A

Report Project On

**Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks**

(Submitted in partial fulfillment of the requirements for the award of Degree) BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

By

N. NITHYANJALI (207R1A66A4)

K. SHREYESH(207R1A6689)

K. PRANAV(207R1A6694)

Under the Guidance of

##### Mr. M. RAVINDRAN

(Assistant Professor)



##### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

**CMR TECHNICAL CAMPUS UGC AUTONOMOUS**

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, NewDelhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

**2020-2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)**



#### CERTIFICATE

This is to certify that the project entitled **“Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks”** being submitted by **N.NITHYANJALI(207R1A66A4), K.SHREYESH(207R1A6689), K.PRANAV(207R1A6694),** partial fulfillment of requirements for the award of the degree of B.Tech in Computer Science and Engineering (AI&ML) to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

MR.M.RAVINDRAN Dr. S RaoChintalapudi

(Assistant Professor) HOD CSE(AI&ML)

INTERNAL GUIDE

**EXTERNAL EXAMINER**

**Submitted for viva voice Examination held on**

##### ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mr. M. RAVINDRAN,** Assisstant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. G. Vinoda Reddy, Dr. K. Mahesh, N. Sateesh & B. Mamatha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. S Rao Chintalapudi,** Head, Department of Computer Science and Engineering (AI&ML) for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the courseof this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

**N. NITHYANJALI (207R1A66A4)**

**K. SHREYESH (207R1A6689)**

**K. PRANAV (207R1A6694)**

##### ABSTRACT

The botnet attack is a multi-stage and the most prevalent cyber-attack in the Internet of Things (IoT) environment that initiates with scanning activity and ends at the distributed denial of service (DDoS) attack. The existing studies mostly focus on detecting botnet attacks after the IoT devices get compromised, and start performing the DDoS attack. Similarly, the performance of most of the existing machine learning based botnet detection models is limited to a specific dataset on which they are trained. As a consequence, these solutions do not perform well on other datasets due to the diversity of attack patterns. Therefore, in this work, we first produce a generic scanning and DDoS attack dataset by generating 33 types of scan and 60 types of DDoS attacks. In addition, we partially integrated the scan and DDoS attack samples from three publicly-available datasets for maximum attack coverage to better train the machine learning algorithms. Afterwards, we propose a two-fold machine learning approach to prevent and detect IoT botnet attacks. In the first fold, we trained a state-of-the-art deep learning model, i.e., ResNet-18 to detect the scanning activity in the premature attack stage to prevent IoT botnet attacks. While, in the second fold, we trained another ResNet-18 model for DDoS attack identification to detect IoT botnet attacks. Overall, the proposed two-fold approach manifests 98.89% accuracy, 99.01% precision, 98.74% recall, and 98.87% f1-score to prevent and detect IoT botnet attacks. To demonstrate the effectiveness of the proposed two-fold approach, we trained three other ResNet-18 models over three different datasets for detecting scan and DDoS attacks and compared their performance with the proposed two-fold approach. The experimental results prove that the proposed two-fold approach can efficiently prevent and detect botnet attacks as compared to

other trained models.

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| 3.1 | Project Architecture for Two fold machine learning approach to prevent and detect IOT botnet attacks | 11 |
| 3.2 | Use Case Diagram for Two fold machine learning approach to prevent and detect IOT botnet attacks | 12 |
| 3.3 | Class Diagram for Two fold machine learning approach to prevent and detect IOT botnet attacks | 13 |
|  |  |  |
| 3.4 | Sequence diagram for Two fold machine learning approach to prevent and detect IOT botnet attacks | 14 |
| 3.5 | Activity diagram for Two fold machine learning approach to prevent and detect IOT botnet attacks | 15 |

|  |  |  |
| --- | --- | --- |
| **SCREENSHOT NO.** | **SCREENSHOT NAME** | **PAGE NO**. |
| 5.1 | Home page | 33 |
| 5.2 | Remote user | 33 |
| 5.3 | Admin login | 34 |
| 5.4 | Remote users list | 34 |
| 5.5  5.6  5.7  5.8  5.9  5.10  5.11 | Algorithm accuracy(pie chart)  Algorithm accuracy(line chart)  Botnet accuracy(line chart)  Botnet accuracy(pie chart)  Botnet detection  Botnet detection type  Botnet attack ratio | 35  35  36  36  37  37  38 |

**ABSTRACT** i

LIST OF FIGURES ii

LIST OF SCREENSHOTS iii

1. [INTRODUCTION 1](#_bookmark0)
   1. [PROJECT SCOPE 1](#_bookmark1)
   2. [PROJECT PURPOSE 1](#_bookmark2)
   3. [PROJECT FEATURES 1](#_bookmark3)
2. [SYSTEM ANALYSIS 2](#_bookmark4)
   1. [PROBLEM DEFINITION 2](#_bookmark5)
   2. [EXISTING SYSTEM/LITERATURE REVIEW 2](#_bookmark6)
      1. [LIMITATIONS OF EXISTING SYSTEM 3](#_bookmark7)
   3. [PROPOSED SYSTEM 3](#_bookmark7)

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

2.4 FEASIBILITY STUDY

2.4.1 ECONOMIC FEASIBILITY

2.4.2 TECHNICAL FEASIBILITY

2.4.3 SOCIAL FEASIBILITY

* 1. [SYSTEM DESIGN AND DEVELOPMENT 4](#_bookmark8)

2.6 [HARDWARE & SOFTWARE REQUIREMENTS 5](#_bookmark11)

2.6.2 [HARDWARE REQUIREMENTS 5](#_bookmark12)

* + 1. [SOFTWARE REQUIREMENTS 6](#_bookmark13)

1. [ARCHITECTURE 7](#_bookmark14)
   1. [PROJECT ARCHITECTURE 8](#_bookmark15)
   2. [DESCRIPTION 8](#_bookmark16)
   3. [USE CASE DIAGRAM 9](#_bookmark17)

3.4 [CLASS DIAGRAM 10](#_bookmark18)

3.5 [SEQUENCE DIAGRAM 11](#_bookmark19)

3.6 [ACTIVITY DIAGRAM 12](#_bookmark20)

1. [IMPLEMENTATION 13](#_bookmark21)
   1. LINEAR REGRESSION [14](#_bookmark22)
   2. SUPPORT VECTOR MACHINE
   3. RANDOM FOREST
   4. GRADIENT BOOSTING ALGORITHM
   5. SAMPLE CODE
   6. RESULT ANALYSIS
2. [SCREENSHOTS 26](#_bookmark23)
3. [TESTING 30](#_bookmark24)
   1. [INTRODUCTION TO TESTING 31](#_bookmark25)
   2. [TYPES OF TESTING 31](#_bookmark26)
      1. UNIT TESTING
      2. INTEGRATION TESTING
      3. USER ACCEPTANCE TESTING
      4. OUTPUT TESTING
      5. VALIDATION TESTING
   3. USER TRAINING
   4. MAINTANANCE
   5. TESTING STRATEGY
4. **CONCLUSION**
   1. PROJECT CONCLUSION
   2. FUTURE SCOPE
5. **BIBLIOGRAPHY**
   1. REFERENCES
   2. GITHUB LINK

# INTRODUCTION

#### INTRODUCTION

##### PROJECT SCOPE

##### This project aims to develop a robust security mechanism for Internet of Things (IoT) devices to prevent and detect botnet and Distributed Denial of Service (DDOS) attacks. It involves the analysis of prevalent attack techniques, creation of a comprehensive dataset, and implementation of a two-fold machine learning approach using ResNet-18 models. The focus is on addressing security vulnerabilities in IoT devices during the scanning and attack stages to mitigate the potential risks posed by botnet attacks.

##### PROJECT PURPOSE

The purpose of this project is to enhance the security of IoT devices by developing an effective defense mechanism against botnet and DDOS attacks. By leveraging machine learning techniques and building upon existing research, the project aims to provide proactive detection and prevention capabilities, thereby safeguarding IoT networks from malicious activities. The ultimate goal is to contribute to the resilience of IoT infrastructure and protect against the escalating threats posed by cyber-attacks targeting insecure IoT devices.

