A

Major Project

On

Robust Spammer Detection using Collaborative Neural Network in Internet of Thing Applications

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

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

BIRADAR NITESH (207R1A05M4)

POLABOINA SREEJA (207R1A05M6)

MARELLI RAMYA (207R1A05M1)

Under the Guidance of

NAJEEMA AFRIN

(Assistant Professor)

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 **"Robust Spammer Detection using Collabaritive Neural Network in Internet of Thing Applications "** being submitted by **BIRADAR NITESH(207R1A05M4), POLABOINA SREEJA(207R1A05M6) & MARELLI RAMYA (207R1A05M1)** in partial fulfillment 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 2023-24.

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

MRS. NAJEEMA AFRIN
(Assistant Professor)
INTERNAL GUIDE

Dr. A. RAJI REDDY
DIRECTOR

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 **NAJEEMA AFRIN,** Assistant Professor for her exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by her 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.

B. NITESH (207R1A05M4)

P.SREEJA (207R1A05M6)

M.RAMYA (207R1A05M1)

ABSTRACT

Spamming is emerging as a key threat to Internet of Things (IoT)-based social media applications. It will pose serious security threats to the IoT cyberspace. To this end, artificial intelligence-based detection and identification techniques have been widely investigated. The literature works on IoT cyberspace can be categorized into two categories: 1) behavior pattern-based approaches; and 2) semantic pattern-based approaches. However, they are unable to effectively handle concealed, complicated, and changing spamming activities, especially in the highly uncertain environment of the IoT. To address this challenge, in this paper, we exploit the collaborative awareness of both patterns, and propose a Collaborative neural network-based Spammer detection mechanism (Co-Spam) in social media applications. In particular, it introduces multi-source information fusion by collaboratively encoding long-term behavioral and semantic patterns. Hence, a more comprehensive representation of the feature space can be captured for further spammer detection. Empirically, we implement a series of experiments on two real-world datasets under different scenario and parameter settings. The efficiency of the proposed Co-Spam is compared with five baselines with respect to several evaluation metrics. The experimental results indicate that the Co-Spam has an average performance improvement of approximately 5% compared to the baselines.

LIST OF TABLES/FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1.INTRODUCTION

1.1 PROJECT SCOPE

The project scope for robust spammer detection using collaborative neural networks in IoT applications is comprehensive and multifaceted. It begins with a clear definition of the problem statement, highlighting the increasing threat of spamming activities within IoT ecosystems and the necessity for an efficient detection system. Objectives are set to achieve high accuracy, real-time detection capabilities, and scalability to handle diverse IoT devices and data sources. The scope is defined, outlining the targeted spamming activities, the range of IoT devices considered, and the boundaries of the project. The system architecture is designed to encompass key components such as data acquisition, preprocessing, collaborative neural networks, decision-making mechanisms, feedback loops, integration with IoT platforms, and security measures.

1.2 PROJECT PURPOSE

The purpose of the project on robust spammer detection using collaborative neural networks in IoT applications is to address the growing security challenges posed by spamming activities within IoT ecosystems. With the proliferation of interconnected devices and sensors in IoT environments, the risk of malicious actors exploiting vulnerabilities to propagate spam has increased significantly. The project aims to develop an effective and scalable solution to detect and mitigate these spamming activities in real-time.

1.3 PROJECT FEATURES

The project on robust spammer detection using collaborative neural networks in IoT applications incorporates several key features to address the complex challenges of detecting and mitigating spamming activities within IoT ecosystems. Firstly, the system integrates advanced machine learning techniques, particularly collaborative neural networks, to analyze diverse data sources collected from IoT devices. These networks collaborate to effectively identify patterns indicative of spam behavior, enabling accurate and efficient detection of spam messages. Secondly, the system is designed to be scalable and adaptable to diverse IoT environments, accommodating a wide range of devices, data types, and communication protocols.

2.SYSTEM ANALYSIS

2.1 SYSTEM ANALYSIS

Firstly, the requirements of the system are analyzed, taking into account the needs of stakeholders, the characteristics of IoT environments, and the nature of spamming activities. This involves gathering and prioritizing functional and non-functional requirements to guide the design and development process. Next, the existing infrastructure and technologies within IoT ecosystems are evaluated to identify any potential limitations or compatibility issues. This includes assessing the capabilities of IoT devices, communication protocols, data formats, and integration points with external systems.

2.2 EXISTING SYSTEM

In the context of robust spammer detection using collaborative neural networks in IoT applications, the existing systems often rely on traditional spam detection methods, which may lack the sophistication and adaptability required to effectively mitigate spamming activities within IoT ecosystems. These conventional approaches typically involve rule-based filtering, pattern matching, and heuristic analysis, which may struggle to keep pace with the evolving tactics employed by spammers.

