NETWORK ANOMALY DETECTION ALERTS USING AWS SNS

A Major Project Report submitted to

Rajiv Gandhi University of Knowledge and Technologies-Ongole

In the partial fulfilment for the award of the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Molli Bhanu Prakash	**O190092**
Pondugula Thriveni	**O190921**
Kagithala Vathsalya	**O191055**
Gajjalakonda Venkateswarlu	**O190420**
Vadithya Manoj Naik	**O190755**

Under the guidance of

Mr.T.SREEKANTH M.Tech.,(PhD)

Assistant Professor

Computer science and engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES ONGOLE

2024-2025

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "NETWORK ANOMALY DETECTION ALERTS USING AWS SNS" Submitted by Molli Bhanu Prakash(O190092), Pondugula Thriveni(O190921), Kagithala Vathsalaya(O191055), Gajjalakonda Venkateswarlu(O190420),Vadithya Manoj Naik(O190755) partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering is a record of bonafide project work carried out under my supervision during the academic year 2024-2025.

We are indebted to Mr.T. Sreekanth M.Tech., (PhD), Our project guide for conscientious guidance and encouragement to accomplish this project. I extremely thankful and pay my gratitude to Mr.N.Mallikarjuna M.Tech(I/C) Head of the Department CSE, for this valuable guidance and support on the completion of this project.

The report hasn't been submitted previously in part or full to this or any other university or institution for the award of any degree.

Mr.T. Sreekanth **M.Tech., (PhD),**	**Mr.N. Mallikarjuna** **M.Tech.,.**
Assistant Professor,	Assistant Professor,
Department of CSE,	Head of the Department(I/C),
RGUKT, ONGOLE.	Department of CSE,
	RGUKT, ONGOLE

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "NETWORK ANOMALY DETECTION ALERTS USING AWS SNS" submitted by Molli Bhanu Prakash(O190092), Pondugula Thriveni(O190921), Kagithala Vathsalya(O191055), Gajjalakonda Venkateswarlu(O190420),Vadithya Manoj Naik(O190755) to the Department of Computer Science and Engineering, Rajiv Gandhi University of Knowledge Technologies, Ongole, during the academic year 2024-2025 is a partial fulfilment for the award of Under graduate degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work record carried out by them under my supervision during the academic year 2024-25.

The Report hasn't been submitted previously in part or in full to this or any other university or institution for the award of any degree.

Mr.T.Sreekanth M.Tech.,(PhD)	Mr.N.Mallikarjuna M.Tech.
Assistant Professor	Assistant Professor
Department of CSE	Head of the Department
RGUKT ONGOLE	Department of CSE
	RGUKT ONGOLE

APPROVAL SHEET

This report entitled "NETWORK ANOMALY DETECTION ALERTS USING AWS SNS" submitted by Molli Bhanu Prakash(O190092), Pondugula Thriveni(O190921), Kagithala Vathsalaya(O191055), Gajjalakonda Venkateswarlu(O190420),Vadithya Manoj Naik(O190755) to Mr.T. Sreekanth M.Tech., (Ph.D), Assistant Professor, RGUKT, Ongole approved for the degree of Bachelor of Technology in COMPUTER SCIENCE AND ENGINEERING.

Examiner

____

____

Supervisor

____

____

____

____

Date: ______________________

Place: ______________________

DECLARATION

We declare that this written submission represents our ideas in our own words where other ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honestly and integrity and have not misrepresented or fabricated or falsified any idea /data/fact/source in my submission.

We understand that any violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

	Signature
MOLLI BHANU PRAKASH(O190092)	____________________
PONDUGULA THRIVENI(O190921)	____________________
KAGITHALA VATHSALYA(O191055)	____________________
GAJJALAKONDA VENKATESWERALU(O190420)	____________________
VADITHYA MANOJ NAIK(O190755)	____________________

DATE:____________________

PLACE:____________________

ACKNOWLEDGEMENT

We take this opportunity to express deep gratitude to the people who have been instrumental in the successful completion of the project.

We would like to thank our project guide **Mr.T.SREEKANTH M.Tech.,(PhD)** which was greatly helpful in the progression and smoothness of the entire project work. We would like to show our gratitude to our beloved Head of the Department, **Mr.N.Mallikarjuna M.Tech** for his encouragement in the aspect of our course, **RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES,** who gave us official support for the progress of our project.

We are deeply indebted to pay our sincere thanks to our Academic Dean **Mr. M. RUPAS KUMAR**, Asst. Professor , Dept. Of Civil Engineering,for showing some exuberant consent for this project and providing me with all the facilities in the department to complete the project.

We convey our profound thanks and gratitude to **Prof. BHASKAR PATEL**, Director of RGUKT – ONGOLE for providing me an excellent academic climate which made this endeavour possible.

We would also like to extend our gratitude and sincere thanks to all the faculty members and Non-Teaching Staff in the Department of Computer Science and Engineering for supporting us. We would also like to thank our parents and friends, who have willingly helped us out with their abilities in completing the project work.

With Sincere Regards

M.BHANU PRAKASH(O190092)

P.THRIVENI(O190921)

K.VATHSALYA(O191055)

G.VENKATESWARLU(O190420)

V.MANOJ NAIK(O190755)

Date: ________________________

ABSTRACT

The Network Anomaly Detection System using the Random Forest algorithm and AWS SNS is a machine learning-based solution that identifies unusual network behavior and sends real-time alerts to system administrators. By applying the Random Forest algorithm, the system can classify network traffic patterns as normal or anomalous with high accuracy, using a decision-tree ensemble to reduce errors and improve prediction reliability. The integration with AWS SNS allows instant notifications through various channels, ensuring that administrators can respond promptly to potential threats. Deployed on AWS infrastructure, this system is scalable and cost-effective, capable of monitoring large networks in real-time. The solution also provides data-driven insights through visualizations, helping organizations improve network security and reduce vulnerabilities.