##### PROJECT FEATURES

Two-Fold Machine Learning Approach: Implementing a two-pronged strategy utilizing ResNet-18 models for both scanning attack detection and DDOS attack detection, aiming to intercept malicious activities at early stages and mitigate potential botnet threats effectively.

Adaptability Across Multiple Datasets: Demonstrating the efficacy of the proposed approach across different datasets to validate its robustness and effectiveness in detecting and preventing IoT botnet attacks, thereby ensuring scalability and applicability in diverse IoT environments.

## SYSTEM ANALYSIS

##### SYSTEM ANALYSIS

**SYSTEM ANALYSIS**

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst hasa firm understanding of what is to be done.

##### PROBLEM DEFINITION

The proliferation of Internet of Things (IoT) devices has led to an increased risk of cyber-attacks, particularly from botnets and Distributed Denial of Service (DDOS) attacks. Existing methodologies primarily focus on detecting scanning activity and DDoS attacks, neglecting other stages of IoT botnet attacks. Moreover, IoT botnet attacks may not always initiate with scanning activity and conclude with DDoS attacks. This presents a significant challenge in effectively securing IoT networks against evolving cyber threats..

##### EXISTING SYSTEM

Nguyen et al. [16] introduced a graph-based approach for IoT botnet detection, utilizing PSI graphs to extract high-level features from function call graphs. They trained a convolutional neural network (CNN) over these graphs for detection. Similarly, Wang et al. [24] presented BotMark, employing flow-based and graph-based analysis for botnet detection. Yassin et al. [25] focused on Mirai attacks, utilizing graph visualization and rules generation. Nugraha et al. [32] evaluated deep learning models for botnet attack detection.

###### LIMITATIONS OF EXISTING SYSTEM

* Existing methodologies primarily focus on detecting scanning activity and DDoS attacks, overlooking other stages of IoT botnet attacks.
* IoT botnet attacks may not always initiate with scanning activity and conclude with DDoS attacks.

##### PROPOSED SYSTEM

The proposed system addresses the limitations of existing methods by analyzing various scanning and DDoS attack techniques to generate a comprehensive dataset. It introduces a two-fold machine learning approach for both inbound and outbound botnet attack detection in IoT networks. This approach intercepts scanning activity to prevent attackers from advancing to further attack stages. Additionally, the system demonstrates its effectiveness across multiple datasets by training ResNet-18 models.

###### ADVANTAGES OF THE PROPOSED SYSTEM

* Novel two-fold machine learning approach enhances IoT botnet attack prevention and detection.
* Early interception during scanning activity prevents attackers from progressing to subsequent attack stages, enhancing network security.

##### FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**
  + 1. **ECONOMIC FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

###### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

###### SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

* 1. **SYSTEM DESIGN AND DEVELOPMENT**

**2.5.1 INPUT DESIGN**

* **Problem Statement and Motivation:**
  + Botnet attacks pose a significant threat to Internet of Things (IoT) devices.
  + Existing studies primarily focus on detecting botnet attacks after devices are compromised and DDoS attacks begin.
  + However, the performance of machine learning models is often limited to specific training datasets, hindering their effectiveness across diverse attack patterns.
* **Dataset Creation:**
  + You generated a generic scanning and DDoS attack dataset.
  + This comprehensive dataset includes 33 types of scans and 60 types of DDoS attacks.
  + Integration of samples from publicly-available datasets enhances attack coverage.
* **Two-Fold Approach:**

**Fold 1: Scanning Activity Detection:**

* Trained a ResNet-18 deep learning model to detect scanning activity during the early attack stage.
* This proactive step helps prevent IoT botnet attacks by identifying suspicious behavior before compromise.

**Fold 2: DDoS Attack Identification:**

* Trained another ResNet-18 model to identify ongoing DDoS attacks.
* This enables real-time detection of IoT botnet attacks during the DDoS phase.

**Fold 1: Scanning Activity Detection:**

* **Input:**
* Network traffic data: Captured from IoT devices during their normal operation.
* Features: Extracted from network packets, including source/destination IP addresses, port numbers, packet size, and protocol type.

**Fold 2: DDoS Attack Identification:**

* + **Input:**
  + Network traffic data: Collected during ongoing IoT device communication.
  + Features: Similar to Fold 1, including packet-level information.
    1. **OUTPUT DESIGN**
* **Output:**
* **Binary classification: Scanning activity detection**
* 1 (Positive): Indicates scanning activity (potential botnet reconnaissance).
* 0 (Negative): Represents normal network behavior.
* **Output:**
* **Binary classification: DDOS attack identification**
* 1 (Positive): Indicates an ongoing DDoS attack (botnet activity).
* 0 (Negative): Signifies legitimate network traffic.

##### HARDWARE & SOFTWARE REQUIREMENTS

###### HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

* ➢ Processor - i3 or above
* ➢ RAM - 4 GB (min)
* ➢ Hard Disk - 20 GB
* ➢ Key Board - Standard Windows Keyboard
* ➢ Mouse - Two or Three Button Mouse
* ➢ Monitor - SVGA( Super video graphics array)

##### SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

* Operating system : Windows 7 Ultimate.
* Coding Language : Python.
* Back-End : Django-ORM
* Designing : Html, css, javascript.
* Data Base : MySQL (WAMP Server).

## ARCHITECTURE

##### 3.ARCHITECTURE

* 1. **PROJECT ARCHITECTURE**

This project architecture shows the procedure followed for classification,starting from input to final prediction.

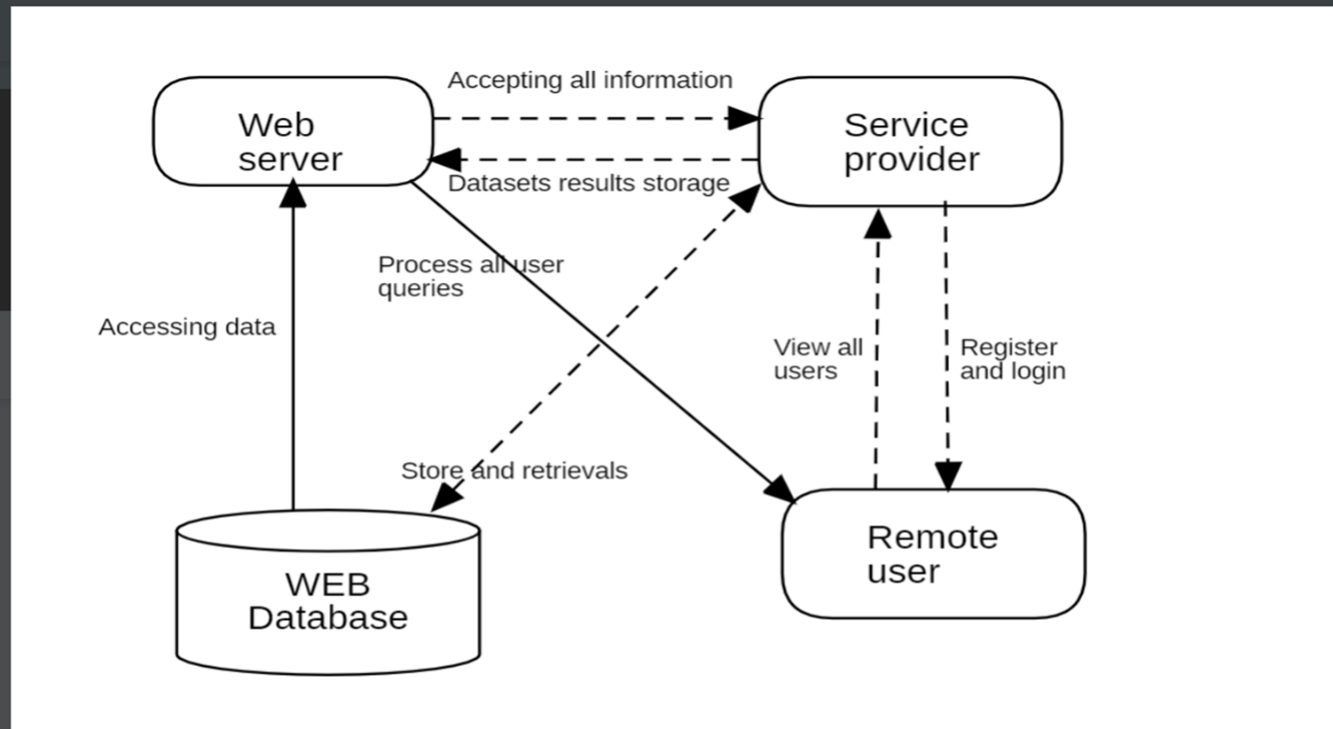


Figure 3.1: Project Architecture of Two fold machine learning approach to prevent and detect IOT botnet attacks

###### DESCRIPTION

The system architecture employs ResNet-18 models for two-fold machine learning approach: scanning attack detection and DDoS attack detection. Data preprocessing extracts features from IoT network traffic. A ResNet-18 model trained for scanning activity identifies precursors to botnet attacks, while another model detects DDoS attacks by analyzing traffic patterns. Continuous model updates adapt to emerging threats. Seamless integration with existing security infrastructure enhances defense capabilities.

###### USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted

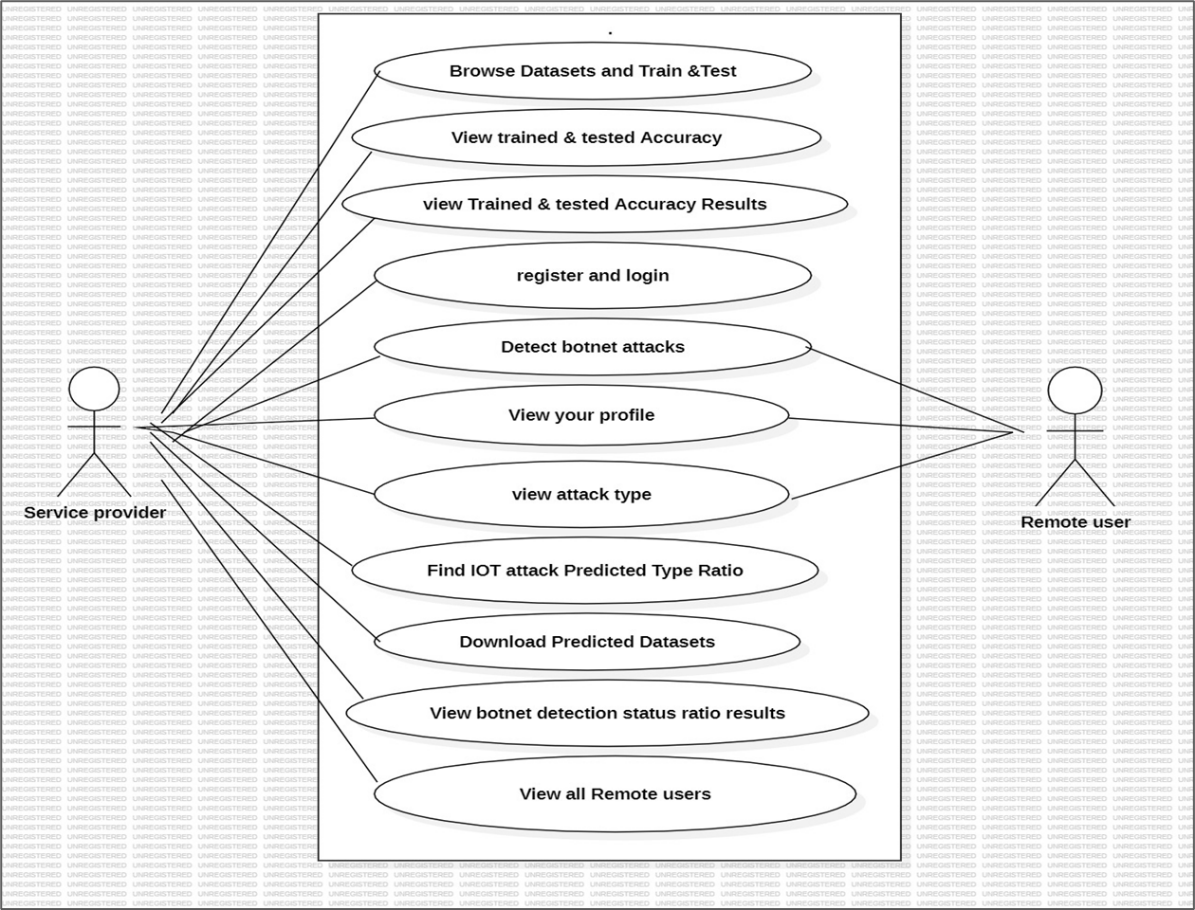


Figure 3.2: Use Case Diagram for Two fold machine learning approach to prevent and detect IOT botnet attacks

.

##### CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

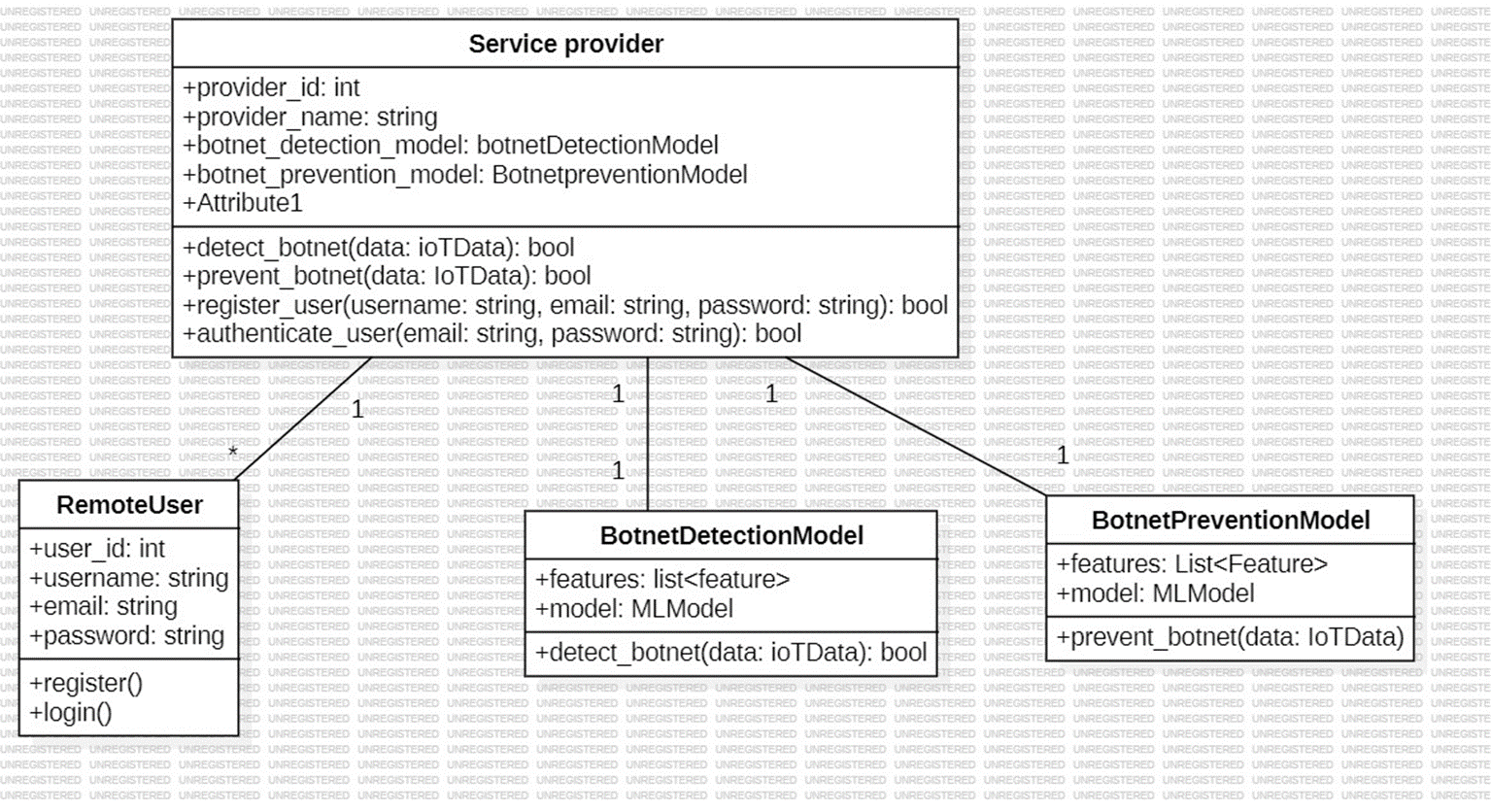


Figure 3.3: Class Diagram for Two fold machine learning approach to prevent and detect IOT botnet attacks

##### SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

A diagram of a server

Description automatically generated

Figure 3.4: Sequence Diagram for Two fold machine learning approach to prevent and detect IOT botnet attacks

###### 3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

A diagram of a process

Description automatically generated

Figure 3.5: Activity Diagram for Two fold machine learning approach to prevent and detect IOT botnet attacks

## 4. IMPLEMENTATION

**4.1 LINEAR REGRESSION**

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between a dependent variable and one or more independent features. When the number of the independent feature, is 1 then it is known as Univariate Linear regression, and in the case of more than one feature, it is known as multivariate linear regression.