2.2.1 DISADVANTAGES OF EXISTING SYSTEM

The following are the Disadvantages of the Existing system:

- Rule-Based Rigidity
- Limited Scalability
- False Positives and Negatives
- Dependency on Labeled Data

2.3 PROPOSED SYSTEM

The proposed system for robust spammer detection in Internet of Things (IoT) applications introduces a Collaborative Neural Network model to overcome the limitations inherent in existing systems. At its core, the system leverages advanced machine learning techniques to enhance spam detection accuracy. The Collaborative Neural Network is designed for robustness, capable of adapting to diverse spamming techniques and evolving patterns through collaborative learning mechanisms. Real-time detection is a focal point, enabling the system to promptly identify and mitigate spam attacks as they occur.

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

The following are the Advantages of Proposed system:

- Enhanced Accuracy
- Robustness Against Evolving Threats
- Real-Time Detection and Responsiveness
- Scalability
- Seamless IoT Integration
- Comprehensive Documentation and User Interface

2.4 FEASIBILITY STUDY

The study encompasses a thorough examination of technical, economic, operational, legal, and scheduling aspects. From a technical perspective, the study evaluates the feasibility of implementing the CNN model, considering the availability of technology and required expertise. Operational feasibility is assessed by examining the practicality of integrating the system into existing processes, including staffing and training requirements.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.4.1 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.

2.4.2 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.

2.4.3 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.

2.5 HARDWARE AND SOFTWARE REQUIREMENTS

2.5.1 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 : Pentium –IV
- RAM : 4 GB (min)
- Hard Disk : 20 GB
- Key Board : Standard Windows Keyboard
- Mouse : Two or Three Button Mouse
- Monitor : SVGA

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

3.ARCHITECTURE

3.1 PROJECT ARCHITECTURE

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

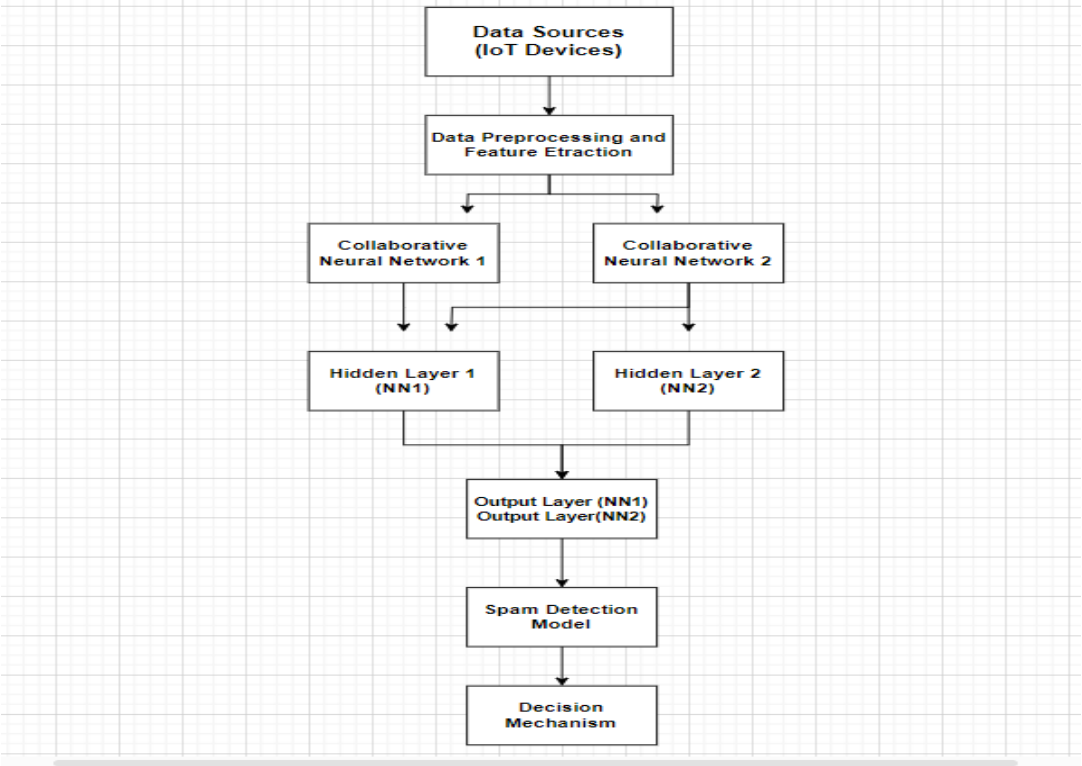


Figure 3.1 : Project Architecture of Robust Spammer Detection using Collaborative Neural Network in Internet of Things Application

3.2 DESCRIPTION

Robust spammer detection using collaborative neural networks in IoT applications involves deploying sophisticated machine learning techniques to combat the increasing threat of spamming activities within interconnected IoT ecosystems. This innovative approach harnesses the power of collaborative neural networks to analyze diverse data streams generated by IoT devices, such as sensors, cameras, and smart appliances. By leveraging collaborative intelligence, these neural networks collaboratively learn and adapt to identify patterns indicative of spam behavior, enabling accurate and timely detection of spam messages or activities.

