A report on

PREDICTION OF DDOS ATTACK USING DEEP LEARNING

Submitted in partial fulfillment of the

requirementsfor the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering

By

N. JYOTHI	**(204G1A0546)**
G. MAHESH KUMAR	**(204G1A0551)**
V. MEGHANA	**(204G1A0556)**
J. ASHOK	**(204G1A0517)**

Under the Guidance of

Dr. C. Sasikala, M.Tech, Ph.D.

Associate Professor

Department of Computer Science & Engineering

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY
ANANTAPURAMU
(Autonomous)

(Affiliated to JNTUA, Accredited by NAAC with 'A' Grade, Approved by AICTE, New Delhi & Accredited by NBA (EEE, ECE & CSE))

2022-2023

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY
(Autonomous)
(Approved by AICTE & Affiliated to JNTU, Anantapur)
Rotarypuram Village , B K Samudram Mandal , Ananthapur - 515701

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

This is to certify that the Project Stage 1 report entitled **Predicting of DDoS attack using Deep Learning** is the bonafide work carried out by **N. Jyothi** bearing Roll Number 204G1A0546, G. Mahesh Kumar bearing Roll Number 204G1A0551, **V. Meghana** bearing Roll Number 204G1A0556, J. Ashok bearing Roll Number 204G1A0517 in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2023-2024.

Guide	**Head of the Department**
Dr. C. Sasikala, M.Tech, Ph.D.	Mr. P. Veera Prakash, M.Tech, (Ph.D.)
Associate Professor	Assistant Professor

Date:

Place: Rotarypuram

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express our gratitude for all of them.

It is with immense pleasure that we would like to express our indebted gratitude to our Guide **Dr. C. Sasikala** **M.Tech., Ph.D**, **Associate Professor**, **Computer Science & Engineering Department**, who has guided us a lot and encouraged us in every step of the project work. We thank her for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this project work.

We are very much thankful to **Mr. P. Veera Prakash** **M.Tech. (Ph.D)**, **Assistant Professor & Head Of the Department, Computer Science & Engineering,** for his kind support and for providing necessary facilities to carry out the work.

We wish to convey our special thanks to **Dr. G. Balakrishna** **M.Tech. Ph.D**, **Principal** of **Srinivasa Ramanujan Institute of Technology** for giving the required information in doing our project work. Not to forget, we thank all other faculty and non-teaching staff, and our friends who had directly or indirectly helped and supported us in completing our project in time.

We also express our sincere thanks to the Management for providing excellent facilities**.**

Finally, we wish to convey our gratitude to our family who fostered all the requirements and facilities that we need.

Project Members

DECLARATION

We, Ms. N. Jyothi bearing reg no: 204G1A0546, Mr. G. Mahesh Kumar bearing reg no: 204G1A0551, Ms. V. Meghana bearing reg no: 204G1A0556, Mr. J. Ashok bearing reg no: 204G1A0517, students of SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram, hereby declare that the dissertation entitled "PREDICTING DDOS ATTACK USING DEEP LEARNING" embodies the report of our project work carried out by us during IV Year Bachelor ofTechnology under the guidance of Dr. C. Sasikala M.Tech., Ph.D, Department of CSE and this work has been submitted for the partial fulfillment of the requirements for the award of Bachelor of Technology degree.

The results embodied in this project report have not been submitted to any other Universities of Institute for the award of Degree.

N. JYOTHI	Reg no: 204G1A0546
G. MAHESH KUMAR	Reg no: 204G1A0551
V. MEGHANA	Reg no: 204G1A0556
J. ASHOK	Reg no: 204G1A0517

Contents **Page No**

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DFD	Data flow digaram
OO	Object oriented
UML	Unified Modeling Language
DL	Deep learning
ML	Machine Learning
DNN	Deep Neural Networks
RNN	Recurrent Neural Networks
DOS	Denial of Service
DDoS	Distributed Denial of Service
TCP	Transition Control Protocol
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
SDN	Standard Normal Deviation
IoT	Internet Of Things
ANN	Artificial neural networks

ABSTRACT

In recent years, Internet services have been increased in public and business ventures for production tasks. So internet applications need a lot of security to secure the data of other businesses and also itself. DDoS attacks are major security risks in the application environment. It happens by sending thousands of requests to flood the server and prevent it from processing requests. DDoS attack is a significant cybersecurity challenge that makes a particular system or network out of reach and unusual for some time. It affects the server's resources. The proposed system is used to detect such types of attacks by utilizing LSTM algorithm and a high level of accuracy. Hence, this work aims to solve this issue by applying a Long Short-term memory(LSTM) algorithm with a high degree of accuracy to detect these types of assaults. The suggested technique, which has a 93% accuracy rate in identifying DDoS attacks, will be evaluated and simulated using Python and it is compared with the exsisting machine learning algorithms.

Keywords:

Deep Learning, Long-short term memory, Distributed Denial-of-service(DDoS)attacks

CHAPTER 1

INTRODUCTION

A distributed denial-of-service (DDoS) attack is a deliberate attempt to disrupt normal operations of a server, service, or network by overloading the target or the infrastructure around it with too much traffic. The effectiveness of DDoS assaults originates from their capacity to leverage a large number of compromised computer systems as attack traffic sources. Machines that are networked and have Internet of Things devices could be deemed as exploited machines. At a high level, a denial-of-service attack (DDoS) might be likened to an unexpected traffic jam that closes a highway and prevents regular traffic from getting to its intended destination. DDoS assaults make use of computer networks that are online.

These networks are made up of computers and other devices (such Internet of Things devices) that have been infected with malware, enabling an attacker to remotely manipulate them. These standalone devices are known as bots (sometimes called zombies), and a collection of bots is known as a botnet. An attacker can control an attack by remotely instructing each bot in the botnet once it has been set up as shown in Figure 1. Every bot that is sent to an IP address that is the target of a botnet attack sends queries to the IP address of the victim, which may overwhelm the server or network and cause a denial of service to regular traffic. It might be challenging to distinguish between attack and legal traffic because every bot is an Internet device.

The most noticeable sign of a denial-of-service assault is when a website or service suddenly becomes unreliable or slow. However, since several factors, such a real traffic increase, can result in comparable performance problems, more research is typically necessary. You can detect some of these obvious indicators of a DDoS assault with the aid of traffic analytics tools. Unusual volumes of traffic coming from a single IP address or range, a deluge of traffic from users with similar device types, geolocation settings, or web browser versions, or an inexplicable spike in requests to a single page or endpoint are all signs of suspicious activity. Unusual traffic patterns, such spikes at strange times of day or patterns that seem out of the ordinary (like a spike every 10 minutes).

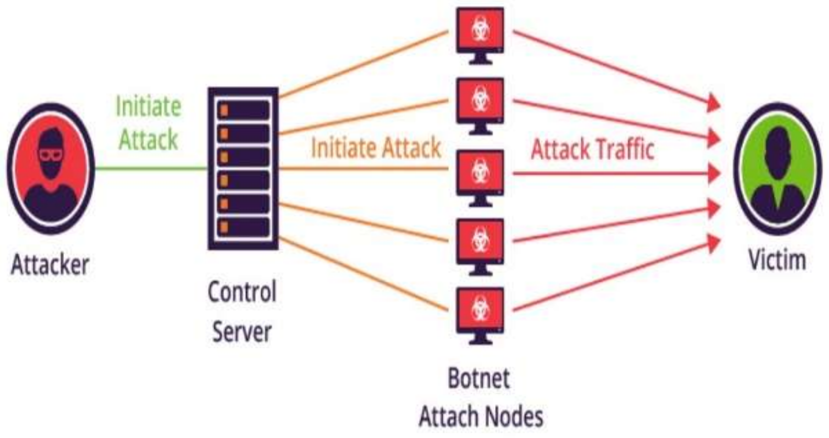


Figure 1.1 : DDoS Attack using Botnet[13]

Some of the famous DDoS attacks on some organizations such as, on 28 February 2018 The largest-ever DDoS attack was launched against GitHub, a well-known online code management site utilized by millions of developers. platform was not ready for the enormous influx of traffic, which peaked at a record-breaking 1.3 terabits per second, even though it was accustomed to high levels of traffic. The GitHub attack used a technique called memcaching, a database caching solution meant to speed up networks and websites, rather than botnets. After successfully impersonating GitHub, the attackers significantly increased the volume of traffic going to the platform. Thanks to the DDoS protection solution that GitHub was utilizing, the attack was contained and prevented from spreading in less than ten minutes after it started.[14]

October 2016 saw the second-largest DDoS attack against major DNS operator Dyn. The hack caused significant disruption, bringing down the websites of over 80 of its clients, including Reddit, Amazon, Netflix, Airbnb, Spotify, Twitter, and PayPal. Hackers built a vast botnet of 100,000 Internet of things (IoT) devices to launch their attack using a malware known as Mirai. Radios, smart TVs, and printers were among the gadgets that were set up to bombard Dyn with requests and cause traffic congestion. Approximately 14,500 domains stopped using Dyn's services immediately after the attack, which is estimated to have caused $110 million in damage even though it was contained in a single day.[14]

Ransomware and DDoS assaults were identified as the top two threats affecting businesses in 2018 by the UK's National Crime Agency. They saw a sharp rise in attacks and recommended that organizations take urgent action to fortify themselves against this escalating danger.

This lengthy list makes it clear that DDoS assaults have the power to bring down entire corporate websites, networks, and, as the Dyn attack showed, nearly the whole internet.

Businesses ought to think about utilizing a DDoS protection service, which can identify unusual traffic patterns and divert DDoS attacks off the network. Additional security precautions include using firewalls, VPNs, anti-spam software, and additional DDoS defense layers to safeguard network infrastructure.

1.1.Real-time of attacker disrupting user:

DDoS attacks involve malicious efforts to overwhelm a target system or network with an excessive volume of traffic, causing disruption and rendering the services inaccessible to legitimate users. The dynamic and distributed nature of cloud infrastructures further complicates the detection and mitigation of such attacks. Traditional security measures, while effective to some extent, are often insufficient in addressing the evolving sophistication of DDoS attacks.

In recent years, the rapid proliferation of cloud computing has revolutionized the way organizations manage and deploy their IT infrastructure. Cloud environments offer scalability, flexibility, and cost efficiency, making them an attractive choice for hosting critical applications and services. However, this widespread adoption has also exposed cloud systems to an escalating threat landscape, with Distributed Denial of Service (DDoS) attacks emerging as a formidable challenge.

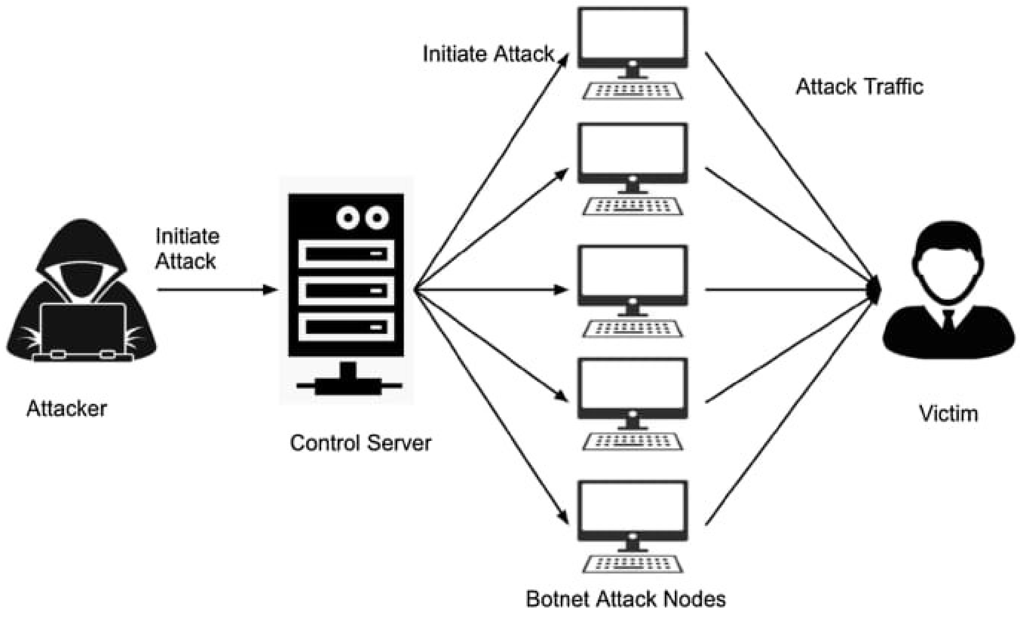


Figure 1.2: DDoS attacker disrupting the user

The figure 2 explains how the attacker attacks the network using a botnet and disrupts the usage of the normal user to access the server. By using botnet attackers increase the traffic over the internet and server, which makes the user unable to reach the server.

1.2.Deep Learning

This work focus on leverage the power of deep learning techniques for the prediction and early detection of DDoS attacks in cloud environments. Deep learning, a subset of machine learning, has demonstrated remarkable capabilities in extracting intricate patterns and features from complex data sets. By harnessing the inherent adaptability of deep learning algorithms, this project seeks to enhance the ability to identify and respond to DDoS threats in real time, thereby fortifying the security posture of cloud-based systems.

1.3 Objective:

The objectives of this study encompass the development of a robust deep-learning model trained on historical data to recognize subtle patterns indicative of impending DDoS attacks. Additionally, the project will explore the integration of anomaly detection mechanisms to augment the model's ability to discern abnormal network behavior. The ultimate goal is to create an intelligent and proactive defense system capable of predicting and mitigating DDoS attacks before they can inflict significant damage. The objectives of the paper is three methodologies:

i. To develop a Deep Learning model (LSTM) to detect the attack.

ii. To compare the model with the Machine Learning models.

iii. To create a model which can be used for real-time detection of DDos attacks.

Through this work, aspire to contribute to the advancement of security by providing a predictive framework that empowers organizations to safeguard their critical assets and ensure uninterrupted service delivery in the face of evolving cyber threats. The outcomes of this project hold the potential to redefine the landscape of DDoS defense in the cloud, fostering a more resilient and secure digital environment for businesses and individuals alike.