The interpretability of linear regression is a notable strength. The model’s equation provides clear coefficients that elucidate the impact of each independent variable on the dependent variable, facilitating a deeper understanding of the underlying dynamics. Its simplicity is a virtue, as linear regression is transparent, easy to implement, and serves as a foundational concept for more complex algorithms.

Linear regression is not merely a predictive tool; it forms the basis for various advanced models. Techniques like regularization and support vector machines draw inspiration from linear regression, expanding its utility. Additionally, linear regression is a cornerstone in assumption testing, enabling researchers to validate key assumptions about the data.

**Types of Linear Regression**

There are two main types of linear regression:

**Simple Linear Regression**

This is the simplest form of linear regression, and it involves only one independent variable and one dependent variable.

**Multiple Linear Regression**

This involves more than one independent variable and one dependent variable

**4.2 SUPPORT VECTOR MACHINE ALGORITHM**

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships.

SVM algorithms are very effective as we try to find the maximum separating hyperplane between the different classes available in the target feature.

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it’s best suited for classification. The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

**4.3 Gradient Boosting**

Gradient Boosting is a popular boosting algorithm in machine learning used for classification and regression tasks. Boosting is one kind of ensemble Learning method which trains the model sequentially and each new model tries to correct the previous model. It combines several weak learners into strong learners. Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

In contrast to AdaBoost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of the predecessor as labels. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

**4.4 Random Forest Algorithm**

A Random Forest Algorithm is a supervised machine learning algorithm that is extremely popular and is used for Classification and Regression problems in Machine Learning. We know that a forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is based on the concept of ensemble learning which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model.

**Essential Features of Random Forest**

Miscellany: Each tree has a unique attribute, variety and features concerning other trees. Not all trees are the same.

Immune to the curse of dimensionality: Since a tree is a conceptual idea, it requires no features to be considered. Hence, the feature space is reduced.

Parallelization: We can fully use the CPU to build random forests since each tree is created autonomously from different data and features.

Train-Test split: In a Random Forest, we don’t have to differentiate the data for train and test because the decision tree never sees 30% of the data.

Stability: The final result is based on Bagging, meaning the result is based on majority voting or average.

##### 4.5 SAMPLE CODE

##### 4.5.1 Remote User

admin.py:

from django.contrib import admin

# Register your models here.

**apps.py:**

from django.apps import AppConfig

class ClientSiteConfig(AppConfig):

name = 'Remote\_User'

**forms.py:**

from django import forms

from Remote\_User.models import ClientRegister\_Model

class ClientRegister\_Form(forms.ModelForm):

password = forms.CharField(widget=forms.PasswordInput())

email = forms.EmailField(required=True)

class Meta:

model = ClientRegister\_Model

fields = ("username","email","password","phoneno","country","state","city")

**models.py:**

from django.db import models

# Create your models here.

from django.db.models import CASCADE

class ClientRegister\_Model(models.Model):

username = models.CharField(max\_length=30)

email = models.EmailField(max\_length=30)

password = models.CharField(max\_length=10)

phoneno = models.CharField(max\_length=10)

country = models.CharField(max\_length=30)

state = models.CharField(max\_length=30)

city = models.CharField(max\_length=30)

gender= models.CharField(max\_length=30)

address= models.CharField(max\_length=30)

class detect\_iot\_botnet\_attacks(models.Model):

Sender\_IP= models.CharField(max\_length=3000)

Sender\_Port= models.CharField(max\_length=3000)

Target\_Ip= models.CharField(max\_length=3000)

Target\_Port= models.CharField(max\_length=3000)

Transport\_Protocol= models.CharField(max\_length=3000)

Duration= models.CharField(max\_length=3000)

AvgDuration= models.CharField(max\_length=3000)

PBS= models.CharField(max\_length=3000)

AvgPBS= models.CharField(max\_length=3000)

TBS= models.CharField(max\_length=3000)

PBR= models.CharField(max\_length=3000)

AvgPBR= models.CharField(max\_length=3000)

TBR= models.CharField(max\_length=3000)

Missed\_Bytes= models.CharField(max\_length=3000)

Packets\_Sent= models.CharField(max\_length=3000)

Packets\_Received= models.CharField(max\_length=3000)

SRPR= models.CharField(max\_length=3000)

Prediction= models.CharField(max\_length=3000)

class detection\_accuracy(models.Model):

names = models.CharField(max\_length=300)

ratio = models.CharField(max\_length=300)

class detection\_ratio(models.Model):

names = models.CharField(max\_length=300)

ratio = models.CharField(max\_length=300)

**tests:**

from django.test import TestCase

# Create your tests here.

**Views.py:**

from django.db.models import Count

from django.db.models import Q

from django.shortcuts import render, redirect, get\_object\_or\_404

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.metrics import accuracy\_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import VotingClassifier

# Create your views here.

from Remote\_User.models import ClientRegister\_Model,detect\_iot\_botnet\_attacks,detection\_ratio,detection\_accuracy

def login(request):

if request.method == "POST" and 'submit1' in request.POST:

username = request.POST.get('username')

password = request.POST.get('password')

try:

enter = ClientRegister\_Model.objects.get(username=username,password=password)

request.session["userid"] = enter.id

return redirect('ViewYourProfile')

except:

pass

return render(request,'RUser/login.html')

def index(request):

return render(request, 'RUser/index.html')

def Add\_DataSet\_Details(request):

return render(request, 'RUser/Add\_DataSet\_Details.html', {"excel\_data": ''})

def Register1(request):

if request.method == "POST":

username = request.POST.get('username')

email = request.POST.get('email')

password = request.POST.get('password')

phoneno = request.POST.get('phoneno')

country = request.POST.get('country')

state = request.POST.get('state')

city = request.POST.get('city')

address = request.POST.get('address')

gender = request.POST.get('gender')

ClientRegister\_Model.objects.create(username=username, email=email, password=password, phoneno=phoneno,

country=country, state=state, city=city,address=address,gender=gender)

obj = "Registered Successfully"

return render(request, 'RUser/Register1.html',{'object':obj})

else:

return render(request,'RUser/Register1.html')

def ViewYourProfile(request):

userid = request.session['userid']

obj = ClientRegister\_Model.objects.get(id= userid)

return render(request,'RUser/ViewYourProfile.html',{'object':obj})

def Predict\_Drug\_Response(request):

if request.method == "POST":

if request.method == "POST":

Sender\_IP= request.POST.get('Sender\_IP')

Sender\_Port= request.POST.get('Sender\_Port')

Target\_Ip= request.POST.get('Target\_Ip')

Target\_Port= request.POST.get('Target\_Port')

Transport\_Protocol= request.POST.get('Transport\_Protocol')

Duration= request.POST.get('Duration')

AvgDuration= request.POST.get('AvgDuration')

PBS= request.POST.get('PBS')

AvgPBS= request.POST.get('AvgPBS')

TBS= request.POST.get('TBS')

PBR= request.POST.get('PBR')

AvgPBR= request.POST.get('AvgPBR')

TBR= request.POST.get('TBR')

Missed\_Bytes= request.POST.get('Missed\_Bytes')

Packets\_Sent= request.POST.get('Packets\_Sent')

Packets\_Received= request.POST.get('Packets\_Received')