At the heart of the system lies a collaborative neural network architecture, comprised of multiple interconnected neural network nodes. Each node specializes in analyzing specific data modalities or features, such as text, images, sensor readings, or network traffic. Through collaborative learning, these nodes exchange information and insights, enhancing the overall detection accuracy and robustness of the system.

The process begins with data acquisition, where raw data from IoT devices is collected and preprocessed to extract relevant features and normalize the data for input into the neural network models. The collaborative neural networks then analyze the preprocessed data, leveraging advanced machine learning algorithms to detect spamming activities based on learned patterns and anomalies.

3.3 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.

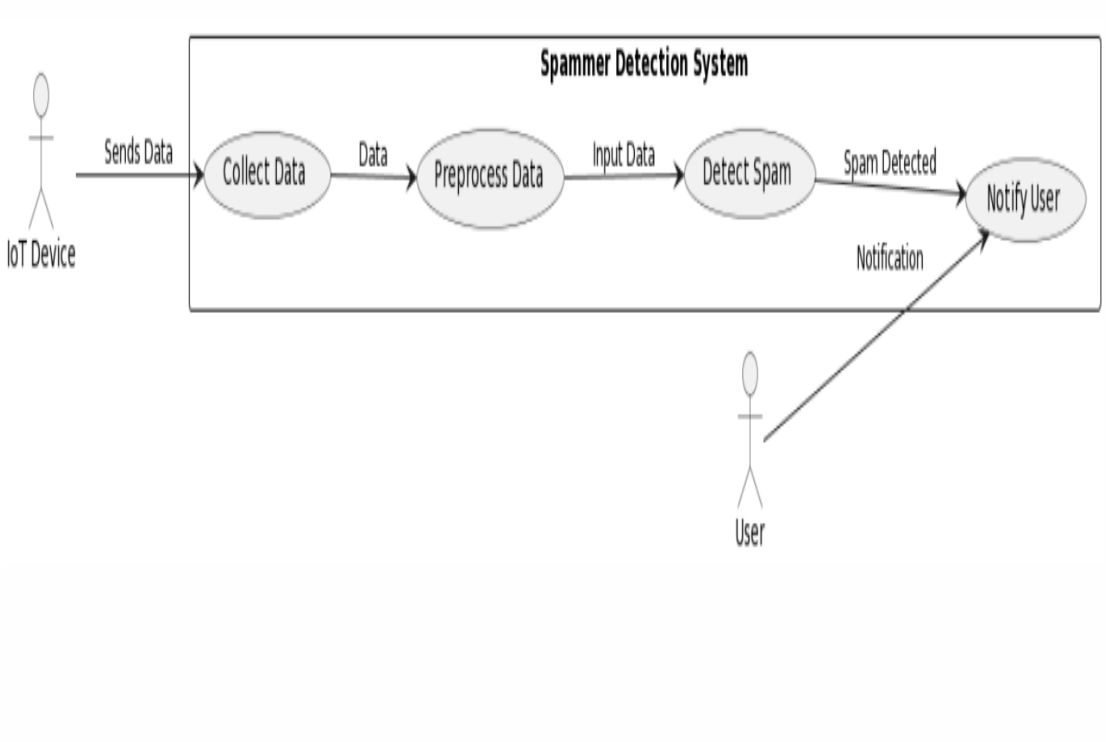


Figure 3.2: Use Case Diagram of Robust Spammer Detection using Collaborative Neural Network in Internet of Things Application

3.4 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.