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 About Our Project

Our project focuses on building a robust Network Anomaly Detection System that leverages machine learning and AWS services for real-time alerts. The system uses a Random Forest algorithm to classify network traffic and predict anomalies based on patterns derived from historical data. Random Forest, a powerful ensemble learning method, ensures high accuracy and reliability in identifying potential threats or unusual activities. Upon detecting an anomaly, the system integrates with AWS Simple Notification Service (SNS) to send real-time alerts via email, SMS, or other notification channels. This combination of predictive modeling and cloud-based alerting provides an efficient, scalable solution for monitoring and securing network environments.

1.2 Motivation of the Project

The motivation behind the Network Anomaly Detection and Alerts System is driven by the increasing complexity and volume of network traffic, coupled with the rising threats of cyberattacks and data breaches. Traditional rule-based systems often fail to adapt to new and evolving attack patterns, making it essential to leverage machine learning for dynamic and accurate anomaly detection. The use of Random Forest provides a reliable and scalable method for identifying irregularities in network behavior. Additionally, integrating with AWS SNS ensures real-time notifications, enabling swift response to potential threats. This project aims to enhance network security, reduce downtime, and minimize the impact of cyber incidents, ensuring a more resilient and proactive defense mechanism.

1.3 Problem definition

The problem addressed by this project is the difficulty in identifying and responding to network anomalies in real-time due to the increasing complexity of network environments and the sophistication of cyberattacks. Traditional detection methods, such as static rules or signature-based systems, struggle to adapt to new and evolving threats, leading to false positives, missed detections, and delayed responses.the lack of an efficient alerting mechanism hampers timely action,increasing the risk of data breaches, service disruptions, and financial losses. The challenge lies in developing a system that can accurately predict anomalies using advanced machine learning techniques, like Random Forest, and promptly notify administrator through AWS SNS for immediate intervention.

1.4 Objectives of project

The objective of the project is to develop an intelligent and scalable network anomaly detection system that combines machine learning and cloud-based notification services to enhance network security. Specifically, the system aims to:

- **Detect and classify anomalies**: Use the Random Forest algorithm to accurately identify abnormal network activities by analyzing traffic patterns.
- **Reduce false positives and negatives**: Improve detection accuracy compared to traditional rule- based methods to ensure reliable anomaly identification.
- **Provide real-time alerts**: Leverage AWS SNS to send timely notifications (via email, SMS, etc.) to administrators, enabling rapid response to potential threats.
- **Ensure scalability and efficiency**: Design the system to handle high volumes of network traffic and adapt to evolving patterns, making it suitable for diverse network environments.
- **Enhance proactive threat management**: Minimize the risk of data breaches, service disruptions, and cyberattacks by enabling early detection and swift action.

CHAPTER-2

LITERATURE SURVEY

2.1 Literature Review

The literature review provides the conceptual framework that guides the research and connects it to existing knowledge. Here's how theoretical elements apply to your topic of network anomaly detection using AWS SNS and the Random Forest algorithm:

[A] DARPA Intrusion Detection System

Proposed by Dr. David L. Tennenhouse.

This project employed machine learning algorithms like KNN and decision trees on the DARPA/KDD dataset to identify network intrusions and set standards for anomaly detection systems.

[B] University of New Brunswick - CICIDS Projects

Led by Dr. Ali Ghorbani.

These projects created the CICIDS datasets, utilizing Random Forest and neural networks to detect anomalies in network traffic and provide real-time threat alerts.

[C] IBM QRadar Network Anomaly Detection

Authored by Dr. Rob Clyde.

This project combined Random Forest and clustering algorithms with real-time alerting monitor enterprise network traffic and detect suspicious activities.

[D] Cisco Stealthwatch Security Analytics

Proposed by Dr. Gee Rittenhouse.

Using machine learning models such as Random Forest and unsupervised clustering, Stealthwatch identifies anomalies in network flows and generates real-time security alerts to help prevent cyberattacks.

[E] Adobe's VAST (Visual Analytics Security Toolkit)

Authored by Dr. Abhay Parasnis.

This toolkit applies machine learning techniques, including Random Forest and deep learning, to detect anomalies in network logs. Alerts are sent through platforms like Slack and email for immediate action.

[F] UCLA's NetSage for Anomaly Detection

Proposed by Professor Mani Srivastava.

This project leverages neural networks and time-series analysis to identify anomalies network traffic and deliver detailed visual analytics.

[G] **Nokia Bell Labs Network Anomaly Monitor**

Led by Dr. Marcus Weldon.

This initiative combines Random Forest and Autoencoders to monitor and classify anomalies intelecommunications networks in real time.

[H] **Sandia National Laboratories Network Detection Project**

Authored by Dr. Cynthia Phillips.

This project applies Random Forest and decision tree algorithms to detect cyber threats within critical infrastructure networks, paired with advanced alerting mechanisms.

[I] **Carnegie Mellon CERT Network Situational Awareness Tool**

Proposed by Dr. Richard Pethia.

Utilizing clustering and Bayesian networks, this tool identifies anomalies in enterprise traffic improve network threat detection and response.

CHAPTER-3
ANALYSIS

3.1 Existed System

The DARPA project which is Proposed by Dr. David L. Tennenhouse., although foundational, relied heavily on early machine learning techniques like decision trees and KNN, which may not adapt as well to real-time detection needs in modern, high-speed network environments.This project lacked advanced alerting systems, as it primarily focused on static, rule-based systems for analysis rather than dynamic or automated alerts.

3.2 Proposed System

The proposed system leverages machine learning, specifically the Random Forest algorithm, integrated with AWS Simple Notification Service (SNS) to detect and alert anomalies in network traffic. This system is designed for real-time detection, scalability, and efficient alerting.

1. **Data Collection and Preprocessing**

   Network traffic data is gathered from sources like firewalls, intrusion detection systems, and logs. Preprocessing involves cleaning, feature extraction (e.g., packet size, IP address, protocol types), and labeling data as “normal” or “anomalous.” Historical data is used to train the model, ensuring its effectiveness in supervised learning tasks.

