



A Data driven generalized approach for DDoS Detection using LSTM Algorithm

A PROJECT WORK PHASE II REPORT

Submitted by

RAMYAA M (113119UG03079)

MONIKA M (113119UG03059)

NIVETHA D (113119UG03064)

RENUKA R (113119UG03081)

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VEL TECH MULTI TECH Dr. RANGARAJAN Dr. SAKUNTHALA

ENGINEERING COLLEGE, ALAMATHI ROAD, AVADI, CHENNAI-62

ANNA UNIVERSITY, CHENNAI 600 025.

MAY 2023

ANNA UNIVERSITY, CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report of title “**Data driven generalized approach for DDoS Detection using LSTM algorithm** ”is the bonafide work of **RAMYAA M (113119UG03079), MONIKA M (113119UG03059), NIVETHA D (113119UG03064), RENUKA R(113119UG03081)** who carried out the project work under my supervision.

SIGNATURE

SIGNATURE

HEAD OF THE DEPARTMENT

SUPERVISOR

Dr.R.Saravanan,B.E,M.E(CSE).,Ph.D.
PROFESSOR,
Department of Computer Science and
Engineering,
Vel Tech Multi Tech Dr. Rangarajan Dr.
Sakunthala Engineering College, Avadi,
Chennai-600 062.

Ms.D.Vaitheeswari,M.E(CSE),
Asst.Professor,
Department of Computer Science and
Engineering,
Vel Tech Multi Tech Dr. Rangarajan
Dr.Sakunthala Engineering College,
Avadi, Chennai-600 062.

CERTIFICATE FOR EVALUATION

This is to certify that the project entitled “**Data driven generalized approach for DDoS Detection using LSTM algorithm**” is the bonafide record of work done by following students to carry out the project work under our guidance during the year 2022-2023 in partial fulfilment for the award of Bachelor of Engineering degree in Computer Science and Engineering conducted by Anna University, Chennai.

RAMYAA M	(113119UG03079)
MONIKA M	(113119UG03059)
NIVETHA D	(113119UG03064)
RENUKA R	(113119UG03081)

This project report was submitted for viva voce held on _____

At Vel Tech Multi Tech Dr. Rangarajan and Dr.Sakunthala Engineering College.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We wish to express our sincere thanks to Almighty and the people who extended their help during the course of our work.

We are greatly and profoundly thankful to our honourable Chairman, **Col. Prof.Vel. Shri Dr.R.Rangarajan B.E.(ELEC), B.E.(MECH), M.S.(AUTO)., D.Sc.,** & Vice Chairman, **Dr.Mrs.Sakunthala Rangarajan M.B.B.S.,** for facilitating us with this opportunity.

We also record our sincere thanks to our honorable Principal, **Dr.V.Rajamani M.E.,Ph.D.,** for his kind support to take up this project and complete it successfully.

We would like to express our special thanks to our Head of the Department, **Dr.R.Saravanan, B.E,M.E(CSE).,Ph.D.** Department of Computer Science and Engineering and our project supervisor **Ms.D.Vaitheeswari, M.E(CSE),** for their moral support by taking keen interest on our project work and guided us all along, till the completion of our project work and also by providing with all the necessary information required for developing a good system with successful completion of the same.

Further, the acknowledgement would be incomplete if we would not mention a word of thanks to our most beloved Parents for their continuous support and encouragement all the way through the course that has led us to pursue the degree and confidently complete the project work.

(RAMYAA M)

(MONIKA M)

(NIVETHA D)

(RENUKA R)

ABSTRACT

This project, titled “Data driven generalized approach for DDoS Detection using LSTM algorithm”. DDoS attacks also known as distributed denial of service (DDoS) attacks have emerged as one of the most serious and fastest-growing threats on the Internet. Denial-of-service (DDoS) attacks are an example of cyber-attacks that target a specific system or network in an attempt to render it inaccessible or unusable for a period of time. As a result improving the detection of diverse types of DDoS cyber threats with better algorithms and higher accuracy while keeping the computational cost under control has become the most significant component of detecting DDoS cyber threats. This attack originates from many sources scattered over multiple network locations. DoS attacks are primarily motivated by the desire to significantly degrade the performance or completely consume a certain resource and a process to exploit a machine defect and cause failure of a processing or exhausting the system resources by exploiting a system flaw. Yet another method of assaulting the target system is to flood the network and monopolize it so preventing anyone else from utilizing it.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF FIGURES	V
	LIST OF TABLES	VI
	LIST OF ABBREVIATIONS	VII
1.	INTRODUCTION	12
	1.1. OBJECTIVE	13
	1.2. SCOPE OF THE PROJECT	13
	1.3. LITERATURE SURVEY	13
2.	SYSTEM ANALYSIS	16
	2.1. EXISTING SYSTEM	16
	2.1.1.DISADVANTAGES	17
	2.2. PROPOSED SYSTEM	18
	2.2.1. ADVANTAGES	18
3.	SYSTEM SPECIFICATIONS	19
	3.2. SOFTWARE SPECIFICATIONS	20
	3.2.1 HARDWARE SPECIFICATIONS	20
4.	SOFTWARE DESCRIPTION	21
	4.1. FRONT END	22
	4.2. BACK END	23

5.	MODULE DESCRIPTION	26
	5.1. PROJECT DEFINITION	27
	5.2. OVERVIEW OF THE PROJECT	28
	5.3. METHODOLOGY	28
	5.4. ARCHITECTURE DIAGRAM	29
	5.5. MODULES	30
	5.5.1. DATA PRE-PROCESSING	30
	5.5.2. FEATURE EXTRACTION	30
	5.5.3. MODEL TRAINING	31
	5.5.4. EVALUATION	31
	5.5.5. LOCAL STORAGE	32
	5.6.INPUT DESIGN	33
	5.7.OUTPUT DESIGN	34
6.	SYSTEM TESTING	35
	6.1.CODE REVIEW	36
	6.2 .TESTING PROCESS	36
	6.2.1. UNIT TESTING	37
	6.2.2. BLACKBOX TESTING	38
	6.2.3. INTEGRATION TESTING	38
	6.2.4. SYSTEM TESTING	39
	6.2.5. SANITY TESTING	40
	6.2.6. REGRESSION TESTING	41
	6.2.7. PERFORMANCE TESTING	42

7.	SYSTEM IMPLEMENTATION	43
	7.1. IMPLEMENTATION PROCEDURE	44
	7.2. LONG SHORT TERM MEMORY ALGORITHM	44
	7.3.RANDOM FOREST ALGORITHM	45
8.	CONCLUSION AND FUTURE ENHANCEMENTS	46
	8.1. CONCLUSION	47
	8.2. FUTURE ENHANCEMENTS	47
	APPENDICES	48
	APPENDIX-1 SCREENSHOTS	49
	APPENDIX-2 IMPLEMENTATION CODE	52
	REFERENCES	64

LIST OF FIGURES

FIGURE NO.	NAME	PAGE NO.
A.1	ARCHITECTURE DIAGRAM	29
A.2	IMPORTING PACKAGES	30
A.3	DDoS ATTACKS TYPE COUNT	49
A.3	LABELING THE ATTACK TYPES	50
A.4	DDoS ATTACKS TYPE COUNT	50
A.5	DDoS ATTACK ACCURACY	51

List of Tables

TABLE NO.	NAME	PAGE NO.
A.5	DDoS ATTACK ACCURACY SS52	45

CHAPTER 1

INTRODUCTION

INTRODUCTION

Software-Defined Networking (SDN) is a new technology that facilitates management and programmability of the network system. SDN makes the network more reliable by centralizing it through separating the control plane from the data plane. However the emerging paradigm is subjected to many security vulnerabilities and new faults that can be used by attackers to create different types of malicious attacks. Further the common threats and attacks which can exploit the classical network infrastructure can also exist in the SDN environment. Moreover these attacks can impact the whole SDN system that includes multi-devices from different vendors unlike in the traditional network where in general an attack mainly crashes a part of the network devices from a single vendor only without affecting the entire network. There are many attack vectors that can exploit the SDN network. As the number and complexity of cybersecurity attacks increase at a tremendous pace on a daily basis defenders are in need to find more effective protection measures that rely on machine intelligence. To this account a recent trend in information security is the adoption of solutions based on Artificial Neural Networks (ANNs) to analyze network traffic and the behavior of software running on computers to identify possible compromised systems or unauthorized access attempts. Compared to traditional signature based and anomaly-based approaches ANNbased threat detection methods are more resilient to variations on attack patterns and are not constrained by the requirement to define thresholds for attack detection. However training and updating an ANN model for effective threat detection is a non-trivial task especially when dealing with zero-day vulnerabilities and attack vectors due to the complexity and variability of emerging attacks and the lack of data with relevant and upto-date attack profiles.

Reflection-based DDoS: Are those kinds of attacks in which the identity of the attacker remains hidden by utilizing legitimate third-party component. The packets are sent to reflector servers by attackers with source IP address set to target victim's IP address to overwhelm the victim with response packets

1.1 OBJECTIVE

To develop a productive and embodied architecture. To achieve our computations in less time with adequate prediction performance To use the second derivative to minimize model error. To ensure classification algorithms do not discriminate against protected groups. To select weights from the distribution and scale the weight-dependent parameters accordingly.