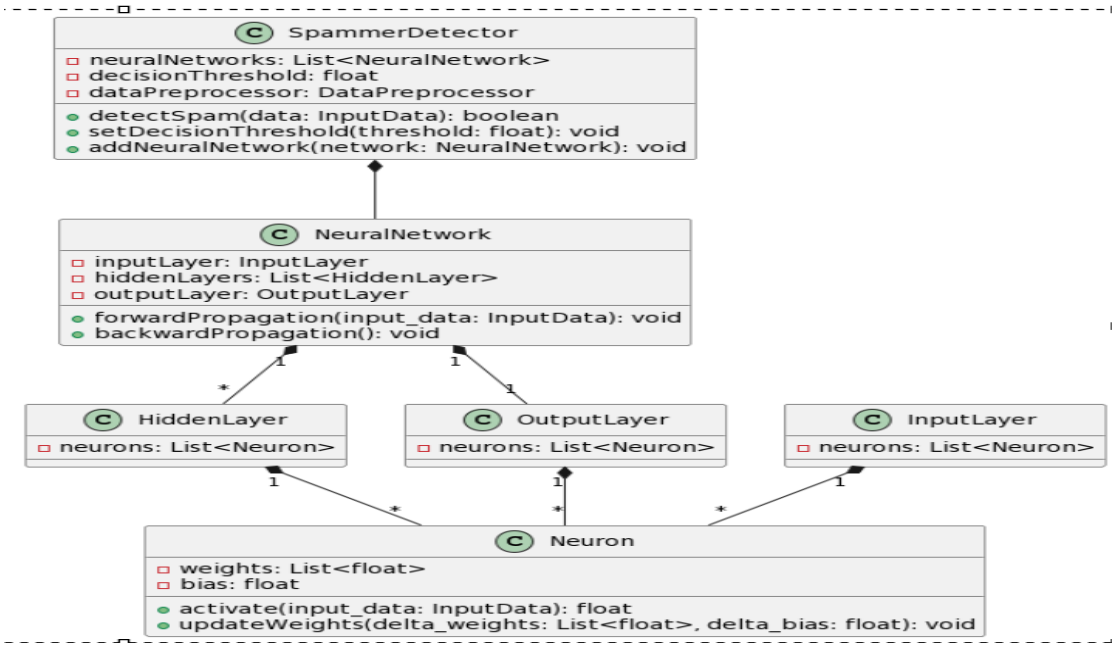


Figure 3.3: Class Diagram of Robust Spammer Detection using Collaborative Neural Network in Internet of Things Applications

3.5 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.

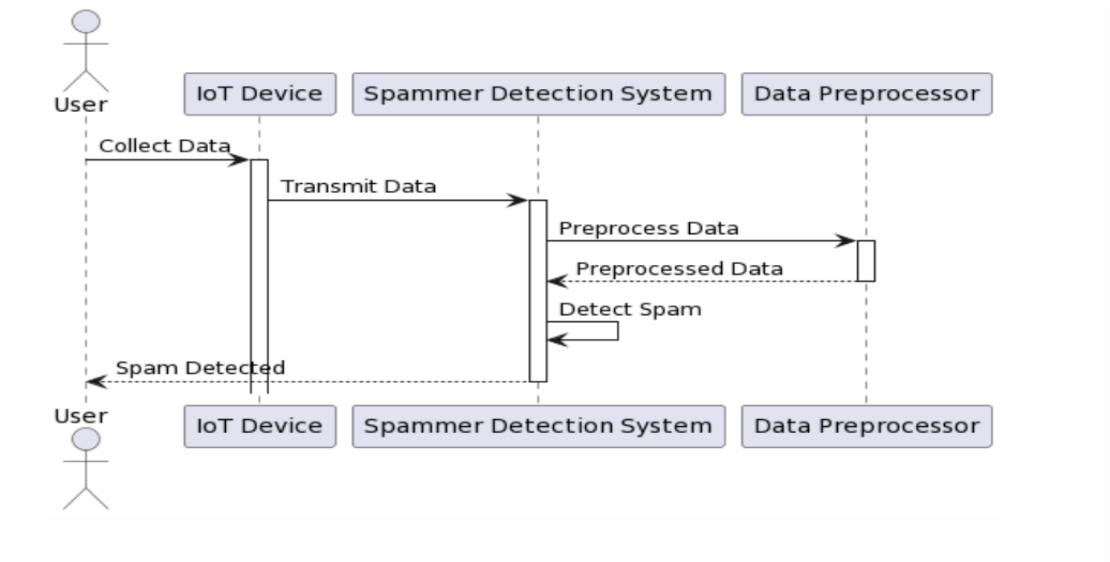


Figure 3.4: Sequence Diagram of Robust Spammer Detection using Collaborative Neural Network in Internet of Things Application

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.