2. **Machine Learning-Based Detection**

   The Random Forest algorithm is employed for its high accuracy and ability to handle high-dimensional data. Using multiple decision trees, it classifies network activities into normal or anomalous patterns. The model is trained and validated using labeled datasets to minimize false positives while ensuring robust anomaly detection.

3. **Real-Time Monitoring and Alerting**

   Real-time network traffic is continuously analyzed using the trained model. Detected anomalies trigger an AWS SNS topic to send immediate alerts. Notifications are sent to stakeholders via email, SMS, or other channels. This ensures swift response to potential threats, minimizing downtime and damage.

3.3 HARDWARE REQUIREMENTS

- Operating System – Ubuntu 20.04 LTSDevelopment – Visual Studio Code,Text Editor
- Laptop
- Mouse
- 1TB HDD Hard Disk

3.4 SOFTWARE SPECIFICATIONS

- Browser
- Python Code
- Streamlit
- Ubuntu 20.04
- AWS SNS (Simple Notification Service)
- Boto3 (AWS SDK for Python)

3.4.1 OVERALL DESCRIPTION

The project "Network Anomaly Detection Alerts using AWS SNS and Random Forest Algorithm" focuses on identifying unusual patterns in network traffic that could indicate potential security threats. It uses machine learning to classify network data as normal or anomalous, and AWS Simple Notification Service (SNS) to alert system administrators when anomalies are detected.

1. **Data Collection and Preprocessing:** Network traffic data is gathered from sources like system logs or network monitoring tools. This data includes information like IP addresses, packet sizes, and timestamps. It needs to be cleaned and processed before it can be used in machine learning models. This might involve removing errors, handling missing data, and scaling numerical values to ensure that the data is ready for analysis.

2. **Anomaly Detection with Random Forest:** The Random Forest algorithm is used to detect anomalies because it can handle large, complex datasets and is effective at distinguishing between normal and abnormal patterns. The model is trained using labeled data, where the traffic is already marked as normal or anomalous. After training, the model can classify new data based on what it has learned. If it detects an anomaly, it flags it as a potential security threat.

3. **Real-time Detection:** Once deployed, the system monitors live network traffic continuously. As new data arrives, the Random Forest model evaluates it in real time, classifying it as normal or anomalous. If an anomaly is detected, it triggers an alert.

4. **AWS SNS for Alerting:** When an anomaly is detected, the system uses AWS SNS to send an alert. SNS can send messages via email, SMS, or other channels to notify administrators of the potential issue. This allows for quick responses to any security threats.

5. **Visualization:** A dashboard can be added to visualize the data and anomaly trends, helping administrators understand the network's health and track any detected issues.

CHAPTER-4

SYSTEM DESIGN

The Unified Modelling Language (UML) diagrams are drawn for the project with the best possible interpretation of the project

4. UML DIAGRAMS

UML is the short form of Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The important goal for UML is to create a common modelling language for the sake of Object-Oriented Software engineering. In its current form UML consists of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modelling and other non- software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

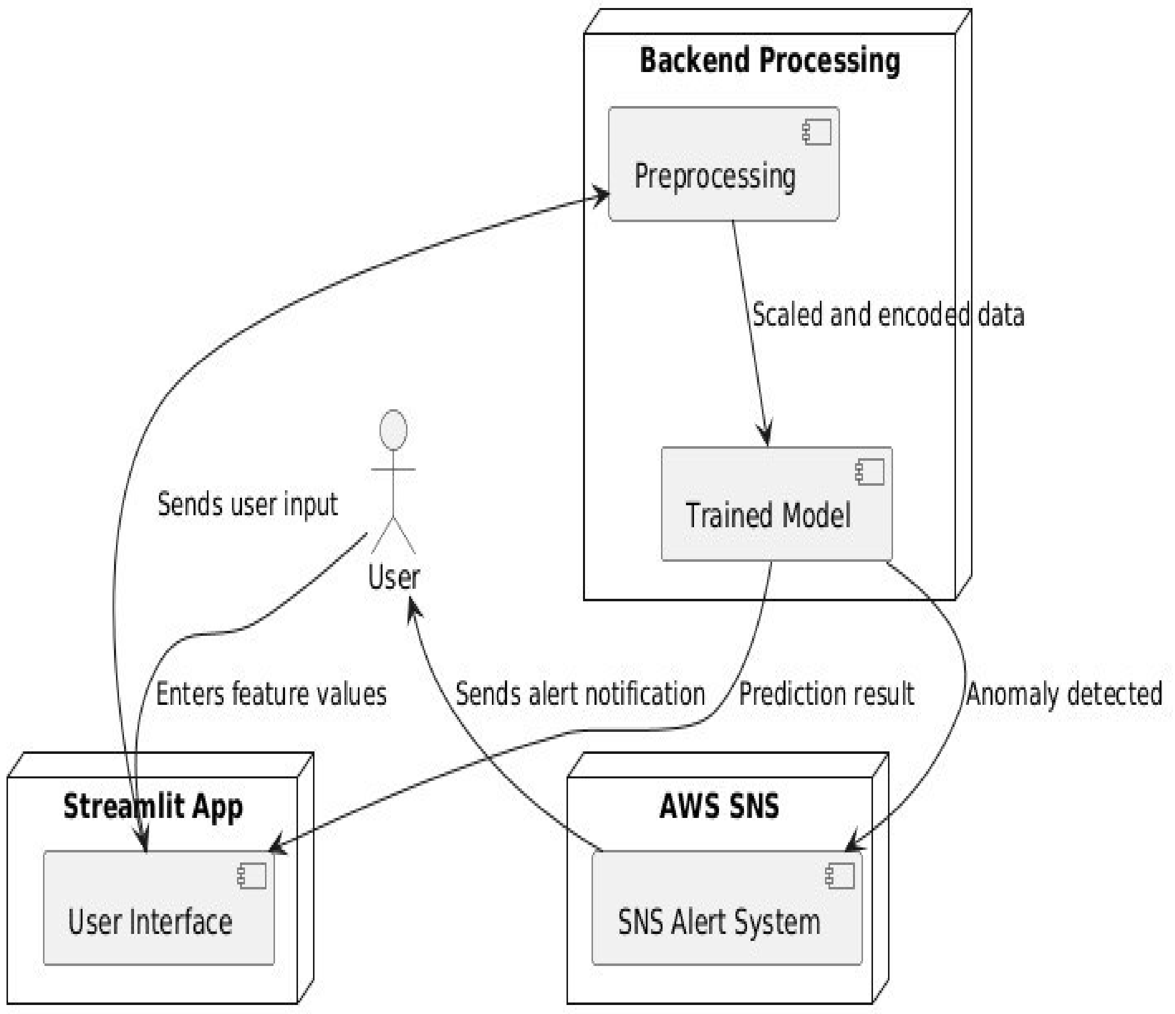
GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modelling language
5. Encourage the growth of the Object-Oriented tools market.
6. Support higher level development concepts such as collaborations frameworks, patterns and components and Integrate best practices.
7. A UML Diagram is based on UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. The UML diagrams are divided into Structural and Behavioural UML.

