0]
 [  4 2446  4 253  0]
 [  0  63  0  4  0]]
```



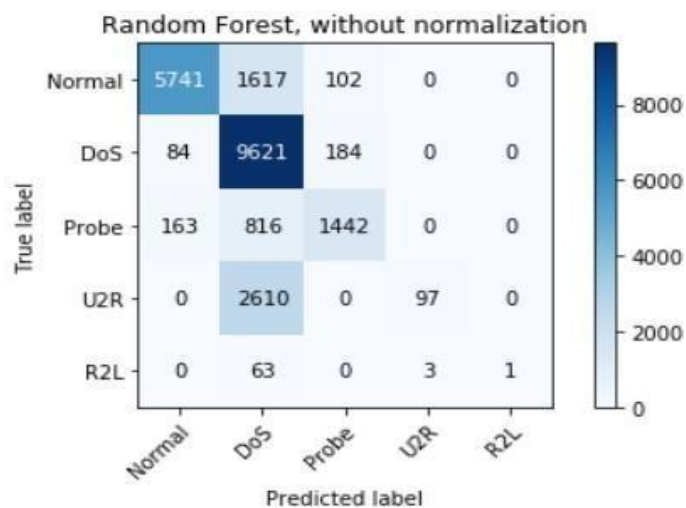
Confusion matrix, without normalization

```
[[6220 1198  42  0  0]
 [ 95 9609 185  0  0]
 [ 355 890 1176  0  0]
 [ 3 2413  0 291  0]
 [ 0 63  0 4  0]]
```



Confusion matrix, without normalization

```
[[5741 1617 102  0  0]
 [ 84 9621 184  0  0]
 [ 163 816 1442  0  0]
 [ 0 2610  0 97  0]
 [ 0 63  0 3  1]]
```



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 (IJET), 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