EFFECTIVE INTRUSION DETECTION SYSTEM USING HYBRID ENSEMBLE METHOD FOR CLOUD COMPUTING

A PROJECT REPORT

Submitted by,

ARAVENTH M

GANGARAJU THARUN

SOURABH SONNY

RAJ KUMAR J

in partial fulfillment for the award of the degree
of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING GUINDY

ANNA UNIVERSITY CHENNAI 600 025

MAY & 2023

ANNA UNIVERSITY CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“EFFECTIVE INTRUSION DETECTION SYSTEM USING HYBRID ENSEMBLE METHOD FOR CLOUD COMPUTING”** is the bonafide work of **“ARAVENTH M, GANGARAJU THARUN, SOURABH SONNY, RAJ KUMAR J”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
DR. S VALLI **HEAD OF THE DEPARTMENT**	**DR. S BOSE** **PROFESSOR**
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
ANNA UNIVERSITY CHENNAI 600 025	ANNA UNIVERSITY CHENNAI 600 025

TABLE OF CONTENTS

ABSTRACT

Due to its adaptability and pay-per-use services, cloud computing has grown in popularity among businesses, but security and privacy issues are still very much present. Intruders can exploit vulnerabilities in the open and dispersed nature of cloud environments, leading to attacks that can damage entire projects within a short period of time. To address this issue, organizations need to implement effective intrusion detection systems (IDS) that can detect and alert administrators of any suspicious activities. There are three widely used methods for IDS: signature-based detection, anomaly-based detection, and hybrid detection. Hybrid detection, which combines the strengths of signature-based and anomaly-based detection, has been shown to produce superior results. IDS can be categorized into host-based IDS (HIDS), network-based IDS (NIDS), hypervisor-based IDS, and distributed IDS (DIDS), each with their own unique characteristics and benefits. The CICIDS2017 dataset provides a diverse set of attacks and benign traffic for researchers and practitioners to develop and evaluate IDS systems. Overall, putting in place a strong intrusion detection system is critical for maintaining the security and privacy of cloud-based projects, as well as ensuring their availability.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	DESCRIPTION
VM	Virtual Machine
HIDS	Host Intrusion Detection System
NIDS	Network Intrusion Detection System
DIDS	Distributed Intrusion Detection System
CIDS	Cloud Intrusion Detection System
CICIDS2017	Canadian Institute for Cybersecurity Intrusion Detection Dataset 2017
CICFlowmeter	Canadian Institute for Cybersecurity Flowmeter
DOS	Denial of Service
DDOS	Distributed Denial of Service
PSO	Particle Swarm Optimization
SCAE	Stacked Contractive Autoencoder
DT	Decision Tree
LSTM	Long Short Term Memory
RMSE	Root Mean Square Error

CHAPTER 1

INTRODUCTION

Cloud computing seems to be one of the most emerging technologies in recent years. Despite the fact that there are now a lot more cloud projects than there were a few years ago, it is still important and difficult to conduct research on how to guarantee the security and availability of project information, offerings, and resources. Attacks that are generated to hack the cloud have the ability to deplete entire cloud resources, get information about its bandwidth, and ruin an entire cloud program. Since every cloud platform offers flexibility, pay to go services, it is the preferred option for every organization. Even so, security and privacy are important challenges to its success owing to its open and distributed design, which is vulnerable to infiltration. Cloud computing is made up of three underlying layers: the application layer, the environment layer, and the system layer. Whereas the **last two layers are concerned with virtual machines (VM) and operating systems, the first layer covers cloud applications** such as web apps that are connected with databases. The cloud architecture is made up of two distinct components.

One of these is cloud security, which the majority of organizations view as a major impediment to cloud adoption. This is because the cloud environment is open and completely scattered, which increases its susceptibility to security risks and flaws. As a result, hackers are incited to prepare potential attacks against the cloud or its associated devices. So, in order to safeguard the data, an intrusion detection system is required.

An intrusion detection system (IDS) is a type of technology used to identify harmful behavior on a computer or network. It may be used to keep an eye out for unusual activity on the network or cloud and notify administrators when it is found. The system may also take preventive measures to protect the network from further damage. There are three widely used methods**: hybrid detection, anomaly detection**, **and signature detection**. Signature detection, sometimes referred to as misuse detection, locates intrusions by comparing the obtained data with a database of patterns of well-known attacks or a pre-defined set of criteria. The latter method's primary flaw is that it can only identify known attacks. Anomaly-based detection, on the other hand, finds intrusions by comparing the gathered data to a predetermined baseline. When behavior deviates from the norm, it raises the warning that a hostile assault may be imminent. The ability to identify both known and unidentified assaults is the main benefit of anomaly detection. Signature and anomaly detection are combined in a hybrid detection approach. According to reports, CIDS that employ hybrid detection produce superior results than those produced by conventional approaches. The traffic for the entire network will be captured by **networks-based IDSs (NIDS)**, which will then analyse it to find any potential intrusions like port scanning or occasionally DoSS attacks. The NIDS generally performs such intrusion detection by swiftly reading all of the captured system packets' IP and network layer headers. Following the collection of the network packets, a correlation between all the signatures for the multiple known assaults is discovered. This correlation is then utilised to compare user activity with known profiles. The abnormality will be used in this way. We also need to know the location of its NIDS, which is normally concealed from its attacker if this has to be executed in stealth mode. The network has a number of active hosts that are protected from all intruders by properly implemented NIDS. When the traffic is encrypted, the NIDS is unable to conduct any analysis.

Hypervisor-based IDS will be deployed in the hypervisor layer, and the hypervisor will offer various levels of interaction with the VMs. This aids in the examination of the information that is already accessible for spotting unusual user behavior. Such information has relied on multi-level communication within a virtual network powered by a hypervisor. **Distributed IDS (DIDS)** is a group of IDSs like NIDS or HIDS that are installed in a network to analyze traffic for invasive detection behavior. The correlation manager will get the collected data in a 5-standard format from the detection component after it has examined the system's behavior. The correlation manager will aggregate the information from several IDS to provide a high-level warning that keeps track of an attack.

The **CICIDS2017 dataset** provides a realistic simulation of a real-world network environment with a diverse set of attacks and benign traffic. It is widely used by researchers and practitioners to create and assess intrusion detection systems and to advance the field of network security. The dataset contains both benign traffic and various types of attacks, including **DoS (Denial of Service), DDoS (Distributed Denial of Service), Port Scan, Botnet, Brute Force, Infiltration, and Web Attack**. The attacks were generated using various tools and techniques commonly used by attackers to compromise or disrupt network systems.

1.1 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 algorithm examines enormous amounts of data to identify patterns and detect potential threats before they can cause any 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 identifying and avoiding cybersecurity hazards, the intrusion detection system (IDS) aids in reducing the risk of data theft, financial loss, and 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.

1.2 PROBLEM STATEMENT

Cloud computing is often used by organizations to store data and applications, so it is very important to make sure that the platform is safe or not. While the cloud environment offers many benefits, there are also several security challenges that organizations face when using cloud services. Data breaches are a significant concern for cloud service providers, as sensitive data stored in the cloud can be vulnerable to theft or unauthorized access. **DoS attacks can cause significant disruptions** to cloud services by overloading the system with traffic. This can prevent legitimate users from accessing cloud resources, causing significant financial and reputational damage to the cloud service provider. Most of the datasets used to train the IDS are not large datasets, so **there should be the use of large datasets in the training of the intrusion detection system.**

An intrusion detection system (IDS) is an important security tool in cloud environments that can detect and alert users to potential security breaches or threats. IDS can examine network data, system logs, and user activity to detect and identify particular security incidents or attacks. There is a **constant need for optimization of algorithms and enhancement in IDS due to the increasing threats of attacks on the cloud.**

1.3 ORGANIZATION OF THESIS

In chapter 2, some of the works related to intrusion detection using various algorithms are discussed. In chapter 3, the overall architecture is explained briefly. All the modules in the project are explained with flowcharts and algorithms. In chapter 4, all the implementation details are provided with corresponding screenshots. Model evaluation, performance metrics analysis and Results are discussed briefly in chapter 4. Future enhancements are explained and conclusion is provided in chapter 5.

