A

Major Project

On

**AN EFFICIENT SPAM DETECTION TECHNIQUE FOR IOT DEVICES USING MACHINE LEARNING**

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

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

G.VENKATESHWARLU (207R1A05L3)

Under the guidance of

**DR.K. SRUJAN RAJU**

(Professor of CSE)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## CMR TECHNICAL CAMPUS UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) 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**



**CERTIFICATE**

This is to certify that the project entitled **“AN EFFICIENT SPAM DETECTION TECHNIQUE FOR IOT DEVICES USING MACHINE LEARNING ”** being submitted by **G.VENKATESHWARLU (207R1A05L3)** in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering 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 2023-2024.

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

|  |  |
| --- | --- |
| **Dr. K. Srujan Raju**  (Professor of CSE) | **Dr. A. Raji Reddy**  **DIRECTOR** |
| **INTERNAL GUIDE** |  |
| **Dr. K. Srujan Raju**  HOD | **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 **Dr . K.Srujan Raju,** Professor of CSEfor 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) **G.Vinesh Shanker, Dr. J. Narasimharao, Ms. Shilpa, & Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering 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 course of 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.

**G. VENKATESWARLU (207R1A05L3)**

# ABSTRACT

"The Internet of Things (IoT) consists of millions of connected devices with sensors and actuators that communicate data through wired or wireless channels. IoT has grown rapidly in the past decade, with over 25 billion devices expected to be connected by 2020. These devices generate a large volume of data, which will continue to increase. This data comes in various forms and qualities, depending on its speed and location. Machine learning can enhance security, detect anomalies, and improve the usability of IoT systems. However, attackers can also exploit machine learning in IoT systems.

This data varies in terms of how quickly it's generated and where it's coming from. Sometimes, it's not easy to tell if the data is trustworthy or if there's any malicious or spammy information mixed in. This is where machine learning, which is a type of computer program that can learn and make decisions, comes into play. It can help ensure the security of these IoT devices, detect anomalies, and make the IoT systems safer to use.

We propose a security solution for IoT devices by using machine learning to detect spam. We introduce a framework called 'Spam Detection in IoT using Machine Learning,' which evaluates five machine learning models using different metrics and input feature sets. Each model assigns a spam score to IoT devices, indicating their trustworthiness under various conditions. We validate our approach using the REFIT Smart Home dataset and demonstrate its effectiveness compared to existing methods."

In the ever-expanding landscape of the IoT, safeguarding against spam and ensuring the security of these devices is paramount. By leveraging machine learning, this framework offers a promising solution to enhance the overall integrity and trustworthiness of IoT systems."

**LIST OF FIGURES/TABLES**

## FIGURE NO FIGURE NAME PAGE NO

Figure 3.1 Project Architecture For Spam 8

Detection In IoT Devices

|  |  |  |
| --- | --- | --- |
| Figure 3.2 | Use Case Diagram For Spam  Detection In IoT Devices | 10 |
| Figure 3.3 | Class Diagram For Spam  Detection In IoT Devices | 11 |
| Figure 3.4 | Sequence Diagram For Spam  Detection In Iot Devices | 12 |
| Figure 3.5 | Activity Diagram For Spam  Detection In Iot Devices | 13 |

**LIST OF SCREENSHOTS**

|  |  |  |
| --- | --- | --- |
| **SCREENSHOT NO.** | **SCREENSHOT NAME** | **PAGE NO**. |
| Screenshot 5.1 | Open Xampp And Start Apache And Mysql Admin | 29 |
| Screenshot 5.2 | Open Command Prompt | 29 |
| Screenshot 5.3 | Spam Detection Model | 30 |
| Screenshot 5.4 | Remote User Login | 30 |
| Screenshot 5.5 | View Your Profile In Remote User | 31 |
| Screenshot 5.6 | Predicting The IoT Message | 31 |
| Screenshot 5.7 | Register For Remote User | 32 |
| Screenshot 5.8 | Login For Service Provider | 32 |
| Screenshot 5.9 | Train And Test For Spam Datasets | 33 |
| Screenshot 5.10 | View Spam Datasets Of Trained And Tested Accuracy In Bar Chart | 33 |
| Screenshot 5.11 | View Tweet Datasets Trained And Tested Accuracy Results | 34 |
| Screenshot 5.12 | View Predicted Spam Detection Type | 34 |
| Screenshot 5.13 | Spam Detection Type Ratio | 35 |
| Screenshot 5.14 | View Spam Detection Ratio Results | 35 |
| Screenshot 5.15 | View All Remote Users | 36 |

**TABLE OF CONTENTS**

**ABSTRACT** i

**LIST OF FIGURES** ii

**LIST OF SCREENSHOTS** iii

1. **INTRODUCTION** 1

1.1 PROJECT SCOPE 1

1.2 PROJECT PURPOSE 1

1.3 PROJECT FEATURES 1

1. **SYSTEM ANALYSIS** 2

2.1 INTRODUCTION 2

2.2 PROBLEM DEFINITION 2

2.3 EXISTING SYSTEM 2

2.3.1 LIMITATIONS OF THE EXISTING SYSTEM 3

2.4 PROPOSED SYSTEM 4

2.4.1 ADVANTAGES OF PROPOSED SYSTEM 5

2.5 FEASIBILITY STUDY 5

2.5.1 ECONOMIC FEASIBILITY 5

2.5.2 TECHNICAL FEASIBILITY 6

2.5.3 SOCIAL FEASIBILITY 6

2.6 HARDWARE & SOFTWARE REQUIREMENTS 6

2.6.1 HARDWARE REQUIREMENTS 6

2.6.2 SOFTWARE REQUIREMENTS 7

1. **ARCHITECTURE** 8

3.1 PROJECT ARCHITECTURE 8

3.2 DESCRIPTION 9

3.3 USECASE DIAGRAM 9

3.4 CLASS DIAGRAM 11

3.5 [SEQUENCE DIAGRAM 12](#_TOC_250011)

3.6 [ACTIVITY DIAGRAM 13](#_TOC_250010)

4. IMPLEMENTATION 14

4.1 SAMPLE CODE 14

5.SCREENSHOTS 29

6.[TESTING](#_TOC_250009) 37

6.1 [INTRODUCTION TO TESTING](#_TOC_250008) 37

6.2 [TYPES OF TESTING 37](#_TOC_250007)

6.2.1 UNIT TESTING 37

6.2.2 [INTEGRATION TESTING 37](#_TOC_250006)

6.2.3 [FUNCTIONAL TESTING 38](#_TOC_250005)

* 1. [TEST CASES 39](#_TOC_250004)

[6.3.1 CLASSIFICATION 39](#_TOC_250003)

7.CONCLUSION & FUTURE SCOPE 40

7.1 PROJECT CONCLUSION 40

7.2 [FUTURE SCOPE 40](#_TOC_250002)

8.BIBLIOGRAPHY 41

8.1 [REFERENCES 41](#_TOC_250001)

8.2 [GITHUB LINK 41](#_TOC_250000)

1. **INTRODUCTION**

**1.INTRODUCTION**

### 1.1 PROJECT SCOPE

This project is titled “An Efficient Spam Detection Technique for IoT Devices

Using Machine Learning”. We can say with the increasing of accuracy and effectiveness of the project, the machine learning algorithms can easily predict many types of Spam attacks. As a result, we can decrease false-positive rate through this model. By considering many machine learning algorithms, we can find the spams in one or other way. It gives Robust features such that it can predict different types of attacks of spam.

### 1.2 PROJECT PURPOSE

### The project could lead to the integration of machine learning-based spam detection techniques into existing security products and services. This integration can enhance the overall cybersecurity. Enhance project's ability to detect and respond to spam attacks in real-time. This can involve optimizing algorithms for implementing streaming data processing, and integrating with network monitoring tools.

### 1.3 PROJECT FEATURES

Exploring spam detection techniques to identify network traffic patterns that might indicate spam attacks. Experiment with data augmentation techniques to increase the diversity and size of your training dataset. This can lead to better generalization and robustness of machine learning models. Conduct extensive testing of the system in a controlled environment to assess its scalability and performance under heavy spam attack scenario.

**2. SYSTEM ANALYSIS**

### 2.SYSTEM ANALYSIS

### 2.1 INTRODUCTION

System Analysis is the important phase in the system development process. The System is studied to the minute details and analysed. 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 has a firm understanding of what is to be done.

### 2.2 PROBLEM DEFINITION

A general statement of spam attack problem can be formulated in the given IOT devices, identify or verify one or more spam attack in the scene or in the network devices using a stored database.

### 2.3 EXISTING SYSTEM

In the realm of IoT (Internet of Things) devices, a multitude of security challenges loom, primarily in the form of cyberattacks. Among these, Denial of Service (DDoS) attacks stand out as a pervasive menace, capable of crippling IoT services. DDoS attacks function by inundating target databases with an overwhelming barrage of undesired requests, typically orchestrated through a network of compromised IoT devices acting as "bots." The relentless onslaught of these malicious requests exhausts the resources allocated by service providers, effectively obstructing legitimate user access and rendering network resources entirely inaccessible. Another notable threat is posed by RFID (Radio-Frequency Identification) attacks, which operate at the physical layer of IoT devices, jeopardizing data integrity. These attacks involve an array of insidious techniques, aimed at tampering with data either while stored within the device or during its transmission across the network. Common attack vectors include attempts to disrupt availability, authenticity, confidentiality, and the audacious endeavor to brute-force cryptography keys. Countering such threats necessitates the implementation of rigorous security measures such as password protection, data encryption, and stringent access control.