1.2.SCOPE OF THE PROJECT

The project will require the collection and analysis of large amounts of network traffic data, as well as the development of machine learning models and algorithms capable of identifying patterns and anomalies in the data. The user interface will need to be designed to provide actionable insights to security personnel, allowing them to quickly and efficiently respond to potential cyber-attacks. Overall, the Fronesis project has the potential to significantly improve the ability of organizations to detect and respond to cyber-attacks in a timely manner, reducing the risk of data breaches and other security incidents.

1.3.LITERATURE SURVEY

1.3.1 .A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning , 2020

LR-DDoS attacks are likely to remain a threat to our systems, particularly those that are centralized (e.g., cloud computing platforms). In this paper, we designed and implemented a modular and exible security architecture to detect and mit-igate LR-DDoS attacks in SDN environments. The modu- larity of the design allows one to easily replace any module without affecting the other modules of the architecture.

1.3.2 DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks, 2020

The Distributed Denial of Service (DDoS) attack has seriously impaired network availability for decades and still there is no effective defense mechanism against it. However, the emerging Software Defined Networking (SDN) provides a new way to reconsider the defense against DDoS attacks. In this paper, we propose two methods to detect the DDoS attack in SDN. One method adopts the degree of DDoS attack to identify the DDoS attack.

1.3.3. Hybrid DDoS Detection Framework Using Matching Pursuit Algorithm, 2020

Although a considerable amount of research has been done on DDoS attacks, it still poses a severe threat to many businesses and internet service providers. DDoS attacks commonly generate a high amount of network traffic.

1.3.4. AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification, 2021

Distributed Denial-of-Service (DDoS) attacks are increasing as the demand for Internet connectivity massively grows in recent years. Conventional shallow machine learning-based techniques for DDoS attack classification tend to be ineffective when the volume and features of network traffic, potentially carry malicious DDoS payloads, increase exponentially as they cannot extract high importance features automatically.

1.3.5. DDoS Attack Mitigation Based on Traffic Scheduling in Edge Computing-Enabled TWDM, 2021

Time-Wavelength Division Multiplexing Passive Optical Network (TWDM-PON) is considered as a promising solution of next generation PON (NG-PON). The integration of Edge Computing (EC) and TWDM-PON can satisfy the QoS requirements

CHAPTER 2

SYSTEM ANALYSIS

SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

Traditional attack detection approaches utilize predefined databases of known signatures about already-seen tools and malicious activities observed in past cyber-attacks to detect future attacks. More sophisticated approaches apply machine learning to detect abnormal behaviour. Nevertheless, a growing number of successful attacks and the increasing ingenuity of attackers prove that these approaches are insufficient. This paper introduces an approach for digital forensics-based early detection of ongoing cyber-attacks called Forensic. The approach combines ontological reasoning with the MITRE ATT&CK framework, the Cyber Kill Chain model, and the digital artifacts acquired continuously from the monitored computer system. Forensic examines the collected digital artifacts by applying rule-based reasoning on the Forensic cyber-attack detection ontology to identify traces of adversarial techniques. The Identified techniques are correlated to tactics, which are then mapped to corresponding phases of the Cyber Kill Chain model, resulting in the detection of an ongoing cyber-attack. Finally, the proposed approach is demonstrated through an email phishing attack scenario.

The proposed Forensic : a digital forensics-based cyber-attack detection approach based on the combined a Dimitris Forensics: Digital Forensics-Based Early Detection of Ongoing Cyber-Attacks utilization of the MITRE ATT&CK knowledge base, Lock- heed Martins Cyber Kill Chain (CKC) intelligence model, and digital artifacts acquired from the monitored system. Digital artifacts are acquired with proper sensors following digital forensics practices to ensure that the integrity of digital artifacts is preserved. Forensics examines the digital artifacts in order to recognize MITRE ATT&CK techniques, based on the traces left by the particular procedures of each technique. The recognized techniques are then associated with their MITRE ATT&CK tactics which are mapped to

corresponding CKC phases. (OWL) and the Semantic Web Rule Language (SWRL) making Forensic a ruled-based detection approach.

The ontology allows for digital artifacts to be represented in an interchangeable and computer processable format, while the rules reason over the facts about artifacts to detect an ongoing cyberattack. MITRE ATT&CK, CKC, OWL, SWRL, and associated rule-based reasoners are open- source specifications and technologies that would allow for broad adoption of Forensic. The proposed detection approach can be implemented as a standalone rule-based detection tool that considers the digital artifacts of the system where it is being operated in order to detect ongoing cyber-attacks.

Finally, the utilization of machine learning (ML) algorithms should be investigated. For instance, Forensic ontology can be used to define similarity measures for machine learning models that will be used in identifying similarities between digital artifacts. ML algorithm will also be investigated for automatic generation of rules that can be used in ontologybased reasoning to extend Forensic. **DISCLAIMER** Any mention of commercial products is for information only; it does not imply recommendation or endorsement by NIST. The authors declare that they have no financial interests.

2.1.1. Disadvantages

- Classification process consumes large amount of computational time.
- Computationally intensive and require relatively large memory space
- It is not an easy-to-use method
- High complexity of installing and maintaining
- Difficulties to obtain better performance

2.2. PROPOSED SYSTEM

We leverage and propose a deep learning approach based on LSTM for detection of DDoS attacks on the SDN. We combine LSTM-CNN with soft max model at the output layer to classify the network traffic into malicious or normal. The trained model behaviour is directly controlled by the values of the hyper-parameters where selecting the best values plays a key role in the success of neural network architecture. However selecting the best values of hyper parameters is still dependent on the best practice or human knowledge. We conducted various experiments to select the optimal values of experiment hyperparameters.

The soft max layer takes the decoder output and classifies the input data into normal or attack traffic. We used categorical- cross entropy as loss function with Adam optimizer and RLU function for activation in all different layers. We evaluate our model using the new released dataset CICDDoS which contain a comprehensive variety of DDoS attacks and addresses the gaps of the existing current datasets.

We benchmark several state-of-the-art ML models that are well known for detection of DDoS attacks and we evaluate our proposed model in terms of precision recall F-score and accuracy. Our proposed method has the best performance.

2.2.1. Advantages

- Ability to process large amount of data quickly in order to detect attacks.
- Better decision-making as to the training time.
- Efficiently improve time efficiency involves the computation time and communication time
- Eliminating the huge workload of traditional methods

CHAPTER 3

SYSTEM SPECIFICATIONS

SYSTEM SPECIFICATIONS

3.1.SOFTWARE SPECIFICATIONS

- Front End : HTML , CSS, BOOTSTRAP
- Back End : Python 3.6, Python Flask , VS Code ,
Anaconda, Jupyter Notebook

3.2. HARDWARE SPECIFICATIONS

- Operating System : Windows XP / Windows Vista
- Processor : 1.4 GHz 64-bit processor
- Disk Space : 100GB Free Space
- Memory : 8GB RAM Minimum
- Resolution : Super VGA (1024 x 768) or
higher -resolution

CHAPTER 4

SOFTWARE DESCRIPTION

SOFTWARE DESCRIPTION

4.1. FRONT END

4.1.1 HTML

The **HyperText Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as **Cascading Style Sheets (CSS)** and scripting languages such as **JavaScript**. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a webpage semantically and originally included cues for the appearance of the document. HTML is the skeleton, CSS is the skin, and JavaScript is the circulatory, digestive, and respiratory systems that brings the structure and the skin to life. You can also look at HTML, CSS, and JavaScript this way: HTML is the structure of a house, CSS is the interior and exterior decor, and JavaScript is the electricity, water system, and many other functional features that make the house livable.

An element consists of the opening tag, a character, the content, and a closing tag. Some elements are empty – that is, they don't have a closing tag but instead have a source or link to content that you want to embed on the web page. HTML elements are often used interchangeably with tags, but there's a small difference between the two. An element is a combination of the opening and closing tag, and then the content between them.

4.1.2 CSS

Cascading Style Sheets (CSS) is a stylesheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

4.2.BACK END

4.2.1.PYTHON 3.6

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

4.2.2 Python Flask

Python Flask is used to develop chatbot applications using python. Flask is mainly used to render and integrate the chatbot application in the browser by providing API. By running the python application, the suitable server domain link is obtained and run in the browser

4.2.3 VS Code

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including C#, Java, JavaScript, Go, Node.js, Python, C++, C, Rust and Fortran. It is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace

4.2.3. ANACONDA

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team

4.2.4. JUPYTER NOTEBOOK

JupyterNotebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyterweb application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats(HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries: IPython, ØMQ, Tornado (web server), jQuery, Bootstrap (front-end framework), MathJax

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

CHAPTER 5

MODULE DESCRIPTION

MODULE DESCRIPTION

.1. PROJECT DEFINITION

DDoS attacks also known as distributed denial of service (DDoS) attacks have emerged as one of the most serious and fastest-growing threats on the Internet. Denial-of-service (DDoS) attacks are an example of cyber-attacks that target a specific system or network in an attempt to render it inaccessible or unusable for a period of time. As a result improving the detection of diverse types of DDoS cyber threats with better algorithms and higher accuracy while keeping the computational cost under control has become the most significant component of detecting DDoS cyber threats. DDoS (distributed denial-of-service) attack originates from many sources scattered over multiple network locations. DoS attacks are primarily motivated by the desire to significantly degrade the performance or completely consume a certain resource and a process to exploit a machine defect and cause failure of a processing or exhausting the system resources by exploiting a system flaw. Yet another method of assaulting the target system is to flood the network and monopolies it so preventing anyone else from utilizing it. DDoS attack has a high impact on crashing the network resources making the target servers unable to support the valid users. The current methods deploy Machine Learning (ML) for intrusion detection against DDoS attacks in the SDN network using the standard datasets. However these methods suffer several drawbacks and the used datasets do not contain the most recent attack patterns - hence lacking in attack diversity.