SRPR= request.POST.get('SRPR')

df = pd.read\_csv('Datasets.csv')

def apply\_response(Label):

if (Label== 0):

return 0 # No Botnet Detection

elif(Label==1):

return 1 # Botnet Detection

df['results'] = df['Label'].apply(apply\_response)

cv = CountVectorizer()

X = df['Sender\_IP']

y = df['results']

print("Sender\_IP")

print(X)

print("Results")

print(y)

cv = CountVectorizer()

X = cv.fit\_transform(X)

models = []

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

X\_train.shape, X\_test.shape, y\_train.shape

# SVM Model

print("SVM")

from sklearn import svm

lin\_clf = svm.LinearSVC()

lin\_clf.fit(X\_train, y\_train)

predict\_svm = lin\_clf.predict(X\_test)

svm\_acc = accuracy\_score(y\_test, predict\_svm) \* 100

print("ACCURACY")

print(svm\_acc)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, predict\_svm))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, predict\_svm))

models.append(('svm', lin\_clf))

print("Logistic Regression")

from sklearn.linear\_model import LogisticRegression

reg = LogisticRegression(random\_state=0, solver='lbfgs').fit(X\_train, y\_train)

y\_pred = reg.predict(X\_test)

print("ACCURACY")

print(accuracy\_score(y\_test, y\_pred) \* 100)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, y\_pred))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, y\_pred))

models.append(('logistic', reg))

print("Gradient Boosting Classifier")

from sklearn.ensemble import GradientBoostingClassifier

clf = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1.0, max\_depth=1, random\_state=0).fit(

X\_train,

y\_train)

clfpredict = clf.predict(X\_test)

print("ACCURACY")

print(accuracy\_score(y\_test, clfpredict) \* 100)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, clfpredict))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, clfpredict))

models.append(('GradientBoostingClassifier', clf))

print("Random Forest Classifier")

from sklearn.ensemble import RandomForestClassifier

rf\_clf = RandomForestClassifier()

rf\_clf.fit(X\_train, y\_train)

rfpredict = rf\_clf.predict(X\_test)

print("ACCURACY")

print(accuracy\_score(y\_test, rfpredict) \* 100)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, rfpredict))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, rfpredict))

models.append(('RandomForestClassifier', rf\_clf))

classifier = VotingClassifier(models)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

Sender\_IP1 = [Sender\_IP]

vector1 = cv.transform(Sender\_IP1).toarray()

predict\_text = classifier.predict(vector1)

pred = str(predict\_text).replace("[", "")

pred1 = pred.replace("]", "")

prediction = int(pred1)

if (prediction == 0):

val = 'No Botnet DDOS Detection'

elif (prediction == 1):

val = 'Botnet DDOS Detection'

print(val)

print(pred1)

detect\_iot\_botnet\_attacks.objects.create(

Sender\_IP=Sender\_IP,

Sender\_Port=Sender\_Port,

Target\_Ip=Target\_Ip,

Target\_Port=Target\_Port,

Transport\_Protocol=Transport\_Protocol,

Duration=Duration,

AvgDuration=AvgDuration,

PBS=PBS,

AvgPBS=AvgPBS,

TBS=TBS,

PBR=PBR,

AvgPBR=AvgPBR,

TBR=TBR,

Missed\_Bytes=Missed\_Bytes,

Packets\_Sent=Packets\_Sent,

Packets\_Received=Packets\_Received,

SRPR=SRPR,

Prediction=val)

return render(request, 'RUser/Predict\_Drug\_Response.html',{'objs': val})

return render(request, 'RUser/Predict\_Drug\_Response.html')

**4.5.2 service provider**

**Admin.py:**

from django.contrib import admin

# Register your models here.

**Apps:**

from django.apps import AppConfig

class ResearchSiteConfig(AppConfig):

name = 'Service\_Provider'

**Model:**

from django.db import models

# Create your models here.

**Tests:**

from django.test import TestCase

# Create your tests here.

**Views:**

from django.db.models import Count, Avg

from django.shortcuts import render, redirect

from django.db.models import Count

from django.db.models import Q

import datetime

import xlwt

from django.http import HttpResponse

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.metrics import accuracy\_score

from sklearn.tree import DecisionTreeClassifier

# Create your views here.

from Remote\_User.models import ClientRegister\_Model,detect\_iot\_botnet\_attacks,detection\_ratio,detection\_accuracy

def serviceproviderlogin(request):

if request.method == "POST":

admin = request.POST.get('username')

password = request.POST.get('password')

if admin == "Admin" and password =="Admin":

detection\_accuracy.objects.all().delete()

return redirect('View\_Remote\_Users')

return render(request,'SProvider/serviceproviderlogin.html')

def View\_Drug\_Response\_Ratio(request):

detection\_ratio.objects.all().delete()

ratio = ""

kword = 'No Botnet DDOS Detection'

print(kword)

obj = detect\_iot\_botnet\_attacks.objects.all().filter(Q(Prediction=kword))

obj1 = detect\_iot\_botnet\_attacks.objects.all()

count = obj.count();

count1 = obj1.count();

ratio = (count / count1) \* 100

if ratio != 0:

detection\_ratio.objects.create(names=kword, ratio=ratio)

ratio12 = ""

kword12 = 'Botnet DDOS Detection'

print(kword12)

obj12 = detect\_iot\_botnet\_attacks.objects.all().filter(Q(Prediction=kword12))

obj112 = detect\_iot\_botnet\_attacks.objects.all()

count12 = obj12.count();

count112 = obj112.count();

ratio12 = (count12 / count112) \* 100

if ratio12 != 0:

detection\_ratio.objects.create(names=kword12, ratio=ratio12)

obj = detection\_ratio.objects.all()

return render(request, 'SProvider/View\_Drug\_Response\_Ratio.html', {'objs': obj})

def View\_Remote\_Users(request):

obj=ClientRegister\_Model.objects.all()

return render(request,'SProvider/View\_Remote\_Users.html',{'objects':obj})

def charts(request,chart\_type):

chart1 = detection\_ratio.objects.values('names').annotate(dcount=Avg('ratio'))

return render(request,"SProvider/charts.html", {'form':chart1, 'chart\_type':chart\_type})

def charts1(request,chart\_type):

chart1 = detection\_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))

return render(request,"SProvider/charts1.html", {'form':chart1, 'chart\_type':chart\_type})

def View\_Prediction\_Of\_Drug\_Response(request):

obj =detect\_iot\_botnet\_attacks.objects.all()

return render(request, 'SProvider/View\_Prediction\_Of\_Drug\_Response.html', {'list\_objects': obj})

def likeschart(request,like\_chart):

charts =detection\_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))

return render(request,"SProvider/likeschart.html", {'form':charts, 'like\_chart':like\_chart})

def Download\_Trained\_DataSets(request):

response = HttpResponse(content\_type='application/ms-excel')

# decide file name

response['Content-Disposition'] = 'attachment; filename="Predicted\_Datasets.xls"'

# creating workbook

wb = xlwt.Workbook(encoding='utf-8')

# adding sheet

ws = wb.add\_sheet("sheet1")

# Sheet header, first row

row\_num = 0

font\_style = xlwt.XFStyle()

# headers are bold

font\_style.font.bold = True

# writer = csv.writer(response)

obj = detect\_iot\_botnet\_attacks.objects.all()

data = obj # dummy method to fetch data.