CHAPTER 2
LITERATURE SURVEY

It is unimaginable that a single attack could result in so significant damage to a computer system or network. However, due to its nature, DDoS will actually bring down the entire network. Its prevention is consequently very difficult to achieve. As a result, there is a huge demand for effective frameworks for DDoS attack detection. Several writers have developed several approaches to identify DDoS attacks in response to this demand. A few of them are detailed it has advantages and disadvantages:

Ankit Agarwal.[1] Most of the methods cannot simultaneously achieve efficient detection with a small number of false alarms. In this case, deep learning techniques are appropriate and effective algorithms to categorize both normal and attacked information. Hence, a novel feature selection-whale optimization algorithm deep neural network (FS-WOA–DNN) method is proposed in this research article to mitigate DDoS attack effectively. Initially, a pre-processing step is carried out for the input dataset where a min–max normalization technique is applied to replace all the input in a specified range. Later on, that normalized information is fed into the proposed FSWOA to select the optimal set of features for ease of the classification process. Those selected features are subjected to a deep neural network classifier to categorize normal and attacked data.

Mohammad Shurman.[2] In this paper, they proposed two methodologies to detect Distributed Reflection Denial of Service (DDoS) attacks in IoT. The first methodology uses a hybrid Intrusion Detection System (IDS) to detect IoT-DoS attacks. The second methodology uses deep learning models, based on Long Short-Term Memory (LSTM) trained with the latest dataset for such kinds of DDoS

Chen Zhibin[3]. In this work, they apply a Hybrid Deep Learning method to detect malicious web traffic in the form of DDoS attacks, controlling the web flow of information reaching a server, and using any dependencies between the different elements of a data stream. An original and cutting-edge Hierarchical Temporal Memory (HTM) hybrid model has been proposed. (e operation of this model is predicated primarily on the portion of the cerebral cortex known as the neocortex. (The neocortex is in charge of various fundamental brain functions, including the perception of senses, the comprehension of language, and the control of movement.

Dong, S., & Sarem, M [4] 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. The other method uses the improved K-Nearest Neighbors (KNN) algorithm based on Machine Learning (ML) to discover the DDoS attack. The results of the theoretical analysis and the experimental results on datasets show that our proposed methods can better detect the DDoS attack compared with other methods.

Abbas, K., & Jain, R[5] Recently, software defined networks (SDNs) and cloud computing have been widely adopted by researchers and industry. However, widespread acceptance of these novel networking paradigms has been hampered by the security threats. Advances in the processing technologies have helped attackers in increasing the attacks too, for instance, the development of Denial of Service (DoS) attacks to distributed DoS (DDoS) attacks which are seldom identified by conventional firewalls. In this paper, we present the state of art of the DDoS attacks in SDN and cloud computing scenarios. Especially, we focus on the analysis of SDN and cloud computing architecture. Besides, we also overview the research works and open problems in identifying and tackling the DDoS attacks.

Wang, Y [6], it is necessary to propose an effective method to detect DDoS attack from massive data traffics. However, the existing schemes have some limitations, including that supervised learning methods, need large numbers of labeled data and unsupervised learning algorithms have relatively low detection rate and high false positive rate. In order to tackle these issues, this paper presents a semi-supervised weighted k-means detection method. Specifically, in this paper, firstly present a Hadoop-based hybrid feature selection algorithm to find the most effective feature sets and propose an improved density-based initial cluster centers selection algorithm to solve the problem of outliers and local optimal. Then, we provide the Semi-supervised K-means algorithm using hybrid feature selection (SKM-HFS) to detect attacks. Finally, we exploit DARPA DDoS dataset, CAIDA "DDoS attack 2007" dataset, CICIDS "DDoS attack 2017" dataset and real-world dataset to carry out the verification experiment. The experiment results have demonstrated that the proposed method outperforms the benchmark in respect of detection performance and technique for order preference by similarity to an ideal solution (TOPSIS) evaluation factor.

As per the above papers, the conclusion is by DDoS attack the functioning of the system gets interrupted and the website stops working for the user. They developed several models but the accuracy was not up to the mark and it is less. So there is a need to improve the accuracy.

CHAPTER 3
DATA PREPROCESSING

3.1 Deep Learning

Deep learning mimics the neural networks within the human brain, employing layers of interconnected nodes to progressively learn hierarchical features and abstract representations. This approach empowers machines to autonomously analyze data, recognize patterns, and make decisions, often surpassing traditional algorithms in handling intricate tasks such as image and speech recognition, natural language processing, and complex decision-making. As we delve into the world of deep learning, this journey will unveil the underlying principles, architectures, and applications that fuel its remarkable capabilities. From convolutional neural networks (CNNs) for image processing to recurrent neural networks (RNNs) for sequential data analysis, deep learning techniques continue to redefine the boundaries of what machines can achieve in the realm of artificial intelligence.

3.2 Dataset Preprocessing

Preparing unprocessed data so that a deep learning model may use it is known as data preparation. To put it another way, when data is acquired in raw format from multiple sources, it is impractical for analysis. It is an essential first step in building a deep learning model. It is not always the case that we find clean, prepared data when developing a deep learning project. Additionally, data must always be cleaned and formatted before being used in any kind of activity. We therefore employ the data-preprocessing job for this. The procedures we must take to change data so that a machine can understand it are referred to as data preprocessing. The main agenda for a model to be accurate and precise in predictions is that the algorithm should be able to easily interpret the data's features.

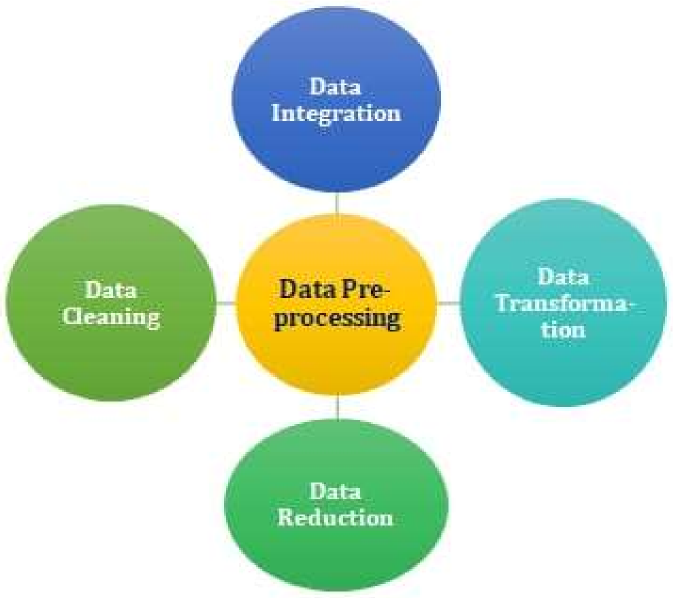


Figure 3.1: Data Preprocessing

3.1.1 Data Cleaning

Data Cleaning is particularly done as part of data preprocessing to clean the data by filling missing values, smoothing the noisy data, resolving the inconsistency, and removing outliers.

1. Missing values

Here are a few ways to solve this issue:

Ignore those tuples

This method should be considered when the dataset is huge and numerous missing values are present within a tuple.

Fill in the missing values

There are many methods to achieve this, such as filling in the values manually, predicting the missing values using regression method, or numerical methods like attribute mean.

2. Noisy Data

It involves removing a random error or variance in a measured variable. It canbe done with the help of the following techniques:

Binning

It is the technique that works on sorted data values to smoothen any noise present in it. The data is divided into equal-sized bins, and each bin/bucket is dealtwith independently. All data in a segment can be replaced by its mean, median or boundary values.

Regression

This data mining technique is generally used for prediction. It helps to smoothen noise by fitting all the data points in a regression function. The linear regression equation is used if there is only one independent attribute; else Polynomial equations are used.

Clustering

Creation of groups/clusters from data having similar values. The values that don't lie in the cluster can be treated as noisy data and can be removed.

3. Removing outliers

Clustering techniques group together similar data points. The tuples that lie outside the cluster are outliers/inconsistent data.

3.1.2 Data Integration

Data Integration is one of the data preprocessing steps that are used to merge thedata present in multiple sources into a single larger data store like a data warehouse. Data Integration is needed especially when we are aiming to solve a real-world scenario like detecting the presence of nodules from CT Scan images. The only option is to integrate the images from multiple medical nodes to form a larger database.

We might run into some issues while adopting Data Integration as one of the Data Preprocessing steps Schema integration and object matching. The data can be present in different formats, and attributes that might cause difficulty in data integration. Removing redundant attributes from all data sources. Detection and resolution of data value conflicts.

3.1.3 Data Transformation

Once data clearing has been done, we need to consolidate the quality data into alternate forms by changing the value, structure, or format of data using the below- mentioned Data Transformation strategies.

Generalization

The low-level or granular data that we have converted to high-level information by using concept hierarchies. We can transform the primitive data in the address like the city to higher-level information like the country.

Normalization

It is the most important Data Transformation technique widely used. The numerical attributes are scaled up or down to fit within a specified range. In this approach, we are constraining our data attribute to a particular container to develop a correlation among different data points.

Attribute Selection

New properties of data are created from existing attributes to help in the data mining process. For example, date of birth, data attribute can be transformed to another property like is_senior_citizen for each tuple, which will directly influence predicting diseases or chances of survival, etc.

Aggregation

It is a method of storing and presenting data in a summary format. For example sales, data can be aggregated and transformed to show as per month and year format.

3.1.4 Data Reduction

The size of the dataset in a data warehouse can be too large to be handled by data analysis and data mining algorithms. One possible solution is to obtain a reduced representation of the dataset that is much smaller in volume but produces the same quality of analytical results.

Data cube aggregation

It is a way of data reduction, in which the gathered data is expressed in a summary form.

Dimensionality reduction

Dimensionality reduction techniques are used to perform feature extraction. The dimensionality of a dataset refers to the attributes or individual features of the data.

This technique aims to reduce the number of redundant features we consider in machine learning algorithms. Dimensionality reduction can be done using techniques like Principal Component Analysis etc.

Data compression

By using encoding technologies, the size of the data can significantly reduce. But compressing data can be either lossy or non-lossy. If original data can be obtained after reconstruction from compressed data, this is referred to as lossless reduction; otherwise, it is referred to as lossy reduction.

Discretization

Data discretization is used to divide the attributes of the continuous nature into data with intervals. This is done because continuous features tend to have a smaller chance of correlation with the target variable. Thus, it may be harder to interpret the results. After discretizing a variable, groups corresponding to the target can be interpreted. For example, attribute age can be discretized into bins like below 18, 18- 44, 44-60, above 60.

Numerosity reduction

The data can be represented as a model or equation like a regression model. Thiswould save the burden of storing huge datasets instead of a model.

Attribute subset selection

It is very important to be specific in the selection of attributes. Otherwise, it might lead to high dimensional data, which are difficult to train due to underfitting/overfitting problems. Only attributes that add more value towards model training should be considered, and the rest all can be discarded.

CHAPTER 4

PROPOSED SYSTEM

4.1 Existing System

In the existing system, the approach to tackling DDoS attacks involves a specific flow and relies on a set of techniques for model development. However, it has become evident that this approach encounters limitations, particularly in terms of memory utilization and the accuracy of results. To enhance the effectiveness of this system, incorporating machine learning methods, specifically DecisionTreeClassifier and GradientBoostingClassifier, offers several advantages over traditional techniques.

4.1.1 Disadvantages

Data Imbalance: DDoS attack data is often heavily imbalanced, with a significant disparity between normal traffic and attack data. This imbalance can lead to biased models that struggle to accurately predict attacks due to the scarcity of attack samples.

Labeling Challenges: Labeling DDoS attack data for training deep learning models can be difficult and time-consuming. Accurate labeling is essential for supervised learning, but the dynamic and evolving nature of DDoS attacks can make it challenging to keep datasets up to date.

Overfitting: Deep learning models are prone to overfitting, especially when dealing with relatively small DDoS attack datasets. Overfit models perform well on training data but may not generalize effectively to new, unseen attacks or variations in attack patterns.

Resource Intensiveness: Training deep learning models for DDoS prediction in the cloud can be computationally intensive and may require substantial resources, including powerful hardware and large datasets. This can increase operational costs and complexity.

False Positives/Negatives: Deep learning models can produce false positives (flagging normal traffic as attacks) and false negatives (failing to detect real attacks). This trade-off between sensitivity and specificity can be challenging to balance, potentially leading to alert fatigue or missed attacks.

4.2 Proposed System

We propose an innovative application that can be considered a highly useful system, as it addresses the limitations commonly encountered with traditional and other existing methods for DDoS attack detection. The primary objective of this study is to develop a fast and reliable method that accurately detects the effects of Distributed Denial of Service (DDoS) attacks. In designing this system, we have leveraged the capabilities of a powerful algorithm in a Python-based environment, which includes the integration of Long Short-Term Memory (LSTM) neural networks.

4.2.1 Advantages

Early Threat Detection: Deep learning models can analyze network traffic patterns and anomalies in real-time, enabling early detection of potential DDoS attacks. This allows for timely mitigation before the attack disrupts services.

Improved Accuracy: Deep learning algorithms can learn and adapt to evolving attack strategies, making them more accurate in distinguishing between legitimate and malicious traffic. This reduces false positives and minimizes the chances of blocking legitimate users.

Scalability: Cloud-based deep learning solutions are highly scalable, allowing them to handle large volumes of network traffic. As cloud environments can experience sudden spikes in traffic, the ability to scale the prediction system is crucial.

Automation: Deep learning models can automatically trigger countermeasures when a DDoS attack is detected, such as diverting traffic through a scrubbing center or implementing access controls. This reduces the need for manual intervention and speeds up the response time.

Behavior-Based Detection: Deep learning models can analyze the behavior of incoming traffic, looking for patterns associated with DDoS attacks, rather than relying on static rules. This makes them more adaptive and effective in identifying novel attack vectors.

4.3 work Flow of Proposed system

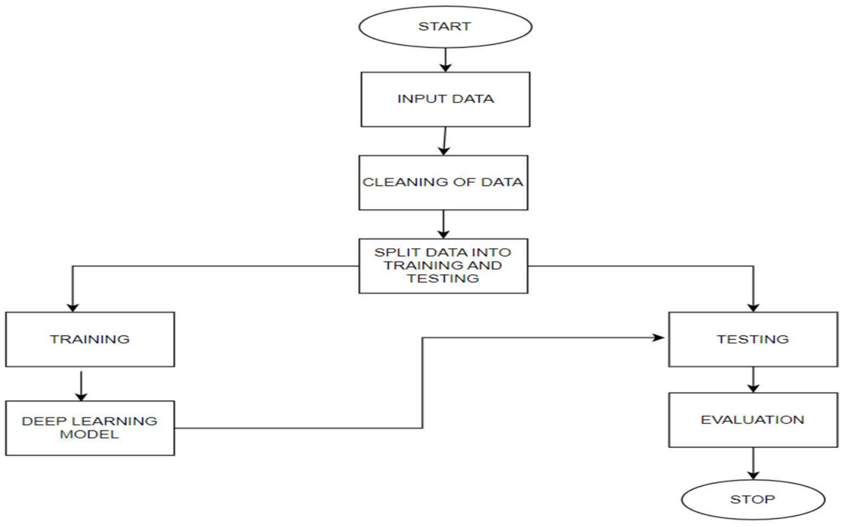


Figure 4.1 Work Flow of the Proposed System

4.4 Architecture of DDoS attack

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the availability of a website, server, or network by overwhelming it with a flood of incoming traffic from multiple sources.