CHAPTER 2

SUMMARY OF RELATED WORK

Cloud Intrusion Detection Method Based on Stacked Contractive Auto-Encoder and Support Vector Machine

W. Wang et al. developed a hybrid system that combines the SVM classification method for identifying hazardous assaults with a stacked contractive auto-encoder (SCAE) for feature extraction. By comparing the proposed SCAE+SVM-IDS model to three cutting-edge techniques, they empirically showed that in six measures, it provides excellent results for classification. They did this by utilizing the NSL-KDD and KDD Cup 99 intrusion detection datasets.

An anomaly detection method to detect web attacks using Stacked Auto-Encoder

A. M. Vartouni et al. employs the SAE technique in order to extract meaningful characteristics from HTTP request logs for anomaly detection in web apps. As a one-class learner, Isolation Forest was used to distinguish between attack and normal data. The findings show that deep learning in general and deep models perform differently with various SAE structures.

Feature Selection and Intrusion Detection in Cloud Environment Based on Machine Learning Algorithms

A. Javadpour et al. proposed a unique technique based on intrusion detection systems and their varied designs to improve the efficacy of intrusion detection in cloud computing. The two methods utilized in this article were Pearson Linear Correlation and Mutual Information; plugins and irrelevant features were removed. In this paper, the linear correlation and mutual information feature selection strategies were combined to evaluate the proposed strategy using the KDD99 database. The data was classified using a variety of methods, including decision trees, random forests, the CART algorithm, and neural networks.

A review on intrusion detection techniques for cloud computing and security challenges

G. Kene et al. proposed the various intrusions that compromise the cloud's availability, secrecy, and integrity. They have provided a thorough description of the various IDS types utilized in cloud environments. They have supplied an overview of intrusion detection methods in the way of charts and tables, which make it simple to comprehend the entire cloud computing environment. They came to the conclusion that while several IDS techniques have been suggested and have helped in the detection of intrusions in the cloud, they don’t provide total protection.

Intrusion Detection in Cloud Computing

M. Ficco et al. proposed a methodology for creating distributed intrusion detection systems in cloud computing is proposed. It is an open-source solution that enables the creation and deployment of security probes on the virtual assets and cloud infrastructure of the customer. The implemented architecture serves as the initial iteration of a cloud-based IDS management system.

Proxy Network Intrusion Detection System for cloud computing

U. Oktay et al. proposed a proxy NIDS design that performs network intrusion detection operations through the gateway. It aims to make it simple for customers and service providers to choose where to locate their protection mechanism in this way. The Proxy Network Intrusion Detection System solution allows providers to place NIDS on mesh topology gateways. As a result, providers will seek to secure their infrastructure with less resource consumption, in addition to users choosing this strategy to utilize and pay a lower value of resources.

CIDS: A Framework for Intrusion Detection in Cloud Systems

H. A. Kholidy et al. proposed a framework for "CIDS," a cloud intrusion detection system in order to address the shortcomings of existing IDSs. To analyze the alarms and tell the cloud administrator, CIDS additionally offers a component. The CIDS architecture lacks a central coordinator and is elastic and scalable. The advantages of CIDS are discussed in this study, along with its components, architecture, and detection models.

Genetic Algorithm Based Feature Selection Algorithm for Effective Intrusion Detection in Cloud Networks

A. Kannan et al. proposed a novel genetically based feature selection method for cloud networks is presented in this paper. Using the KDD cup data set, this method is used to determine the ideal amount of attributes for intrusion detection. The KDD Cup dataset has been used to successfully classify intrusions using a system for intrusion detection that uses this feature selection technique and then applies the current Fuzzy SVM to protect cloud networks.

A parallel technique for improving the performance of signature-based network intrusion detection system

B. Shanmugam et al. proposed the parallel approach presented that was recommended to improve the performance of signature, network intrusion detection systems. The use of an effective string-matching algorithm, hardware acceleration, and finally parallelism have all been suggested in the past because a subpar passive system misses many attacks and drops many packets on a high-speed network.

A Cooperative Intrusion Detection System Framework for Cloud Computing Networks

C. -C. Lo et al. proposed a collaborative intrusion detection method for networks using cloud computing in order to lessen the effects of DoS attacks. By doing this, cooperative IDS alerts other IDS systems if a DoS attack occurs in one of the cloud computing zones. The same assault sent by one IDS could be gathered by another IDS. The reliability of this alarm message is then assessed using the majority vote approach. The suggested system prevents the failure in the IDS system. Early detection and preventive methods are implemented by these agents working together. IDSs placed in cloud computing zones, apart from the victim one, could therefore stop this kind of attack from happening.