.2. OVERVIEW OF THE PROJECT

we propose novel detection system against DDoS attacks in SDN environments. Our method is based on Deep Learning (DL) technique combining the LSTM. We evaluate our model using the newly released dataset CICDDoS which contains a comprehensive variety of DDoS attacks and addresses the gaps of the existing current datasets. We obtain a significant improvement in attack detection as compared to other benchmarking methods. Hence our model provides great confidence in securing these networks. Are those kinds of attacks in which the identity of the attacker remains hidden by utilizing legitimate third-party component. The packets are sent to reflector servers by attackers with the source IP address set to the target victim's IP address to overwhelm the victim with response packets. These attacks can also be carried out through application layer protocols using transport layer protocols i.e. TCP and UDP. TCP based exploitation attacks include SYN flood and UDP based attacks include UDP flood and UDP-Lag. UDP flood attack is initiated on the remote host by sending a large number of UDP packets.

.3. METHODOLOGY

- Data Pre-processing
- Feature Extraction
- Model Training
- Evaluation

4. ARCHITECTURE DIAGRAM

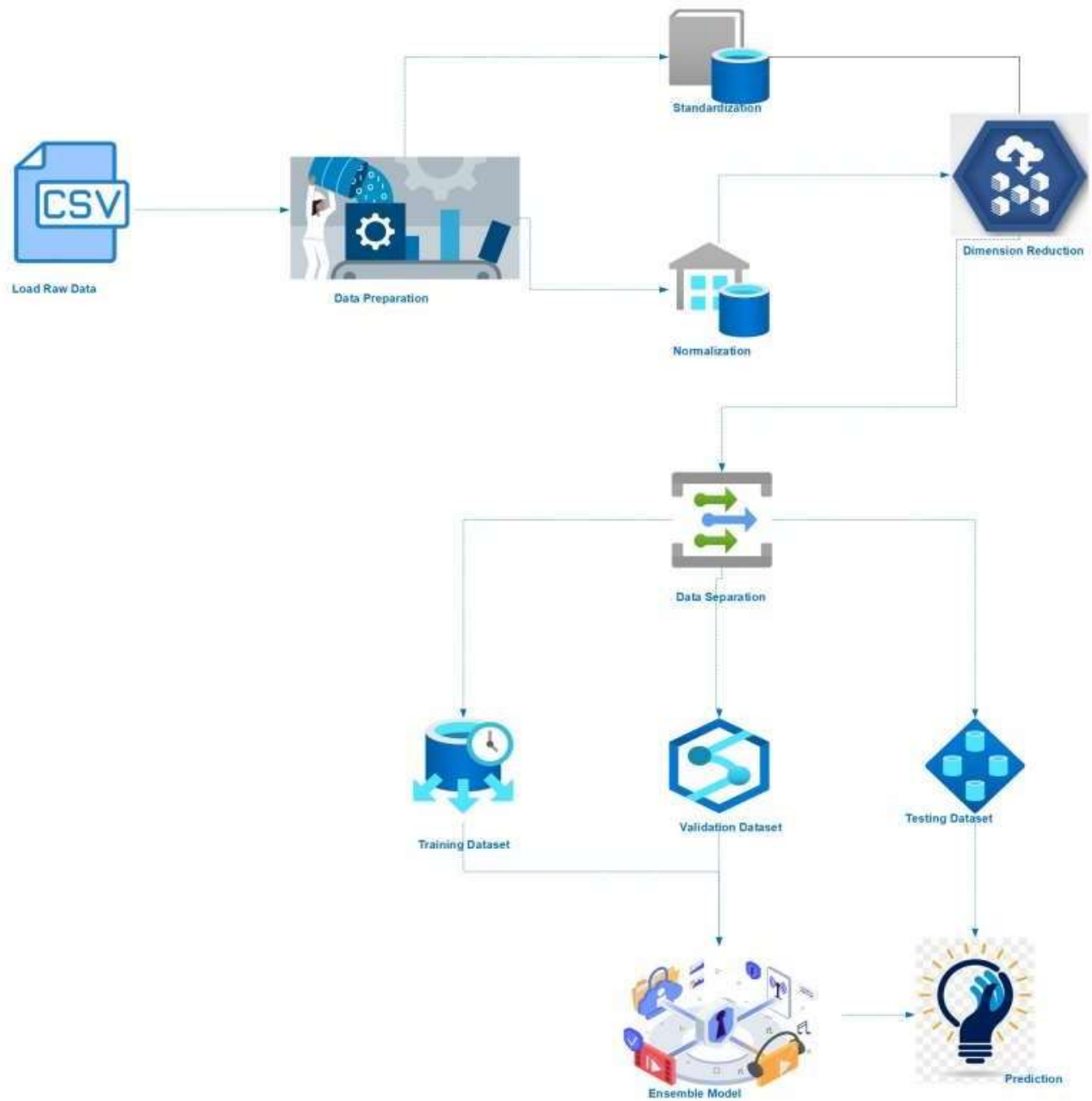


Fig. 5.1: Architecture diagram

5. MODULE

5.5.1 Data pre-processing

The operation steps for data pre-processing can be summarized as below:

- Replacing null and infinite values: The Dataset contains many infinite and null values. Therefore, we replaced null values with an average value and also infinite values with maximum values.
 - Removing remaining null values: This Dataset also contains non-numerical type null values, which cannot be replaced, and since there are enough records, as a consequence, these null values can be removed too.
 - Encoding categorical columns: Encoding nonnumerical values to numerical values is essential for preparing the Dataset for experimental operations.
- Removing columns with zero variance: When a columns variance or standard deviation is equal to zero, it means that all the values are the same, so that feature has no impact on the result.

Removing columns with low correlation: For dimension reduction, overfitting and making our model faster, it is necessary to remove negligible features.

5.5.2 Feature Extraction

Feature selection is used to reduce the number of variables and obtain a simpler model. It is an important pre-processing step in a number of machine learning applications. It employs a two-step process to extract numerical or categorical information (features) of the traffic observed, i.e., packet grouping and statistics computation. The former involves aggregating into flows packets generated between same pairs of applications, which can be achieved by monitoring origin, destination, and protocol fields.

5.5.3 Model Trainig

We split the dataset into three sections of training, validation, and testing subset. We build the model using the training set to adjust the weight on the neural network. The validation set is used to fine-tune the experiment parameters i.e.classifier architecture (not weights) like the number of hidden layers in the proposed model. Besides, the test set is used to estimate the model accuracy or performance. Inthis paper, we used the train ,test split technique for the model evaluation instead of the k-fold cross-validation technique. Although cross-validation is a widely used method in classification evaluation, we do not use it in our case. The inherent serial correlation of the time series data makes the use of cross-validation less suitable for these problems.The convolution layer (CONV) that manages the data from a receivercell. There are three hyperparameters to dimension the volume of the convolution layer: the depth, stride, and zero-padding. The pooling layer (POOL), which enables to reduce the size of the intermediate image by compressing the information and operates on each feature map independently. The correction layer (Rectified Linear Unit, RELU), which is often referred to as the RELU in reference to the activation function.

The hidden states from two units are concatenated so that future informationis accessible at the current timestamp t , which can potentially benefit the downstreamclassification task. Since our targets are sequences with similar malicious flows, it is the hidden states at the two ends of the structures that can acquire most information. The hidden representations in the middle of a sequence from two (Conv-)LSTM unitsyield certain amount of redundant information when the same architecture is used.

5.5.4 EVALUATION

We focus on the following important performance indicators: false alarm rate (FAR), precision, F-score, detection rate (DR), recall, True Negative Rate (TNR), False Accept Rate (FAR), ROC Curve, and accuracy. We use the Receiver Operating Characteristic (ROC) curve to evaluate how well the model performs accurately. The ROC curve indicates the relation between two parameters: True and False classes. The area underneath the ROC Curve (AUC) measures separability between false positive and true positive rate. We also use various metrics to evaluate our proposed model, such as precision, recall, precision, F-score and accuracy, in order to have a systematic benchmarking analysis with other related approaches.

5.5.5.Local Storage

The Local Storage is responsible for defining the detailed database , including tables , indexes , views , constraints , triggers , stored procedures, and other database-Specific constructs needed to store , retrieve , and delete persistent objects. A good database design is , therefore , one that: Divides your information it requires to join the information in the tables together as needed . helps support and ensures the accuracy and integrity of your information. The requirements for a bank management system provide a complete description of the system behaviour and are based on the business expectations. The functional requirements are sign in with login and password, change password , register new bank customers , view customer information , manage customer accounts. The phase of database design are analysis, conceptual design , logical and physical design, implementation , testing , operation.