The interconnected nature of IoT devices also exposes them to Internet attacks, masterminded by spammers and cybercriminals. These malicious actors harbor intentions of either pilfering sensitive information from other systems or orchestrating incessant traffic to their target websites, typically through the deployment of spamming techniques. A prominent strategy in this realm is Ad fraud, which manufactures artificial clicks on specific websites with the sole purpose of monetary gain, emblematic of the activities perpetrated by cybercriminals. Furthermore, NFC (Near Field Communication) attacks have surfaced as a substantial concern, primarily entailing electronic payment fraud. These attacks encompass a spectrum of nefarious activities, including the manipulation of unencrypted traffic, eavesdropping on sensitive data, and the modification of NFC tags. The solution to these security challenges often lies in the implementation of conditional privacy protection. This approach ensures that attackers struggle to replicate user profiles, as it relies on the use of random public keys generated by trusted service managers. The comprehensive threat landscape highlights the critical need for robust and proactive security measures within IoT ecosystems to safeguard the integrity and functionality of connected devices.

#### 2.3.1 DISADVANTAGES OF EXISTING SYSTEM

* The system is less effective.
* Less Accurate.
* This system is more time taking process.
* There is Absence of Supervised Machine Learning Algorithm.
* Easily Attacked through Unauthorised Users.
* It cannot detect the Attackers.

### 2.4 PROPOSED SYSTEM

The digital realm has become intricately intertwined with the proliferation of smart devices, and the integrity of the information harnessed from these devices is paramount. In the multifaceted landscape of the Internet of Things (IoT), where data is drawn from diverse domains, the process of extracting valuable insights presents itself as a formidable challenge. The very essence of IoT lies in the multitude of interconnected devices, resulting in the generation of copious volumes of data that exhibit a distinct and intricate nature characterized by its heterogeneity and diversity. This amalgamation of data sources is collectively referred to as IoT data.

IoT data possesses a constellation of distinctive features, which include real-time data streaming, a multifaceted origin from various sources, richness in content, and, intriguingly, moments of sparsity. The real-time aspect denotes the dynamic nature of this data, as it flows continuously from the myriad sensors and devices within the IoT ecosystem. Furthermore, the multi-source attribute emphasizes that this data emanates from a wide array of devices with differing functionalities, thus enriching the data landscape. The richness of IoT data lies in its capacity to encapsulate a wide spectrum of information, ranging from temperature readings and sensor data to location information and user interactions. However, this wealth of data is counterbalanced by moments of sparsity, where certain data points may be missing or incomplete.

Preserving the integrity and trustworthiness of IoT data is of paramount importance, and one significant aspect of this task is ensuring that it remains devoid of spam or unwanted content. The challenge of spam detection within the IoT environment, given the intricate and multifaceted nature of IoT data, underscores the need for sophisticated and adaptive techniques to safeguard the quality and reliability of information in the digital world.

The proposed system of spam detection is validated using five different machine learning models. An algorithm is proposed to compute the spamicity score of each model which is then used for detection and intelligent decision making. Based upon the spamicity score computed in previous step, the reliability of IoT devices is analyzed using different evaluation metrics. By using supervised machine learning algorithm to detect a spam, by implementing an efficient code in various IoT devices. Here, giving the system with better security and firewalls that will not give access to unauthorized users.

#### 2.4.1 ADVANTAGES OF THE PROPOSED SYSTEM

* Provides Good results by using a supervised machine learning algorithm.
* More Accurate.
* The system is more efficient.
* It takes less time to detect spams.
* It can easily detect the Attackers.

**2.5 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:

● Economic Feasibility

● Technical Feasibility

● Social Feasibility

**2.5.1 ECONOMICAL 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.

### 2.5.2 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.

**2.5.3 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.

### 2.6 HARDWARE & SOFTWARE REQUIREMENTS

**2.6.1 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 : Intel I3 or Above.
* RAM : 4GB or Above.
* Hard Disk : 40 GB.
* Key Board : Standard Keyboard.
* Mouse : Two or Three Button Mouse.
* Monitor : 15 inch VGA Color.
* CPU : 1GHZ.

**2.6.2 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 or Above.
* Coding Language : Python.
* Front-End : Html ,Css, Javascript.
* Back-End : MySQL.
* Framework : Django-ORM.
* Platform : Python Technology.
* Tool : Spyder, Python 3.5.

**3. ARCHITECTURE**

**3.ARCHITECTURE**

**3.1 PROJECT ARCHITECTURE**

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

Service Provider

Upload Post,

View All Uploaded Posts,

View Rating Results,

View Dislike Results,

View Like Results,

View Remote Users,

View Posts Review,

View Trending Posts,

View All Recommended Posts,

View All Spam Detection Details

**Admin**

Accepting all user Information

View user data details

Registering the User

Authorize the Admin

Process all the user queries

**Web Database**

REMOTE USER Server

Tweet Server

REGISTER AND LOGIN,

SEARCH AND VIEW ALL POSTS,

VIEW ALL POST REVIEWS,

VIEW TRENDING POSTS,

VIEW YOUR PROFILE,

VIEW ALL POSTS RECOMMENDED.

**STORE AND RETRIEVALS**

Figure 3.1: Project Architecture for Spam Detection in IoT Devices

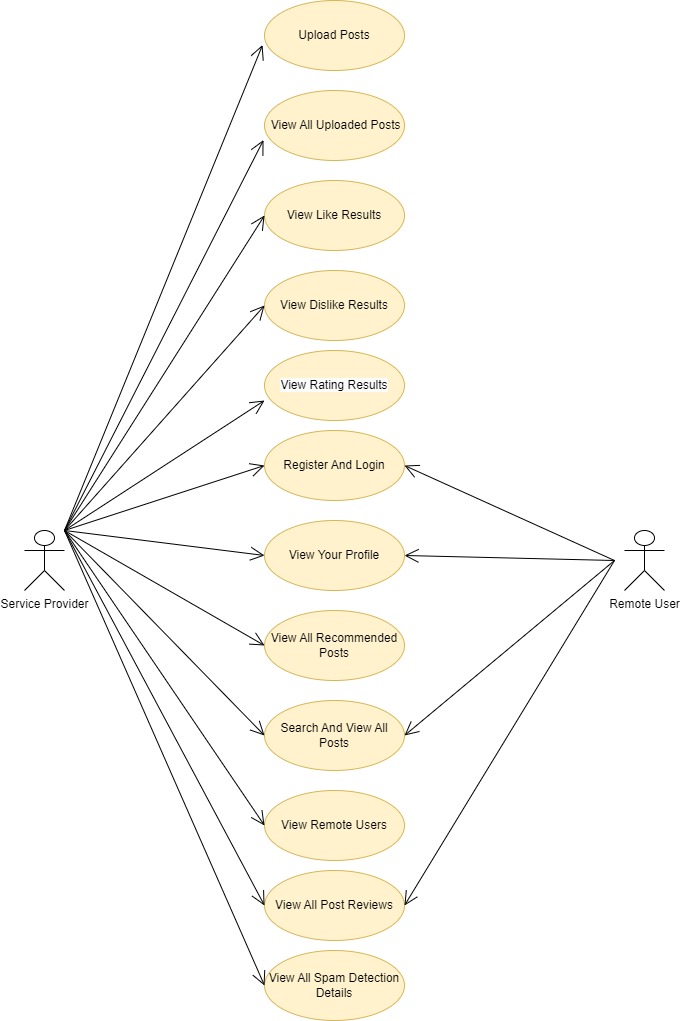
#### 3.2 DESCRIPTION

The use of machine learning algorithms such as SVM, Logistic regression, and Naive Bayes for spam detection in IoT is a novel approach that has gained significant attention in recent years. These algorithms are capable of detecting and preventing attacks in real-time, making them more effective than traditional methods. They can also adapt to changing attack patterns, making them more tough against evolving threats. These algorithms have ability to learn from large amounts of data and identify patterns that may not be immediately apparent to human analysts. This allows them to detect and respond to attacks more quickly and accurately, reducing the risk of damage to IoT devices and networks.

**3.3 USE CASE DIAGRAM**

In the use case diagram, we have basically one actor who is the user in the trained model.

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.



.