for my\_row in data:

row\_num = row\_num + 1

ws.write(row\_num, 0, my\_row.Sender\_IP, font\_style)

ws.write(row\_num, 1, my\_row.Sender\_Port, font\_style)

ws.write(row\_num, 2, my\_row.Target\_Ip, font\_style)

ws.write(row\_num, 3, my\_row.Target\_Port, font\_style)

ws.write(row\_num, 4, my\_row.Transport\_Protocol, font\_style)

ws.write(row\_num, 5, my\_row.Duration, font\_style)

ws.write(row\_num, 6, my\_row.AvgDuration, font\_style)

ws.write(row\_num, 7, my\_row.PBS, font\_style)

ws.write(row\_num, 8, my\_row.AvgPBS, font\_style)

ws.write(row\_num, 9, my\_row.TBS, font\_style)

ws.write(row\_num, 10, my\_row.PBR, font\_style)

ws.write(row\_num, 11, my\_row.AvgPBR, font\_style)

ws.write(row\_num, 12, my\_row.TBR, font\_style)

ws.write(row\_num, 13, my\_row.Missed\_Bytes, font\_style)

ws.write(row\_num, 14, my\_row.Packets\_Sent, font\_style)

ws.write(row\_num, 15, my\_row.Packets\_Received, font\_style)

ws.write(row\_num, 16, my\_row.SRPR, font\_style)

ws.write(row\_num, 17, my\_row.Prediction, font\_style)

wb.save(response)

return response

def train\_model(request):

detection\_accuracy.objects.all().delete()

df = pd.read\_csv('Datasets.csv')

def apply\_response(Label):

if (Label == 0):

return 0 # Bad

elif (Label == 1):

return 1

df['results'] = df['Label'].apply(apply\_response)

cv = CountVectorizer()

X = df['Sender\_IP']

y = df['results']

print("Sender\_IP")

print(X)

print("Results")

print(y)

cv = CountVectorizer()

X = cv.fit\_transform(X)

models = []

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

X\_train.shape, X\_test.shape, y\_train.shape

print(X\_test)

# SVM Model

print("SVM")

from sklearn import svm

lin\_clf = svm.LinearSVC()

lin\_clf.fit(X\_train, y\_train)

predict\_svm = lin\_clf.predict(X\_test)

svm\_acc = accuracy\_score(y\_test, predict\_svm) \* 100

print(svm\_acc)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, predict\_svm))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, predict\_svm))

models.append(('svm', lin\_clf))

detection\_accuracy.objects.create(names="SVM", ratio=svm\_acc)

print("Logistic Regression")

from sklearn.linear\_model import LogisticRegression

reg = LogisticRegression(random\_state=0, solver='lbfgs').fit(X\_train, y\_train)

y\_pred = reg.predict(X\_test)

print("ACCURACY")

print(accuracy\_score(y\_test, y\_pred) \* 100)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, y\_pred))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, y\_pred))

models.append(('logistic', reg))

detection\_accuracy.objects.create(names="Logistic Regression", ratio=accuracy\_score(y\_test, y\_pred) \* 100)

print("Gradient Boosting Classifier")

from sklearn.ensemble import GradientBoostingClassifier

clf = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1.0, max\_depth=1, random\_state=0).fit(

X\_train,

y\_train)

clfpredict = clf.predict(X\_test)

print("ACCURACY")

print(accuracy\_score(y\_test, clfpredict) \* 100)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, clfpredict))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, clfpredict))

models.append(('GradientBoostingClassifier', clf))

detection\_accuracy.objects.create(names="Gradient Boosting Classifier",

ratio=accuracy\_score(y\_test, clfpredict) \* 100)

print("Random Forest Classifier")

from sklearn.ensemble import RandomForestClassifier

rf\_clf = RandomForestClassifier()

rf\_clf.fit(X\_train, y\_train)

rfpredict = rf\_clf.predict(X\_test)

print("ACCURACY")

print(accuracy\_score(y\_test, rfpredict) \* 100)

print("CLASSIFICATION REPORT")

print(classification\_report(y\_test, rfpredict))

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, rfpredict))

models.append(('RandomForestClassifier', rf\_clf))

detection\_accuracy.objects.create(names="Random Forest Classifier", ratio=accuracy\_score(y\_test, rfpredict) \* 100)

csv\_format = 'Results.csv'

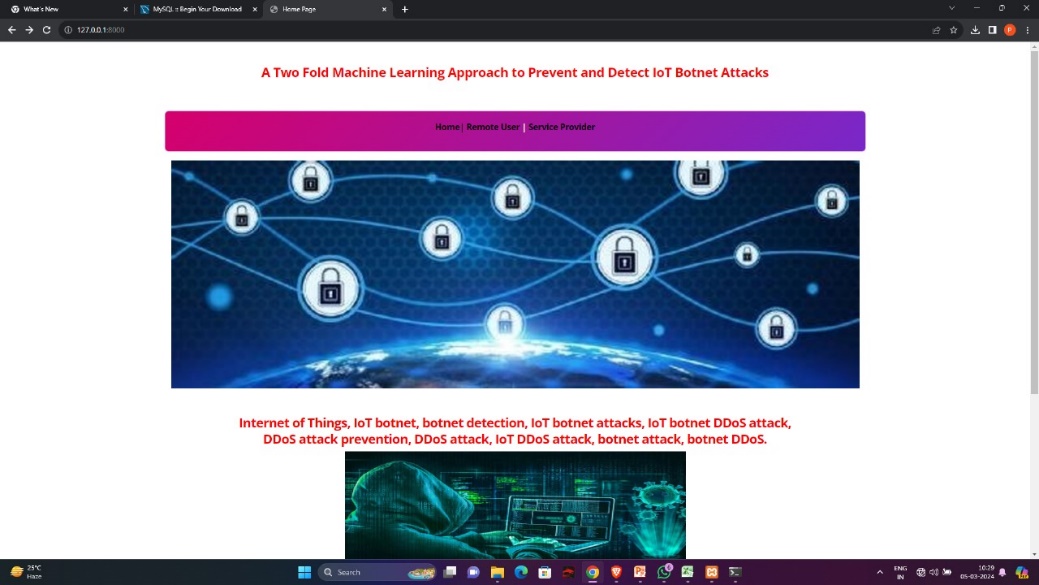
df.to\_csv(csv\_format, index=False)

df.to\_markdown

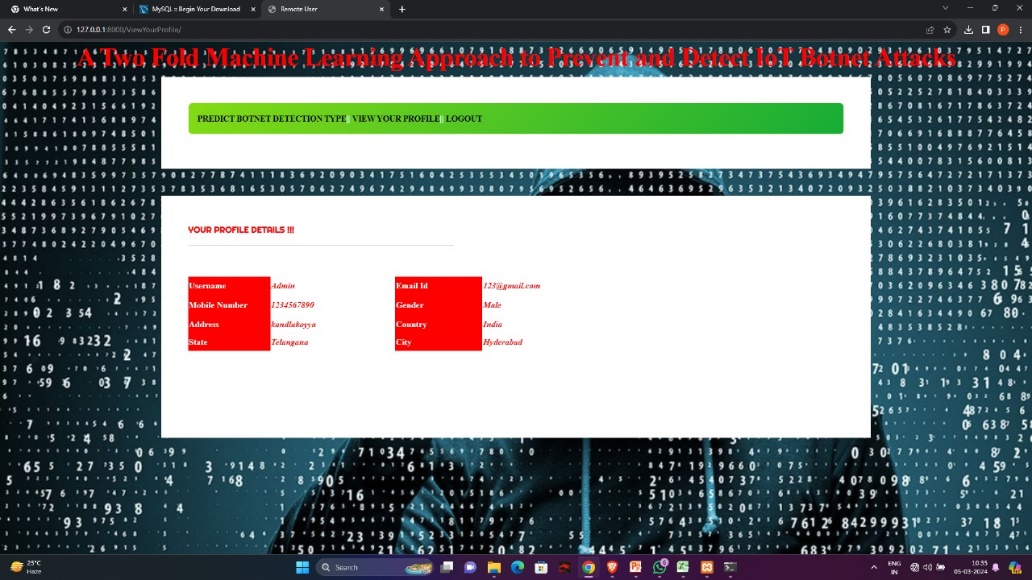
obj = detection\_accuracy.objects.all()

return render(request,'SProvider/train\_model.html', {'objs': obj})