CHAPTER 3

SYSTEM DESIGN

3.1 ARCHITECTURE DIAGRAM

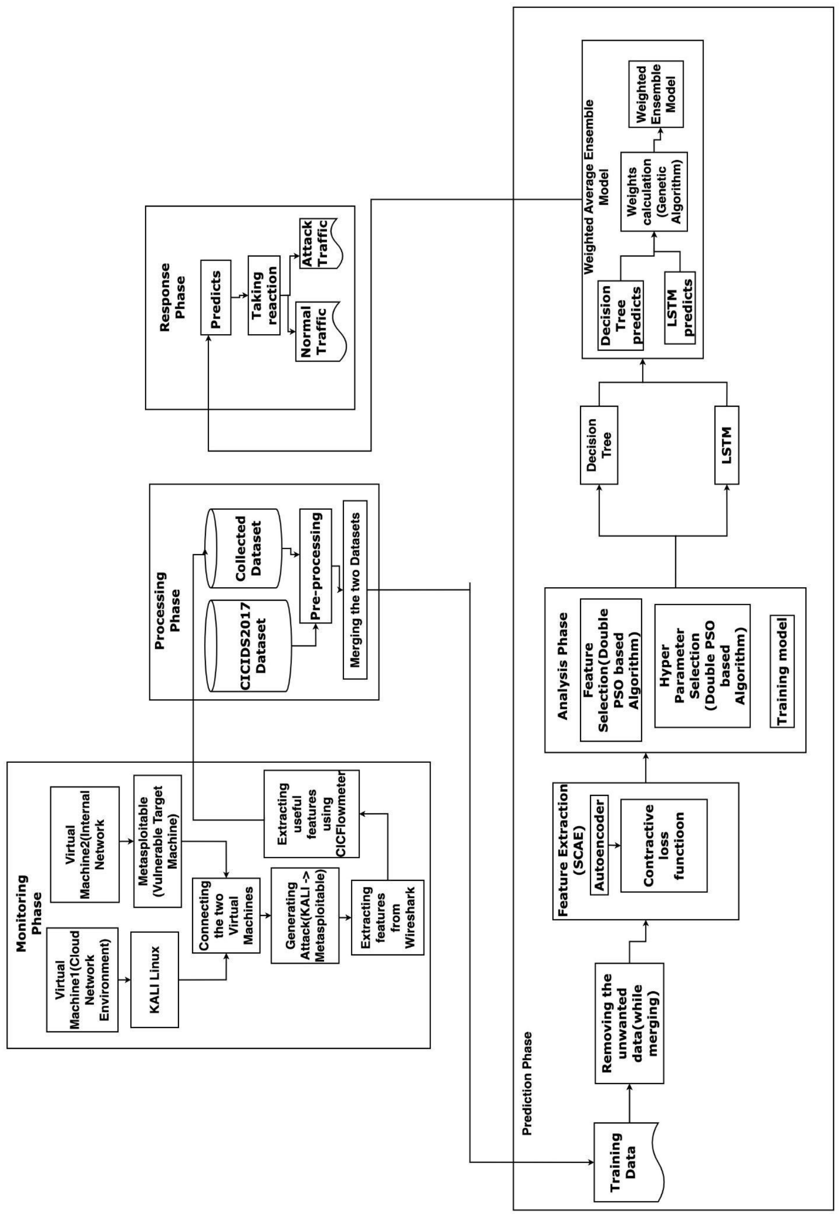

Monitoring Phase
Virtual Machine2(Internal Network
Metasploitable (Vulnerable Target Machine)
Virtual Machine1(Cloud Network Environment)
KALI Linux
Connecting the two Virtual Machines
Generating Attack(KALI -> Metasploitable)
Extracting features from Wireshark
Extracting useful features using CICFlowmeter
Processing Phase
CICIDS2017 Dataset
Collected Dataset
Pre-processing
Merging the two Datasets
Response Phase
Predicts
Taking reaction
Normal Traffic
Attack Traffic
Prediction Phase
Training Data
Removing the unwanted data(while merging)
Feature Extraction (SCAE)
Autoencoder
Contractive loss functioon
Analysis Phase
Feature Selection(Double PSO based Algorithm)
Hyper Parameter Selection (Double PSO based Algorithm)
Training model
Decision Tree
LSTM
Weighted Average Ensemble Model
Decision Tree predicts
LSTM predicts
Weights calculation (Genetic Algorithm)
Weighted Ensemble Model

Fig 3.1 ARCHITECTURE DIAGRAM

The Fig 4.1 is a hybrid IDS that uses both real time data and the CICIDS2017 dataset for training the Models. The Models employed here is an ensemble model combining ML and DL algorithms. As the **Monitoring phase** captures the network traffic in the host machine using CIC Flowmeter, the features are captured and preprocessed in the **Processing phase** such that we are able to detect any known attacks and the rest are labeled as unknown. These known traffics are analyzed in **Analysis phase** using double PSO algorithms and the type of attacks are detected. While the unknown traffic is sent to the **Prediction phase** where an ensemble model is employed to predict the type of traffic or attack. The result of the ensemble model is sent to the **Response phase** where the new unknown traffic is stored in the main database after prediction.

3.2 DETAILS OF MODULE DESIGN

This section represents the list of modules and its details:

- Monitoring phase
- Processing phase
- Analysis phase
- Prediction phase
- Response phase

3.2.1 MONITORING PHASE

The monitoring module's initial job is to capture all incoming and departing data packets transiting the cloud network. The monitoring module analyzes network traffic to help in this operation. Furthermore, the monitoring module **may collect data packets** from various virtual machines 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.

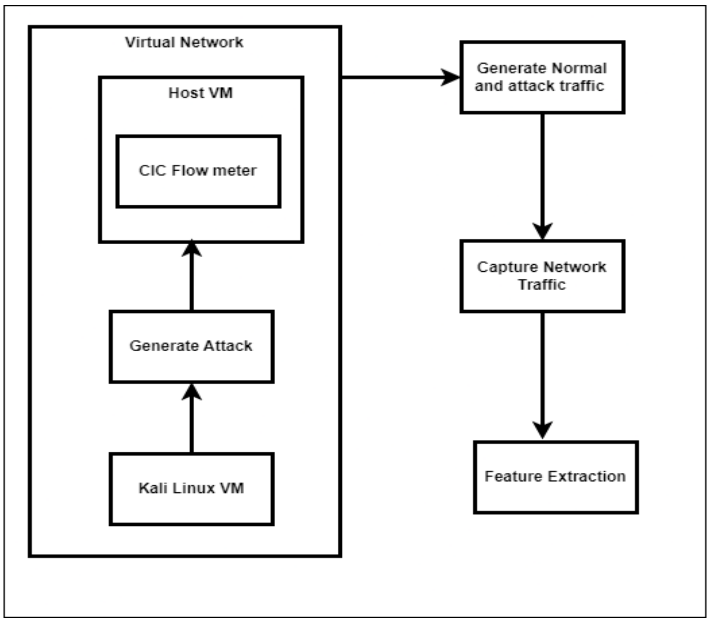


Fig 3.2 MONITORING PHASE

Fig 3.2 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.

Output: Data packets of real-time traffic

Setup of Monitoring Phase:

1. *Private Cloud is set with multiple Virtual Machines.*
2. *One Virtual Machine, say Kali Linux, is used to generate different attacks 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.*