Figure 3.2: Use Case Diagram for An Efficient Spam Detection Technique for IoT Devices

### 3.4 CLASS DIAGRAM

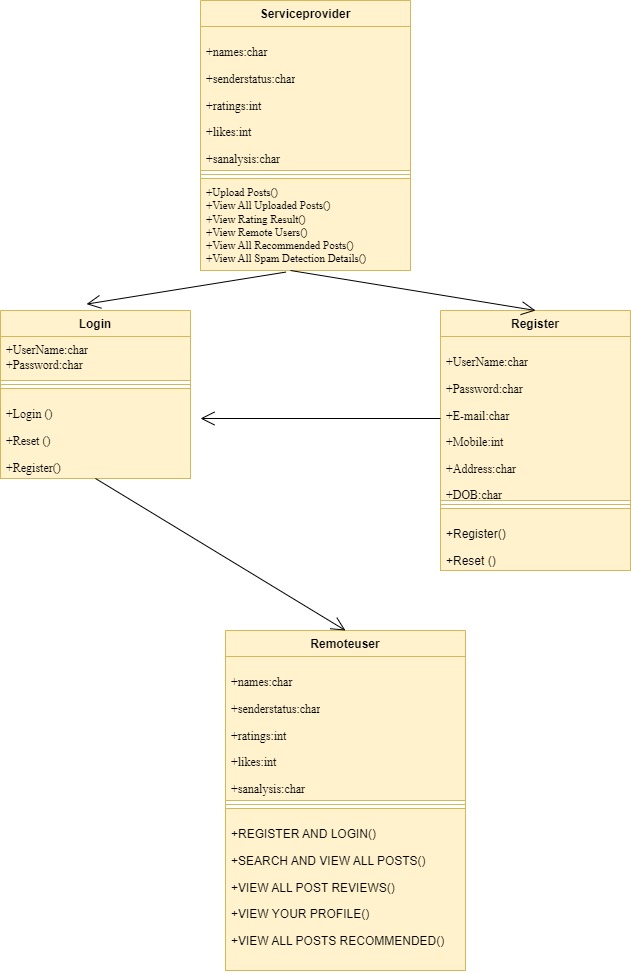
Class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among objects.

Figure 3.3: Class Diagram for An Efficient Spam Detection Technique for IoT Devices

**3.5 SEQUENCE DIAGRAM**

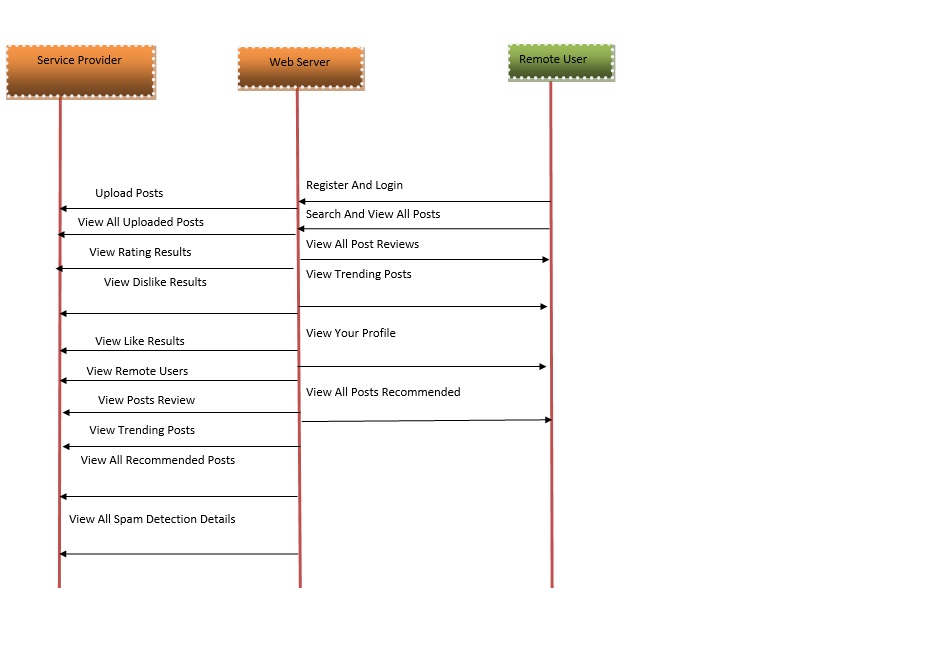
A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

Figure 3.4: Sequence Diagram for An Efficient Spam Detection Technique for IoT Devices

#### 3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

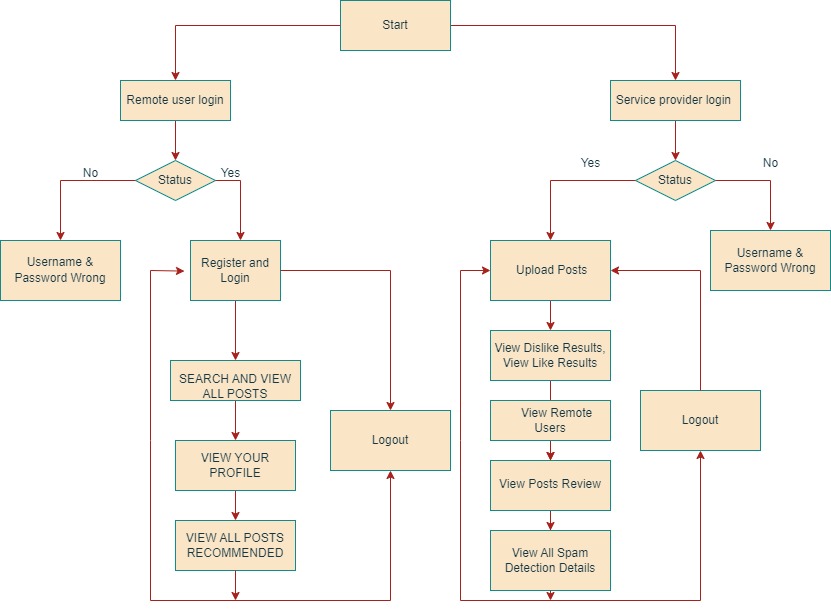


Figure 3.5: Activity Diagram for An Efficient Spam Detection Technique for IoT Devices

**4. IMPLEMENTATION**

**4.1 SAMPLE CODE**

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")

from django.db import models

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)

class Spam\_Prediction(models.Model):

Message\_Id= models.CharField(max\_length=300)

IOT\_Message= models.CharField(max\_length=300000)

Message\_Date= models.CharField(max\_length=300)

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

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)

from django.db.models import Count

from django.db.models import Q

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

import datetime

import openpyxl

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

from wordcloud import WordCloud

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

#NLP tools

import re

import nltk

nltk.download('stopwords')

nltk.download('rslp')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from sklearn.feature\_extraction.text import CountVectorizer

#train split and fit models

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from nltk.tokenize import TweetTokenizer

from sklearn.ensemble import VotingClassifier

#model selection

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

fromRemote\_User.modelsimport ClientRegister\_Model,Spam\_Prediction,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 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')

ClientRegister\_Model.objects.create(username=username,email=email, password=password, phoneno=phoneno,country=country, state=state, city=city)

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

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\_IOT\_Message\_Type(request):

if request.method == "POST":

Tweet\_Message = request.POST.get('Tweet\_Message')

if request.method == "POST":

Tweet\_Message = request.POST.get('Tweet\_Message')

Message\_Id = request.POST.get('Message\_Id')

Message\_Date = request.POST.get('Message\_Date')

IOT\_Message = request.POST.get('IOT\_Message')

data = pd.read\_csv("IOT\_Datasets.csv")

# data.replace([np.inf, -np.inf], np.nan, inplace=True)

mapping = {'ham': 0, 'spam': 1}

data['Results'] = data['Label'].map(mapping)

x = data['Message']

y = data['Results']

# data.drop(['Type\_of\_Breach'],axis = 1, inplace = True)

cv = CountVectorizer()

print(x)

print(y)

x = cv.fit\_transform(data['Message'].apply(lambda x: np.str\_(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("Naive Bayes")

from sklearn.naive\_bayes import MultinomialNB

NB = MultinomialNB()

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

predict\_nb = NB.predict(X\_test)

naivebayes = accuracy\_score(y\_test, predict\_nb) \* 100

print(naivebayes)

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

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

models.append(('naive\_bayes', NB))

# 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))

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("Decision Tree Classifier")

dtc = DecisionTreeClassifier()

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

dtcpredict = dtc.predict(X\_test)

print("ACCURACY")

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

print("CLASSIFICATION REPORT")

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

print("CONFUSION MATRIX")

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

classifier = VotingClassifier(models)

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

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

IOT\_Message1 = [IOT\_Message]

vector1 = cv.transform(IOT\_Message1).toarray()

predict\_text = classifier.predict(vector1)

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

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

prediction = int(pred1)

if prediction == 0:

val = 'Normal'

elif prediction == 1:

val = 'Spam'

Spam\_Prediction.objects.create(Message\_Id=Message\_Id,IOT\_Message=IOT\_Message,Message\_Date=Message\_Date, Prediction=val)

return render(request, 'RUser/Predict\_IOT\_Message\_Type.html',{'objs': val})

return render(request, 'RUser/Predict\_IOT\_Message\_Type.html')

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 numpy as np

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

from wordcloud import WordCloud

from sklearn.pipeline import Pipeline

#to data preprocessing

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

#NLP tools

import re

import nltk

nltk.download('stopwords')

nltk.download('rslp')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from sklearn.feature\_extraction.text import CountVectorizer

#train split and fit models

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from nltk.tokenize import TweetTokenizer

from sklearn.ensemble import VotingClassifier

#model selection

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

fromRemote\_User.modelsimport ClientRegister\_Model,Spam\_Prediction,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\_IOTMessage\_Type\_Ratio(request):

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

rratio = ""

kword = 'Spam'

print(kword)

obj = Spam\_Prediction.objects.all().filter(Q(Prediction=kword))

obj1 = Spam\_Prediction.objects.all()

count = obj.count();

count1 = obj1.count();

ratio = (count / count1) \* 100

if ratio != 0:

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

ratio1 = ""

kword1 = 'Normal'

print(kword1)

obj1 = Spam\_Prediction.objects.all().filter(Q(Prediction=kword1))

obj11 = Spam\_Prediction.objects.all()

count1 = obj1.count();

count11 = obj11.count();

ratio1 = (count1 / count11) \* 100

if ratio1 != 0:

detection\_ratio.objects.create(names=kword1, ratio=ratio1)

obj = detection\_ratio.objects.all()

return render(request, 'SProvider/View\_IOTMessage\_Type\_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 ViewTrendings(request):

topic = Spam\_Prediction.objects.values('topics').annotate(dcount=Count('topics')).order\_by('-dcount')

return render(request,'SProvider/ViewTrendings.html',{'objects':topic})

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\_IOTMessage\_Type(request):

obj =Spam\_Prediction.objects.all()

return render(request, 'SProvider/View\_Prediction\_Of\_IOTMessage\_Type.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\_Data.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 = Spam\_Prediction.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.Message\_Id, font\_style)

ws.write(row\_num, 1, my\_row.Message\_Date, font\_style)

ws.write(row\_num, 2, my\_row.IOT\_Message, font\_style)

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

wb.save(response)

return response

def train\_model(request):

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

data = pd.read\_csv("IOT\_Datasets.csv")

# data.replace([np.inf, -np.inf], np.nan, inplace=True)

mapping = {'ham': 0,

'spam': 1

}

data['Results'] = data['Label'].map(mapping)

x = data['Message']

y = data['Results']

# data.drop(['Type\_of\_Breach'],axis = 1, inplace = True)

cv = CountVectorizer()

print(x)

print(y)

x = cv.fit\_transform(data['Message'].apply(lambda x: np.str\_(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("Naive Bayes")

from sklearn.naive\_bayes import MultinomialNB

NB = MultinomialNB()

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

predict\_nb = NB.predict(X\_test)

naivebayes = accuracy\_score(y\_test, predict\_nb) \* 100

print("ACCURACY")

print(naivebayes)

print("CLASSIFICATION REPORT")

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

print("CONFUSION MATRIX")

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

detection\_accuracy.objects.create(names="Naive Bayes", ratio=naivebayes)

# 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))

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))

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

print("Decision Tree Classifier")

dtc = DecisionTreeClassifier()

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

dtcpredict = dtc.predict(X\_test)

print("ACCURACY")

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

print("CLASSIFICATION REPORT")

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

print("CONFUSION MATRIX")

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

detection\_accuracy.objects.create(names="DecisionTreeClassifier", ratio=accuracy\_score(y\_test, dtcpredict) \* 100)

print("SGD Classifier")

from sklearn.linear\_model import SGDClassifier

sgd\_clf = SGDClassifier(loss='hinge', penalty='l2', random\_state=0)

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

sgdpredict = sgd\_clf.predict(X\_test)

print("ACCURACY")

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

print("CLASSIFICATION REPORT")

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

print("CONFUSION MATRIX")

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

detection\_accuracy.objects.create(names="SGDClassifier", ratio=accuracy\_score(y\_test, sgdpredict) \* 100)

labeled = 'Processed\_data.csv'

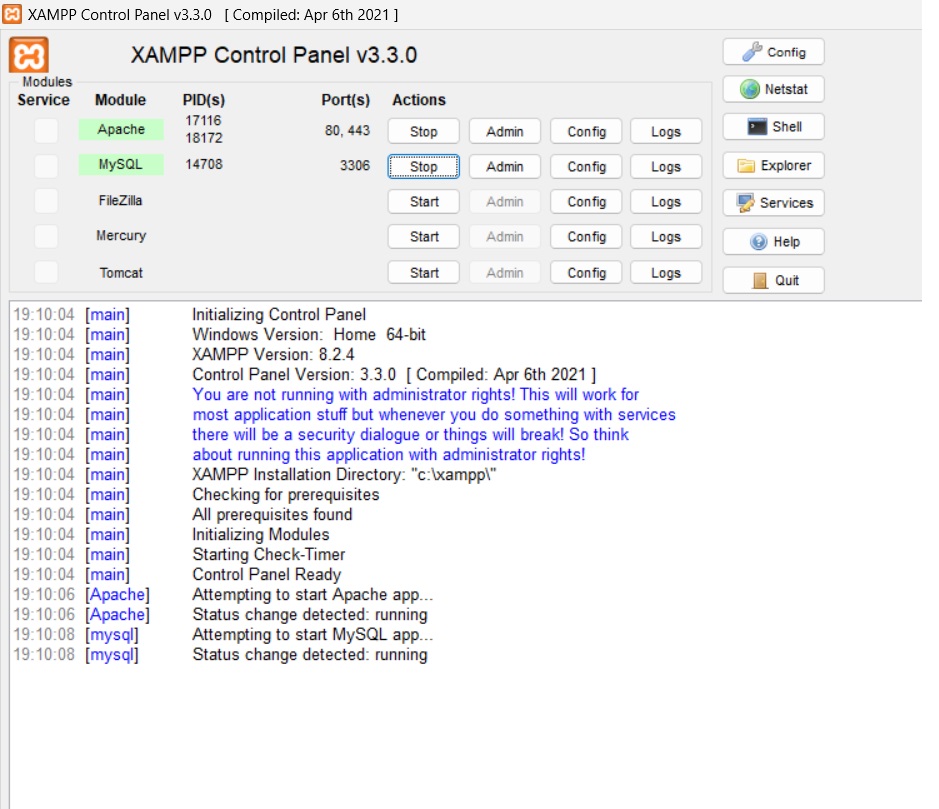
data.to\_csv(labeled, index=False)

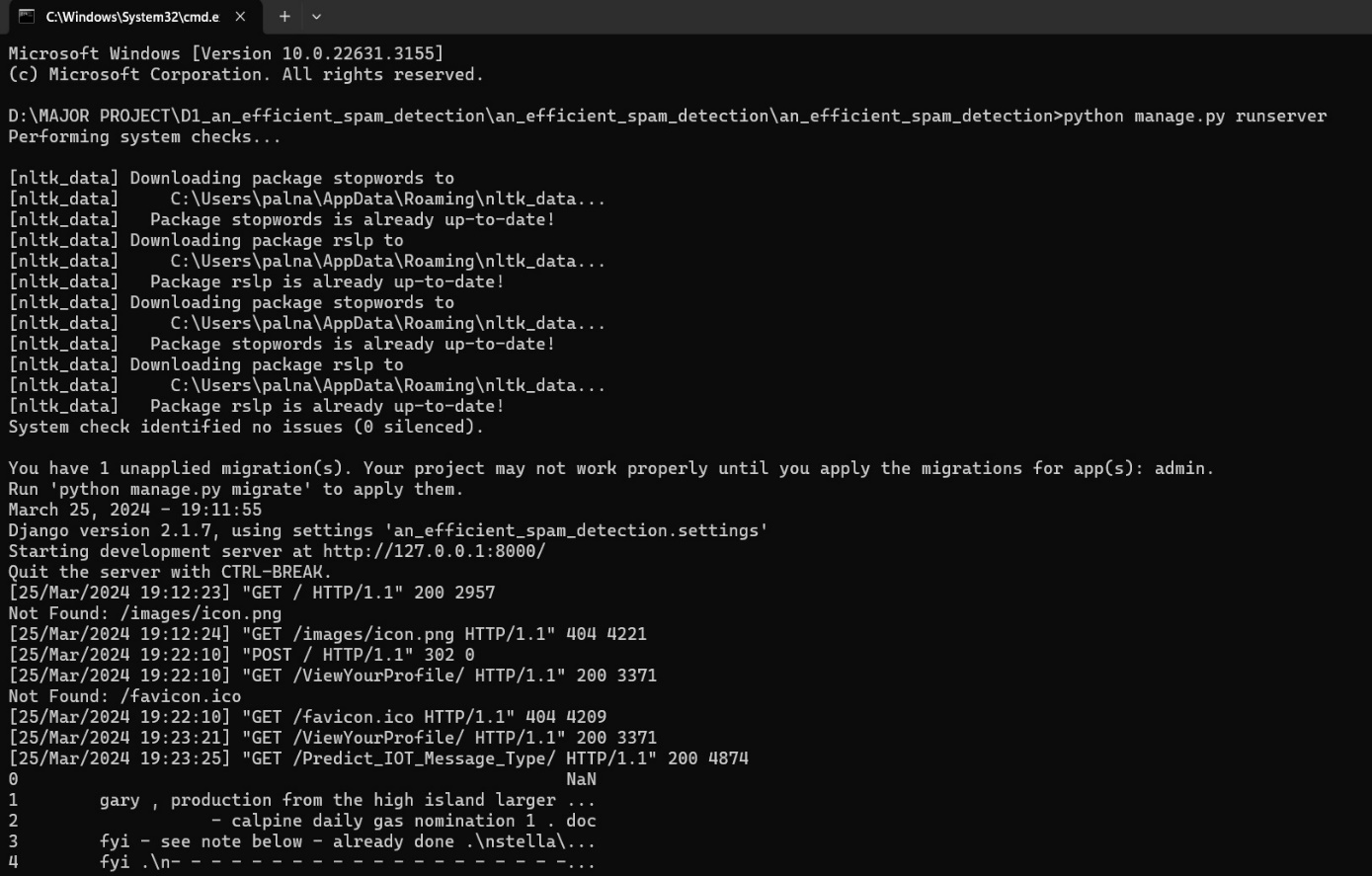
data.to\_markdown

obj = detection\_accuracy.objects.all()