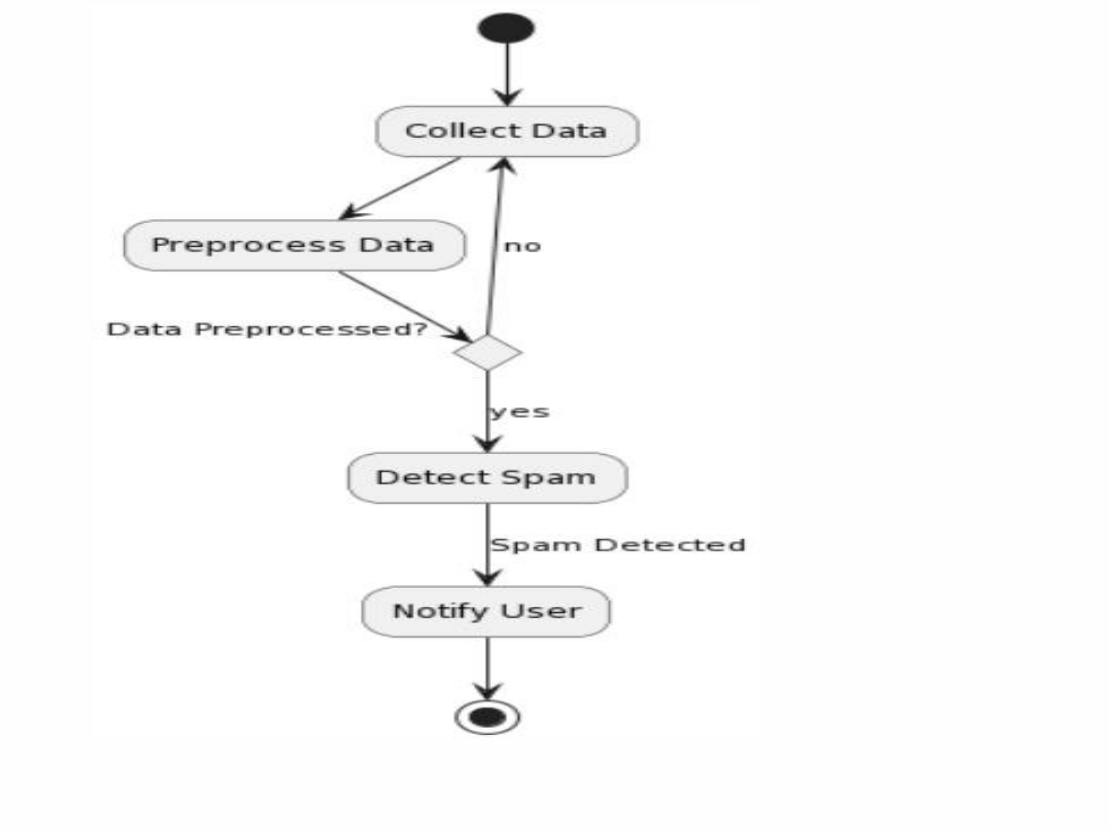


Figure 3.5: Activity Diagram of Robust Spammer Detection using Collaborative Neural Network in Internet of Things Application