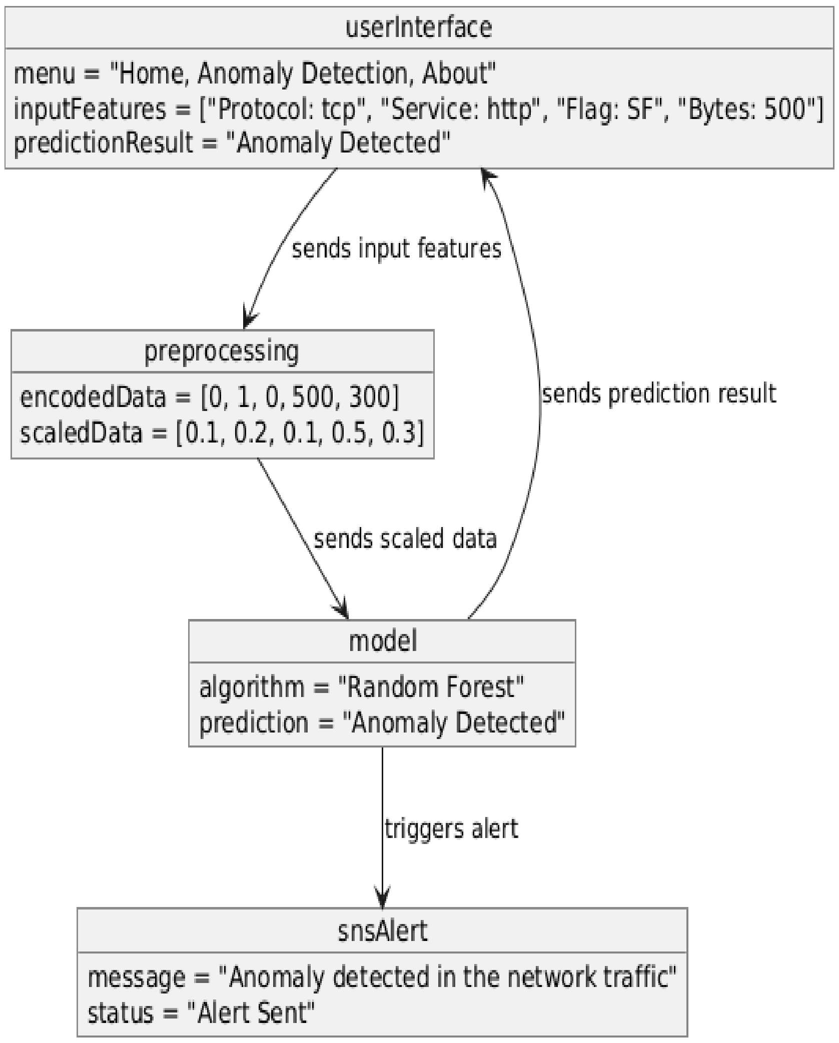
4.1 ARCHITECTURE

The architecture of a project defines the overall structure and organization of its system. It includes the arrangement of software components, their interactions, and data management. It also covers the underlying infrastructure, such as servers or cloud services, supporting the system. The architecture defines how the system meets performance, scalability, and security requirements. A well-designed architecture ensures the project is efficient, maintainable, and adaptable to future needs.