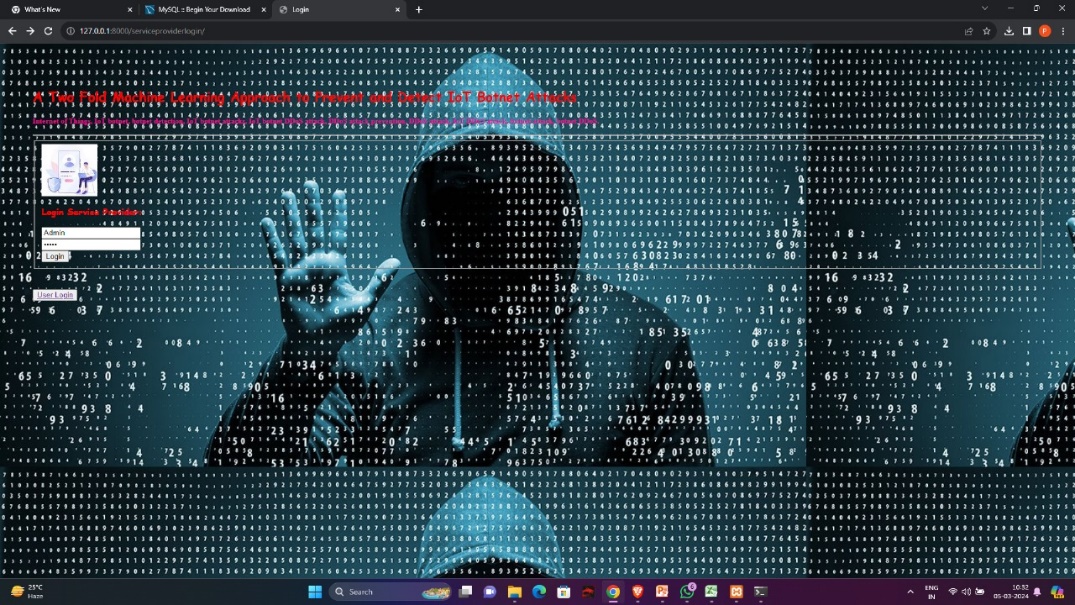
## 2.SCREENSHOTS



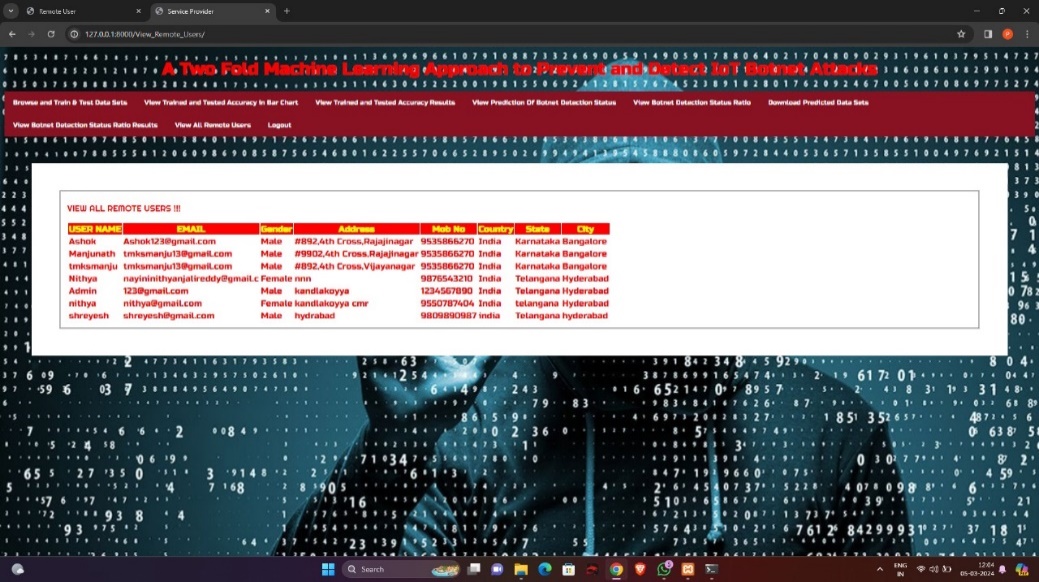
Screenshot 5.1: Home page



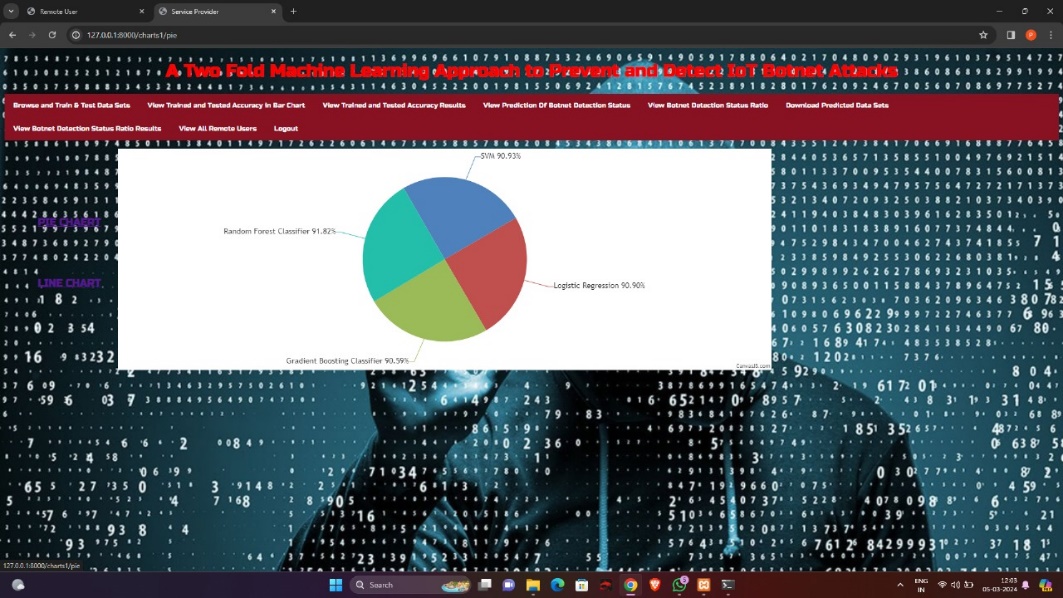
Screenshot 5.2: Remote user



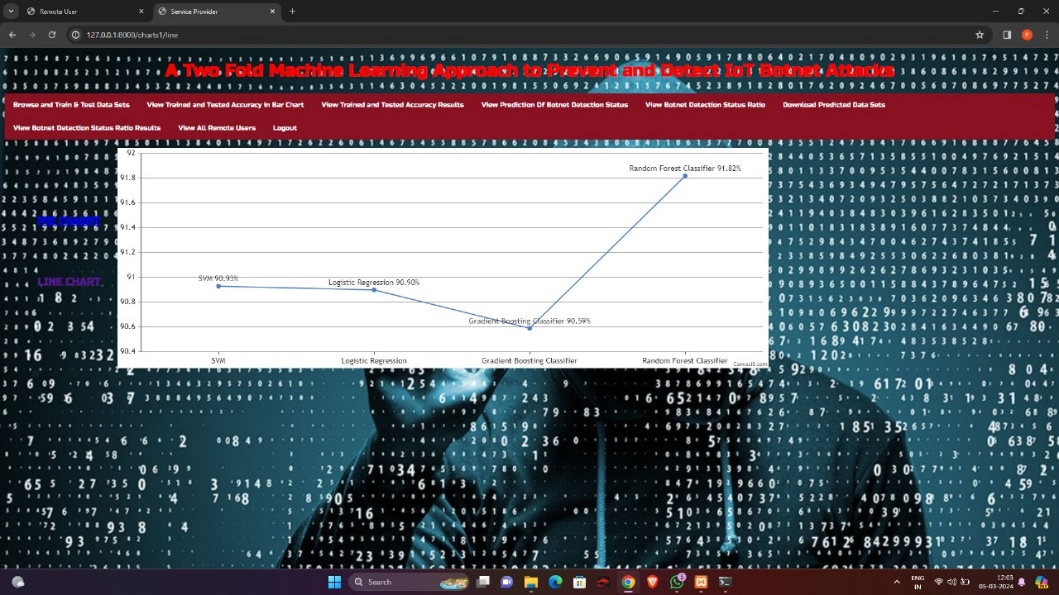
Screenshot 5.3: Admin login



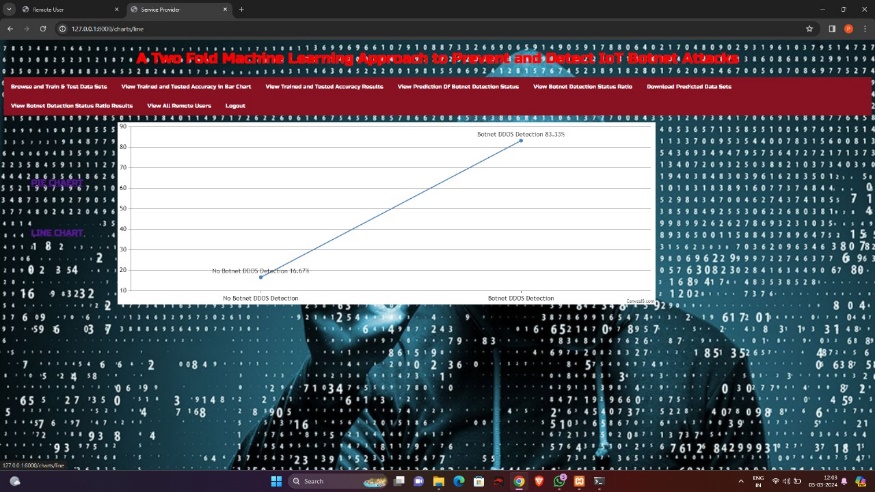
Screenshot 5.4: Remote users



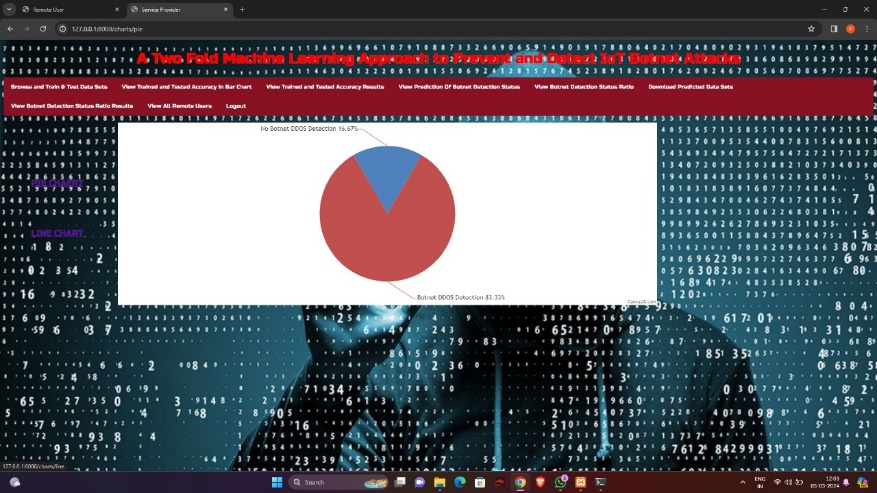
Screenshot 5.5: Algorithm accuracy(Pie chart)



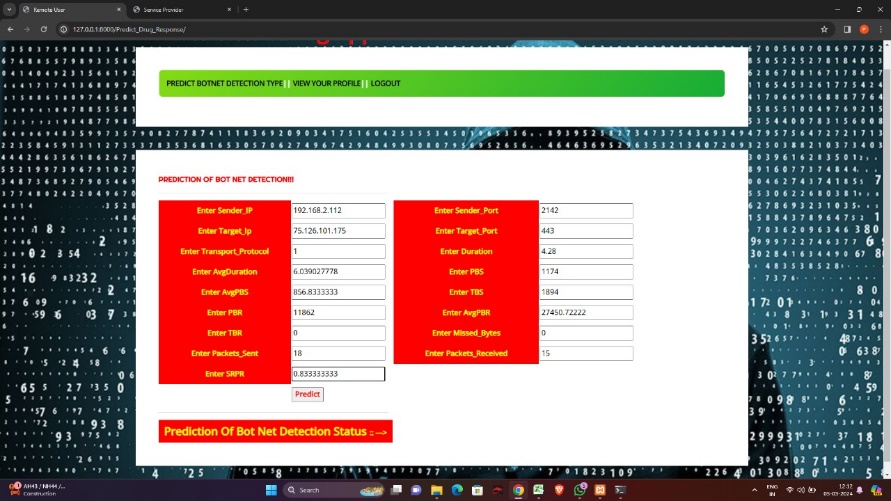
Screenshot 5.6: Algorithm accuracy(line chart)



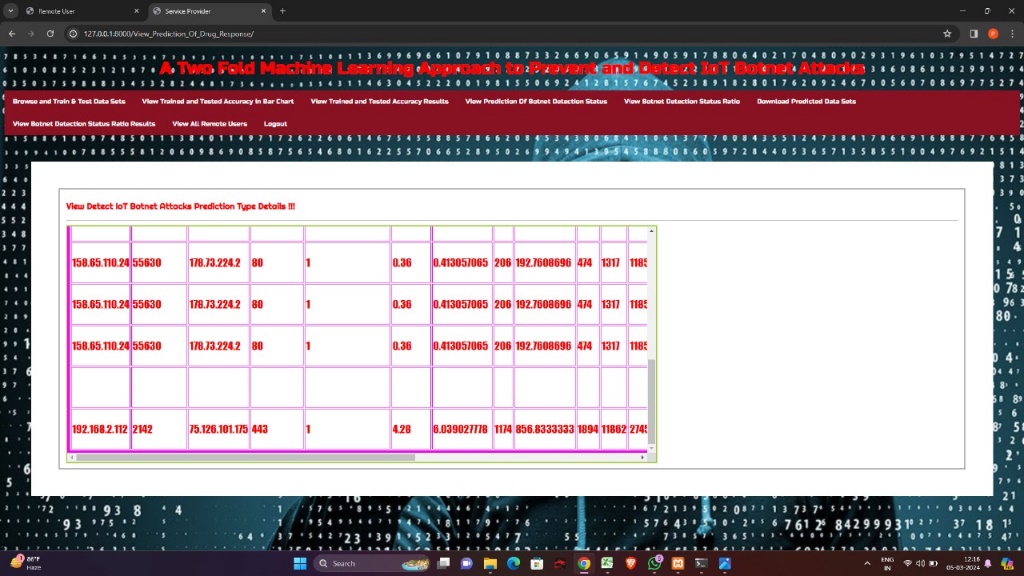
Screenshot 5.7: Botnet accuracy(line chart)



Screenshot 5.8: Botnet accuracy(pie chart)



Screenshot 5.9: Botnet detection



Screenshot 5.10: Botnet detection type



Screenshot 5.11: Botnet attack ratio

## 6.TESTING

##### INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

##### 6.2 TYPES OF TESTING

###### 6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

###### 6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

###### 6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

Features to be tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### 7.CONCLUSION

##### 7.CONCLUSION & FUTURE SCOPE

##### PROJECT CONCLUSION

##### In this study, we introduced a two-fold machine learning approach to prevent and detect IoT botnet attacks. We trained ResNet-18 models, ResNetScan-1 and ResNetDDoS-1, for scanning attack detection and DDoS attack detection