4.IMPLEMENTATION

SAMPLE CODE

```
from django.shortcuts import render
from django.template import RequestContext
from django.contrib import messages
import pymysql
from django.http import HttpResponse
from django.conf import settings
from django.core.files.storage import FileSystemStorage
import matplotlib.pyplot as plt
import re
import cv2
import numpy as np
from string import punctuation
from nltk.corpus import stopwords
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
import os
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
porter = PorterStemmer()

def LCS(l1,l2): #LCS method
    s1 = word_tokenize(l1)
    s2 = word_tokenize(l2)
    dp = [[None]*(len(s1)+1) for i in range(len(s2)+1)]
    for i in range(len(s2)+1):
        for j in range(len(s1)+1):
            if i == 0 or j == 0:
```

```
            dp[i][j] = 0
        elif s2[i-1] == s1[j-1]:
            dp[i][j] = dp[i-1][j-1]+1
        else:
            dp[i][j] = max(dp[i-1][j] , dp[i][j-1])
    return dp[len(s2)][len(s1)]

def cleanPost(doc):
    tokens = doc.split()
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    tokens = [word for word in tokens if word.isalpha()]
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    tokens = [porter.stem(token) for token in tokens]
    tokens = ' '.join(tokens)
    return tokens



text_files = []
text_data = []
image_files = []
image_data = []

def index(request):
    if request.method == 'GET':
        return render(request, 'index.html', {})

def Register(request):
    if request.method == 'GET':
        return render(request, 'Register.html', {})

def Login(request):
```

```
    if request.method == 'GET':
        return render(request, 'Login.html', {})

def UploadSuspiciousFile(request):
    if request.method == 'GET':
        return render(request, 'UploadSuspiciousFile.html', {})


def UploadSuspiciousImage(request):
    if request.method == 'GET':
        return render(request, 'UploadSuspiciousImage.html', {})


def UploadSuspiciousImageAction(request):
    if request.method == 'POST' and request.FILES['t1']:
        output = ''
        myfile = request.FILES['t1']
        fs = FileSystemStorage()
        name = str(myfile)
        filename = fs.save(name, myfile)
        img = cv2.imread(name)
        img = cv2.resize(img,(200,200))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        hist = cv2.calcHist([img], [0], None, [256], [0, 256])
        os.remove(name)
        similarity = 0
        file = 'No Match Found'
        hist1 = 0
        for i in range(len(image_files)):
            metric_val = cv2.compareHist(hist, image_data[i], cv2.HISTCMP_INTERSECT)
            if metric_val > similarity:
                similarity = metric_val
                file = image_files[i]
```

```
            hist1 = image_data[i]
        output = '<table border=1 align=center><tr><th>Source Original Image Name</th><th>Suspicious Image Name</th><th>Histogram Matching Score</th><th>Plagiarism Result</th></tr>'
        result = 'No Plagiarism Detected'
        print(str(name)+" "+str(similarity))
        if similarity >= 39000:
            result = 'Plagiarism Detected'
        output+='<tr><td><font size="" color="white">'+file+'</td><td><font size="" color="white">'+name+'</td>'
        output+='<td><font size="" color="white">'+str(similarity)+'</td><td><font size="" color="white">'+result+'</td></tr>'
        context= {'data':output}
        fig, ax = plt.subplots(2,1)
        ax[0].plot(hist1, color = 'b')
        ax[1].plot(hist, color = 'g')
        plt.xlim([0, 256])
        ax[0].set_title('Original image')
        ax[1].set_title('Plagiarised image')
        plt.show()
        return render(request, 'SuspiciousImageResult.html', context)


def UploadSuspiciousFileAction(request):
    if request.method == 'POST' and request.FILES['t1']:
        output = ''
        myfile = request.FILES['t1']
        fs = FileSystemStorage()
        name = str(myfile)
        filename = fs.save("test.txt", myfile)
        data = ''
        with open("test.txt", "r", encoding='iso-8859-1') as file:
            for line in file:
                line = line.strip('\n')
```

```
            line = line.strip()
            data+=line+" "
        file.close()
        os.remove("test.txt")
        data = cleanPost(data.strip().lower())
        sim = 0
        ff = 'No Match Found'
        for i in range(len(text_data)):
            similarity = LCS(text_data[i],data)
            if similarity > sim:
                sim = similarity
                ff = text_files[i]

        output = '<table border=1 align=center><tr><th>Source Original File Name</th><th>Suspicious File Name</th><th>LCS Score</th><th>Plagiarism Result</th></tr>'
        result = 'No Plagiarism Detected'
        similarity_percent = 0
        if sim >= 0:
            similarity_percent = sim/len(word_tokenize(data))
            if similarity_percent >= 0.60:
                result = 'Plagiarism Detected'
        output+='<tr><td><font size="" color="white">'+ff+'</td><td><font size="" color="white">'+name+'</td>'
        output+='<td><font size="" color="white">'+str(similarity_percent)+'</td><td><font size="" color="white">'+result+'</td></tr>'
        context= {'data':output}
        return render(request, 'SuspiciousFileResult.html', context)


def UploadSourceImage(request):
    if request.method == 'GET':
        if len(image_files) == 0:
```

```
        for root, dirs, directory in os.walk('images'):
            for j in range(len(directory)):
                img = cv2.imread(root+"/"+directory[j])
                img = cv2.resize(img,(200,200))
                img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                hist = cv2.calcHist([img], [0], None, [256], [0, 256])
                image_data.append(hist)
                image_files.append(directory[j])
    output = '<table border=1 align=center><tr><th>Source Image File Name</th><th>Histogram Values</th></tr>'
    for i in range(len(image_files)):
        output+='<tr><td><font size="" color="white">'+image_files[i]+'</td><td><font size="" color="white">'+str(image_data[i])+"</td></tr>"
    context= {'data':output}
    return render(request, 'UploadSourceImage.html', context)

def UploadSource(request):
  if request.method == 'GET':
    if len(text_files) == 0:
        for root, dirs, directory in os.walk('corpus-20090418'):
            for j in range(len(directory)):
                data = ''
                with open(root+"/"+directory[j], "r", encoding='iso-8859-1') as file:
                    for line in file:
                        line = line.strip('\n')
                        line = line.strip()
                        data+=line+" "
                file.close()
                data = cleanPost(data.strip().lower())
                text_files.append(directory[j])
                text_data.append(data)
    output = '<table border=1 align=center><tr><th>Source File Name</th><th>Words in File</th></tr>'
```

```
        for i in range(len(text_files)):
            length = len(text_data[i].split(" "))
            output+='<tr><td><fontsize="" color="white">'+text_files[i]+'</td><td>
<font size="" color="white">'+str(length)+"</td></tr>"
        context= {'data':output}
        return render(request, 'UploadSource.html', context)



def UserLogin(request):
    if request.method == 'POST':
      username = request.POST.get('username', False)
      password = request.POST.get('password', False)
      index = 0
      con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root',
database = 'plagiarism',charset='utf8')
      with con:
          cur = con.cursor()
          cur.execute("select * FROM users")
          rows = cur.fetchall()
          for row in rows:
            if row[0] == username and password == row[1]:
              index = 1
              break
      if index == 1:
       file = open('session.txt','w')
       file.write(username)
       file.close()
       context= {'data':'welcome '+username}
       return render(request, 'UserScreen.html', context)
      else:
       context= {'data':'login failed'}
       return render(request, 'Login.html', context)

def Signup(request):
```