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

**5. SCREENSHOTS**

****

Screenshot 5.1: Open Xampp And Start Apache And Mysql Admin

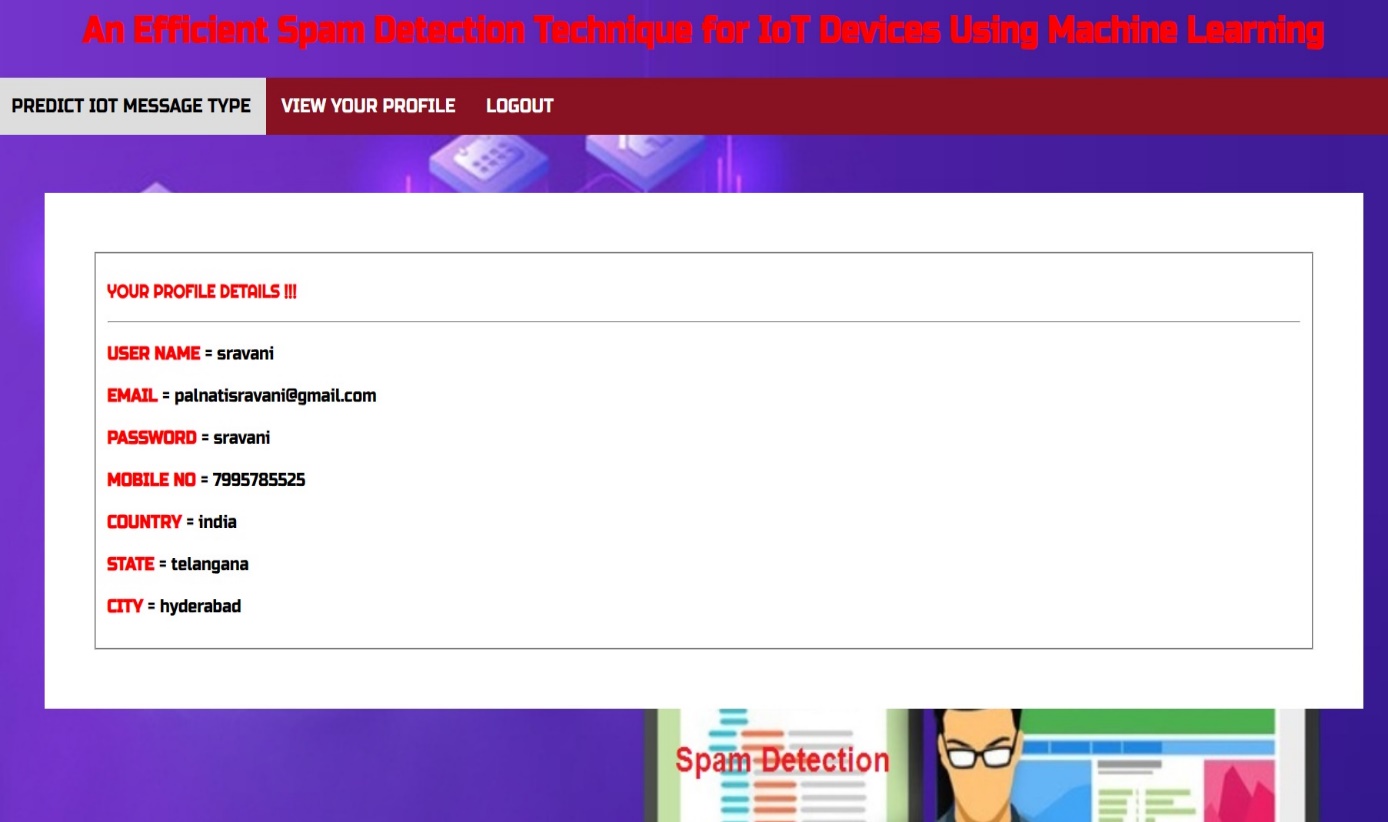
Screenshot 5.2: Open Command Prompt



Screenshot 5.3: Spam Detection Model



Screenshot 5.4: Remote User Login

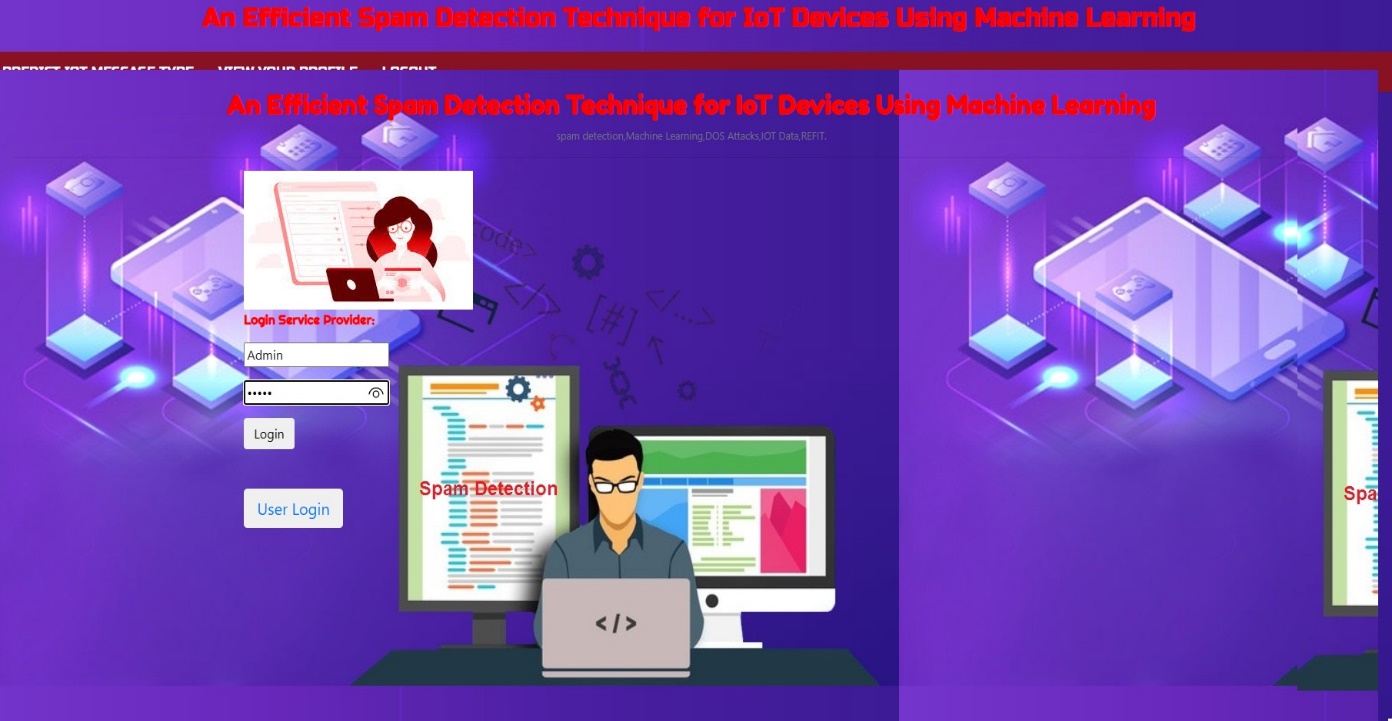


Screenshot 5.5: View Your Profile In Remote User

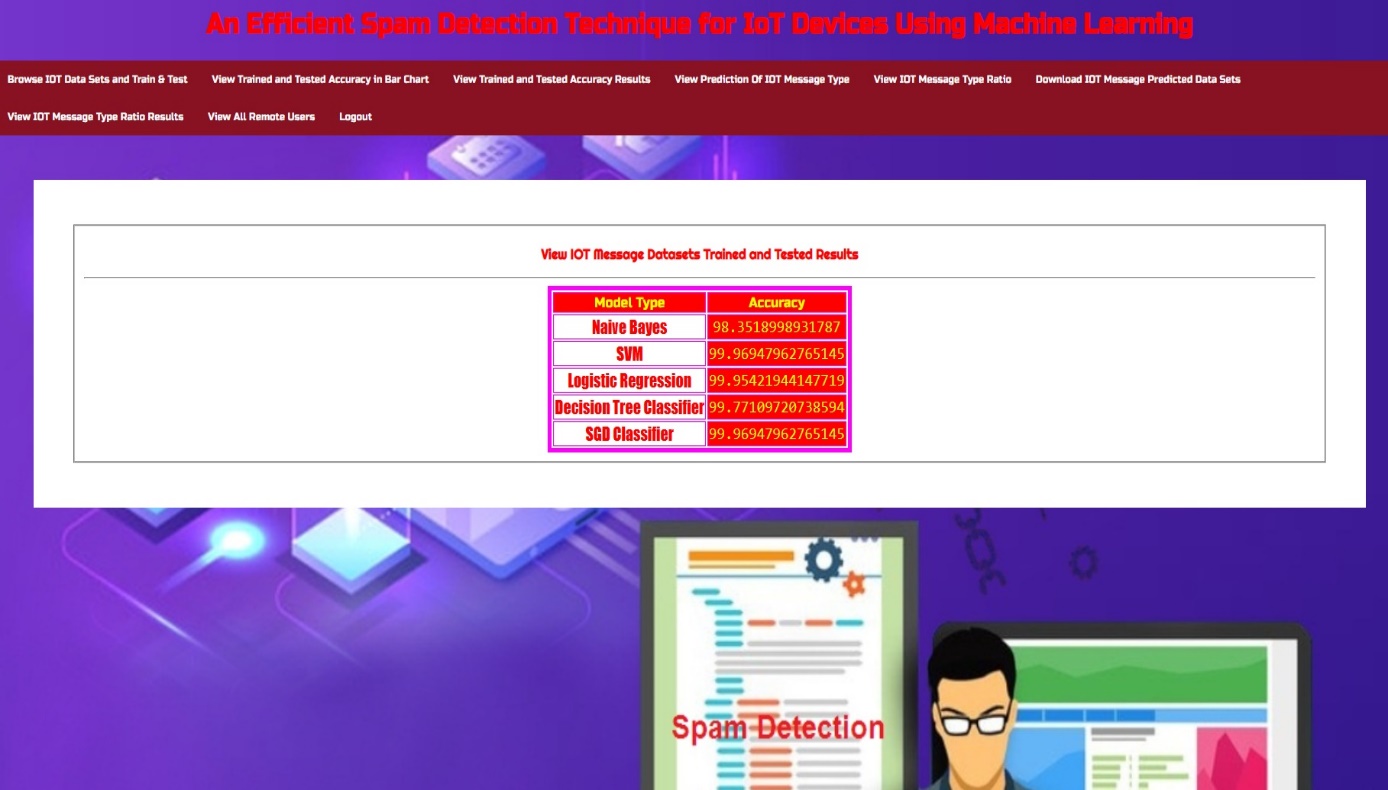
Screenshot 5.6: Predicting The IoT Message