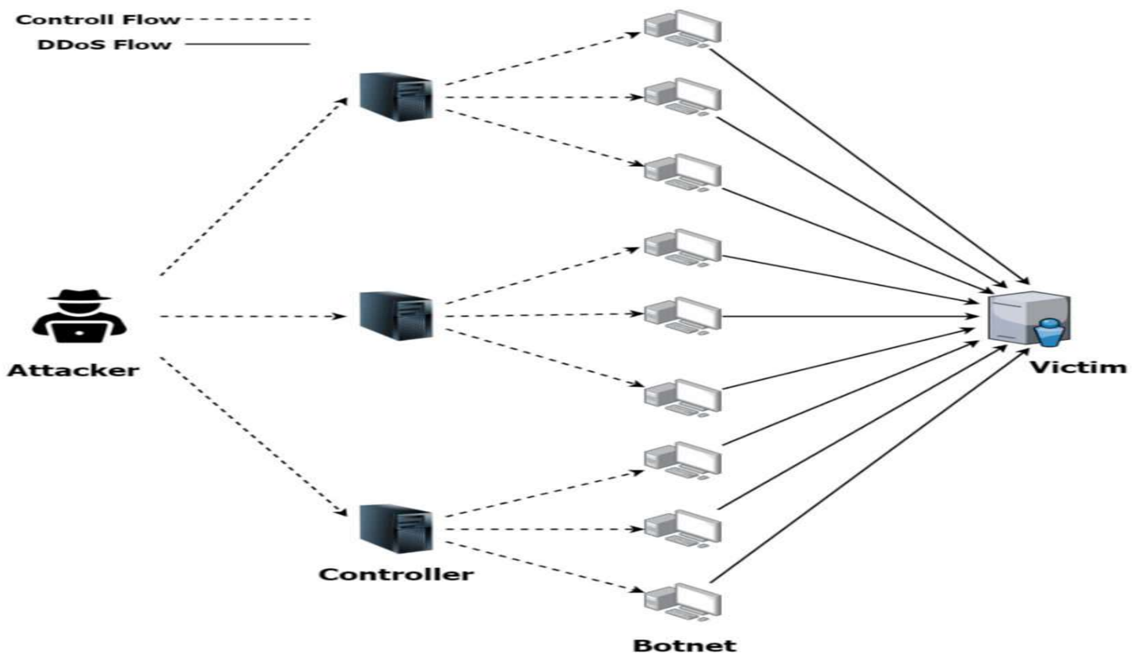


Figure 4.2 : A Architecture of Distributed Denial of Service (DDoS) attack

The architecture of a DDoS attack typically involves several key components:

Botnet: The attacker usually controls a large number of compromised computers, servers, or Internet of Things (IoT) devices that are collectively known as a botnet. These devices are often infected with malware that allows the attacker to control them remotely without the owners' knowledge. The botnet acts as a network of "bots" that canbe instructed to generate and send massive amounts of traffic towards the target system.

Command and Control (C&C) Server: The attacker uses a central server or a group of servers, known as the Command and Control (C&C) server, to send instructions to the botnet. These instructions can include the target IP address or domain name, the typeof attack to be launched, and the duration of the attack.

Attack Tools: The attacker uses specialized software or tools to automate and orchestrate the DDoS attack. These tools can include programs or scripts that control the botnet, generate malicious traffic, and launch various types of DDoS attacks such as volumetric attacks, protocol attacks, or application layer attacks.

Victim System: The target of the DDoS attack is the victim system, which can be a website, server, or network. The victim system is overwhelmed with an excessiveamount of traffic from the botnet, causing it to become unresponsive or unavailable to legitimate users.

Spoofed IP Addresses: To make it difficult to trace the attack back to the original source, the attacker often spoofs or falsifies the IP addresses of the botnet devices. This makes it challenging for the victim system to block the attack based on the source IP addresses, as they appear to be coming from different locations.

Amplification Techniques: In some cases, the attacker may use amplification techniques to magnify the volume of traffic being sent to the victim system. For example, the attacker may use reflective amplification, where they send requests with a spoofed source IP address to vulnerable servers that respond with a much larger response to the victim system, overwhelming its resources.

Coordinated Timing: The attacker may coordinate the timing of the DDoS attack to maximize its impact. For example, launching the attack during peak hours of website traffic or during critical events to cause the most disruption.

In summary, the architecture of a DDoS attack involves a botnet controlled by a C&C server, attack tools to automate and orchestrate the attack, spoofed IP addresses to hide the attacker's identity, amplification techniques to magnify the attack, and coordinated timing for maximum impact. It's important to note that DDoS attacks are illegal and can cause significant harm to the targeted systems and organizations.

CHAPTER 5
METHODOLOGY AND ALGORITHMS

5.1 Module:

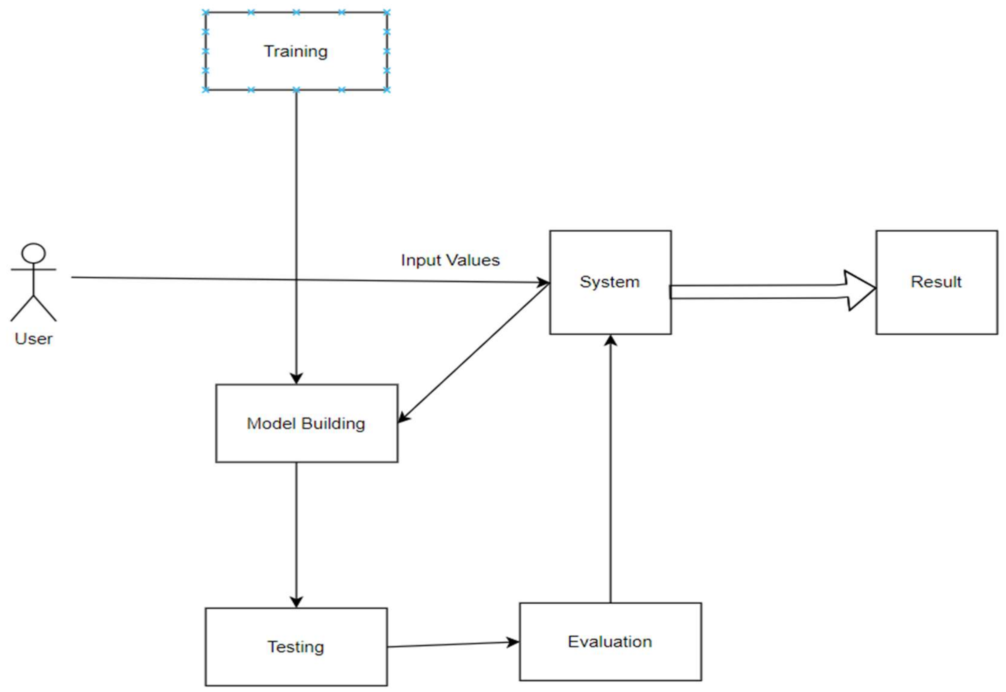


Figure 5.1 : GUI INTERFACE

5.1.1 User:

Upload Dataset (1.1): In this module, users have the capability to upload their dataset, typically in a specified format (e.g., CSV, Excel, or database connection). The system should provide clear instructions on the accepted data format and structure, ensuring a seamless data upload process.

View Dataset (1.2): Users can view the dataset they have uploaded. The system may provide features for data visualization, filtering, and summary statistics to help users understand and explore the dataset before initiating the prediction process.

Input Values for Prediction (1.3): Users need to provide input values relevant to the prediction task. These inputs could include specific data points or variables necessary for the model to make predictions. The system should guide users on what input is required and validate the inputs to ensure they meet the necessary criteria.

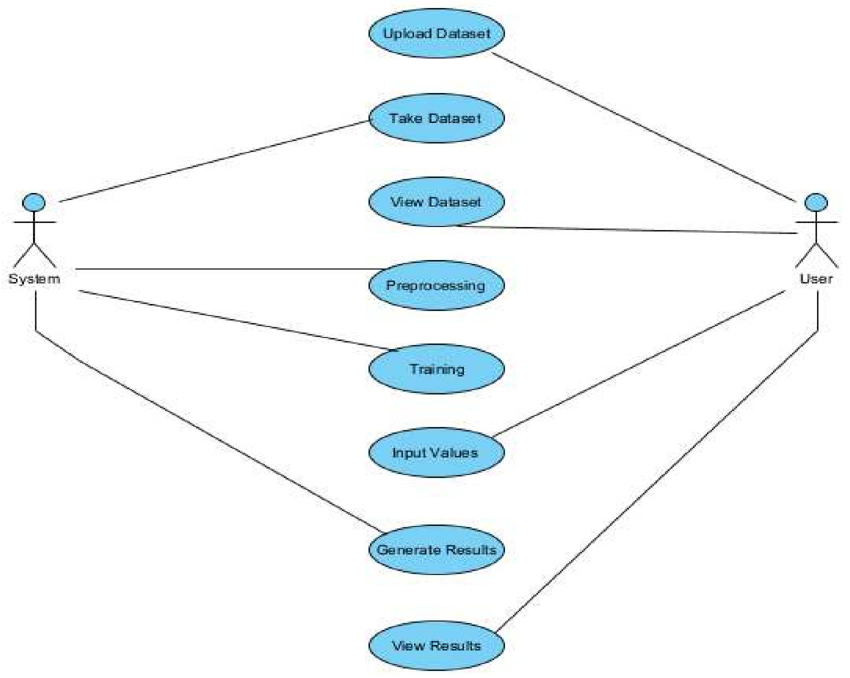


Figure 5.2 Usecase daigram

5.1.2 System:

Take the Dataset (2.1): The system takes the dataset uploaded by the user and stores it securely. It performs data integrity checks and ensures that the dataset is available for further processing.

Preprocessing (2.2): In the preprocessing phase, the system cleans and prepares the data for model building. This involves handling missing values, data transformation, normalization, and feature engineering. It is a critical step to ensure the dataset is ready for training.

Training (2.3): The system utilizes machine learning or deep learning techniques to build a predictive model based on the preprocessed dataset. This may involve splitting the data into training and testing sets, selecting an appropriate algorithm, and training the model on the training data. The model is then evaluated for its performance on the testing data.

Generate Results (2.4): Once the model is trained, the system uses it to generate results. For a DDoS attack prediction system, this could mean evaluating whether the input values provided by the user are indicative of an attack or not. Results may be presented to the user in a user-friendly format, such as a binary classification (e.g., "Attack Detected" or "No Attack Detected") or with probability scores. Users may also receive insights or visualizations that help them understand the model's decisions.

5.2 Algorithm

5.2.1 Gradient boosting:

The gradient boosting algorithm is one of the most powerful algorithms in the field of machine learning. As we know the errors in machine learning algorithms are broadly classified into two categories i.e. Bias Error and Variance Error. As gradient boosting is one of the boosting algorithms it is used to minimize bias error of the model.

Unlike, the Adaboosting algorithm, the base estimator in the gradient boosting algorithm cannot be mentioned by us. The base estimator for the Gradient Boost algorithm is fixed i.e. Decision Stump. Like, AdaBoost, we can tune the n_estimator of the gradient boosting algorithm. However, if we do not mention the value of n estimator, the default value of n estimator for this algorithm is 100.

Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is Mean Square Error (MSE) and when it is used as a classifier then the cost function is Log loss.

5.2.2 Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal.

A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn’t split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can’t ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

5.2.3 RNN :

NEURAL NETWORK:

- A Neural Network consists of different layers connected, working on the structure and function of a human brain. It learns from huge volumes of data and uses complex algorithms to train a neural net.
- Here is an example of how neural networks can identify a dog's breed based on their features.
- The image pixels of two different breeds of dogs are fed to the input layer of the neural network.
- The image pixels are then processed in the hidden layers for feature extraction.
- The output layer produces the result to identify if it's a German Shepherd or a Labrador.
- Such networks do not require memorizing the past output.
- Several neural networks can help solve different business problems. Let's look at a few of them.

- A Recurrent Neural Network works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

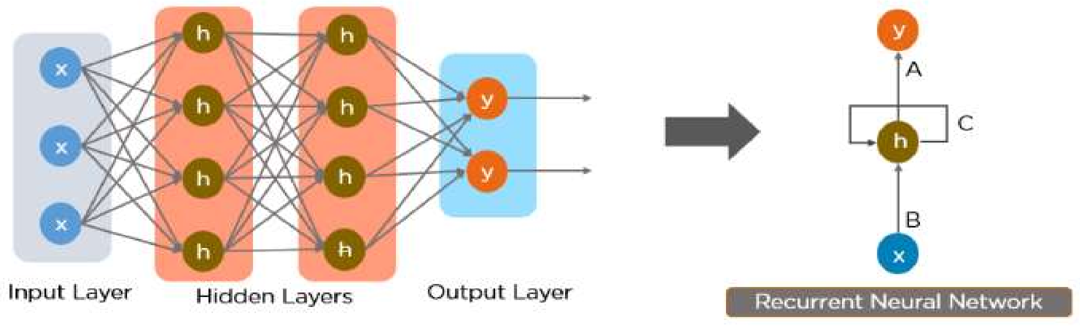


Figure 5.3 : Simple Recurrent Neural Network

Feed-Forward Neural Networks:

- A feed-forward neural network allows information to flow only in the forward direction, from the input nodes, through the hidden layers, and to the output nodes. There are no cycles or loops in the network.
- Below is how a simplified presentation of a feed-forward neural network looks like:

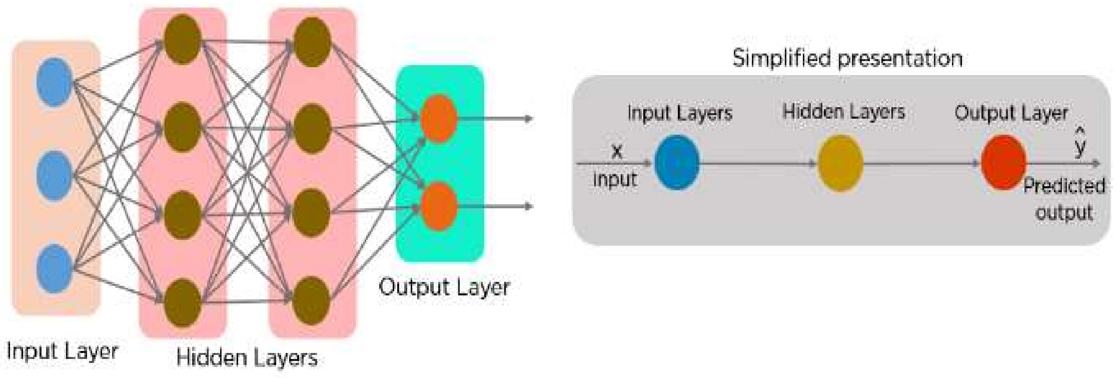


Figure 5.4 : Feed-forward Neural Network

Why Recurrent Neural Networks are better?