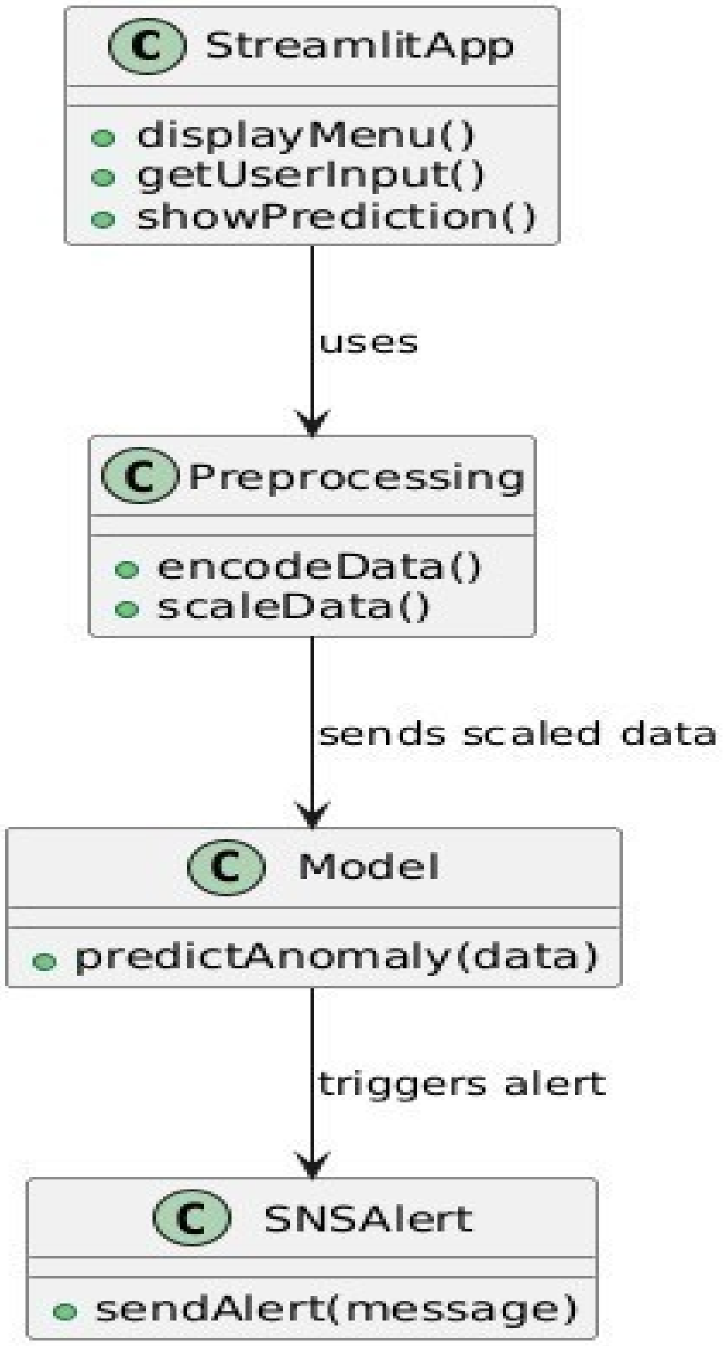
4.1.1 OBJECT DIAGRAM

A UML object diagram represents a specific instance of a class diagram at a certain moment in time.When represented visually, you'll see many similarities to the class diagram. An object diagram focuses on the attributes of a set of objects and how those objects relate to each other.


userInterface
menu = "Home, Anomaly Detection, About"
inputFeatures = ["Protocol: tcp", "Service: http", "Flag: SF", "Bytes: 500"]
predictionResult = "Anomaly Detected"
sends input features
sends prediction result
preprocessing
encodedData = [0, 1, 0, 500, 300]
scaledData = [0.1, 0.2, 0.1, 0.5, 0.3]
sends scaled data
model
algorithm = "Random Forest"
prediction = "Anomaly Detected"
triggers alert
snsAlert
message = "Anomaly detected in the network traffic"
status = "Alert Sent"

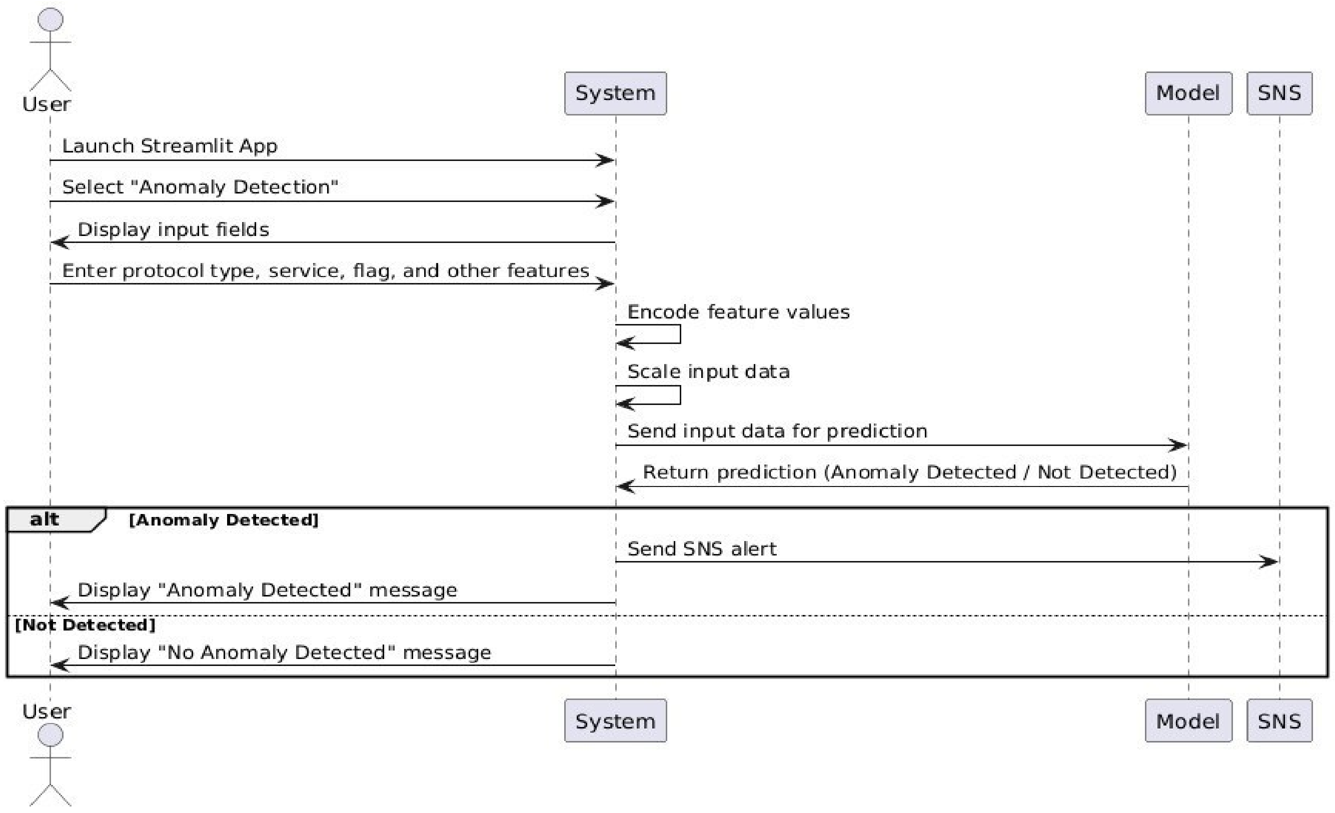
4.1.2 CLASS DIAGRAM

A class diagram is a type of static structure diagram in software engineering that illustrates the structure of a system by depicting its classes, attributes, operations (methods), and the relationships among objects. It provides a visual representation of the classes and their associations within the system, serving as a blueprint for the software design and architecture.