3.2.2 PROCESSING PHASE

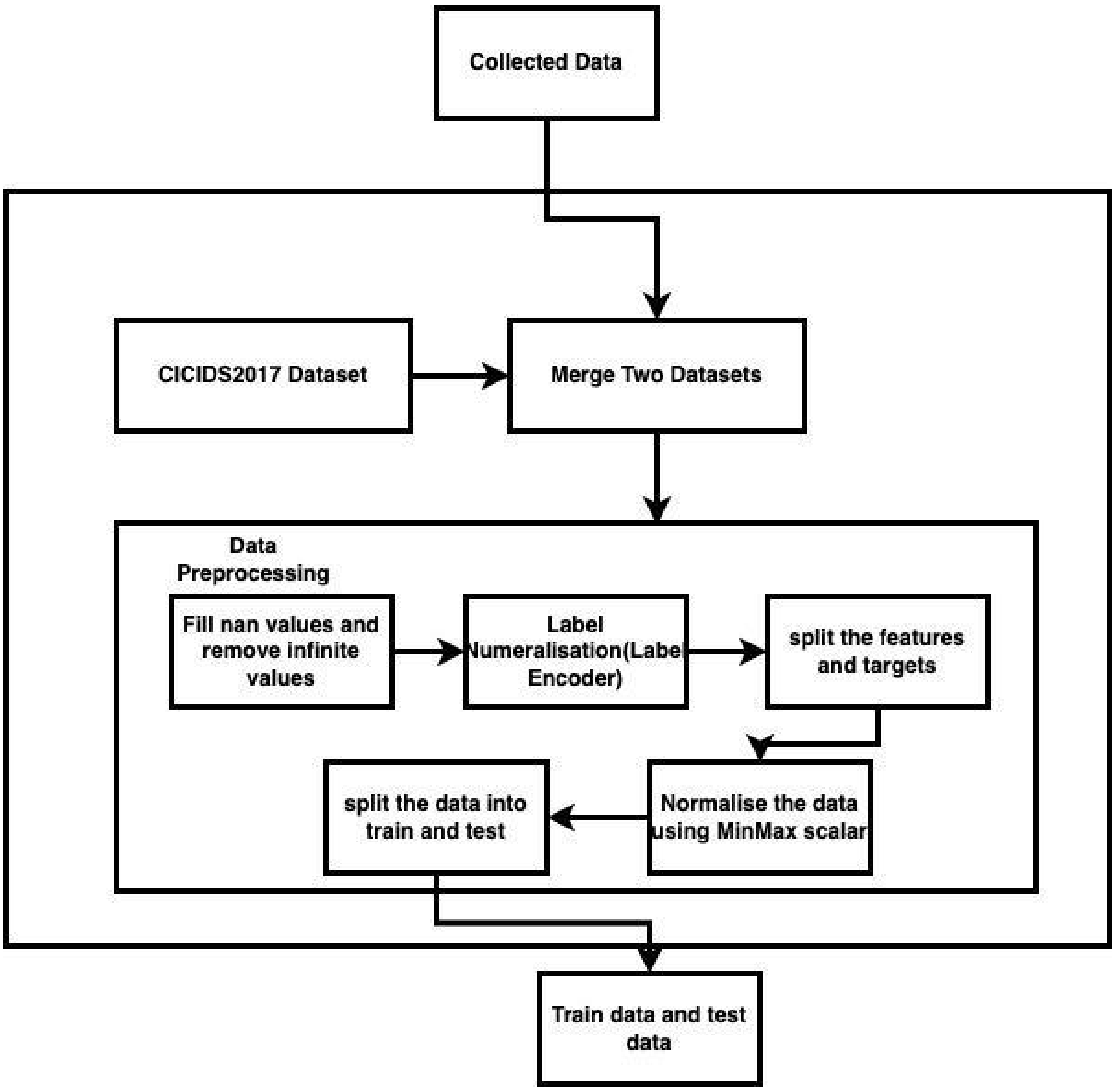

Collected Data
CICIDS2017 Dataset
Merge Two Datasets
Data Preprocessing
Fill nan values and remove infinite values
Label Numeralisation(Label Encoder)
split the features and targets
Normalise the data using MinMax scalar
split the data into train and test
Train data and test data

Fig 3.3 PROCESSING PHASE

In Fig 3.3 the processing module completes every necessary mission, before to the forecast procedure. This initially gets data packets 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. This collected data is preprocessed in such a way that it can be fed to the SCAE model for feature extraction.

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
$x_i = (x_i - min) / (max - min)$.
5.    Split the data into train and test.
```

3.2.3 ANALYSIS PHASE

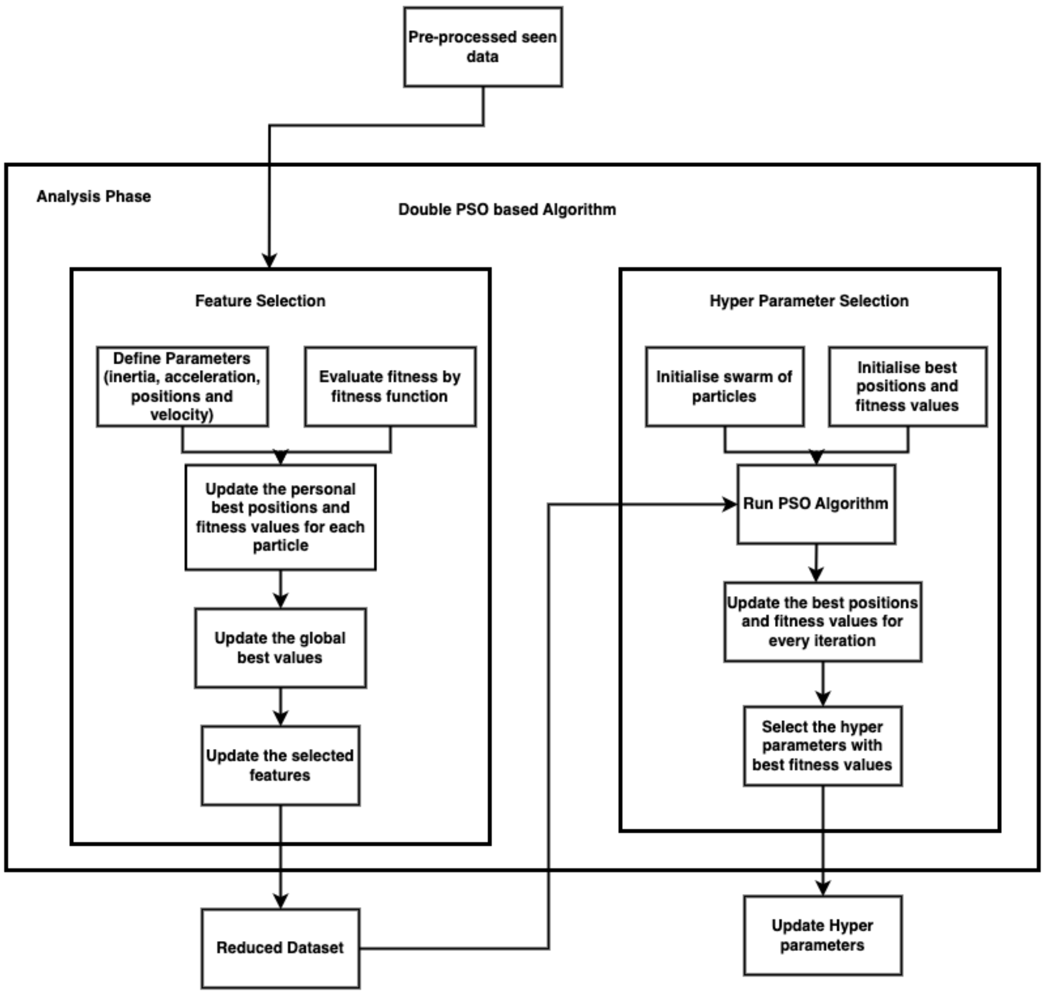

Pre-processed seen data
Analysis Phase
Double PSO based Algorithm
Feature Selection
Define Parameters (inertia, acceleration, positions and velocity)
Evaluate fitness by fitness function
Update the personal best positions and fitness values for each particle
Update the global best values
Update the selected features
Hyper Parameter Selection
Initialise swarm of particles
Initialise best positions and fitness values
Run PSO Algorithm
Update the best positions and fitness values for every iteration
Select the hyper parameters with best fitness values
Reduced Dataset
Update Hyper parameters

Fig 3.4 ANALYSIS PHASE

In Fig 3.4 the analysis module handles the hybrid learning models' pre-training and training stages. 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 data. The upper level of the double PSO-based method is then executed on the main database to determine the best features and store it in an array. The main database is then reduced using only the best characteristics. Then, using the decreased master database, it conducts the bottom phase of the double-PSO-based method to generate the **ideal hyper parameter vector for the Decision tree model**.

Input: Dataset of 32 features (from SCAE)

Output: Reduced dataset (Features)

```
Algorithm - Particle Swarm Optimization Upper Level:

1. Initialize PSO parameters
2. Assess each particle's fitness based on the appropriate feature subset.
3. Update personnel optimum positions and fitness values for each iteration
4. Update global optimum positions and fitness values for the swarm.
5. Update the velocities and positions of the particle.
$V_i = w * v_i + c1 *rand() * (p\text{-}best_{xi} - x_i) + c2 * rand() * (g\text{-}best_x - x_i)$
$x_i = x_i + v_i$
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. Assess each particle's fitness values
4. Update personnel optimum positions and fitness values for each iteration
5. Update global optimum positions and fitness values for the swarm
6. Update the velocity and position of each particle.
    a. Calculate the inertia weight.
            $w_i = w_{start} - (w_{end} - w_{start}) * (iter / max\_iter)$
    b. Compute the cognitive coefficient.
            $v_i = w_i * v_i + c1 * rand() * (pbest_i - x_i)$
    c. Calculate the social coefficient.
            $v_i = w_i * v_i + c2 * rand() * (gbest - x_i)$
    d. Calculate the cognitive component and the social component.
    Cognitive_component = cognitive_coeff * (part [‘personel_best_position’]
                    - part [‘position’]
    Social_component = social_coeff * (part [‘personel_best_position’] - part
                    [‘position’]
    e. Modify the velocity and position of each particle
            Part[‘velocity’] = Part[‘velocity’] * inertia_weight +
                Cognitive_component + Social_component
    f. Modify the particle's place by adjusting its velocity to its existing
       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.
```

3.2.4 PREDICTION PHASE

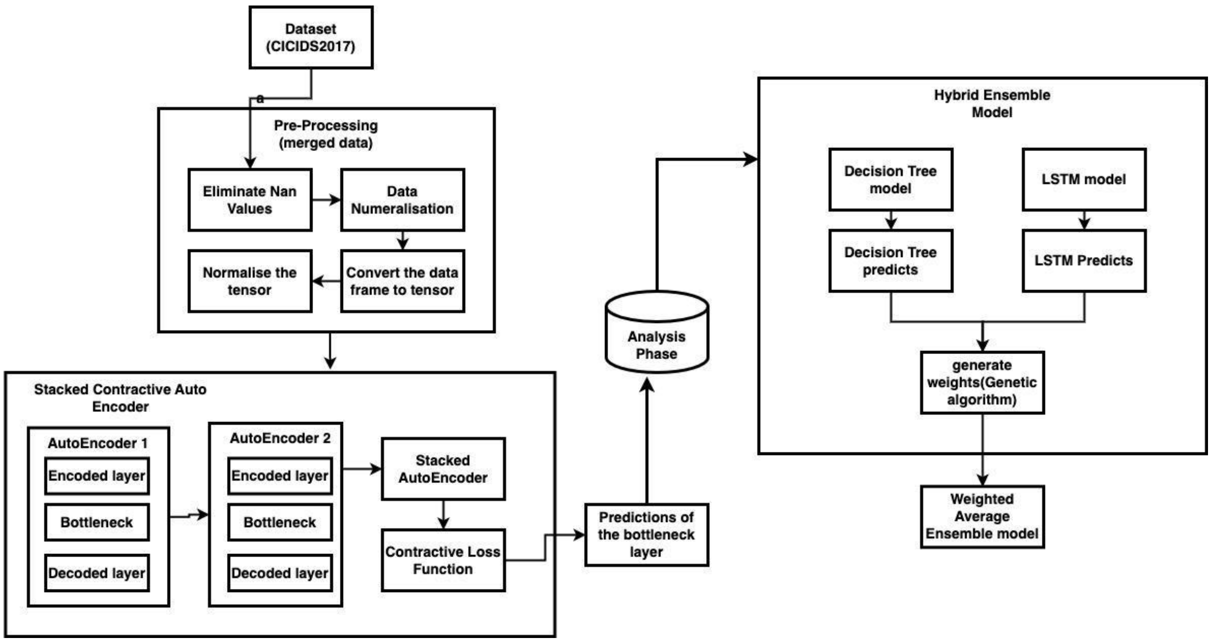


Fig 3.5 PREDICTION PHASE

In Fig 3.5 the prediction module collects the data from the processing module and checks whether there are any vulnerabilities while merging and resolve them. These data 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 packet csv file from the preprocessed, unseen database. Following that, the selected data packet file will be tested using the ensemble model, and the weighted ensemble engine's final judgment is sent to the response module.

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,

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

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 the sum of each element in the 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
           $\beta * contractivepenalty$
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 training 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.
```

3.2.5 RESPONSE PHASE

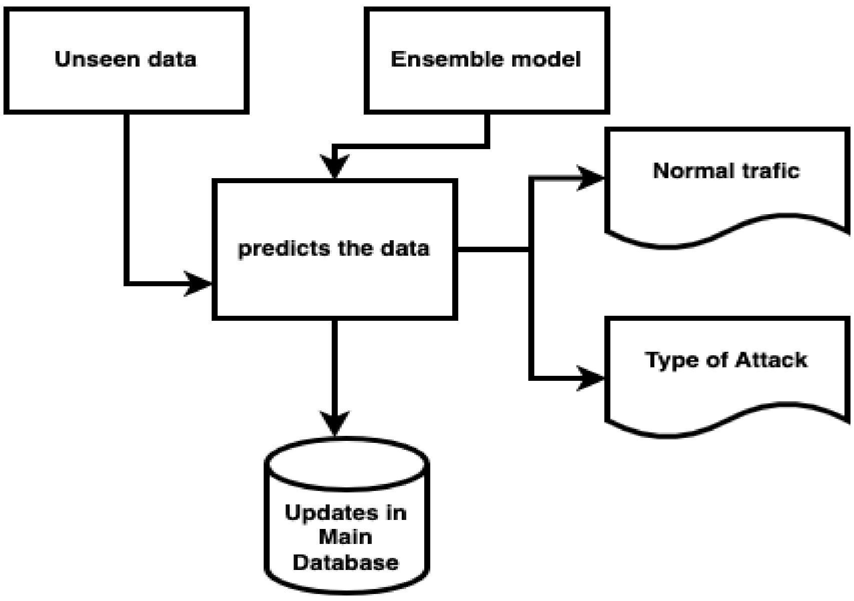


Fig 3.6 RESPONSE PHASE

In Fig 3.6, The tested data packet file and the final model are both delivered to the response module, which then labels the tested data packet with the final model. This function is responsible for responding to the final "Normal" or "Attack" decision. When the label is "Normal," the response module informs the prediction module to go on to the following data packet in the hidden dataset and forecast its label. If, however, the final option selected is "Attack," the response module gives out a warning that the cloud network is being subjected to potentially dangerous activity.

Input: Unlabeled data
Output: Type of Traffic

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 PERFORMANCE METRICS

4.1.1 THRESHOLD METRICS:

- **Classification Rate (CR):** The classification rate is a measure that quantifies a classifier's accuracy, or how effectively it predicts the proper class label of an observation.

Classification Rate = Number of Correctly Predicted instances / Total Number of instances

- **F Measure (FM):** A statistic called F-measure is used to gauge how accurate a classification model is. It is the harmonic mean of recall and accuracy.

F-measure = 2 * (Precision * Recall) / (Precision + Recall)

4.1.2 RANKING METRICS:

- **Accuracy:** It is the proportion of all dataset occurrences with correctly predicted labels.

Accuracy = (Number of Correctly Predicted Instances) / Total Number of Instances

- **Recall:** Recall quantifies the percentage of positive cases that the model properly classifies 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.

Precision = (Number of correctly estimated measurements / Total number of measurements) x 100

4.1.3 PROBABILITY METRICS:

- **Root Mean Square Error (RMSE):** It is the square root of the average of the observed and anticipated values' squared discrepancies.