- Recurrent neural networks were created because there were a few issues in the feed-forward neural network:
- Cannot handle sequential data
- Considers only the current input
- Cannot memorize previous inputs
- The solution to these issues is the Recurrent Neural Network (RNN). An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.

5.2.4 LSTM:

- Why Recurrent Neural Networks?
- Recurrent neural networks were created because there were a few issues in the feed-forward neural network:
- Cannot handle sequential data
- Considers only the current input
- Cannot memorize previous inputs

- The solution to these issues is the Recurrent Neural Network (RNN). An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.
- It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as images.
- The central role of an LSTM model is held by a memory cell known as a 'cell state' that maintains its state over time. The cell state is the horizontal line that runs through the top of the below diagram. It can be visualized as a conveyor belt through which information just flows, unchanged.

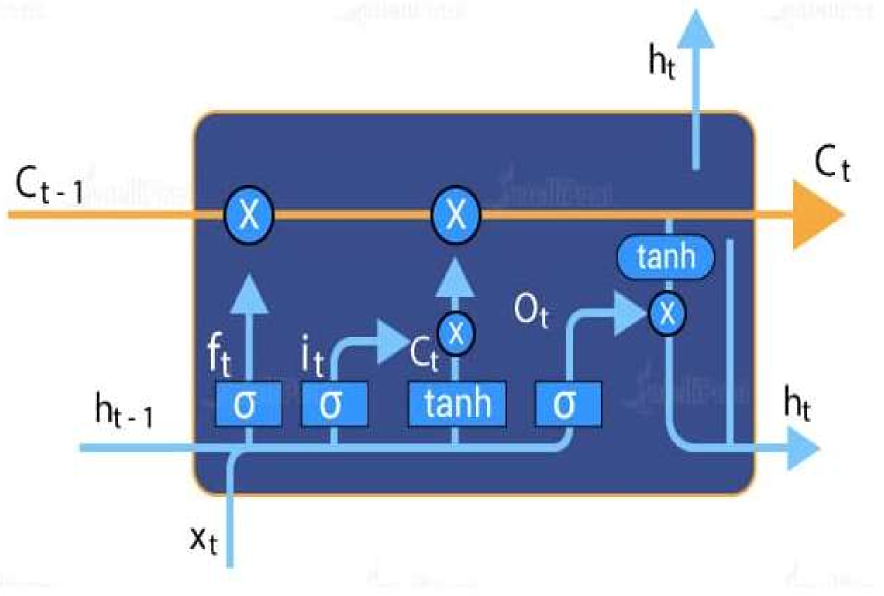


Figure 5.5 LSTM cell with activation functions

- Information can be added to or removed from the cell state in LSTM and is regulated by gates. These gates optionally let the information flow in and out of the cell. It contains a pointwise multiplication operation and a sigmoid neural net layer that assist the mechanism.
- The remember vector is usually called the **forget gate**. The output of the forget gate tells the cell state which information to forget by multiplying 0 to a position in the matrix. If the output of the forget gate is 1, the information is kept in the cell state.
- From equation, sigmoid function is applied to the weighted input/observation and previous hidden state.

- The save vector is usually called the **input gate**. These gates determine which information should enter the cell state / long-term memory. The important parts are the activation functions for each gates.
- The input gate is a sigmoid function and have a range of [0,1]. Because the equation of the cell state is a summation between the previous cell state, sigmoid function alone will only add memory and not be able to remove/forget memory.

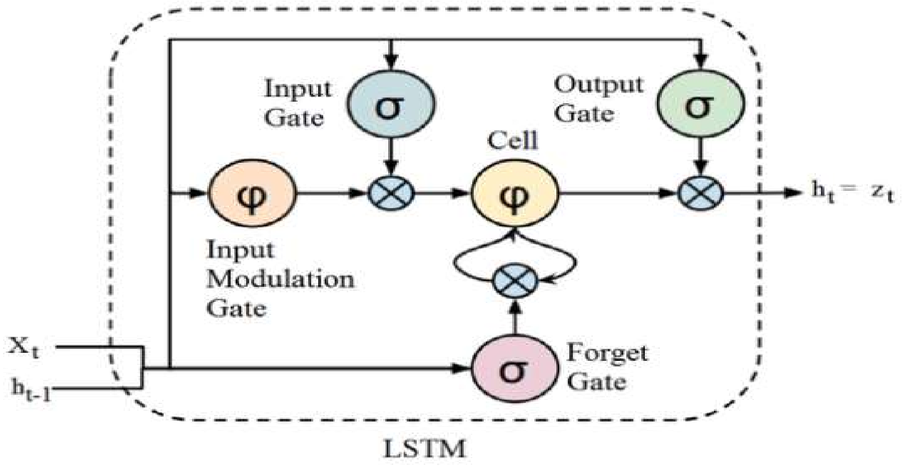


Figure 5.5 LSTM cell

- If you can only add a float number between [0,1], that number will never be zero / turned-off / forget. This is why the input modulation gate has an tanh activation function. Tanh has a range of [-1, 1] and allows the cell state to forget memory.
- The focus vector is usually called the **output gate**. Out of all the possible values from the matrix, which should be moving forward to the next hidden state?
- The first sigmoid activation function is the **forget gate**. Which information should be forgotten from the previous cell state (Ct-1). The second sigmoid and first tanh activation function is our **input gate**. Which information should be saved to the cell state or should be forgotten? The last sigmoid is the **output gate** and highlights which information should be going to the next **hidden state**.

CHAPTER 6

SYSTEM REQURIMENT SPECIFICATION

6.1 Functional and non-functional requirements

Requirement analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

6.1.1 Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

1) Authentication of user whenever he/she logs into the system
2) System shutdown in case of a cyber-attack
3) A verification email is sent to user whenever he/she register for the first time on some software system.

6.1.2 Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

Examples of non-functional requirements:

1) Emails should be sent with a latency of no greater than 12 hours from such an activity.
2) The processing of each request should be done within 10 seconds
3) The site should load in 3 seconds whenever of simultaneous users are > 10000

Hardware Requirements

Processor	**-**
I3/Intel Processor	
Hard Disk	- 160GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- SVGA
RAM	- 8GB

Software Requirements:

Operating System	: Windows 7/8/10
Server side Script	: HTML, CSS, Bootstrap & JS
Programming Language	: Python
Libraries	: Flask, Pandas, Mysql.connector, Os, Smtplib, Numpy
IDE/Workbench	: PyCharm
Technology	: Python 3.6+
Server Deployment	: Xampp Server
Database	: MySQL

6.2 Python

Python is a deciphered, object-situated, significant level prearranging and programming language. Python was first presented in 1991 by Guido van Rossum, a Dutch PC developer who needed to build up a language that could be utilized by anybody. The primary objective of Python was to diminish the expectation to absorb information by picking a grammar that is justifiable as plain English.

The simple syntax rules of the programming language further make it easier for you to keep the code base readable and application maintainable. number of reasons why you should prefer Python to other programming languages. Python is one of the widely used programming languages for image processing. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

6.3 Visual Studio Code

Visual Studio Code (VS Code) is a free and open-source code editor developed by Microsoft. It has gained widespread popularity among developers due to its versatility, lightweight nature, and a rich set of features. Here are some key aspects of Visual Studio Code:

1. Cross-Platform: VS Code is available for Windows, macOS, and Linux, making it a cross-platform code editor that can be used on various operating systems.

2. User Interface: It comes with a clean and intuitive user interface, featuring a sidebar for file navigation, an integrated terminal, and a customizable layout. Users can tailor the appearance and functionality according to their preferences.

3. Language Support: VS Code supports a wide range of programming languages and file types through extensions. You can find extensions for languages like JavaScript, Python, Java, C++, and many more. These extensions enhance the editor's capabilities and provide language-specific features like syntax highlighting, autocompletion, and debugging.

4. Extensions: One of the standout features of VS Code is its extensive library of extensions available through the Visual Studio Code Marketplace. These extensions can add new themes, tools, and language support, enabling developers to customize and extend the functionality of the editor.

5. Integrated Git: VS Code has built-in Git support, allowing developers to manage version control directly from the editor. This includes features such as committing changes, viewing diffs, and resolving merge conflicts.

6. Debugging: The editor provides a robust debugging experience with support for various languages. Developers can set breakpoints, inspect variables, and step through code during the debugging process.

7. IntelliSense: VS Code offers intelligent code completion, known as IntelliSense, which suggests code snippets, variable names, and function signatures based on the context. This feature enhances productivity by reducing manual typing and minimizing errors.

8. Task Automation: With the integrated task runner, developers can automate common tasks like building and running projects. This feature is especially useful for projects with complex build processes.

9. Extensions Marketplace: Visual Studio Code has a thriving ecosystem of extensions available through its marketplace. Developers can easily find and install extensions to tailor the editor to their specific needs.

10. Active Community: The editor has a large and active community of developers contributing to its development and creating extensions. This community support ensures

that VS Code stays up-to-date with the latest technologies and trends.

Overall, Visual Studio Code has become a popular choice for many developers due to its lightweight nature, extensive features, and the support of a vibrant community.

6.4 Modules Used

In Python, Modules are simply files with the ".py" extension containing Python code that can be imported inside another Python Program. In simple terms, we can consider a module to be the same as a code library or afile that contains a set of functions that you want to include in your application.

With the help of modules, we can organize related functions, classes, or any code block in the same file. So, It is considered a best practice while writing bigger codes for production-level projects in Data Science is to split the large Python code blocks into modules containing up to 300–400 lines of code.

The module contains the following components:

- Definitions and implementation of classes,
- Variables, and
- Functions that can be used inside another program.

To incorporate the module into our program, we will use the import keyword, and to get only a few or specific methods or functions from a module, we use the from keyword.

6.4.1 Matplotlib

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.Human minds are more adaptive for the visual representation of data rather than textual data. We can easily understand things when they are visualized. It is better to represent the data through the graph where we can analyze the data more efficiently and make the specific decision according to data analysis.

6.4.2 SkLearn

Scikit-learn (Sklearn) is the most robust machine learning library in Python. It uses a Python consistency interface to provide a set of efficient tools for statistical modeling and machine learning, like classification, regression, clustering, and dimensionality reduction. NumPy, SciPy, and Matplotlib are the foundations of this package, primarily written in Python.

Machine learning academics and data scientistshave flocked to the scikit-learn Python package in the last five years. It includes a collection of tools for tuning model hyperparameters, evaluating, and chaining (pipelines), as well as a unified interface for using models and training.

Machine Learning is the process of teaching a computer to learn and implement tasks without having to write them down explicitly. This indicates that the system is capable of making decisions to some extent. Three types of Machine LearningModels can be implemented using the Sklearn Regression Models Reinforced Learning, Unsupervised Learning, Supervised Learning.

6.4.3 Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them. Seaborn is the only library we need toimport for this simple example. By convention, it is imported with the shorthand sns. Behind the scenes, seaborn uses matplotlib to draw its plots. For interactive work, it's recommended to use a Jupyter/IPython interface in matplotlib mode, or else you'll have to call matplotlib.pyplot.show() when you want to see the plot. This uses the matplotlib rcParam system and will affect how all matplotlib plots look, even if you don't make them with seaborn. Beyond the default theme, there are several other options, and you can independently control the style and scaling of the plot to quickly translate your work between presentation contexts.

6.4.4 Pandas

Pandas is an open source library in Python. It provides ready to use high- performance data structures and data analysis tools. Pandas module runs on top of NumPy and it is popularly used for data science and data analytics. NumPy is a low- level data structure that supports multi-dimensional arrays and a wide range of mathematical array operations. Pandas has a higher-level interface. It also provides streamlined alignment of tabular data and powerful time series functionality. DataFrame is the key data structure in Pandas. It allows us to store and manipulate tabular data as a 2-D data structure. Pandas provides a rich feature-set on the DataFrame. For example, data alignment, data statistics, slicing, grouping, merging, concatenating data, etc. DataFrame is the most important and widely used data structure and is a standard way to store data. DataFrame has data aligned in rows and columns like the SQL table or a spreadsheet database. We can either hard code data into a DataFrame or import a CSV file, tsv file, Excel file, SQL table, etc. We can use the below constructor for creating a DataFrame object.

6.4.5 NumPy

NumPy (Numerical Python) is an open source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

The NumPy library contains multidimensional array and matrix data structures (you'll find more information about this in later sections). It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

6.5 Time:

Regardless of project type, the definition of project time management involves setting time markers against your project and its tasks. It means defining the time value of each such as literature survey, planning, design, algorithm, implementation, testing and documentation task and allocating resource to each step.