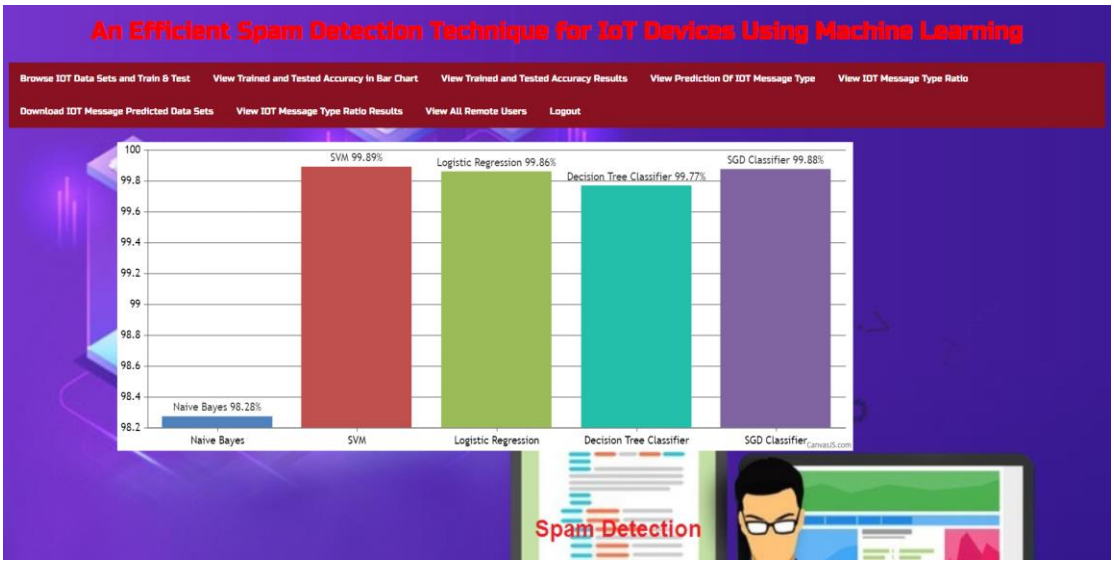
```
    if request.method == 'POST':
      username = request.POST.get('username', False)
      password = request.POST.get('password', False)
      contact = request.POST.get('contact', False)
      email = request.POST.get('email', False)
      address = request.POST.get('address', False)
      db_connection = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root',
password = 'root', database = 'plagiarism',charset='utf8')
      db_cursor = db_connection.cursor()
      student_sql_query = "INSERT INTO
users(username,password,contact_no,email,address)
VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"')"
      db_cursor.execute(student_sql_query)
      db_connection.commit()
      print(db_cursor.rowcount, "Record Inserted")
      if db_cursor.rowcount == 1:
       context= {'data':'Signup Process Completed'}
       return render(request, 'Register.html', context)
      else:
       context= {'data':'Error in signup process'}
       return render(request, 'Register.html', context)
```

Screenshot 5.1: User Login

Screenshot 5.2: Enter Message

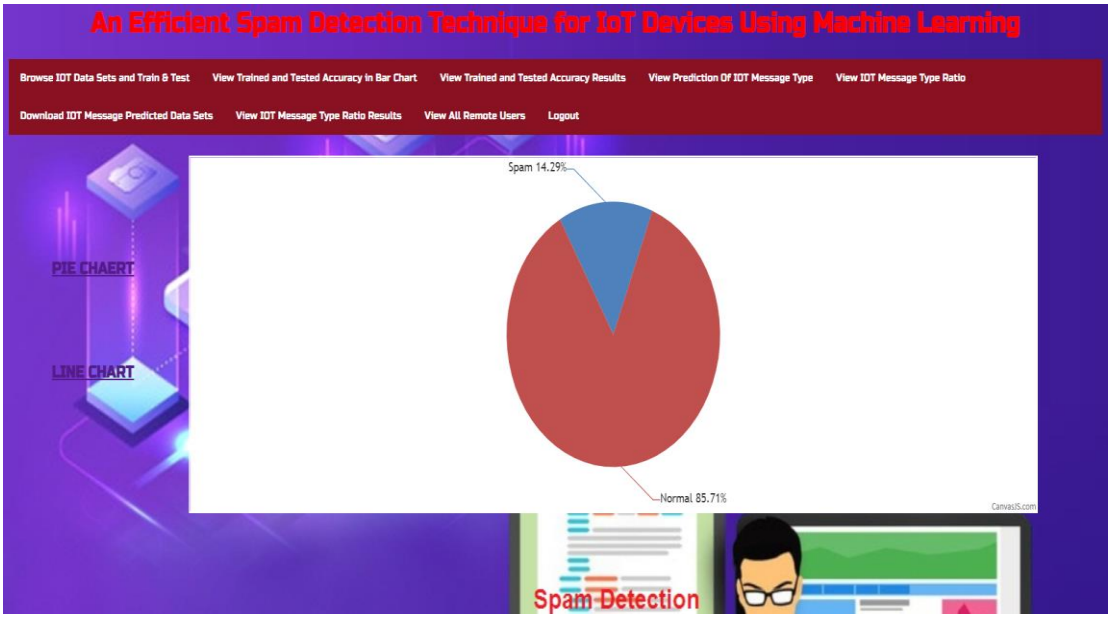
Screenshot 5.3: Predicted IOT Message Type