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

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

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

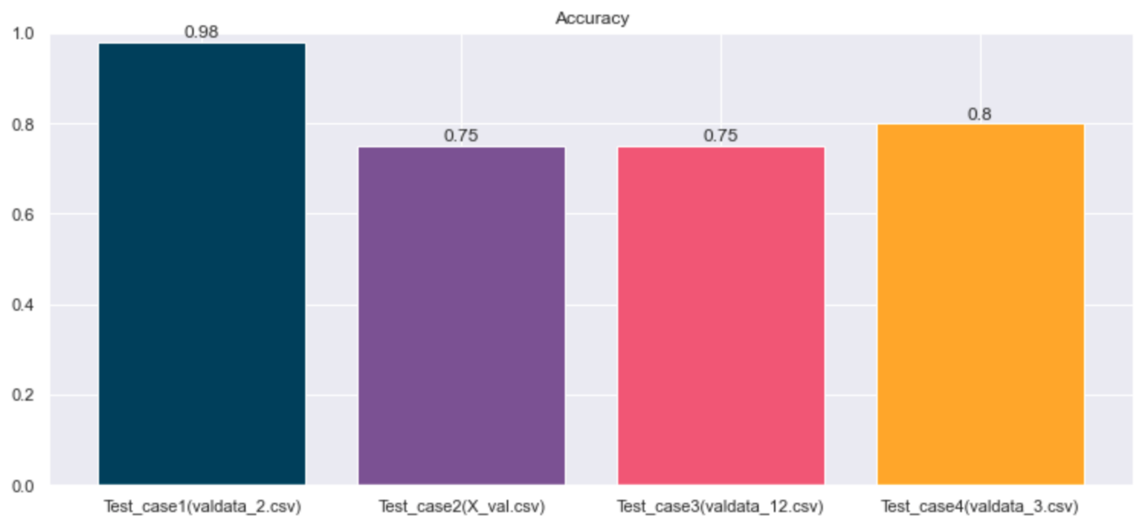


Fig. 4.1 Bar-graph of Accuracy

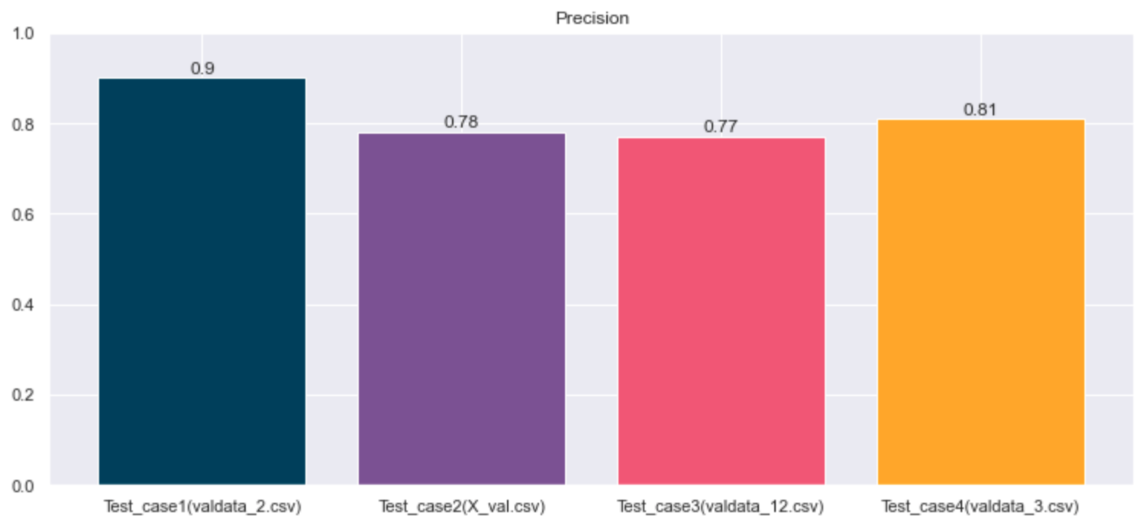


Fig. 4.2 Bar-graph of precision

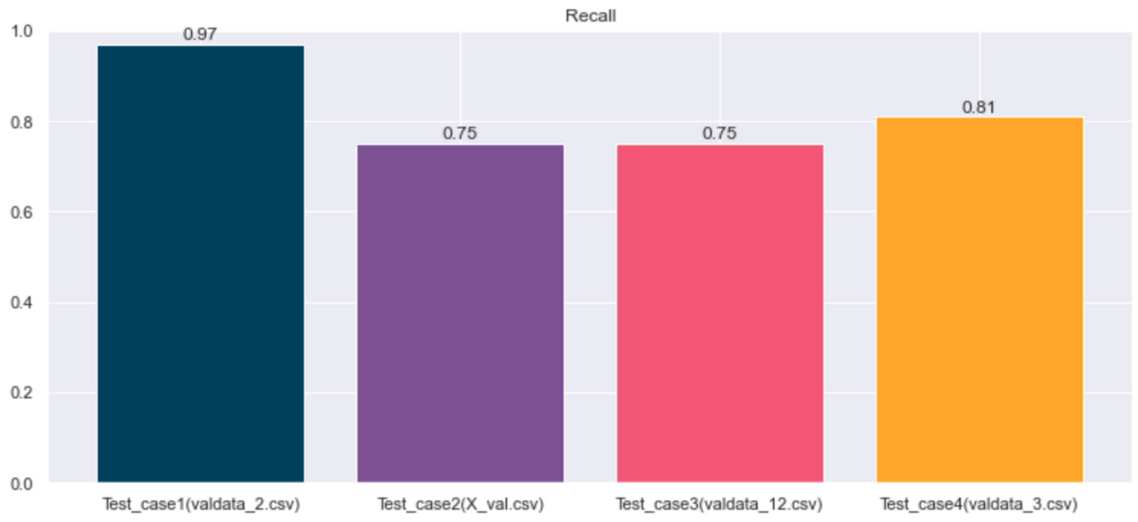


Fig. 4.3 Bar-graph of recall

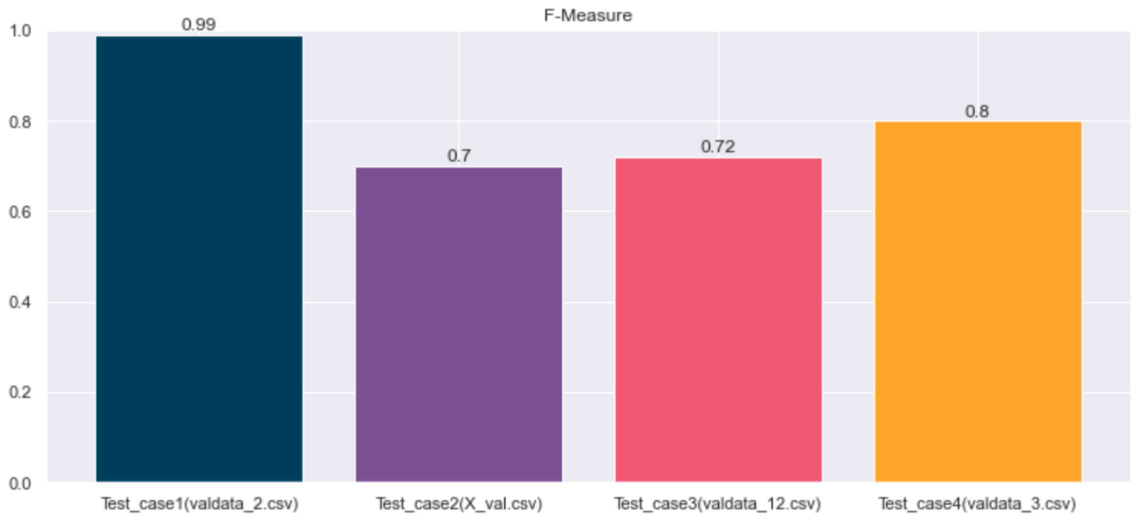


Fig. 4.4 Bar-graph of F-measure

Figs. 4.1, 4.2, 4.3, and 4.4 represent 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