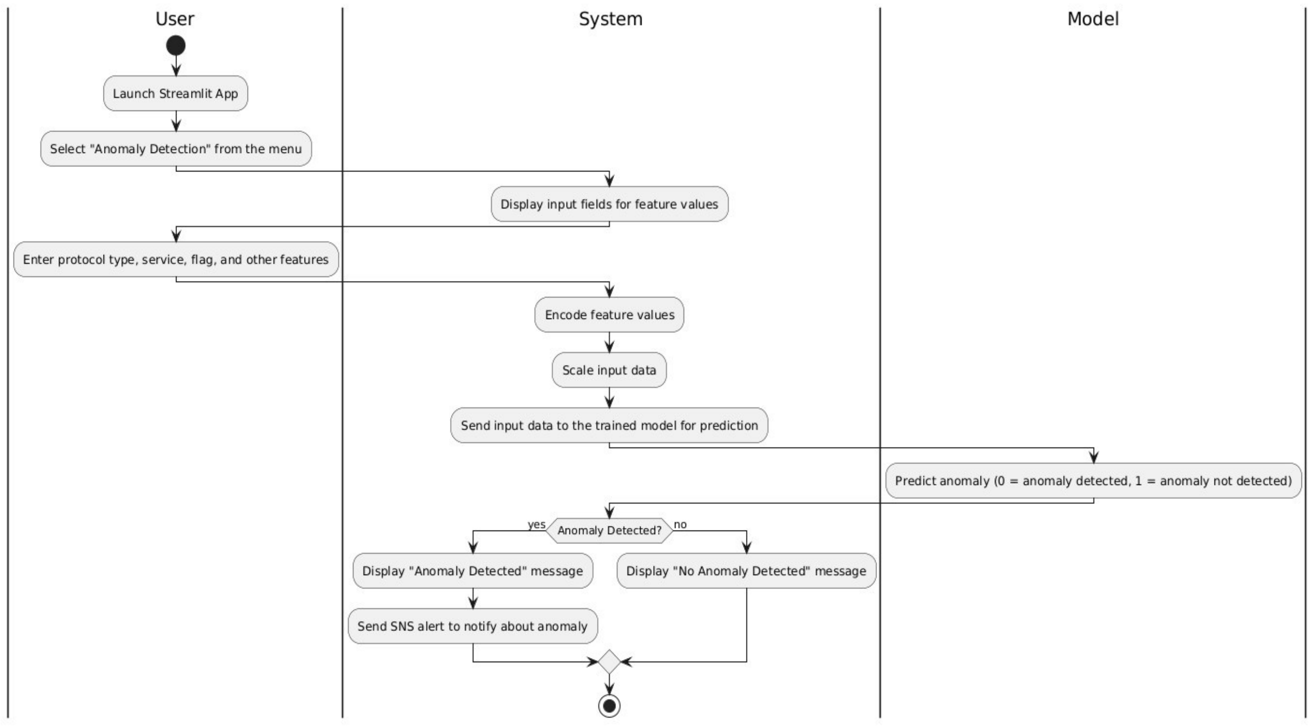
4.1.3 SEQUENCE DIAGRAM

A sequence diagram in UML illustrates how objects interact with each other over time, showing the sequence of messages exchanged between them. It emphasizes the order of events or actions in a specific scenario or process.


User
System
Model
SNS
Launch Streamlit App
Select "Anomaly Detection"
Display input fields
Enter protocol type, service, flag, and other features
Encode feature values
Scale input data
Send input data for prediction
Return prediction (Anomaly Detected / Not Detected)
alt
[Anomaly Detected]
Send SNS alert
Display "Anomaly Detected" message
[Not Detected]
Display "No Anomaly Detected" message