.5. INPUT DESIGN

Input design is the process of converting the user–originated input to a computer based format. The design decision for handling input specify how data are accepted for computer processing. Input design is a part of overall system design that needs careful attention. The collection of input data is considered to be the most expensive part of the system design. Relevant information, extreme care is taken to obtain the pertinent information. If the data going into the system is incorrect then the processing and outputs will magnify these errors. The goal of designing input data is to make data entry as easy, logical and free from errors as possible.

The following are the objectives of input design:

- To produce a cost effective method of input.
- To make the input forms understandable to the end users.
- To ensure the validation of data input

The nature of input data is determined partially during logical system design. However the nature of inputs is made more explicit during the physical design. The impact of inputs on the system is also determined. Effort has been made to ensure that input data remains accurate from the stage at which it is recorded and documented to the stage at which it is accepted by the computer. Validation procedures are also present to detect errors in data input, which is beyond control procedures. Validation procedures are designed to check each record, data item or field against certain criteria.

.6. OUTPUT DESIGN

The output design phase of the system design is concerned with the conveyance of information to the end users in a user friendly manner. The output design should be efficient, intelligible so that the systems relationship with the end user is improved and thereby enhancing the process of decision making. The output design is an ongoing activity almost from the beginning of the project, and follows the principles of form design. Efficient and well-defined output design improves the relation of the system and the user, thus facilitating decision making.

The primary considerations in the design of the output are the requirement of the information and the objectives of the end users. The output design should be efficient, intelligible so that system relationship with the end user is improved and they're by enhancing the process of decision making.

Points to be noted while designing output screen:

- Design output to fit the user.
- Deliver the appropriate quantity of output.
- Assure that output is where it is needed.
- Provide output on time

CHAPTER 6

SYSTEM TESTING

SYSTEM TESTING

6.1. CODE REVIEW

After successful completion of coding, Code review was done with the objective of identifying and correcting deviations from standards, Identifying and fixing logical bugs and fall through and recording code walk through findings .The programs were checked and the code structure was made readable. The variable names were meaningful .It follows certain naming conventions, which makes the program readable.

- Variable names are prefixed with their scope and data type
- Checking out for the correct scope for various function
- All possible explanation for the code were given as comment
- Sufficient labels and comments were included in the code as the description of it for the benefit of the developer and other programmers who might examine it later.
- Checking out the connectivity of the database
- Code optimization was carried out

6.2. TESTING PROCESS

- Testing is the process of executing the program with the intent of finding an error
- A good test has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

The objective is to design tests that systematically uncovers different classes of error and do so with the minimum amount of time and effort .Testing cannot show the absence of defects, it can only show that the software defects are present.

6.2.1. UNIT TESTING .

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

A unit can be almost anything you want it to be -- a line of code, a method, or a class. Generally though, smaller is better. Smaller tests give you a much more granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast. Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development). Unit Testing is the method of verifying the smallest piece of testable code against its purpose. If the purpose or requirement failed then the unit test has failed.

6.2.2. BLACKBOX TESTING

During functional testing, testers verify the app features against the user specifications. This is completely different from testing done by developers which is unit testing. It checks whether the code works as expected. Because unit testing focuses on the internal structure of the code, it is called the white box testing. On the other hand, functional testing checks apps functionalities without looking at the internal structure of the code, hence it is called black box testing.

6.2.3. INTEGRATION TESTING

INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. Test drivers and test stubs are used to assist in Integration Testing. Integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing. Component integration testing Testing performed to expose defects in the interfaces and interaction between integrated components. System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet). Integration tests determine if independently developed units of software work correctly when they are connected to each other. The term has become blurred even by the diffuse standards of the software industry, so I've been wary of using it in my writing. In particular, many people assume integration tests are necessarily broad in scope, while they can be more effectively done with a narrower scope. As often with these things, it's best to start with a bit of history. When I first learned about integration testing, it was in the 1980's and the waterfall was the dominant

6.2.4. SYSTEM TESTING

SYSTEM TESTING is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the systems compliance with the specified requirements. System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not. System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product. System testing is a method of monitoring and assessing the behaviour of the complete and fully-integrated software product or system, on the basis of pre-decided specifications and functional requirements. It is a solution to the A black box testing type, system testing is the first testing technique that carries out the task of testing a software product as a whole. This System testing tests the integrated system and validates whether it meets the specified requirements of the client. System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be covering the end-to-end systems actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenario as per the customer point of view.

6.2.5. SANITY TESTING

Sanity Testing is done when as a QA we do not have sufficient time to run all the test cases, be it Functional Testing, UI, OS or Browser Testing. Sanity testing is a subset of regression testing. After receiving the software build, sanity testing is performed to ensure that the code changes introduced are working as expected. This testing is a checkpoint to determine if testing for the build can proceed or not. The main purpose of this testing is to determine that the changes or the proposed functionality are working as expected. If the sanity test fails, the build is rejected by the testing team to save time and money. It is performed only after the build has cleared the smoke test and been accepted by the Quality Assurance team for further testing. The focus of the team during this testing process is to validate the functionality of the application and not detailed testing.

Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as check. It is done at the build level. It helps not to waste the testing time to simply testing the whole application when the key features work or the key bugs have not been fixed yet. Here our focus will be on primary and core application work flow. To conduct smoke testing, we do not write test cases. We just pick the necessary test cases from already written test cases. As mentioned earlier, here in Smoke Testing, our main focus will be on core application work flow. So we pick the test cases from our test suite which cover major functionality of the application. In general, we pick minimal number of test cases that won't take more than half an hour to execute.

The main aim of Sanity testing to check the planned functionality is working as expected. Instead of doing whole regression testing the Sanity testing is performed

6.2.6. REGRESSION TESTING:

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change.

Regression Testing is a Software Testing type in which test cases are re-executed in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs.

This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix. For regression testing to be effective, it needs to be seen as one part of a comprehensive testing methodology that is cost-effective and efficient while still incorporating enough variety such as well- designed frontend UI automated tests alongside targeted unit testing, based on smart risk prioritization. prevent any aspects of your software applications from going unchecked. These days, many Agile work environments employing workflow practices such as XP (Extreme Programming), RUP (Rational Unified Process), or Scrum appreciate regression testing as an essential aspect of a dynamic, iterative development and deployment schedule. But no matter what software development and quality-assurance process your organization uses, if you take the time to put in enough careful planning up front, crafting a clear and diverse testing strategy with automated regression testing at its core, you can help prevent projects from going over budget, keep your team on track, and, most importantly, prevent unexpected bugs from damaging your products and your company's bottom line.

6.2.7. PERFORMANCE TESTING:

Performance testing gathers all the tests that verify an applications speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application. Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation. Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.

Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by system under varying workloads. Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

CHAPTER 7

SYSTEM IMPLEMENTATION

SYSTEM IMPLEMENTATION

7.1. IMPLEMENTATION PROCEDURE :

Implementation is the stage of the project when the theoretical design is turned out into a working system. Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user. The final stage is to document the entire system which provides components and the operating procedures of the system.

7.2. LONG SHORT TERM MEMORY(LSTM) ALGORITHM:

Long Short Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. LSTM was designed by Hochreiter & Schmidhuber. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying on the basis of time-series data.

ADVANTAGES OF LSTM

- Explicitly designed to deal with the long-term dependency problem
- Improved method of back propagating the error.
- Can be used when dealing with large sequences and accuracy is concerned

7.3 RANDOM FOREST ALGORITHM

"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final Output Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

ADVANTAGES OF RANDOM FOREST ALGORITHM

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION AND FUTURE ENHANCEMENTS

8.1. CONCLUSION

Network virtualization leads to new threats and new exploitable attacks that the ones already existing in the traditional network. The DDoS attack class is considered one of the most aggressive attack types in recent years causing a critical impact on the whole network system. The advent of ubiquitous network-based technologies has increased the associated vulnerabilities. The need for effective network protection tools have never been greater. In this paper we propose an AI-based IDS that is capable of distinguishing between regular and DDoS traffic. In this paper we proposed a new model that is based on DL for the detection of DDoS attacks against SDN network. We used the new released CICDDoS dataset for training and evaluation of our proposed model. The dataset contains comprehensive and most recent DDoS types of attacks. The evaluation of our model showed that gives the highest evaluation metrics in terms of recall precision F-score and accuracy compared to the existing well known classical ML & DL technique.

8.2. FUTURE ENHANCEMENTS

In the future we are interested in testing the performance of our proposed model on other datasets. In this current work we used a binary classification framework to classify the input traffic into normal and malicious types. However it is also necessary to classify each individual attack types separately. We intend to extend our work to a multi-class classification framework. Furthermore we will simulate the SDN network with various types of environments and attack traffics to create a heterogeneous dataset that can effectively represent the current internet traffic.