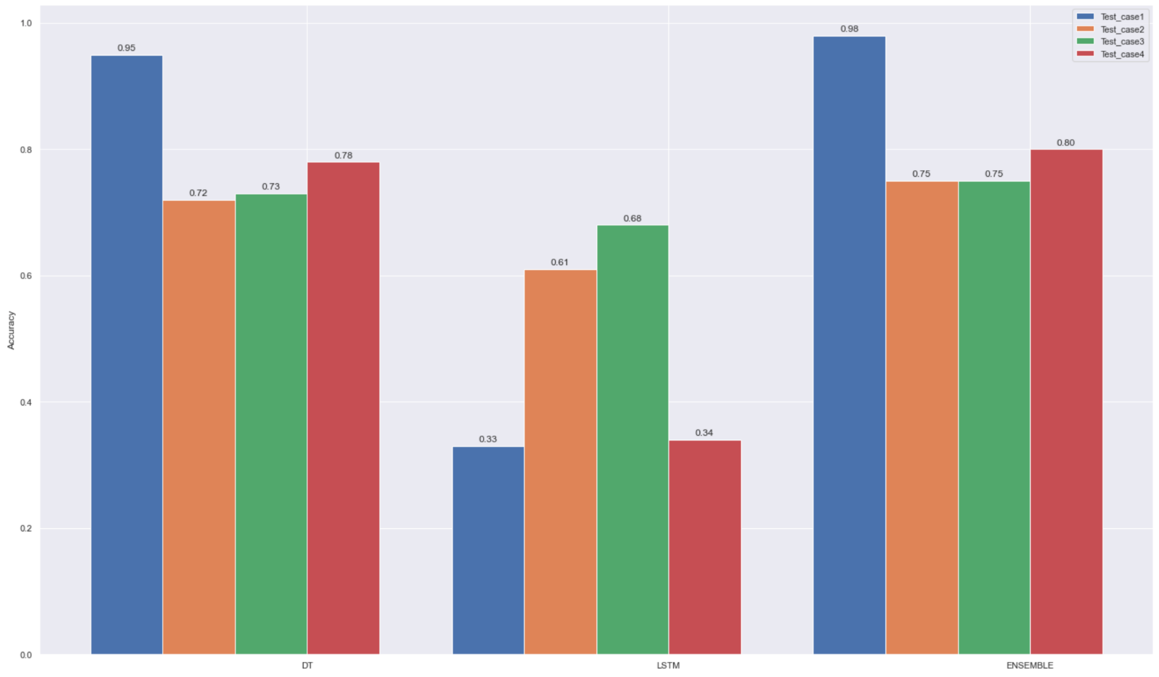


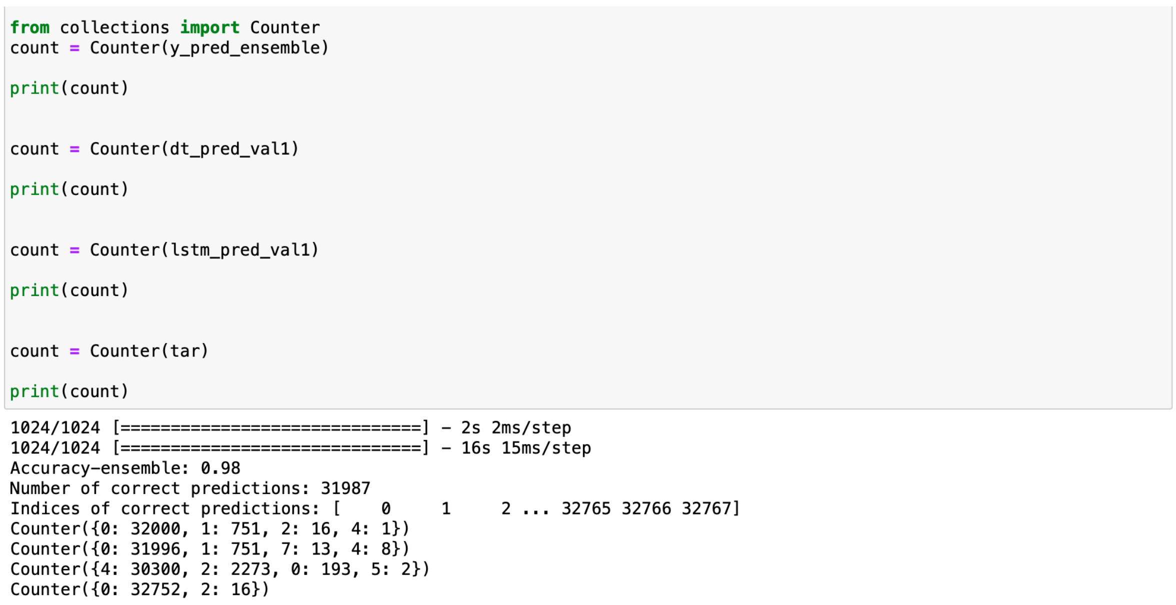
Fig 4.5 Group bar graph of all accuracies of Decision tree, LSTM and Ensemble model for all four test cases

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

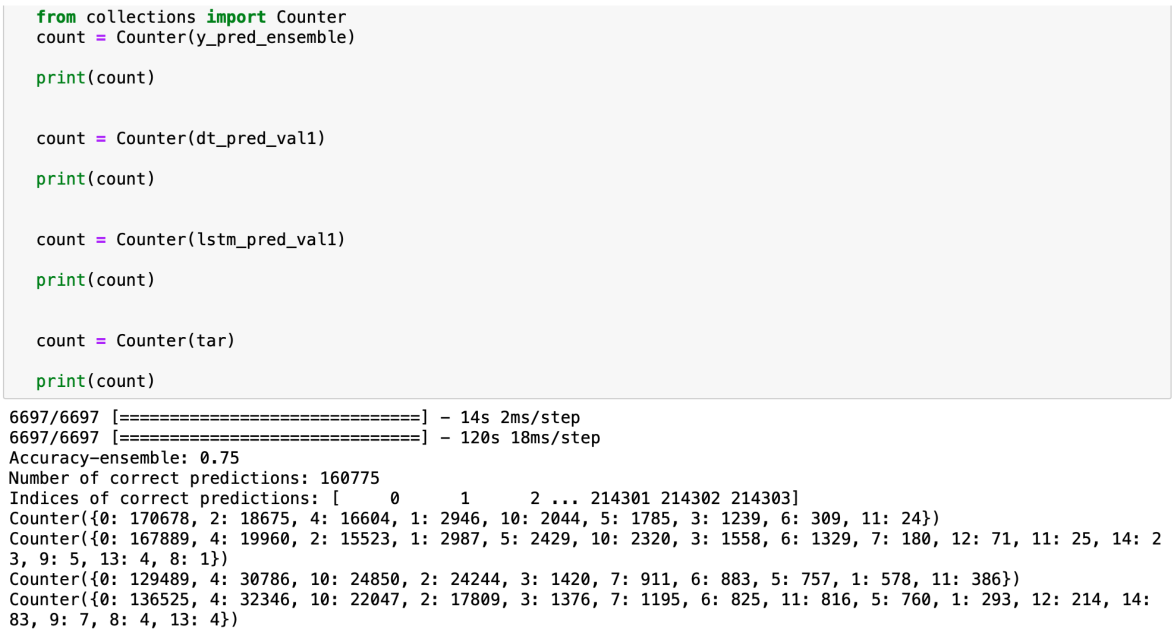
Tb 4.2 Shows the respective info about Counter arrays for each predictions for the below diagrams



```
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})
```