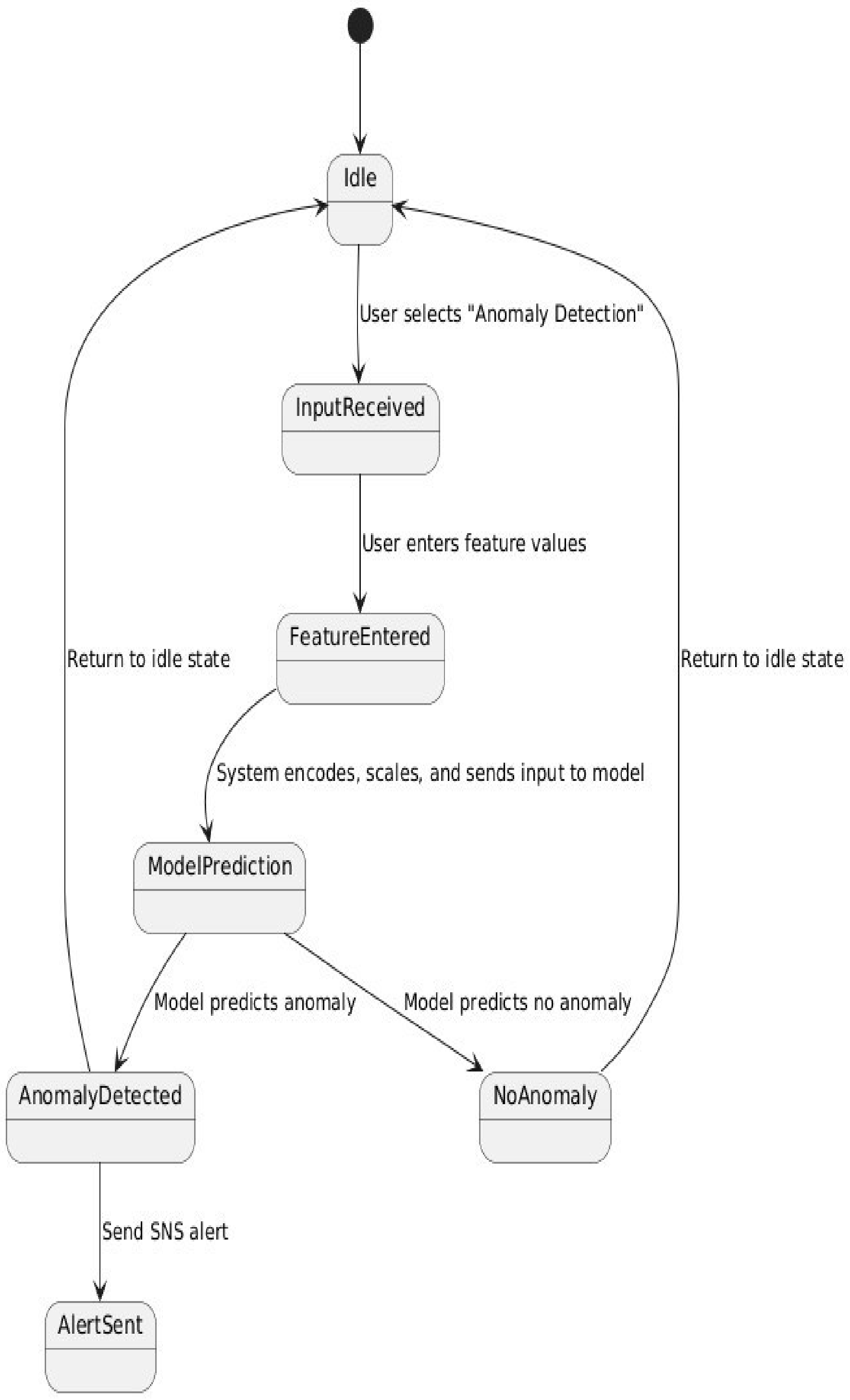
4.1.4 ACTIVITY DIAGRAM

An activity diagram in UML models the flow of activities or tasks within a system, showing the sequence of actions, decisions, and concurrency. It is used to represent workflows, business processes, or use case scenarios.


User
System
Model
Launch Streamlit App
Select "Anomaly Detection" from the menu
Display input fields for feature values
Enter protocol type, service, flag, and other features
Encode feature values
Scale input data
Send input data to the trained model for prediction
Predict anomaly (0 = anomaly detected, 1 = anomaly not detected)
Anomaly Detected?
yes
no
Display "Anomaly Detected" message
Display "No Anomaly Detected" message
Send SNS alert to notify about anomaly

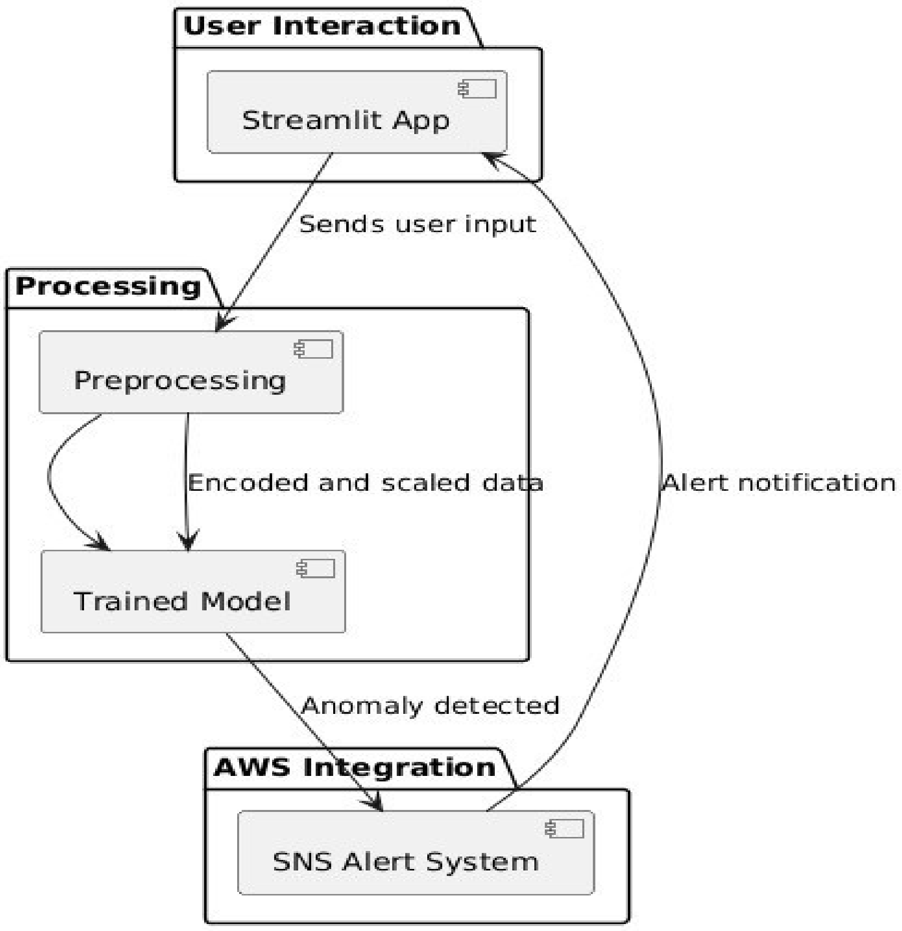
4.1.5 STATE MACHINE DIAGRAM

A state machine diagram in UML is a visual representation of an object's behavior, showcasing its states and transitions. It illustrates how events trigger state changes, often with conditions (guards) and actions, providing a clear understanding of complex systems.



4.1.6 COMPONENT DIAGRAM

A component diagram in UML shows the structure of a system by representing its components (e.g., software modules or hardware devices) and how they interact with each other. It focuses on the organization and dependencies between components.



4.2 STATISTICAL ANALYSIS

4.2.1 Bar Plot for Class Distribution

Description

This bar plot represents the distribution of data points in the dataset, categorized as either “Normal" or "Anomaly."

- **X-axis**: Displays the class names ("Normal" and "Anomaly").
- **Y-axis**: Represents the count or percentage of data points for each.