APPENDICES

APPENDIX - 1

SCREENSHOT

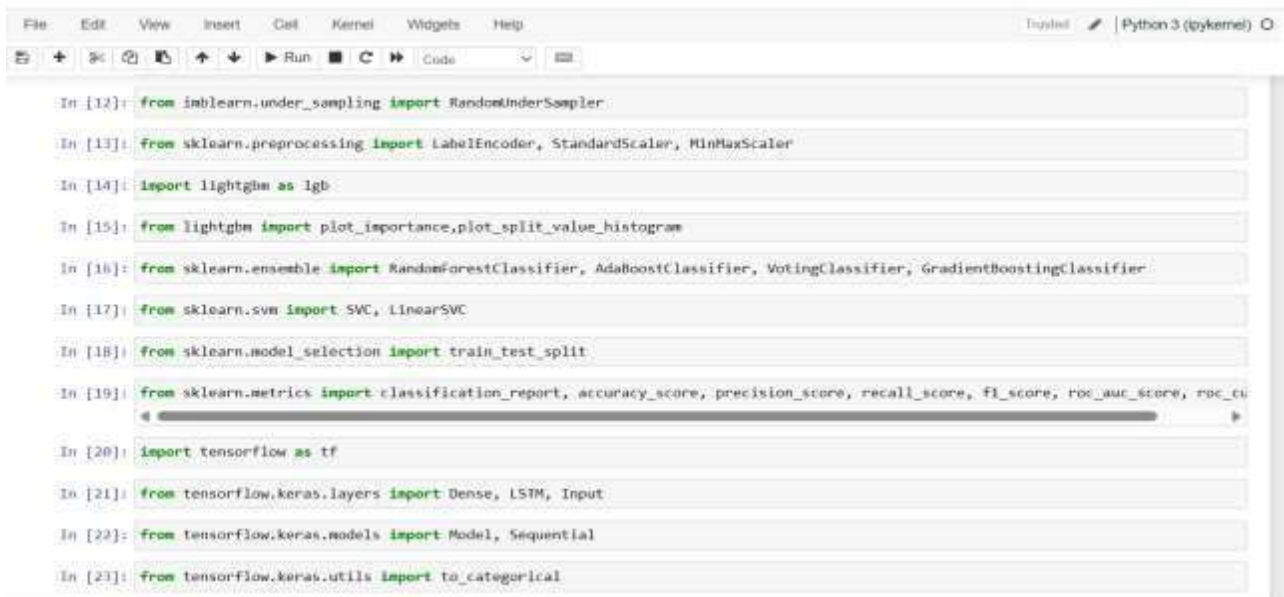


The screenshot shows a Jupyter Notebook interface with a toolbar at the top containing icons for undo, redo, run, and other actions. Below the toolbar is a code editor with the following Python code:

```
In [1]: from IPython.display import display, HTML
display(HTML("""
<style>
.messagebox{
border-radius: 2px;
padding: 1.25em 1.5em;
border: 1px solid;
}
.messagelightgreen{
border-color: hsl(164deg 95% 38%);
color: rgb(5 139 182);
background-color: rgb(236 255 250);
}
.messagelightgreen b{
color:rgb(139 77 5);
}
.messagebrownx
border-color: hsl(35deg 96% 62%);
color: rgb(143 84 4);
background-color: rgb(255 245 234);
}
.messagebrown b{
color: rgb(5 139 182);
}
</style>"""))
```

Below the code editor, there is a status bar that reads "In [1]: Import as a cell with full interactive".

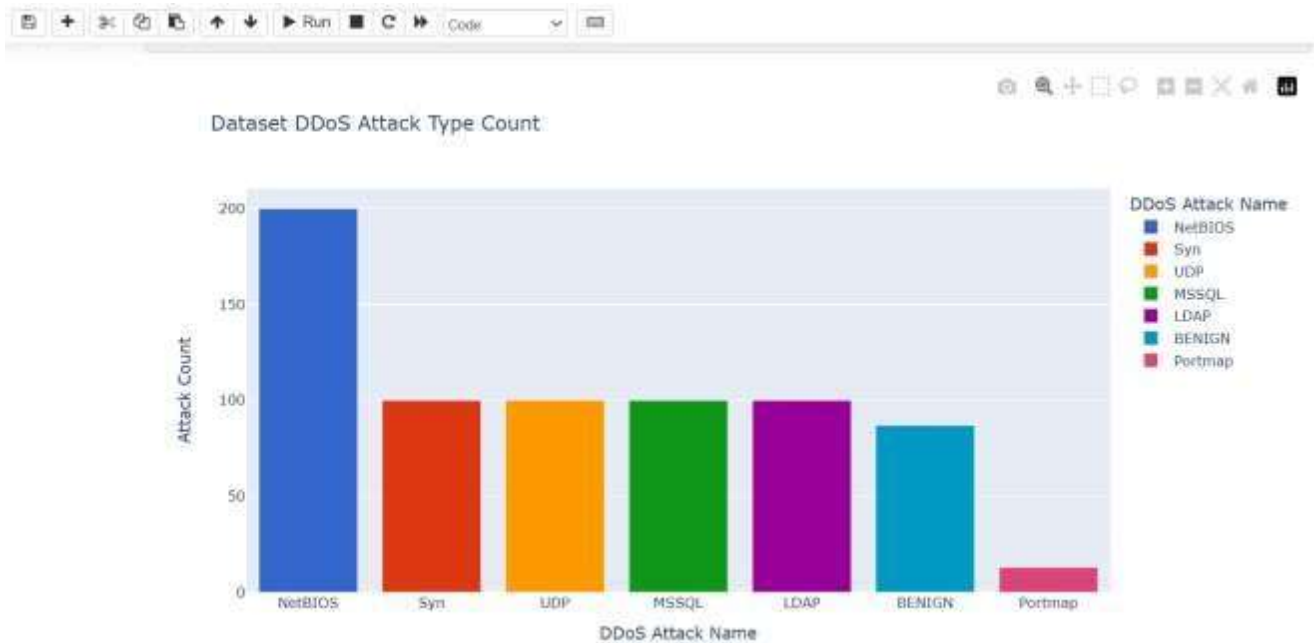
A.1 HTML DISPLAY CODE



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The code editor contains the following import statements:

```
In [12]: from imblearn.under_sampling import RandomUnderSampler
In [13]: from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler
In [14]: import lightgbm as lgb
In [15]: from lightgbm import plot_importance, plot_split_value_histogram
In [16]: from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, VotingClassifier, GradientBoostingClassifier
In [17]: from sklearn.svm import SVC, LinearSVC
In [18]: from sklearn.model_selection import train_test_split
In [19]: from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve
In [20]: import tensorflow as tf
In [21]: from tensorflow.keras.layers import Dense, LSTM, Input
In [22]: from tensorflow.keras.models import Model, Sequential
In [23]: from tensorflow.keras.utils import to_categorical
```