****

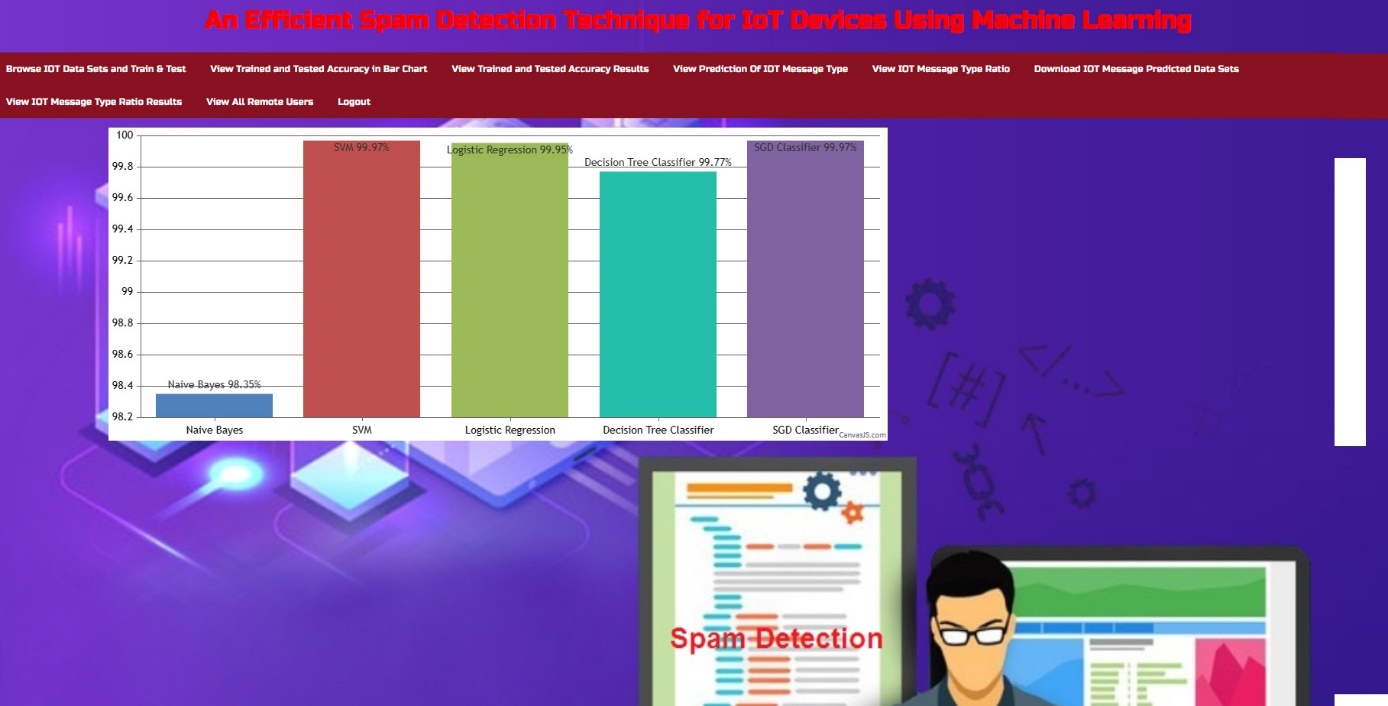
Screenshot 5.7: Register For Remote User



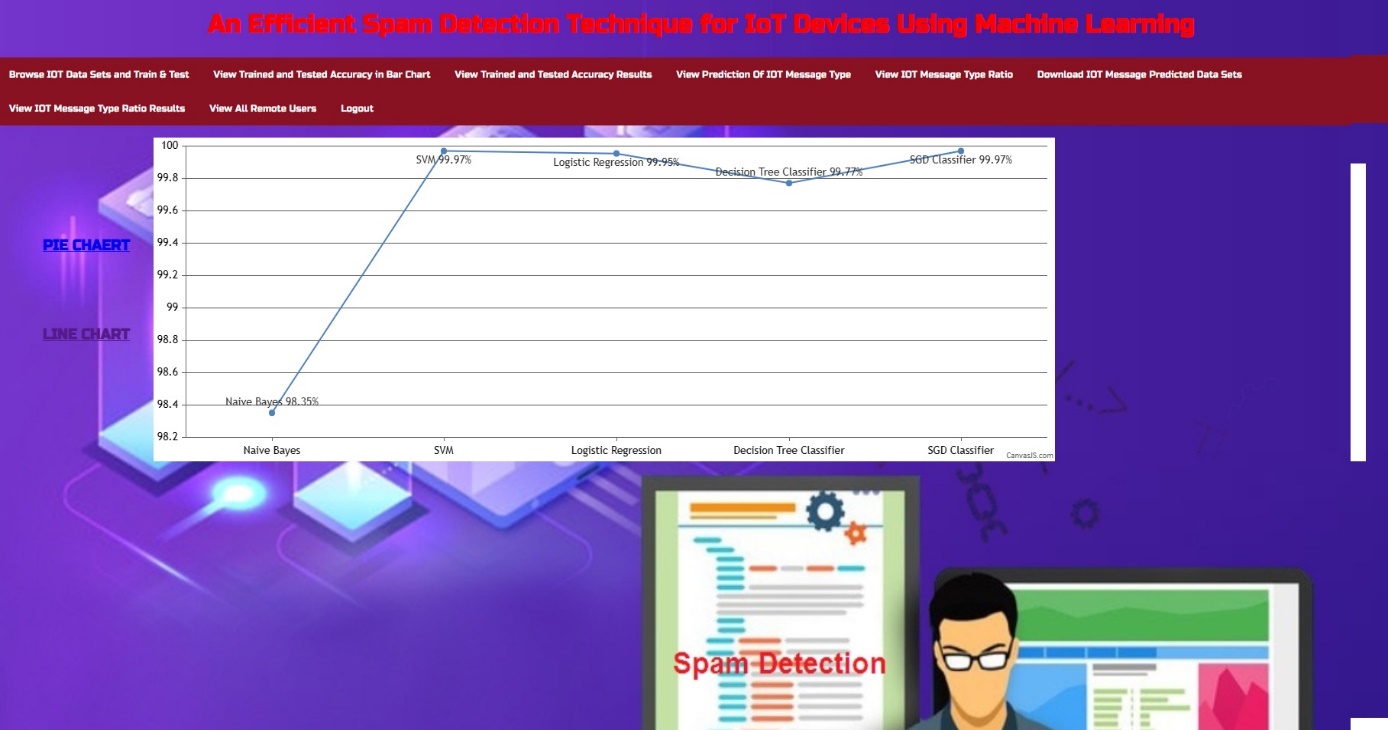
Screenshot 5.8: Login For Service Provider