Screenshot 5.4: Service Provider Login Page



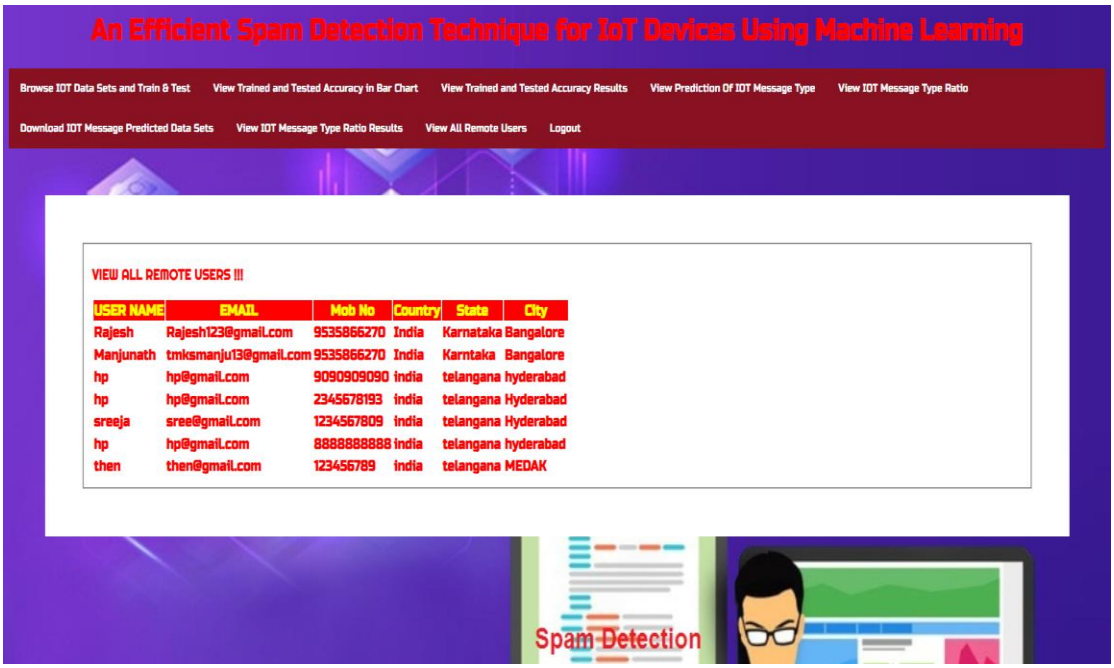
Screenshot 5.5: View Trained and Tested Accuracy in Bar Chart

Screenshot 5.6: View Trained Tested Accuracy Results



Screenshot 5.7: View IOT Message Type Ratio

Screenshot 5.8: View IOT Message Type Ratio in Graph



Screenshot 5.9: View all Remote Users

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, 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.

6.2.4 SYSTEM TESTING

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.

6.2.5 WHITE BOX TESTING

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

6.2.6 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, 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.

6.3 TEST CASES

6.3.1 CLASSIFICATION

Test Case ID	Test Case Name	Purpose	Input	Output
1	Input Data variety Test	The System should Preprocess and analyze various types of data from IoT devices effectively without errors	The user gives the input in the form of dataset	An Output is prediction of message if it is Spam message or Normal Message

7. CONCLUSION AND FUTURE SCOPE

7.1 PROJECT CONCLUSION

In conclusion, robust spammer detection using collaborative neural networks in Internet of Things (IoT) applications offers a powerful solution to mitigate the growing threat of spamming activities within IoT ecosystems. By leveraging advanced machine learning techniques and collaborative intelligence, the system can effectively analyze diverse data sources from IoT devices and identify patterns indicative of spam behavior. Throughout this journey, we have explored the architecture, components, and functionalities of such a system. We have discussed the importance of data preprocessing, the collaboration of neural networks, decision making, feedback mechanisms, integration with IoT platforms, security, privacy, and performance considerations.

7.2 PROJECT FUTURE SCOPE

The future scope for robust spammer detection using collaborative neural networks in IoT applications is vast, with numerous opportunities for advancement and innovation. Research and development efforts can focus on enhancing collaborative learning algorithms to improve detection accuracy and efficiency. Integration of spam detection models with edge computing devices can reduce latency and bandwidth usage, making detection more scalable and responsive. Additionally, exploring multi-modal data fusion techniques can provide richer contextual information for more accurate detection. Real-time response mechanisms, such as automated actions to block spam messages or isolate compromised devices, can further enhance the system's effectiveness. Implementing self-learning and adaptive mechanisms allows the system to continuously improve and adapt to evolving spamming tactics**.**

8. BIBLIOGRAPHY

8.1 BIBLIOGRAPHY

- Cho, J., Lee, J., Kim, J., & Yoo, J. (2019). Robust spammer detection method based on deep learning for IoT applications. IEEE Access, 7, 134239-134248.

- Wang, Y., Zheng, R., & Liu, Y. (2020). Collaborative Neural Networks for Spam Detection in IoT Applications. 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 135-140.

- Rizvi, S. S., & Al-Jumeily, D. (2018). Machine Learning Approaches for Spam Detection in IoT Networks. International Journal of Advanced Computer Science and Applications (IJACSA), 9(12), 63-69.

8.2 GITHUB LINK

https://github.com/sreeja2702/major-project