A.2 IMPORTING PACKAGES



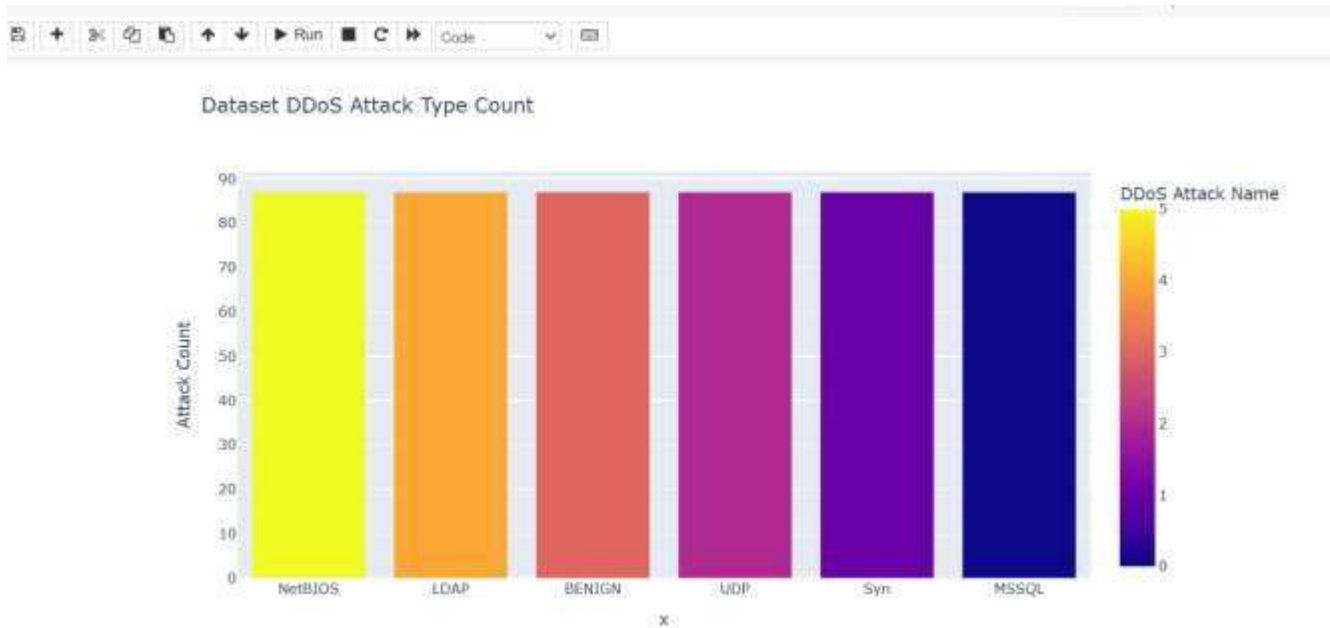
A.3 DDoS ATTACK TYPE COUNT

```

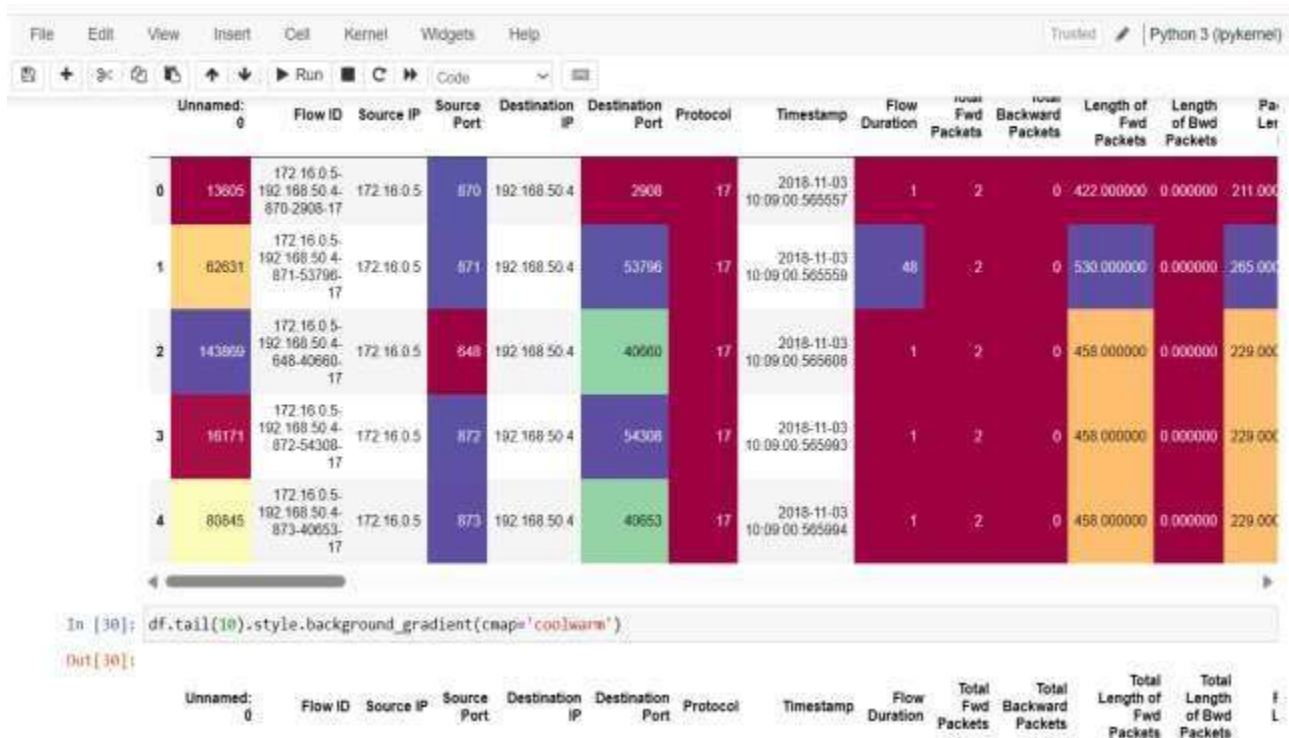
In [56]: attack_types = df["Label"].unique()
In [57]: attack_types
Out[57]: array(['NetBIOS', 'LDAP', 'BENIGN', 'UDP', 'Syn', 'MSSQL'], dtype=object)
In [58]: object_cols = object_cols.tolist()
In [59]: object_cols.remove("Label")
In [60]: label_encoder = LabelEncoder()
In [61]: df['Label'] = label_encoder.fit_transform(df['Label'])
In [62]: df['Label'].unique()
Out[62]: array([3, 1, 0, 5, 4, 2])
In [63]: df.drop(object_cols, axis=1, inplace=True)
In [64]: df.columns
Out[64]: Index(['Unnamed: 0', 'Source Port', 'Destination Port', 'Protocol',

```

A.4 LABELING THE ATTACKS



A.5 DDoS ATTACKS COUNT



A.6 DDoS ATTACK ACCURACY

APPENDIX – 2

IMPLEMENTATION CODE

Index.html

```
from IPython.display import display, HTML
display(HTML("""
<style>
.messagebox{
    border-radius: 2px;
    padding: 1.25em 1.5em;
    border: 1px solid;
}
.messagelightgreen{
    border-color: hsl(164deg 95% 38%);
    color: rgb(5 139 102);
    background-color: rgb(236 255 250);
}
.messagelightgreen b{
    color:rgb(139 77 5);
}
.messagebrownx
    border-color: hsl(35deg 96% 62%);
    color: rgb(143 84 4);
    background-color: rgb(255 245 234);
}
.messagebrown b{
    color: rgb(5 139 102);
```

```

}
</style>"""))
import os, re, time, math, tqdm, itertools
import numpy as np
import pandas as pd
import glob
from pathlib import Path
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
plt.rcParams['font.size'] = 14
plt.rcParams['axes.labelsize'] = 12
plt.rcParams['axes.titlesize'] = 12
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12
plt.rcParams['legend.fontsize'] = 12
plt.rcParams['figure.titlesize'] = 14
plt.rcParams['figure.figsize'] = (20,8)
plt.rcParams["ps.useafm"] = True
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import seaborn as sns
from imblearn.under_sampling import RandomUnderSampler
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler
import lightgbm as lgb
from lightgbm import plot_importance, plot_split_value_histogram
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
VotingClassifier, GradientBoostingClassifier

```

```

from sklearn.svm import SVC, LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score,
precision_score, recall_score, f1_score, roc_auc_score, roc_curve
import tensorflow as tf
from tensorflow.keras.layers import Dense, LSTM, Input
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import plot_model
file_list = glob.glob("Dataset/*.csv")
df_list = []
for file_name in file_list:
    df = pd.read_csv(file_name, nrows=100)
    df_list.append(df)
df = pd.concat(df_list)
df.head().style.background_gradient(cmap='Spectral')
df.tail(10).style.background_gradient(cmap='coolwarm')
df.shape
display(HTML("<h6 class='messagebox messagelightgreen'>No of Rows
Available in Dataset <b>{0}</b></h6>".format(df.shape[0])))
display(HTML("<h6 class='messagebox messagelightgreen'>No of Columns
Available in Dataset <b>{0}</b></h6>".format(df.shape[1])))
df.memory_usage(deep=True)
df.memory_usage(deep=True).sum()
df.columns
df.columns = [col.strip() for col in df.columns]
df.columns
pd.options.mode.use_inf_as_na = True
df.isna()

```

```

df.isna().sum()
df.duplicated().sum()
df.describe()
label_vc = df["Label"].value_counts()
px.bar(label_vc, x=label_vc.index, y = label_vc.values,
color=label_vc.index, title="Dataset DDoS Attack Type Count",
labels={'y': "Attack Count", 'index': "DDoS Attack Name"},
color_discrete_sequence=px.colors.qualitative.G10)
df.replace(to_replace=["Portmap"], value="UDP", inplace=True)
label_vc = df["Label"].value_counts()
px.bar(label_vc, x=label_vc.index, y = label_vc.values,
color=label_vc.index, title="Dataset DDoS Attack Type Count",
labels={'y': "Attack Count", 'index': "DDoS Attack Name"},
color_discrete_sequence=px.colors.qualitative.G10)
numeric_df = df.select_dtypes(include=['int64', 'float64'])
object_df = df.select_dtypes(include=['object'])
numeric_cols = numeric_df.columns
object_cols = object_df.columns
print('Numeric Columns: ')
print(numeric_cols, '\n')
print('Object Columns: ')
print(object_cols, '\n')
print('Number of Numeric Features: ', len(numeric_cols))
print('Number of Object Features: ', len(object_cols))
display(HTML("<h6 class='messagebox messagebrown'>Number of Numeric  
Columns Available in Dataset <b>{0}</b></h6>".format(len(numeric_cols))))
display(HTML("<h6 class='messagebox messagebrown'>Number of Object  
Columns Available in Dataset <b>{0}</b></h6>".format(len(object_cols))))

```

```

object_df.head()
df["SimillarHTTP"].value_counts()
plt.figure(figsize=(19, 17), dpi=80)
plt.barh(list(dict(df["Source IP"].value_counts()).keys()), dict(df["Source IP"].value_counts()).values())
for idx, val in enumerate(dict(df["Source IP"].value_counts()).values()):
plt.text(x = val, y = idx-0.2, s = str(val), color='r', size = 13)
plt.xlabel('Number of Requests')
plt.ylabel('IP addres of sender')
plt.title('Number of all requests')
plt.figure(figsize=(29, 27), dpi=80)
plt.barh(list(dict(df["Destination IP"].value_counts()).keys()), dict(df["Destination IP"].value_counts()).values())
for idx, val in enumerate(dict(df["Destination IP"].value_counts()).values()):
plt.text(x = val, y = idx-0.2, s = str(val), color='r', size = 13)
plt.xlabel('Number of Requests From Destination IP')
plt.ylabel('IP addres of Destination')
plt.title('Number of all requests')
plt.figure(figsize=(12, 17), dpi=80)
plt.barh(list(dict(df["Source IP"].value_counts()).keys()), dict(df["Source IP"].value_counts()).values(), color='lawngreen')
plt.barh(list(dict(df[df.Label != "BENIGN"]["Source IP"].value_counts()).keys()), dict(df[df.Label != "BENIGN"]["Source IP"].value_counts()).values(), color='blue')
for idx, val in enumerate(dict(df["Source IP"].value_counts()).values()):
plt.text(x = val, y = idx-0.2, s = str(val), color='r', size = 13)
for idx, val in enumerate(dict(df[df.Label == "BENIGN"]["Source IP"].value_counts()).values()):
plt.text(x = val, y = idx-0.2, s = str(val), color='w', size = 13)