****

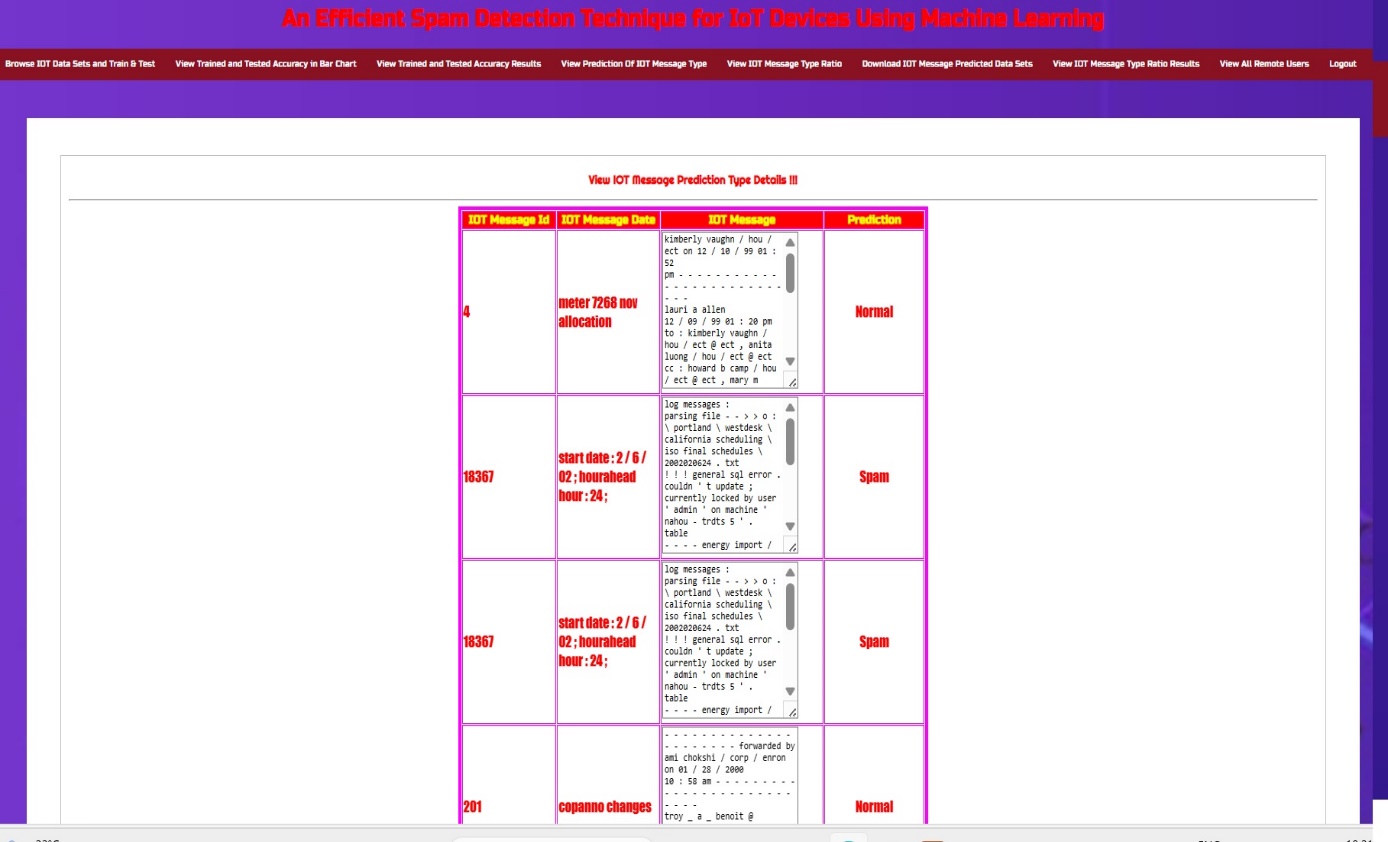
Screenshot 5.9: Train And Test Spam Datasets



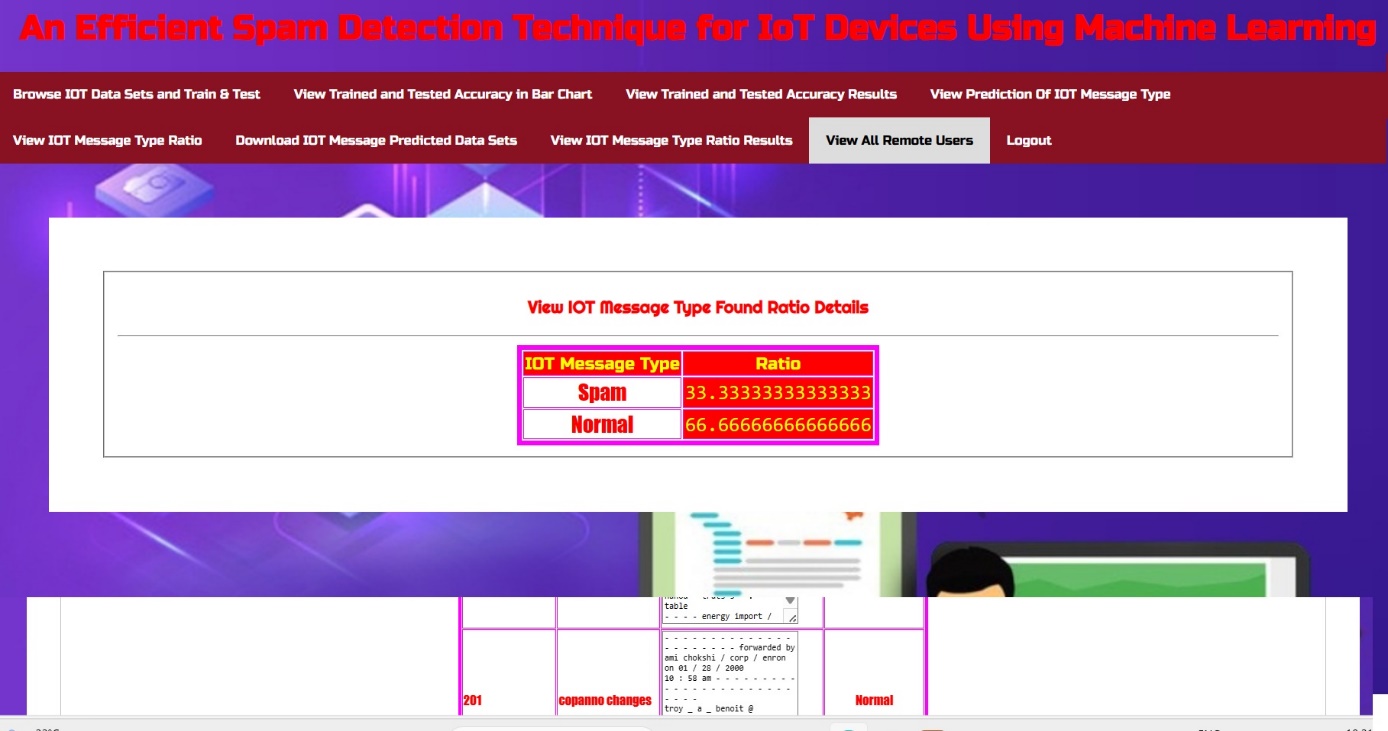
Screenshot 5.10: View Spam Datasets Of Trained And Tested Accuracy In Bar Chart



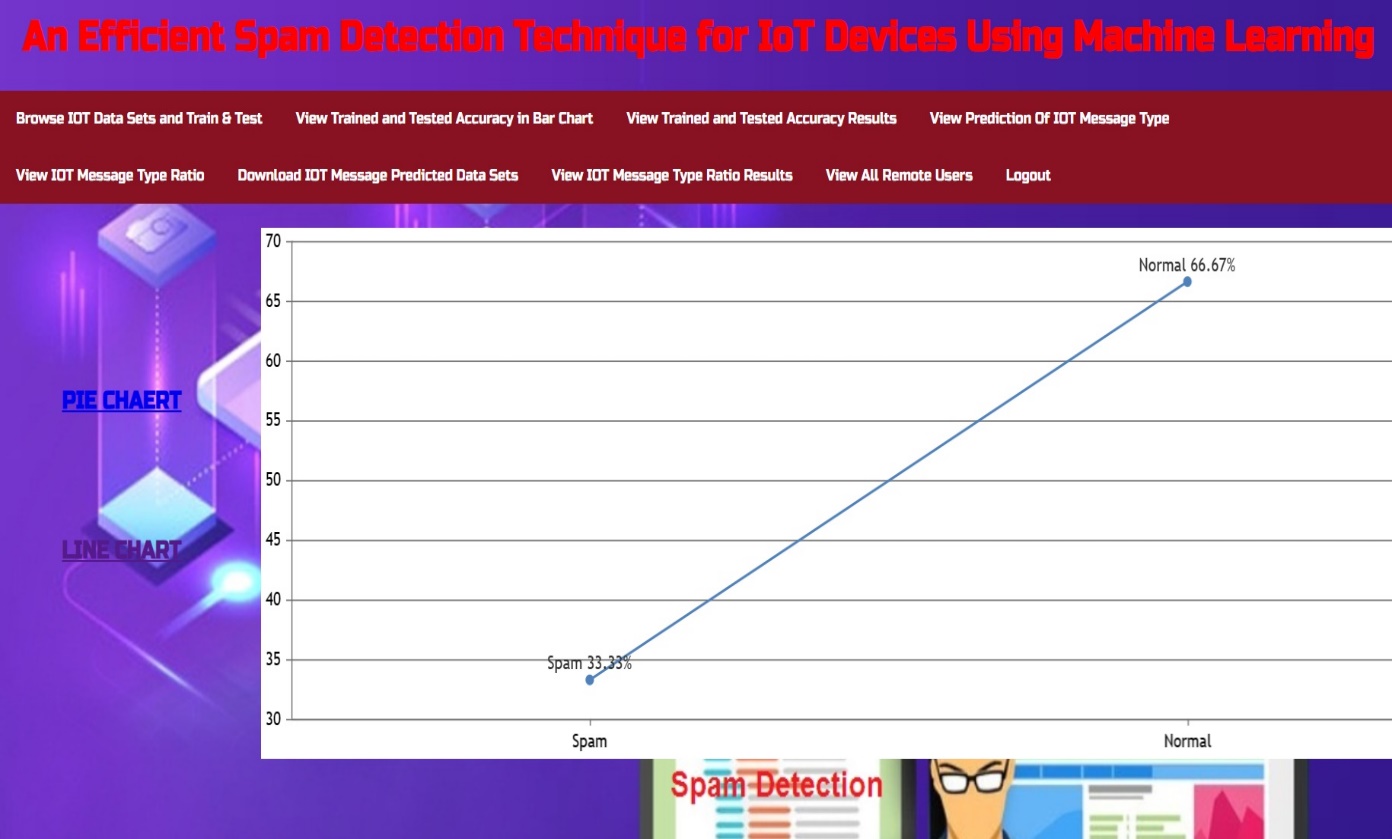
Screenshot 5.11: View Spam Datasets Of Trained And Tested Accuracy Results