Table 4.1: Project Stages

Project Stage	**Duration**
Literature Survey	2 weeks
Planning	1 week
Design	2 weeks
Algorithm	2 weeks
Implementation	3 weeks
Testing	2 weeks
Documentation	1 week

CHAPTER 7
IMPLEMENTATION

The step by step execution of the prediction of DDoS attacks is as follows. The entire project is divided into several modules in which they are dependent on one other. Each module perform unique work in the implementation of the project they are:

1.DataCollection

2.DataPreprocessing

3.Model Creation

4.Model Training

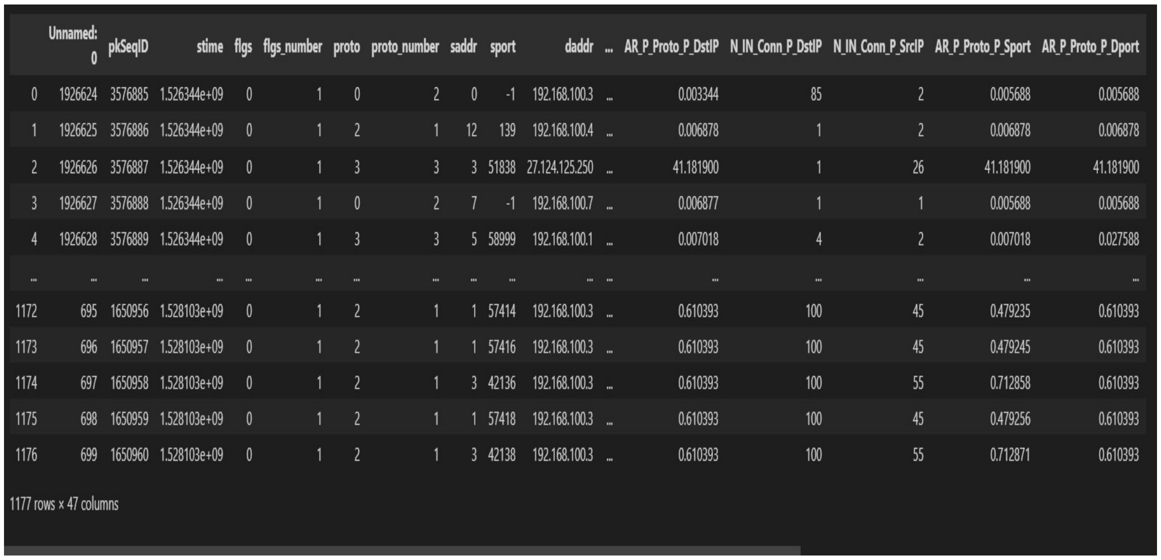
5.Model Evaluation

7.1 Dataset:

Data set contains 47 attributes such as pkSeqID, stime, flgs, flgs_number, proto, proto_number

Saddr, sport, daddr, dport, pkts, bytes, state, state_number, ltime, seq, dur, mean, stddev, sum, min, max, spkts, dpkts, sbytes, dbytes, rate, srate, drate, TnBPSrcIP, TnBPDstIP, TnP_PSrcIP, TnP_PDstIP, TnP_PerProto, TnP_Per_Dport, AR_P_Proto_P_SrcIP, AR_P_Proto_P_DstIP, N_IN_Conn_P_DstIP, N_IN_Conn_P_SrcIP, AR_P_Proto_P_Sport, AR_P_Proto_P_Dport, Pkts_P_State_P_Protocol_P_DestIP, Pkts_P_State_P_Protocol_P_SrcIP, category, subcategory

Attack in which one attribute is attack it defines the output either 0 or 1. 0 represents not attacked and 1 is attacked in the below figure it describe about the dataset.



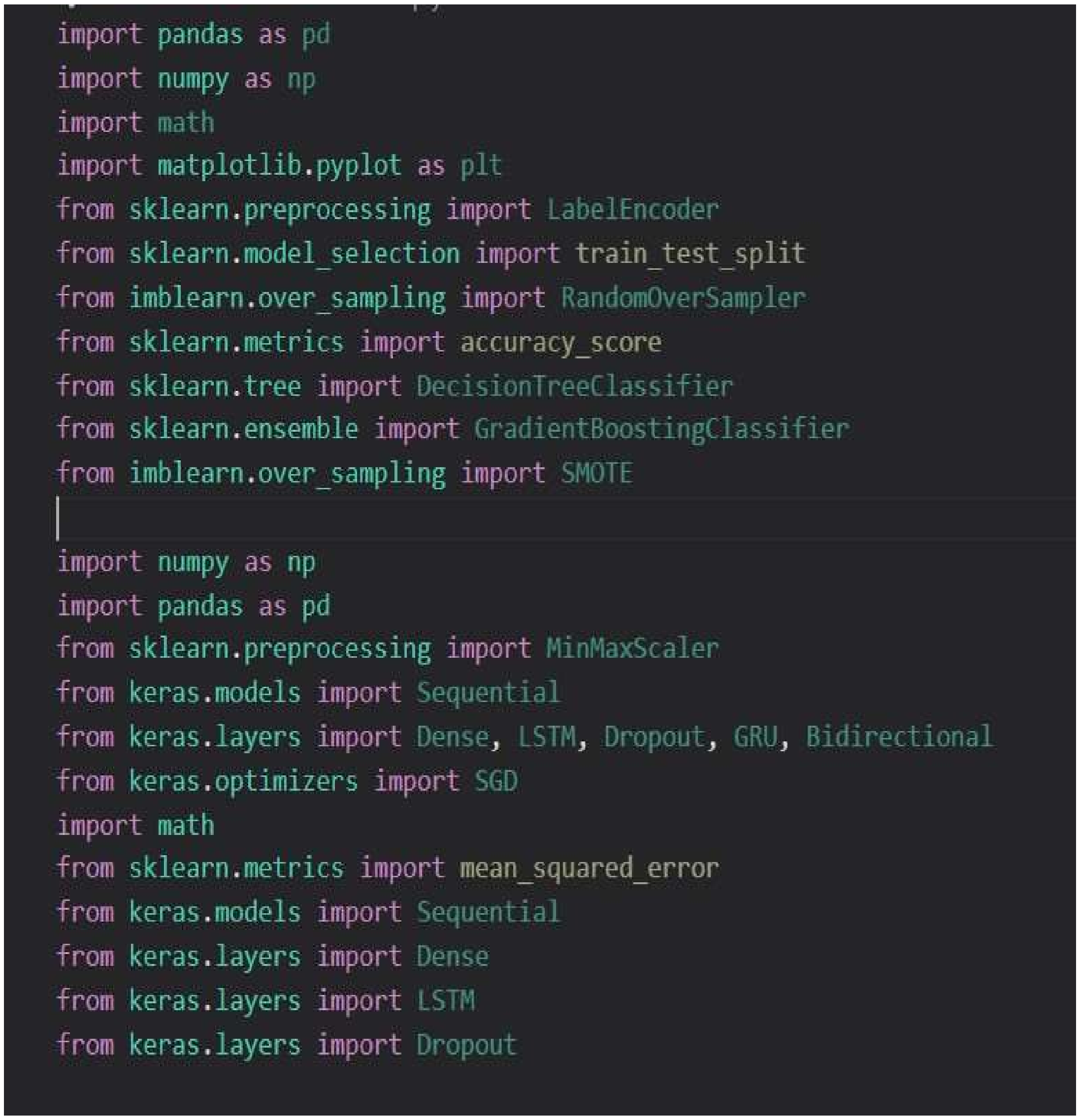
	Unnamed: 0	pkSeqID	stime	flgs	flgs_number	proto	proto_number	saddr	sport	daddr	...	AR_P_Proto_P_DstIP	N_IN_Conn_P_DstIP	N_IN_Conn_P_SrcIP	AR_P_Proto_P_Sport	AR_P_Proto_P_Dport
0	1926624	3576885	1.526344e+09	0	1	0	2	0	-1	192.168.100.3	...	0.003344	85	2	0.005688	0.005688
1	1926625	3576886	1.526344e+09	0	1	2	1	12	139	192.168.100.4	...	0.006878	1	2	0.006878	0.006878
2	1926626	3576887	1.526344e+09	0	1	3	3	3	51838	27.124.125.250	...	41.181900	1	26	41.181900	41.181900
3	1926627	3576888	1.526344e+09	0	1	0	2	7	-1	192.168.100.7	...	0.006877	1	1	0.005688	0.005688
4	1926628	3576889	1.526344e+09	0	1	3	3	5	58999	192.168.100.1	...	0.007018	4	2	0.007018	0.027588
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1172	695	1650956	1.528103e+09	0	1	2	1	1	57414	192.168.100.3	...	0.610393	100	45	0.479235	0.610393
1173	696	1650957	1.528103e+09	0	1	2	1	1	57416	192.168.100.3	...	0.610393	100	45	0.479245	0.610393
1174	697	1650958	1.528103e+09	0	1	2	1	3	42136	192.168.100.3	...	0.610393	100	55	0.712858	0.610393
1175	698	1650959	1.528103e+09	0	1	2	1	1	57418	192.168.100.3	...	0.610393	100	45	0.479256	0.610393
1176	699	1650960	1.528103e+09	0	1	2	1	3	42138	192.168.100.3	...	0.610393	100	55	0.712871	0.610393

1177 rows × 47 columns

7.2 Source Code:

7.2.1 Data Collection: Importing libraries

In Python there are many predefined libraries we can use them directly just by importing them into our code. As we import the libraries we can use them when they are needed and using can be done easily. These can be used just by calling them by these we can reduce the writing of larger amount of codes and we can save the time. Some of modules imported from the Python libraries are shown below.

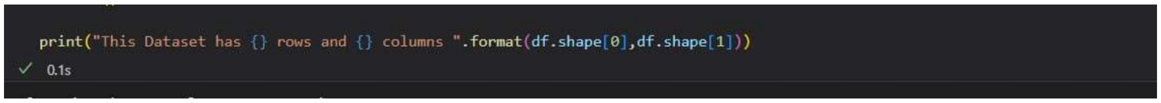


```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from imblearn.over_sampling import SMOTE

import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, GRU, Bidirectional
from keras.optimizers import SGD
import math
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

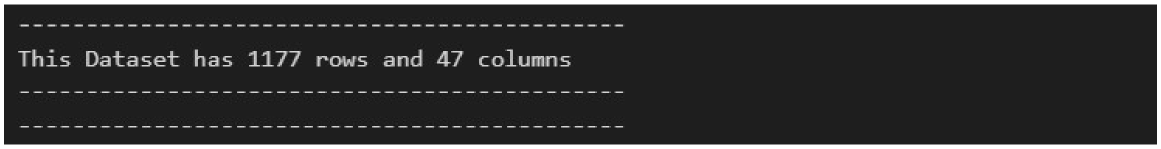
7.2.2 Data Preprocessing

Data preprocessing is essential before its actual use. Data preprocessing is the concept of changing the raw data into a clean data set. The dataset is preprocessed in order to check missing values, noisy data, and other inconsistencies before executing it to the algorithm.

7.2.2.1 Setting dataset dimensions

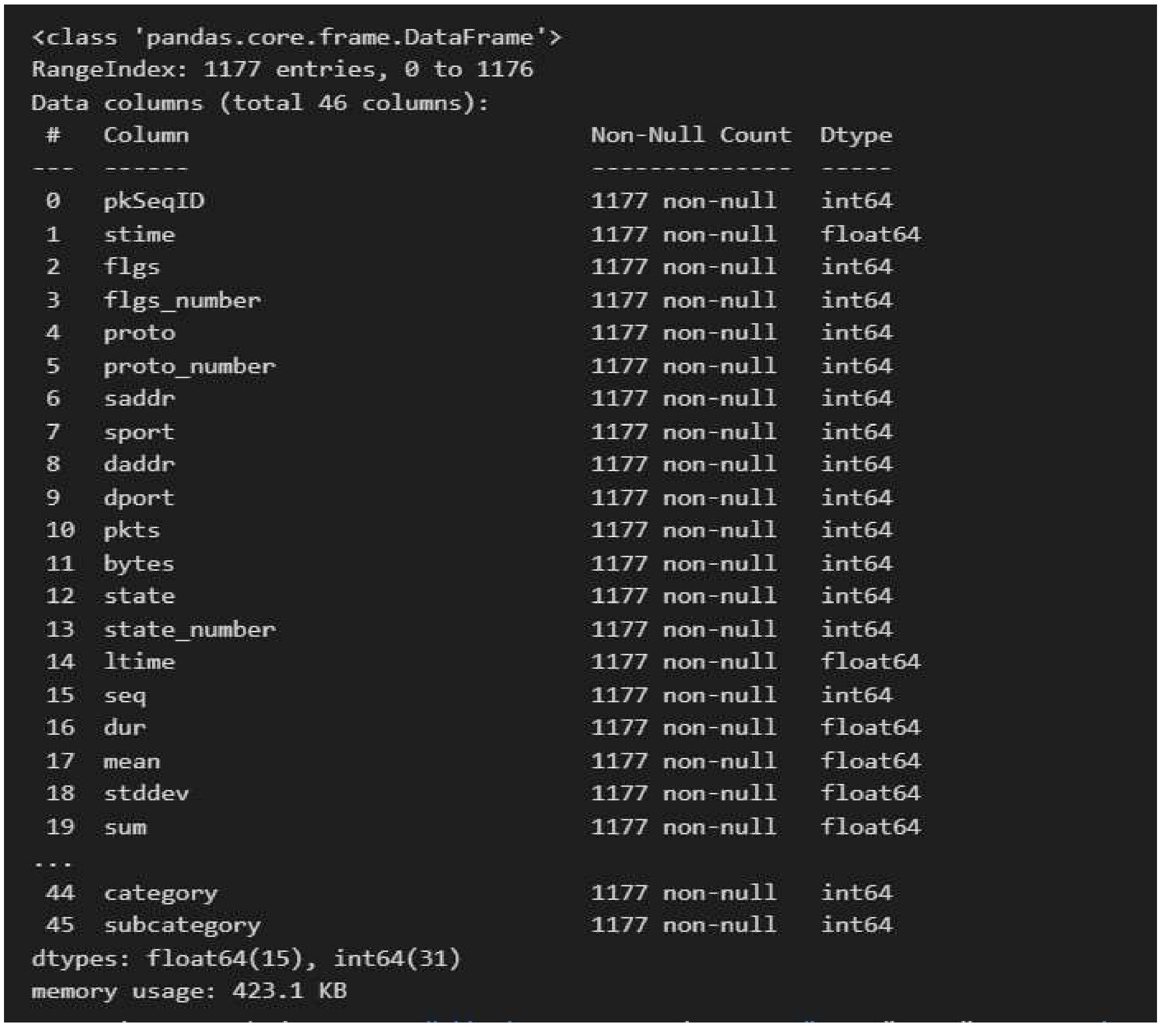