```



```

plt.xlabel('Number of Requests')
plt.ylabel('IP address of sender')
plt.title('Number of requests from different IP address')
attack_types = df["Label"].unique()
attack_types
object_cols = object_cols.tolist()
object_cols.remove("Label")
label_encoder = LabelEncoder()
df['Label']= label_encoder.fit_transform(df['Label'])
df['Label'].unique()
df.drop(object_cols, axis=1, inplace=True)
df.columns
df.drop(["Unnamed: 0"], axis=1, inplace=True)
df.drop(["Unnamed: 0"], axis=1, inplace=True)
len(column_list)
X = df[column_list]
y = df[["Label"]]
rus = RandomUnderSampler(random_state=0)
X, y = rus.fit_resample(X, y)
type(X)
type(y)
df = X
df["Label"] = y
df.head()
label_vc = df["Label"].value_counts()
px.bar(label_vc, x=attack_types, y = label_vc.values,

```

```

color=label_vc.index, title="Dataset DDoS Attack Type Count",labels={'y': "Attack
Count", 'index': "DDoS Attack Name"},
color_discrete_sequence=px.colors.qualitative.G10)
px.bar(label_vc, x=attack_types, y = label_vc.values,
color=label_vc.index, title="Dataset DDoS Attack Type Count",
labels={'y': "Attack Count", 'index': "DDoS Attack Name"},
color_discrete_sequence=px.colors.qualitative.G10)
display(HTML("<h6 class='messagebox messagebrown'>Number of Numeric
Columns Available in Dataset <b>{0}</b></h6>".format(len(numeric_cols))))
display(HTML("<h6 class='messagebox messagebrown'>Number of Object Columns
Available in Dataset <b>{0}</b></h6>".format(len(object_cols))))
params = {
    'boosting_type': 'gbdt',
    'objective': 'multiclass',
    'num_class': 6,
    'learning_rate': 0.01,
    'min_gain_to_split': 0.2,
    'verbose': 1,
    'num_threads':4,
}
dtrain = lgb.Dataset(
    X, label=y
)
bst = lgb.train(
    params, dtrain, num_boost_round=10
)
def plot_features(booster, figsize):
    fig, ax = plt.subplots(1,1,figsize=figsize)
    return plot_importance(booster=booster, ax=ax,max_num_features=20)

```

```

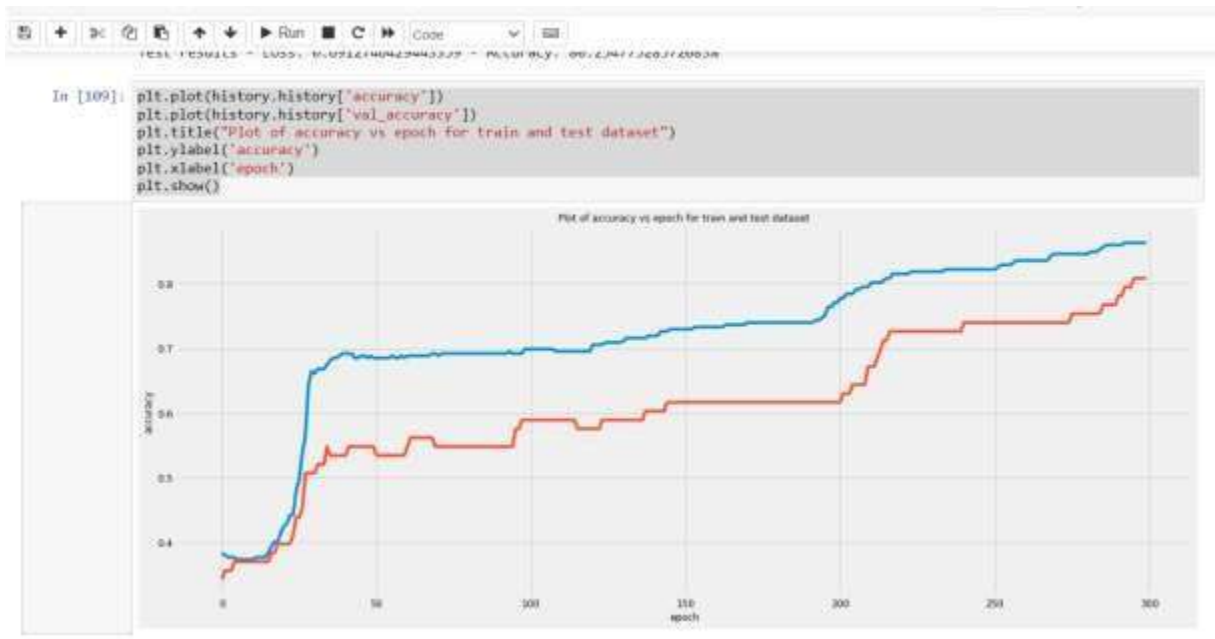
plot_features(bst, (15,15))
df.columns
features = ["Source Port", "Fwd Packet Length Min", "Init_Win_bytes_forward",
"ACK Flag Count", "Protocol", "Destination Port", "Fwd IAT Total", "Active Std"]
X = df[features]
X.head()
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
pd.DataFrame(X, columns=features)
y = df["Label"]
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.70,
random_state=2, stratify=y)
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
y_pred=rfc.predict(X_test)
rfc_ac=accuracy_score(y_test, y_pred)*100
display(HTML("<h6 class='messagebox messagelightgreen'>Random Forest
Accuracy Score <b>{0}</b></h6>".format(rfc_ac)))
print(classification_report(y_test, y_pred,target_names=attack_types))
X_train.shape, X_test.shape
len(attack_types)
def create_lstm_model():
model = Sequential()
model.add(Dense(units=50, activation='relu'))
model.add(Dense(6,activation='softmax'))
return model
model = create_lstm_model()
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',

```

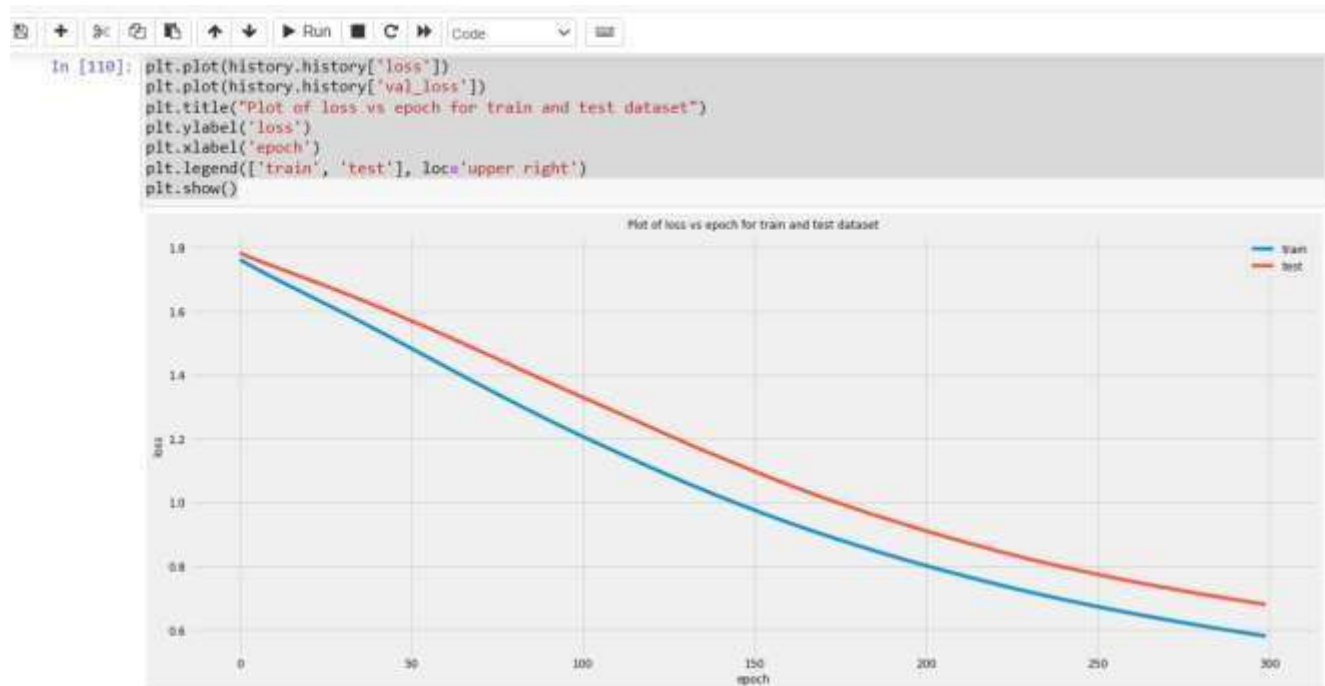
```

metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=300,
batch_size=5000,validation_split=0.2)
test_results = model.evaluate(X_test, y_test, verbose=1)
print(f'Test results - Loss: {test_results[0]} - Accuracy: {test_results[1]*100}%')
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title("Plot of accuracy vs epoch for train and test dataset")
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("Plot of loss vs epoch for train and test dataset")
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()
y_pred = model.predict(X_test)
y_test
y_pred_cm = np.argmax(y_pred, axis=1)
print(classification_report(y_test, y_pred_cm,target_names=attack_types))

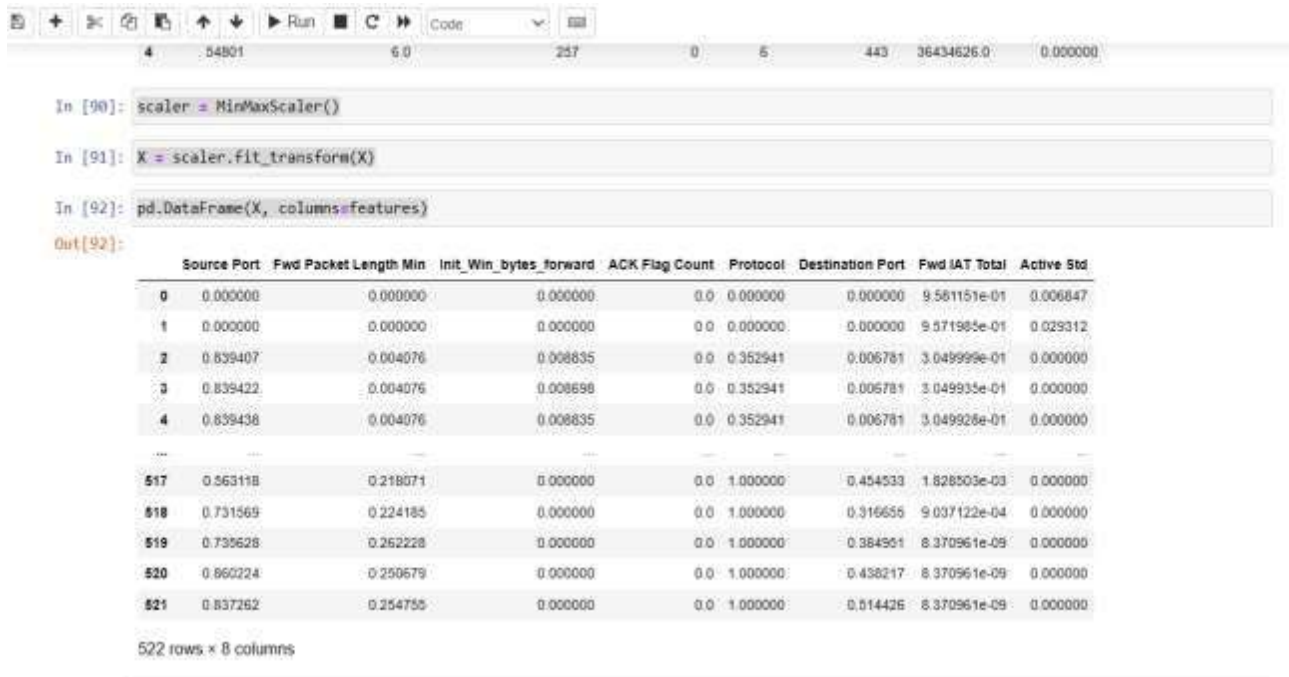
```