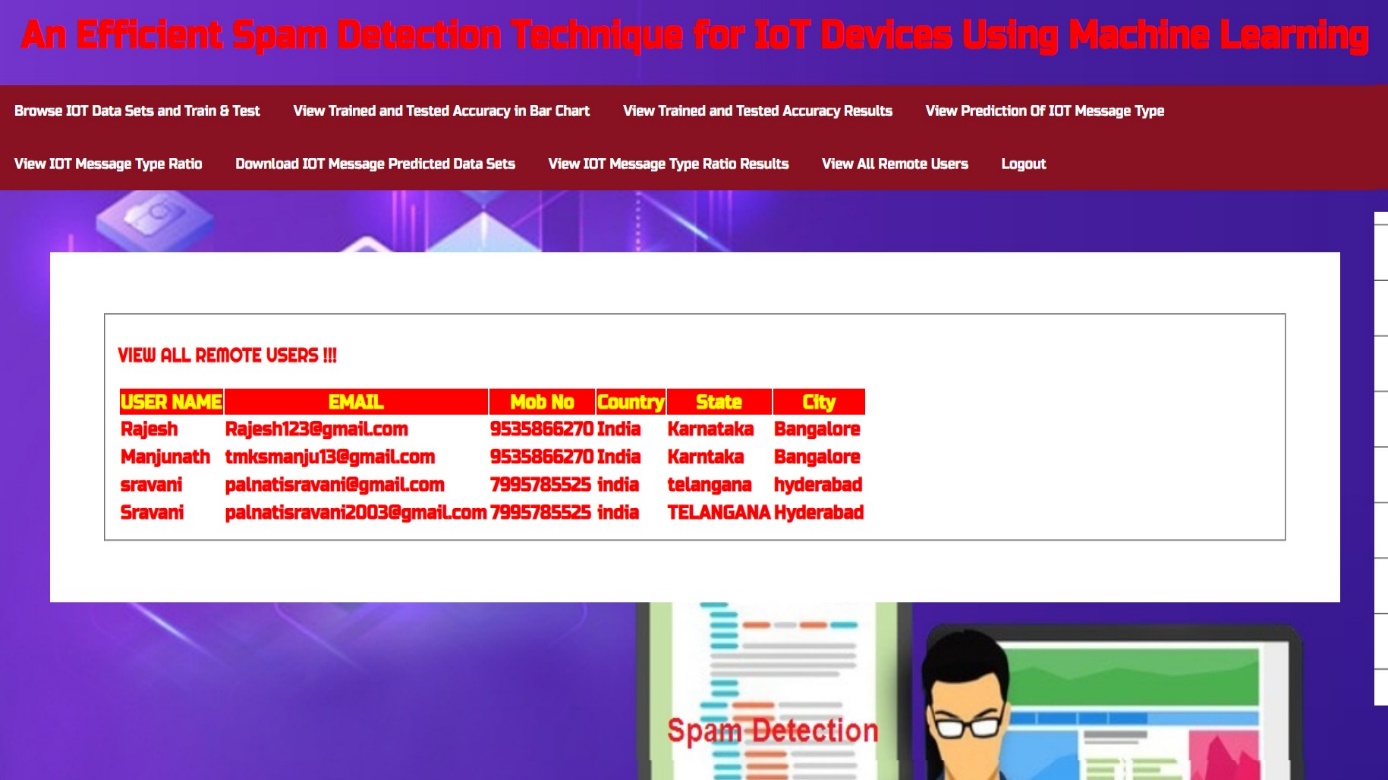
Screenshot 5.12: View Predicted Spam Detection Type

****

Screenshot 5.13: Spam Detection Type Ratio



Screenshot 5.14: View Spam Detection Ratio Results

****

Screenshot 5.15: View All Remote Users

**6.TESTING**

**6.TESTING**

### 6.1 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, subassemblies, 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 tests. 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. Integration tests demonstrate that although the components were individually satisfactory, 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 : identified classes of invalid input must Input be rejected.

Functions : identified functions must be exercised.

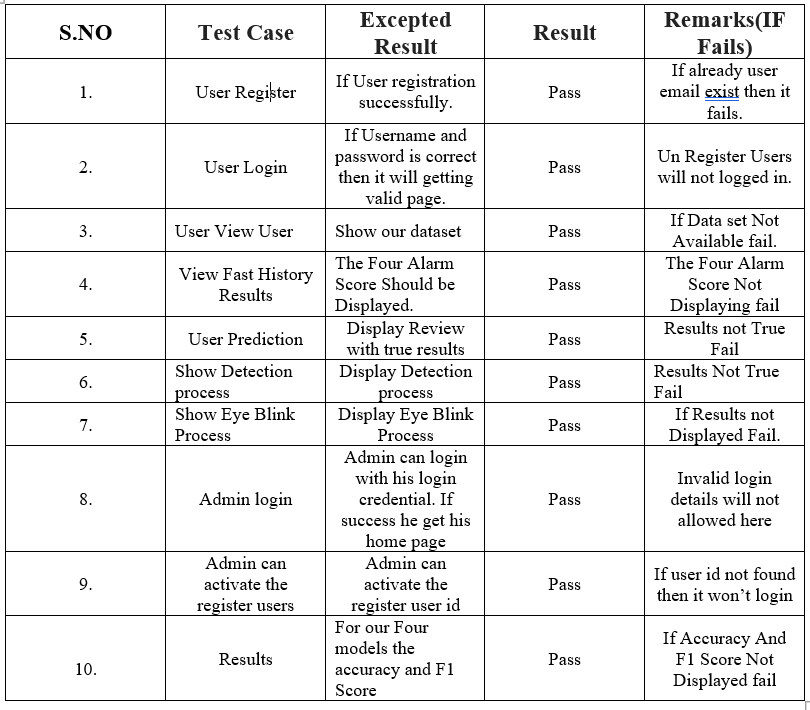
Output : identified classes of application outputs must be exercised.

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.

**6.3 TEST CASES**

**6.3.1 CLASSIFICATION**



**7.CONCLUSION**

**7.CONCLUSION & FUTURE SCOPE**

### 7.1 PROJECT CONCLUSION

Machine learning algorithms like SVM, Logistic Regression and Naive Bayes provide significant advantages for Spam detection techniques in IoT. These algorithms are capable of detecting and preventing attacks in real-time, making them more effective than traditional methods. Additionally, they can adapt to changing attack patterns, making them more resilient against evolving threats. By implementing these algorithms, organizations can significantly improve their security posture and protect their IoT devices from malicious attacks. As the number of connected devices continues to grow, it is essential to consider these advanced security measures to ensure the safety and privacy of sensitive data.

### 7.2 FUTURE SCOPE

**Integration with Existing Security Solutions**:

The project could lead to the integration of machine learning-based Spam mitigation techniques into existing security products and services. This integration can enhance the overall cybersecurity.

**Real-time Detection**:

Enhance project's ability to detect and respond to Spam attacks in real-time. This can involve optimizing algorithms for implementing streaming data processing, and integrating with network monitoring tools.

**Data Augmentation**:

Experiment with data augmentation techniques to increase the diversity and size of your training dataset. This can lead to better generalization and robustness of machine learning models.

**Large-scale Testing**:

Conduct extensive testing of the system in a controlled environment to assess its scalability and performance under heavy Spam attack scenarios.

**8.BIBLIOGRAPHY**

**8.BIBLIOGRAPHY**

### 8.1 REFERENCES

[1] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, “IoT security: ongoing challenges and research opportunities,” in 2014 IEEE 7th international conference on service-oriented computing and applications. IEEE, 2014, pp. 230–234**.**

[2]E. Bertino and N. Islam, “Botnets and internet of things security,” Computer, no. 2, pp. 76–79, 2017.

[3] W. Kim, O.-R. Jeong, C. Kim, and J. So, “The dark side of the internet: Attacks, costs and responses,” Information systems, vol. 36, no. 3, pp.675–705, 2011.

[4] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine learning in wireless sensor networks: Algorithms, strategies, and applications,” IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 1996–2018, 2014.

[5] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, “Evaluation of machine learning classifiers for mobile malware detection,” Soft Computing, vol. 20, no. 1, pp. 343–357, 2016.

### 8.2 GITHUB LINK

### https://github.com/Sravani2510/Spam-Detection-Technique-in-IoT-Devices