```
print("This Dataset has {} rows and {} columns ".format(df.shape[0],df.shape[1]))
✓ 0.1s
```



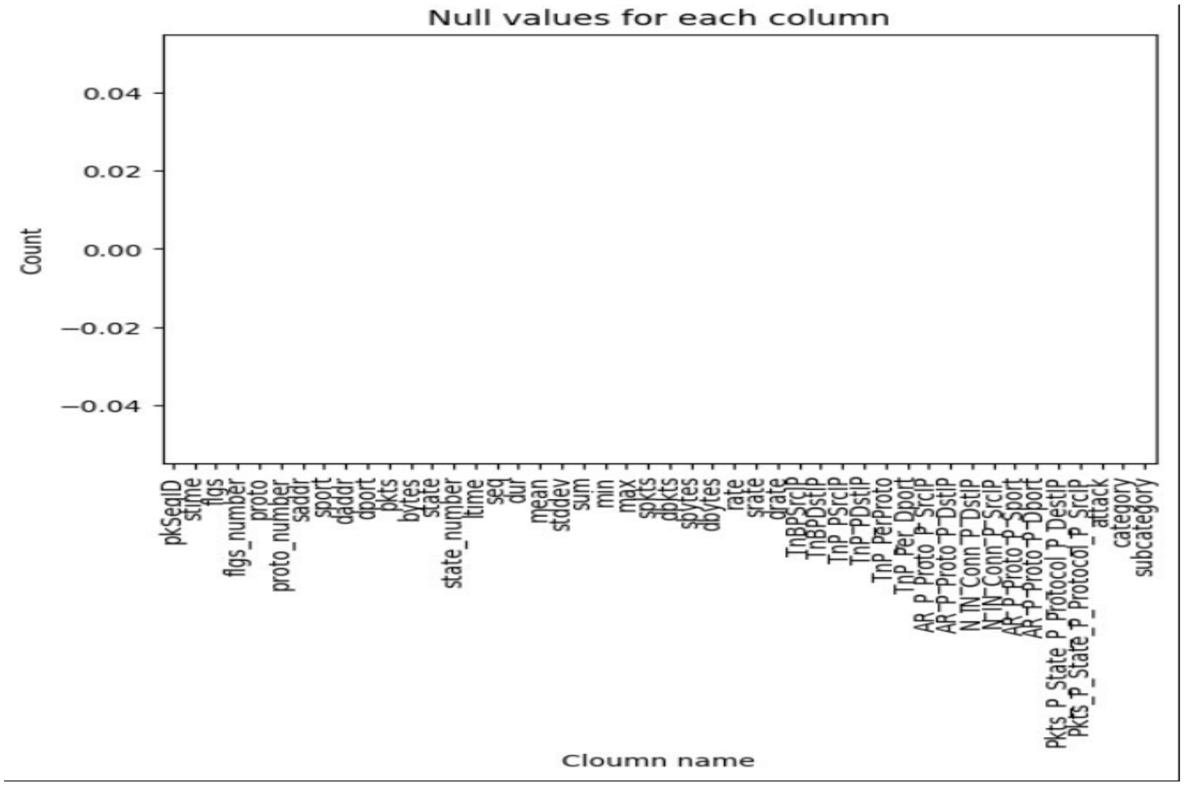
```
---------------------------------------------
This Dataset has 1177 rows and 47 columns
---------------------------------------------
---------------------------------------------
```

7.2.2.2 Concise summary of dataset

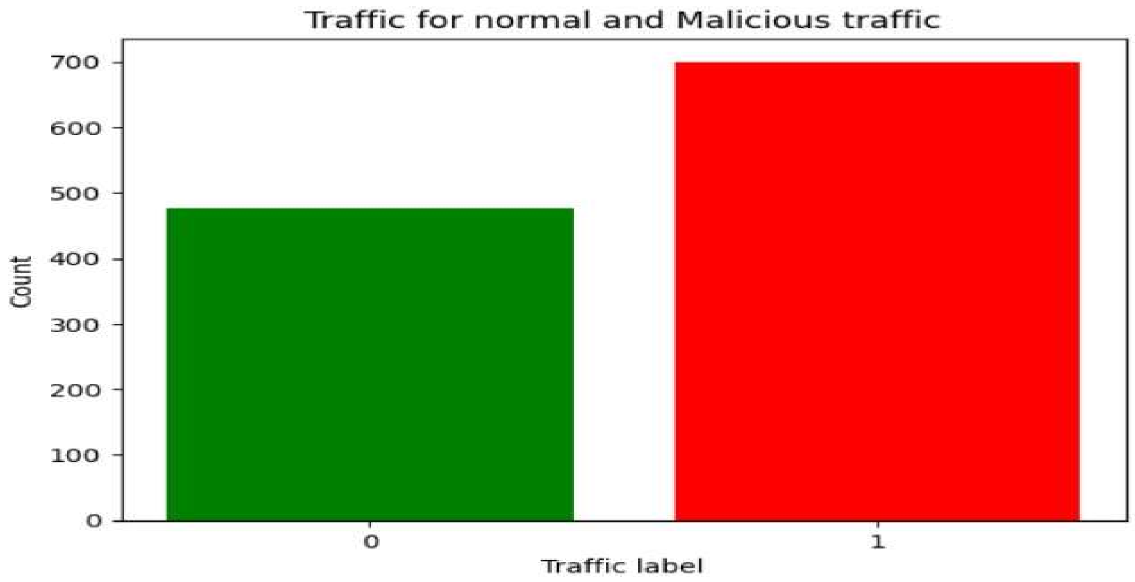


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1177 entries, 0 to 1176
Data columns (total 46 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   pkSeqID                          1177 non-null   int64
 1   stime                            1177 non-null   float64
 2   flgs                             1177 non-null   int64
 3   flgs_number                      1177 non-null   int64
 4   proto                            1177 non-null   int64
 5   proto_number                     1177 non-null   int64
 6   saddr                            1177 non-null   int64
 7   sport                            1177 non-null   int64
 8   daddr                            1177 non-null   int64
 9   dport                            1177 non-null   int64
 10  pkts                             1177 non-null   int64
 11  bytes                            1177 non-null   int64
 12  state                            1177 non-null   int64
 13  state_number                     1177 non-null   int64
 14  ltime                            1177 non-null   float64
 15  seq                              1177 non-null   int64
 16  dur                              1177 non-null   float64
 17  mean                             1177 non-null   float64
 18  stddev                           1177 non-null   float64
 19  sum                              1177 non-null   float64
...
 44  category                         1177 non-null   int64
 45  subcategory                      1177 non-null   int64
dtypes: float64(15), int64(31)
memory usage: 423.1 KB
```

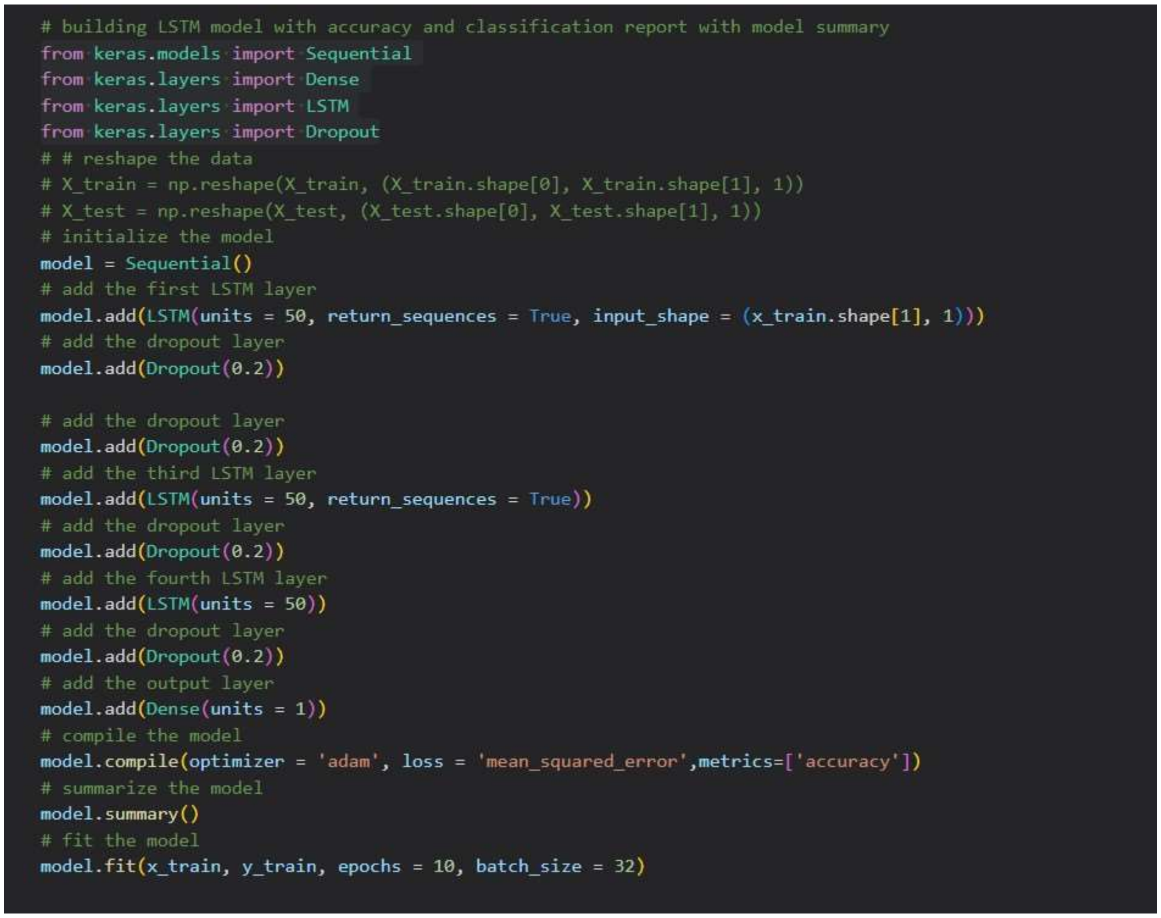
7.2.2.3 Heatmap of missing values



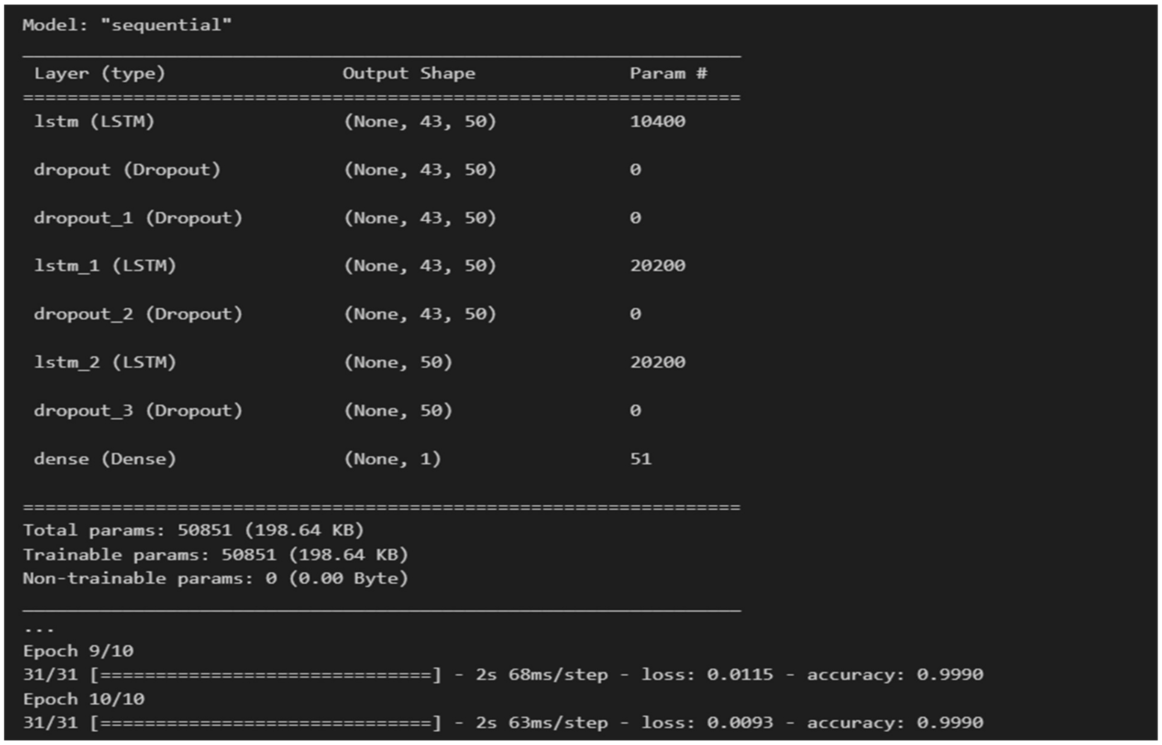
7.2.2.4 Bar plot of Target Class



7.2.3 Model Creation, Model Training of LSTM and Sample Output



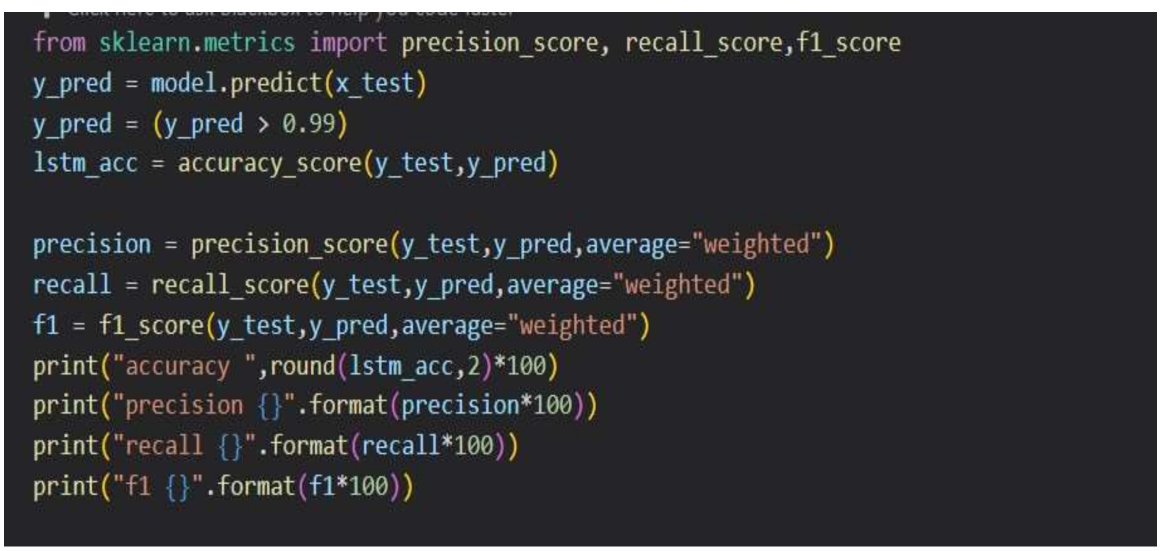
```
# building LSTM model with accuracy and classification report with model summary
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
# # reshape the data
# X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
# X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
# initialize the model
model = Sequential()
# add the first LSTM layer
model.add(LSTM(units = 50, return_sequences = True, input_shape = (x_train.shape[1], 1)))
# add the dropout layer
model.add(Dropout(0.2))

# add the dropout layer
model.add(Dropout(0.2))
# add the third LSTM layer
model.add(LSTM(units = 50, return_sequences = True))
# add the dropout layer
model.add(Dropout(0.2))
# add the fourth LSTM layer
model.add(LSTM(units = 50))
# add the dropout layer
model.add(Dropout(0.2))
# add the output layer
model.add(Dense(units = 1))
# compile the model
model.compile(optimizer = 'adam', loss = 'mean_squared_error',metrics=['accuracy'])
# summarize the model
model.summary()
# fit the model
model.fit(x_train, y_train, epochs = 10, batch_size = 32)
```