respectively. Experimental results demonstrated the superior performance of our proposed models compared to existing ones, even when tested on datasets they were not trained on. This underscores the effectiveness and robustness of our approach in detecting a wide range of attack patterns.

##### FUTURE SCOPE

While our current work covers a substantial number of scanning and DDOS attack techniques, future endeavors will aim to expand this coverage for more comprehensive training of our framework. Additionally, deploying our two-fold approach within an Intrusion Detection System (IDS) will enable us to assess its effectiveness in real-world scenarios, analyzing live network traffic to further validate its capabilities in detecting and preventing IoT botnet and DDOS attacks.

### 8.BIBLIOGRAPHY

##### . BIBLIOGRAPHY

##### REFERENCES

* S. Sikkanan and M. Kasthuri, ‘‘Denial-of-service and botnet analysis, detection, and mitigation,’’ in Forensic Investigations and Risk Management in Mobile and Wireless Communications. Hershey, PA, USA: IGI Global, 2020, pp. 114–151.
* F. Tang, Y. Kawamoto, N. Kato, K. Yano, and Y. Suzuki, ‘‘Probe delay based adaptive port scanning for IoT devices with private IP address behind NAT,’’ IEEE Netw., vol. 34, no. 2, pp. 195–201, Mar. 2020
* S. Dange and M. Chatterjee, ‘‘IoT botnet: The largest threat to the IoT network,’’ in Data Communication and Networks. Singapore: Springer, 2020, pp. 137–157.
  + 1. **GITHUB LINK**