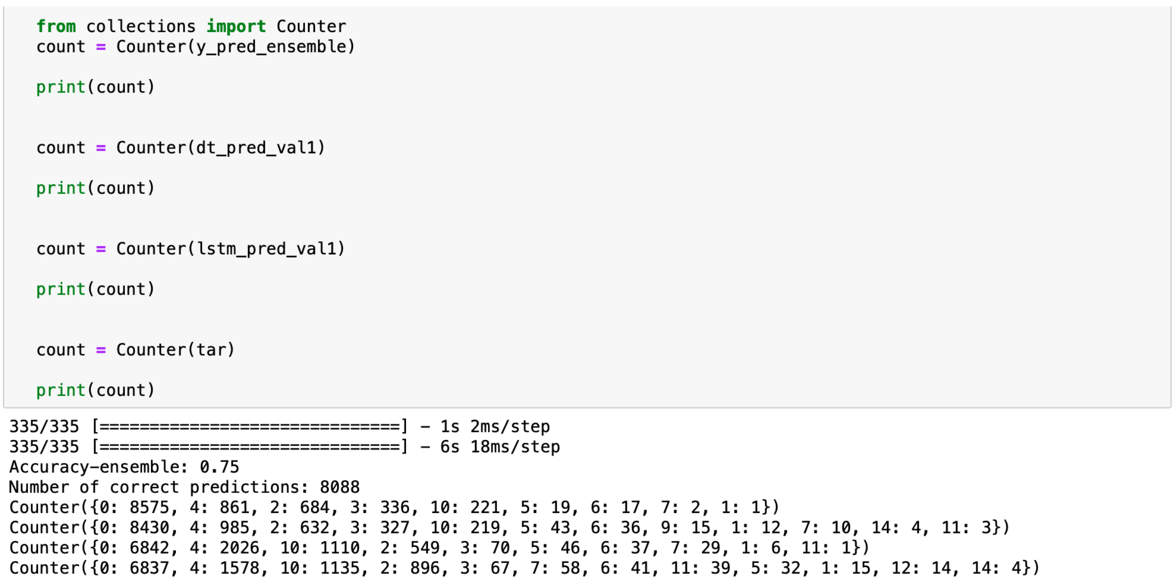
Fig 4.6 shows the predictions of all 3 models and True predictions for valdata_2.csv dataset



```
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})
```

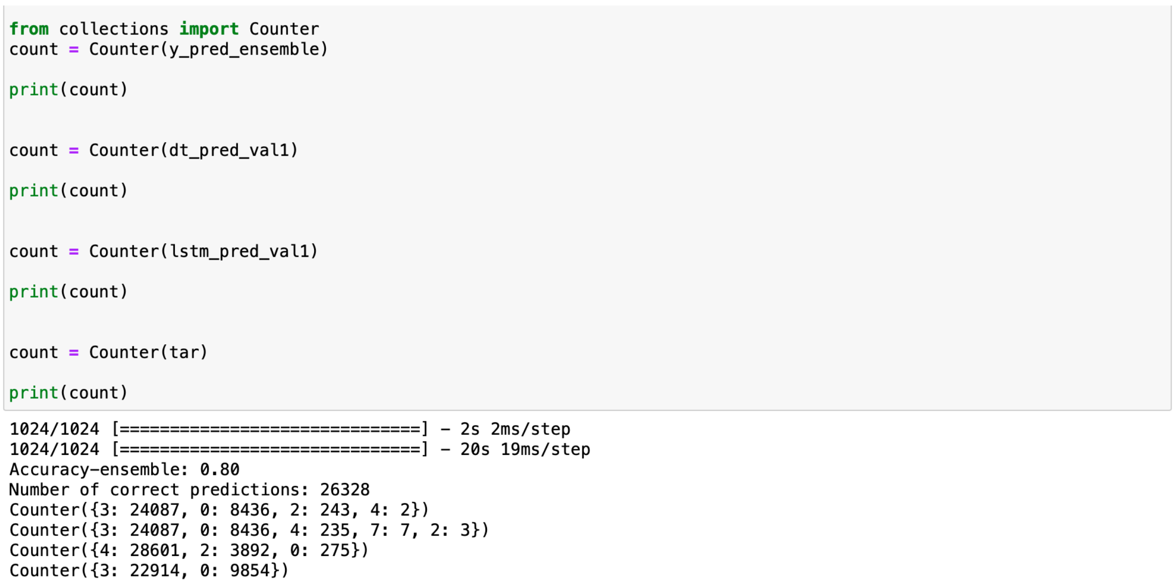
Fig 4.7 shows the predictions of all 3 models and True predictions for X_val.csv dataset



```
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.

Dataset	No of instances	No of Normal Data	No of attack Data	No of correct predictions(ensemble model)
Valdata_2.csv	32,768	32300	468	31,987
X_val.csv	2,14,304	1,36,525	77,779	1,60,775
Valdata_12.csv	10,716	6,001	4715	8,088
Valdata_3.csv	32,768	9,854	22914	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

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.

APPENDICES

In this study, we performed attacks on virtual machines running Kali Linux with Metasploitable 2 and manually gathered some attack data. We then integrated that data with the CICIDS2017 dataset and trained the combined dataset. Wireshark and CICFlowmeter services were installed on an Azure Virtual machine, which was then used to capture some typical traffic from a cloud virtual machine. We created an ensemble model and deployed it on the cloud. This model will predict the data if we provide a dataset as input. The gathered normal traffic data is utilized for model validation. Below Fig 5.1 represents the training dataset of our model.

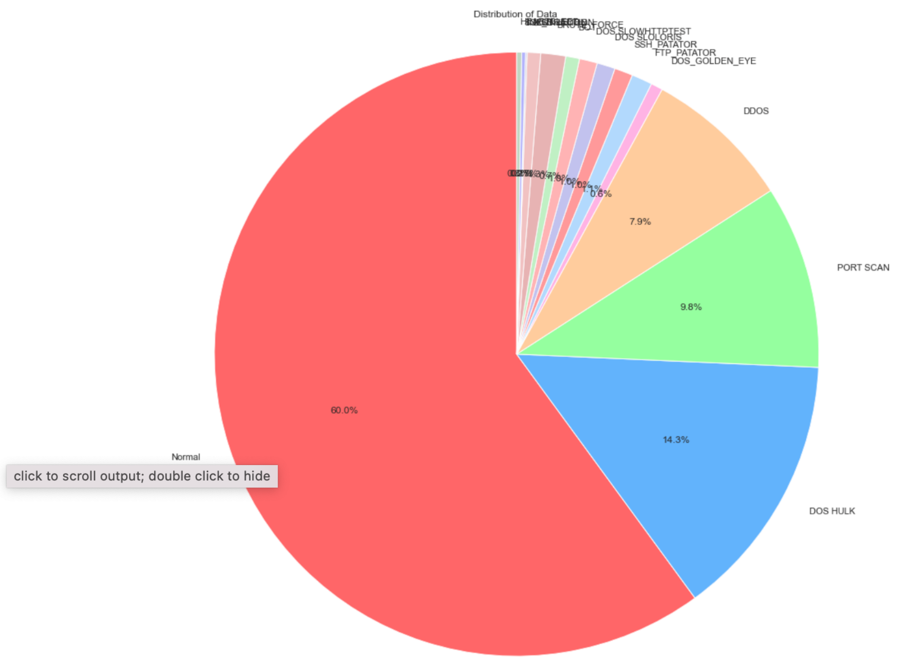


Fig. 5.1 Pie chart of the dataset

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

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 (TAEECE), 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.

9. 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