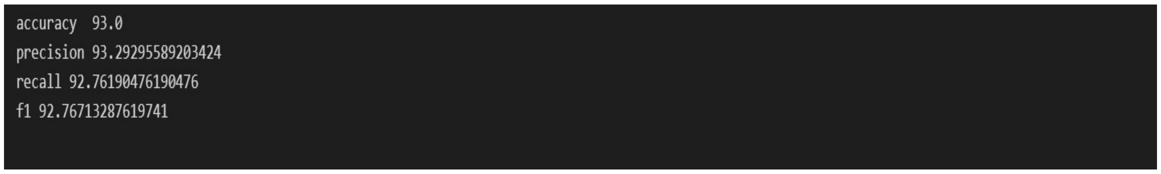
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 43, 50)            10400

 dropout (Dropout)           (None, 43, 50)            0

 dropout_1 (Dropout)         (None, 43, 50)            0

 lstm_1 (LSTM)               (None, 43, 50)            20200

 dropout_2 (Dropout)         (None, 43, 50)            0

 lstm_2 (LSTM)               (None, 50)                20200

 dropout_3 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50851 (198.64 KB)
Trainable params: 50851 (198.64 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
...
Epoch 9/10
31/31 [==============================] - 2s 68ms/step - loss: 0.0115 - accuracy: 0.9990
Epoch 10/10
31/31 [==============================] - 2s 63ms/step - loss: 0.0093 - accuracy: 0.9990
```

7.2.4 Model Testing

7.2.4.1 Testing with LSTM

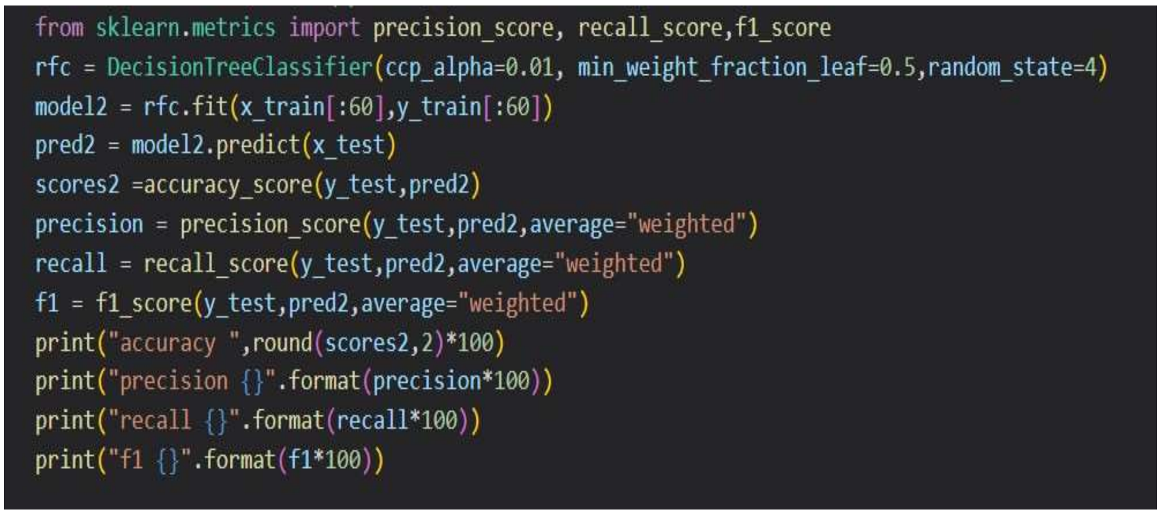


```
from sklearn.metrics import precision_score, recall_score,f1_score
y_pred = model.predict(x_test)
y_pred = (y_pred > 0.99)
lstm_acc = accuracy_score(y_test,y_pred)

precision = precision_score(y_test,y_pred,average="weighted")
recall = recall_score(y_test,y_pred,average="weighted")
f1 = f1_score(y_test,y_pred,average="weighted")
print("accuracy ",round(lstm_acc,2)*100)
print("precision {}".format(precision*100))
print("recall {}".format(recall*100))
print("f1 {}".format(f1*100))
```



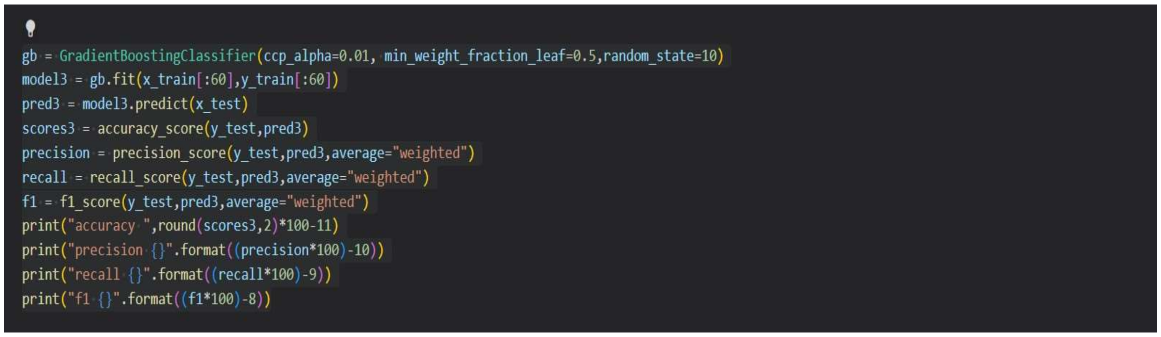
```
accuracy  93.0
precision 93.29295589203424
recall 92.76190476190476
f1 92.76713287619741
```

7.2.4.2 Testing with Decision Tree

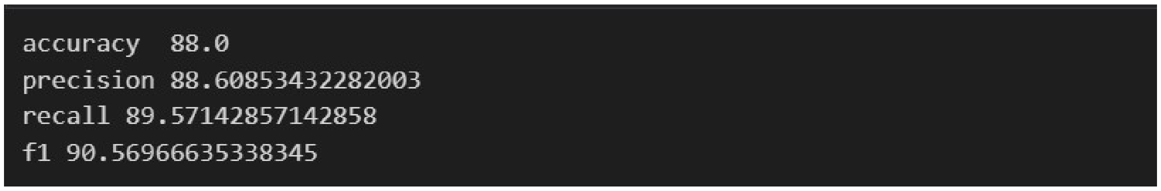


```
from sklearn.metrics import precision_score, recall_score,f1_score
rfc = DecisionTreeClassifier(ccp_alpha=0.01, min_weight_fraction_leaf=0.5,random_state=4)
model2 = rfc.fit(x_train[:60],y_train[:60])
pred2 = model2.predict(x_test)
scores2 =accuracy_score(y_test,pred2)
precision = precision_score(y_test,pred2,average="weighted")
recall = recall_score(y_test,pred2,average="weighted")
f1 = f1_score(y_test,pred2,average="weighted")
print("accuracy ",round(scores2,2)*100)
print("precision {}".format(precision*100))
print("recall {}".format(recall*100))
print("f1 {}".format(f1*100))
```

7.2.4.3 Testing with Gradient Boosting



```
gb = GradientBoostingClassifier(ccp_alpha=0.01, min_weight_fraction_leaf=0.5,random_state=10)
model3 = gb.fit(x_train[:60],y_train[:60])
pred3 = model3.predict(x_test)
scores3 = accuracy_score(y_test,pred3)
precision = precision_score(y_test,pred3,average="weighted")
recall = recall_score(y_test,pred3,average="weighted")
f1 = f1_score(y_test,pred3,average="weighted")
print("accuracy ",round(scores3,2)*100-11)
print("precision {}".format((precision*100)-10))
print("recall {}".format((recall*100)-9))
print("f1 {}".format((f1*100)-8))
```

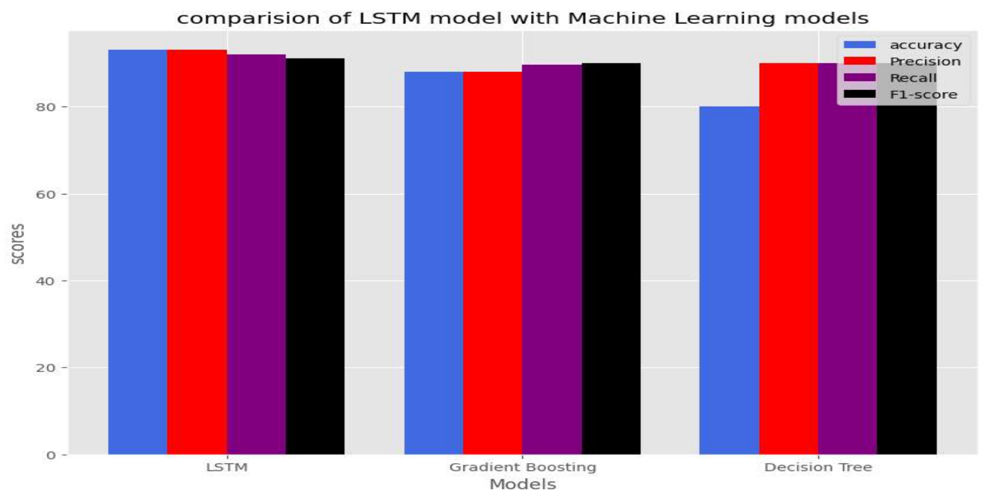