A.7 ACCURACY VS EPOCH



A.8 LOSS VS EPOCH



A.9 DATA



A.10 BAR GRAPH

```

241 2
373 4
142 1
Name: Label, Length: 157, dtype: int32

In [113]: y_pred_cm = np.argmax(y_pred, axis=1)

In [114]: print(classification_report(y_test, y_pred_cm, target_names=attack_types))

```

	precision	recall	f1-score	support
NetBIOS	0.80	0.59	0.68	27
LDAP	1.00	0.96	0.98	26
BENIGN	1.00	0.50	0.67	26
UDP	0.59	1.00	0.74	26
Syn	0.80	0.92	0.86	26
MSSQL	0.88	0.85	0.86	26
accuracy			0.80	157
macro avg	0.85	0.80	0.80	157
weighted avg	0.84	0.80	0.80	157

A.11 OUTPUT



A.12 ATTACK TYPE

REFERENCES

- [1] A. Kuzmanovic and E. W. Knightly, Low-rate TCP-targeted denial of service attacks: The shrew vs. The mice and elephants, in Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun., New York, NY, USA, 2003, p. 75.
- [2] N. Agrawal and S. Tapaswi, Defense mechanisms against DDoS attacks in a cloud computing environment: State-of-the-Art and research chal- Oct. 2019.
- [3] M. S. Bonm, K. L. Dias, and S. F. L. Fernandes, Integrated NFV/SDN architectures: A systematic literature review, ACM Comput. Surveys, [10] J. Cao, Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, and Y. Yang, The crosspath attack: Disrupting the \$SDN\$ control channel via shared links, in Proc.
- [4] T. Apostolovic, N. Stankovic, K. Milenkovic, and Z. Stanisavljevic, DDoSSimSystem for visual representation of the selected distributed denial of service attacks, in Proc. Zooming Innov. Consum. Technol. Conf.
- [5] Dominus. (2018). Hulk Ddos Attack Script Created Using Python Libs..
- [6] Y. Zhang, Z. Morley Mao, and J. Wang, Low-rate TCP-targeted dos attack disrupts Internet routing, in Proc. 14th Annu. Netw. Distrib. Syst. Secur.
- [7] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, Information metrics for low-rate DDoS attack detection: A comparative evaluation, in Proc.
- [8] S. Floyd and V. Jacobson, Random early detection gateways for conges- [30] M. V. Kieu, D. T. Nguyen, and T. T. Nguyen, Using CPR metric to detect and lter low-rate DDoS ows, in Proc. 8th Int. Symp. Inf. Commun. Technol., 2017, p. 325.
- [9] N. Meti, D. G. Narayan, and V. P. Baligar, Detection of distributed denial of service attacks using machine learning algorithms in software dened networks, in Proc. Int. Conf. Adv. Comput., Commun. Informat.
- [10] S. L. Salzberg, C4.5: Programs for machine learning by J. Ross Quinlan. 2001. J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, A DDoS attack detection method based on SVM in software dened network, Secur. Commun
- [11] Ethereum: A Secure Decentralised Generalised Transaction Ledge.
- [12] Blockchain for Financial Services. Accessed: Jul. 1, 2019. [Online].
- [13] Etherscan. The Ethereum Block Explorer: Ropsten (Revival) Testnet.
- [14] P. Ferguson and D. Senie, Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoong (BCP 38), document RFC 2827, 2000. ietf.org/html/rfc2827

- [15] F. Guo, J. Chen, and T.-C. Chiueh, Spoof detection for preventing DoS put. Syst. (ICDCS), Washington, DC, USA, Jul. 2006, p. 37. doi: 10.1109/ICDCS.2006.78.
- [16] REST API. Accessed: www.sowrt.com/reference.php Jul. 1, 2019.
- [17] B. B. Zarpelao, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, A survey of intrusion detection in Internet of Things, *J. Netw. Comput.*
- [18] R. Renk, L. Saganowski, W. Holubowicz, and M. Choras, Intrusion detection system based on matching pursuit, in *Proc. 1st Int. Conf. Intell.*
- [19] W. Lu and A. A. Ghorbani, Network anomaly detection based on Dec. 2008. N. Z. Bawany, J. A. Shamsi, and K. Salah, DDoS attack detection and mitigation using SDN: Methods, practices, and solutions, *Arabian J. Sci.*
- [20] S. Shin and G. Gu, Attacking software-defined networks: A feasibility study, in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw.*
- [21] L. Zhou, M. Liao, C. Yuan, and H. Zhang, Low-rate DDoS attack detection using expectation of packet size, *Secur. Commun. Netw.*, Oct. 2017, Art. no. 3691629.
- [22] Y. Xie and S. Z. Yu, Monitoring the application-layer DDoS attacks for Feb. 2009. L. Nie, D. Jiang, and Z. Lv, Modeling network traffic for traffic matrix estimation and anomaly detection based on Bayesian network in cloud 2017.
- [23] M. Arlitt and T. Jin. (Aug. 1998). 1998 World Cup Web Site Access Logs.
- [24] M. Arlitt and T. Jin, A workload characterization study of the 1998 World on the top one million visited Web pages, in *Proc. 8th Euro-NF Conf.*
- [25] H. V. Nguyen and Y. Choi, Proactive detection of DDoS attacks utilizing k-NN classifier in an anti-DDoS framework, *Int. J. Elect., Comput., Syst.*
- [26] J. Cheng, M. Li, X. Tang, V. S. Sheng, Y. Liu, and W. Guo, Flow correlation degree optimization driven random forest for detecting DDoS Art. no. 6459326.
- [27] X. Ma and Y. Chen, DDoS detection method based on chaos analysis of Jan. 2014.
- [28] M. Baskar, T. Gnanasekaran, and S. Saravanan, Adaptive IP traceback *Emerg. Trends Comput., Commun. Nanotechnol. (ICECCN)*, Tirunelveli, [30] S. Behal and K. Kumar, Detection of DDoS attacks and ash events using Apr. 2017
- [29] N. A. Singh, K. J. Singh, and T. De, Distributed denial of service attack detection using naive Bayes classifier through info gain feature selection, in *Proc. Int. Conf. Inform. Anal.*, Aug. 2016, p. 54.