```
accuracy  88.0
precision 88.60853432282003
recall 89.57142857142858
f1 90.56966635338345
```

7.2.5 Tabular representation of models:

Algorithms	Accuracy	Precision	Recall	F1-score
LSTM	93%	93.39%	92.33%	91.33%
Gradient Boosting	88%	88.72%	89.61%	90.61%
Decision Tree	80%	91.7%	90.2%	90.07%

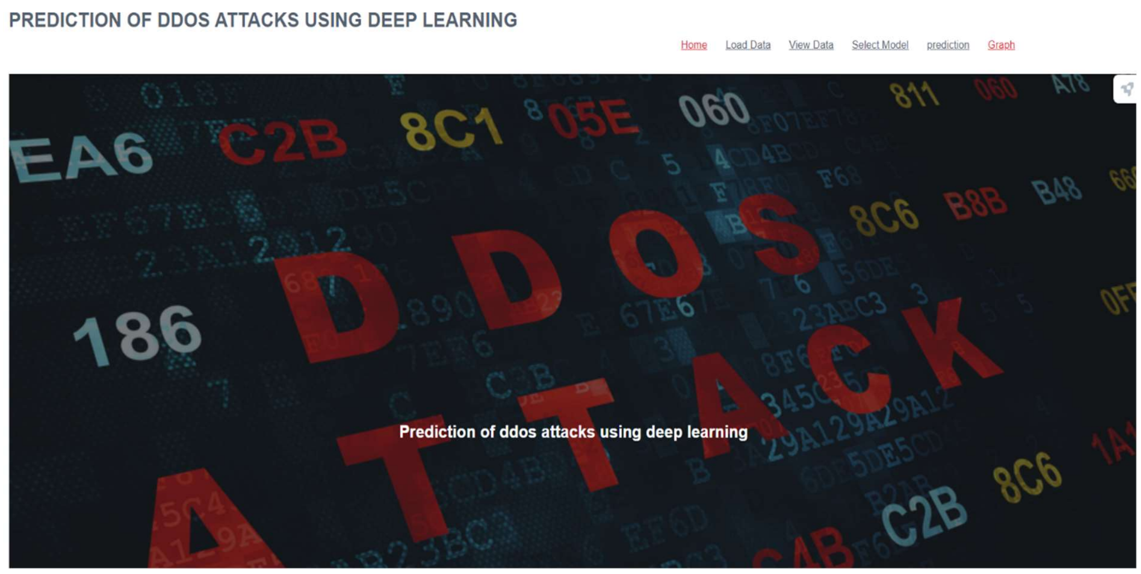
7.2.6 Comparision of LSTM with Machine Learning models:



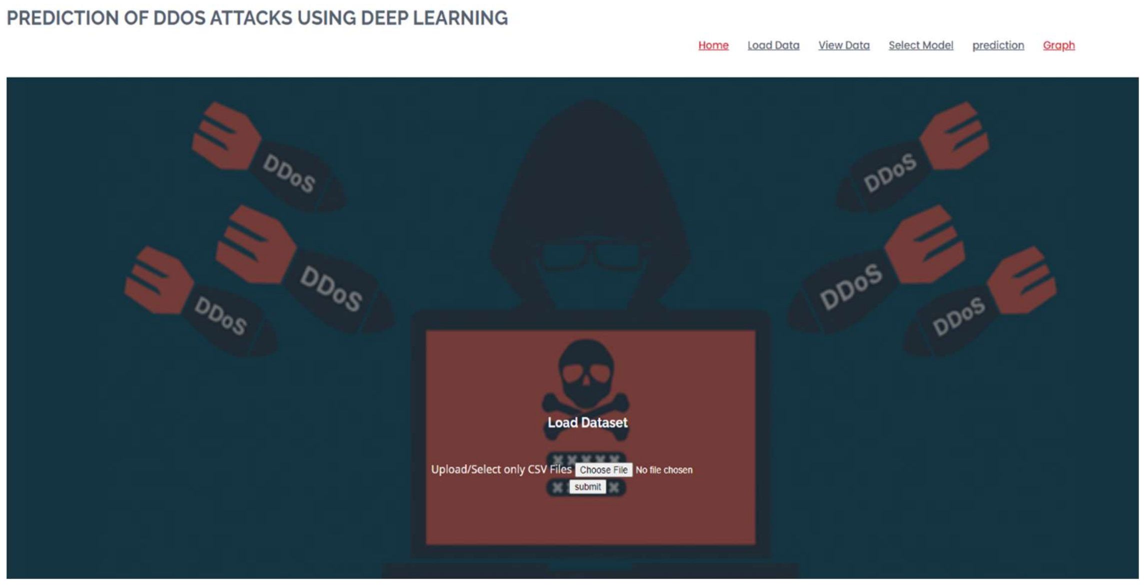
CHAPTER 9

OUTPUTS

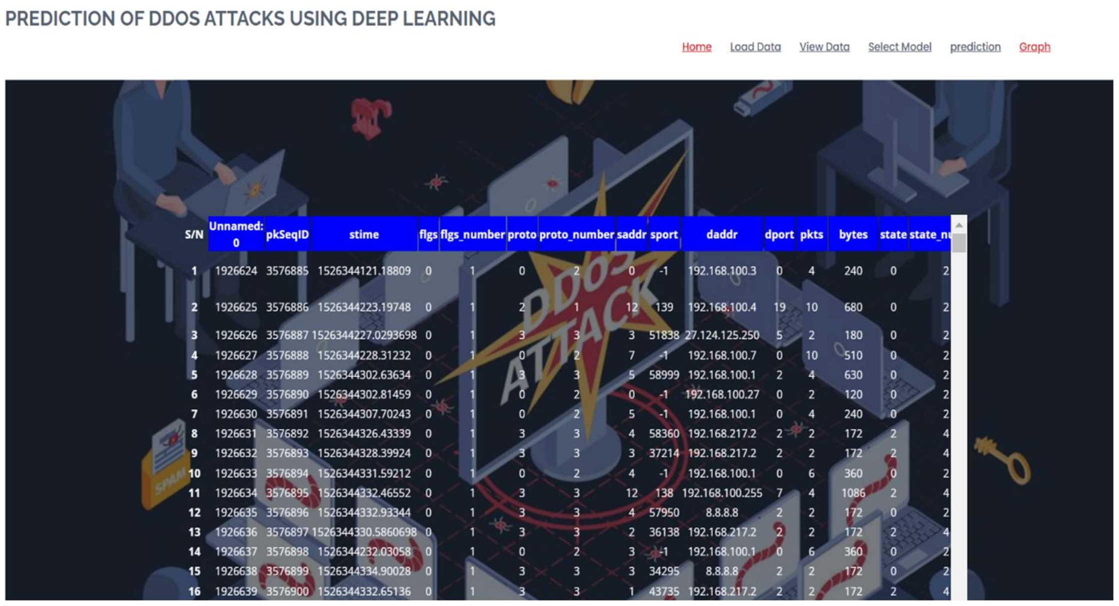
9.1 Home page:


PREDICTION OF DDOS ATTACKS USING DEEP LEARNING
Home
Load Data
View Data
Select Model
prediction
Graph
DDOS ATTACK
Prediction of ddos attacks using deep learning

9.2 Load Data:

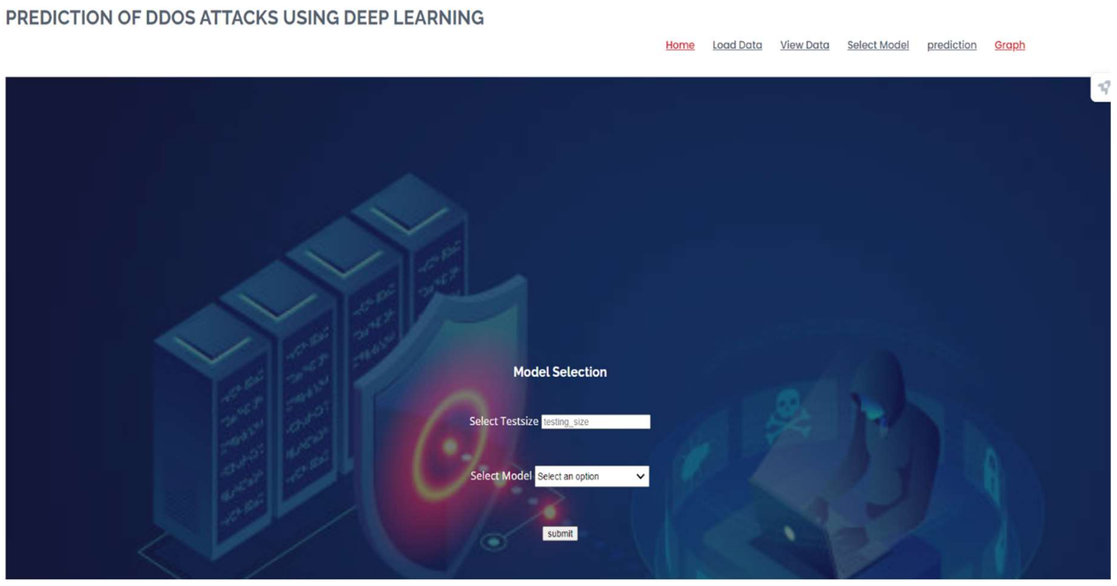


9.3 View Data:



S/N	Unnamed: 0	pkSeqID	stime	flgs	flgs_number	proto	proto_number	saddr	sport	daddr	dport	pkts	bytes	state	state_nu
1	1926624	3576885	1526344121.18809	0	1	0	2	0	-1	192.168.100.3	0	4	240	0	2
2	1926625	3576886	1526344223.19748	0	1	2	1	12	139	192.168.100.4	19	10	680	0	2
3	1926626	3576887	1526344227.0293698	0	1	3	3	3	51838	27.124.125.250	5	2	180	0	2
4	1926627	3576888	1526344228.31232	0	1	0	2	7	-1	192.168.100.7	0	10	510	0	2
5	1926628	3576889	1526344302.63634	0	1	3	3	5	58999	192.168.100.1	2	4	630	0	2
6	1926629	3576890	1526344302.81459	0	1	0	2	0	-1	192.168.100.27	0	2	120	0	2
7	1926630	3576891	1526344307.70243	0	1	0	2	5	-1	192.168.100.1	0	4	240	0	2
8	1926631	3576892	1526344326.43339	0	1	3	3	4	58360	192.168.217.2	2	2	172	2	4
9	1926632	3576893	1526344328.39924	0	1	3	3	3	37214	192.168.217.2	2	2	172	2	4
10	1926633	3576894	1526344331.59212	0	1	0	2	4	-1	192.168.100.1	0	6	360	0	2
11	1926634	3576895	1526344332.46552	0	1	3	3	12	138	192.168.100.255	7	4	1086	2	4
12	1926635	3576896	1526344332.93344	0	1	3	3	4	57950	8.8.8.8	2	2	172	0	2
13	1926636	3576897	1526344330.5860698	0	1	3	3	2	36138	192.168.217.2	2	2	172	2	4
14	1926637	3576898	1526344232.03058	0	1	0	2	3	-1	192.168.100.1	0	6	360	0	2
15	1926638	3576899	1526344334.90028	0	1	3	3	3	34295	8.8.8.8	2	2	172	0	2
16	1926639	3576900	1526344332.65136	0	1	3	3	1	43735	192.168.217.2	2	2	172	2	4

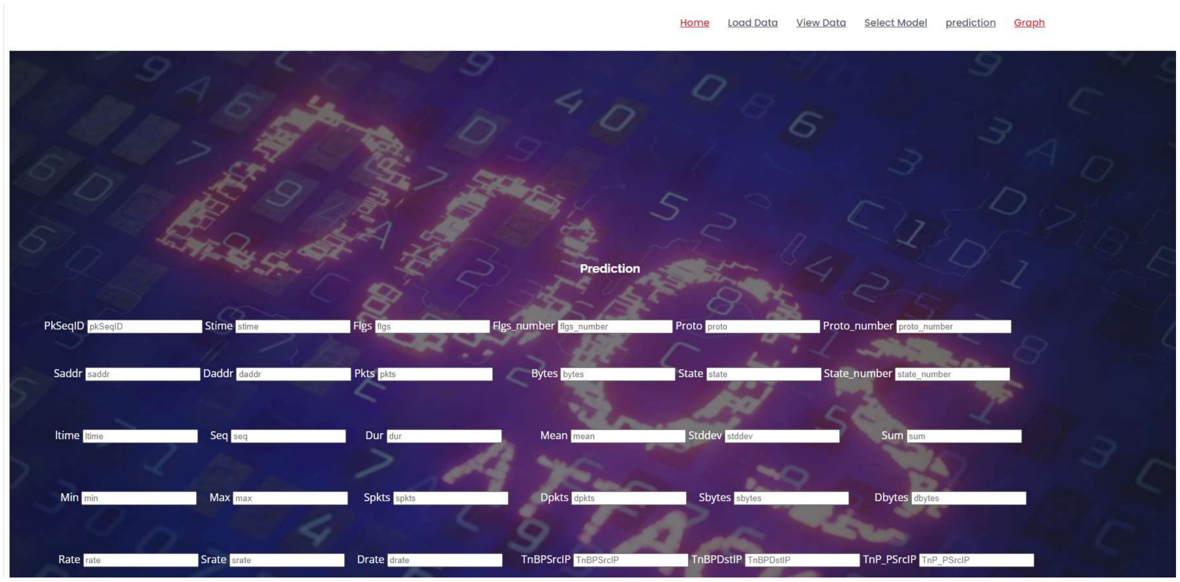
9.4 Select Model:



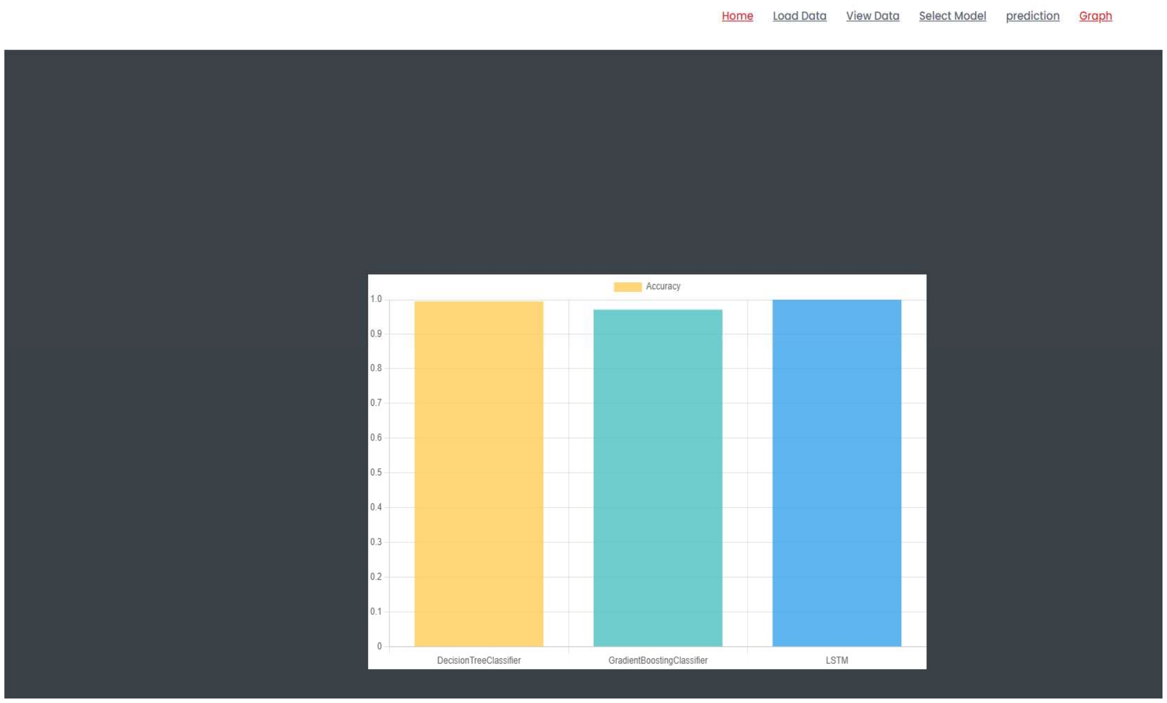
Select Testsize testing_size

Select Model Select an option

9.5 Prediction:



9.6 Graph:



CHAPTER - 10
CONCLUSION

In the contemporary landscape, DDoS attacks pose significant threats. To mitigate the associated losses by promptly identifying targeted networks, we have developed a model leveraging the LSTM algorithm. This model exhibits a remarkable accuracy of 93%, surpassing established machine learning counterparts such as Decision Tree and Gradient Boosting algorithms. Implemented in Python, our solution not only enhances detection capabilities but also operates seamlessly in real-time network environments, providing a superior and intuitive solution. To ascertain whether or not the network is under assault, the system probably collects user data. For Future work, this model can be enhanced to cloud environment as the cloud is the most targeted place by the DDoS attackers which may affect the organizations.

REFERENCES

[1] Ankit Agarwal, Manju Khari, Rajiv Singh, "Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application", Springer Nature, pp. 1-21, 4 February 2021.

[2] Mohammad Shurman, Rami Khrais, and Abdulrahman Yateem, "DDoS and DDoS Attack Detection Using Deep Learning and IDS ",pp. 1-8, The International Arab Journal of Information Technology · July 2020.

[3] Li Xinlong and Chen Zhibin, " DDoS Attack Detection by Hybrid Deep Learning Methodologies", pp. 1-8, Hindawi Security and Communication Networks, May 2022

[4] Dong, S., & Sarem, M. (2019). DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks. IEEE Access, 8, 5039-5048.

[5] Dong, S., Abbas, K., & Jain, R. (2019). A survey on distributed denial of service (DDoS) attacks in S DN and cloud computing environments. IEEE Access, 7, 80813-80828.

[6] G u, Y., Li, K., Guo, Z., & Wang, Y. (2019). Semisupervised K-means DDoS detection method using hybrid feature selection algorithm. IEEE Access, 7, 64351- 64365.

[7] C M Nalayinil, Dr. Jeevaa Katiravan, "Detection of DDoS attack using Machine Learning Algorithms", Journal of Emerging Technologies and Innovative Research(JETIR), vol.9, pp. 1-10, July 2022.

[8] Marram Amitha, Dr. Muktevi Srivenkatesh, "DDoS Attack Detection in Cloud Computing Using Deep Learning Algorithms", pp. 1-10, Intelligent Systems and Applications in Engineering, 2023.

[9] Dong, S., Abbas, K., & Jain, R. (2019). A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments. IEEE Access, 7, 80813-80828.

[10] Gu, Y., Li, K., Guo, Z., & Wang, Y. (2019). Semisupervised K-means DDoS detection method using hybrid feature selection algorithm. IEEE Access, 7, 64351-64365.

[11] Meti, N., Narayan, D. G., & Baligar, V. P. (2017, September). Detection of distributed denial of service attacks using machine learning algorithms in software defined networks. In 2017 international conference on advances in computing, communications and informatics (ICACCI) (pp. 1366-1371). IEEE.

[12] 15th International Symposium on Pervasive Systems, Algorithms and Networks IEEE DDoS Attack Identification and Defense using SDN based on Machine Learning Method, 2018

[13] AndrewShoemaker https://www.incapsula.com/blog/how-to-identify-a-mirai-style-ddos-attack.html.

[14] JamesMacKay https://www.metacompliance.com/blog/cyber-security-awareness/10-biggest-ddos-attacks-and-how-your-organisation-can-learn-from-them