Analysis

The bar plot provides insights into class distribution

- Helps determine if the dataset is balanced or imbalanced.
- An imbalanced dataset may require preprocessing techniques like oversampling or undersampling.

Visualization

- **X-axis:** Class names (e.g., "Normal," "Anomaly").
- **Y-axis:** Count or percentage of data points.
- **Normal Class:** Shown with a green bar.
- **Anomaly Class:** Shown with a blue bar.

Usage

This bar plot is useful for understanding the dataset's class distribution. It aids in:

- Quickly identifying class imbalances.
- Planning preprocessing steps for model training.
- Explaining data insights in reports and presentations.

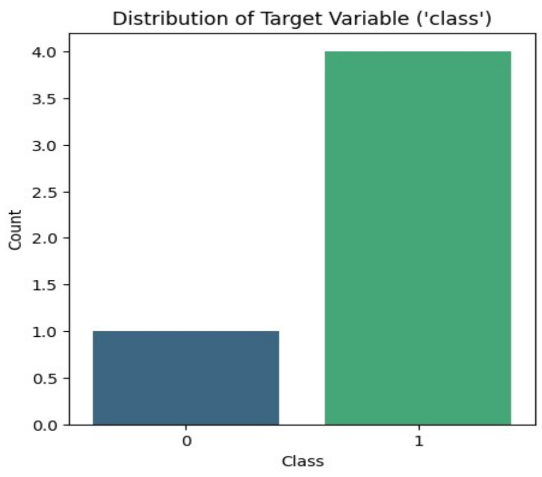


Fig:4.2.1 Bar Plot for Class Distribution

4.2.2 Visualization for Source and Destination Bytes by Class

a. Box Plot for Source Bytes

Description

The box plot groups the source byte values based on their class (Normal or Anomaly).

- **X-axis**: Represents the class (Normal and Anomaly).
- **Y-axis**: Represents the source byte values.

Analysis

- Highlights the distribution of source bytes for normal and anomalous traffic.
- Shows differences in the range, median, and outliers for each class.
- Useful for identifying whether anomalous data exhibits distinct patterns compared to normal data.

Visualization

- **X-axis**: Class (Normal, Anomaly).
- **Y-axis**: Source Byte Values.
- Box plot elements include median,interquartile range (IQR), and potential outliers for each class.

Usage

- Understand differences in source byte distributions for normal and anomalous traffic.
- Detect potential indicators of anomalies through byte patterns.

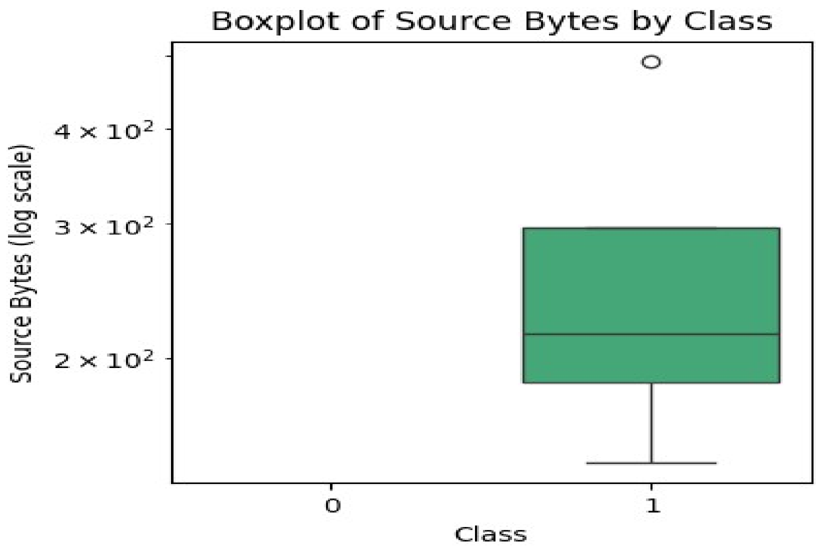


Fig:4.2.2 a) Box Plot for Source Bytes

B. Box Plot for Destination Bytes

Description

This box plot visualizes the distribution of destination byte values grouped by class (Normal or Anomaly).

- **X-axis**: Represents the class (Normal and Anomaly).
- **Y-axis**: Represents the destination byte values.

Analysis

- Displays variations in destination byte traffic for normal and anomalous data.
- Highlights potential outliers and variations in traffic patterns between classes.

Visualization

- **X-axis**: Class (Normal, Anomaly).
- **Y-axis**: Destination Byte Values.
- Includes key box plot elements such as median, IQR, and outliers for each class.

Usage

- Analyze differences in destination byte traffic patterns for normal and anomalous data.
- Identify significant features for anomaly detection models.
- Gain insights into traffic behavior that may indicate anomalies.

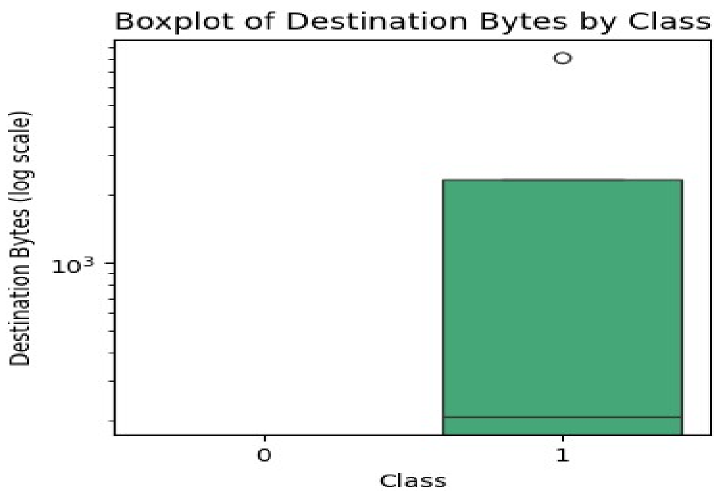


Fig:4.2.2 b) Boxplot of Destination Bytes by class

4.2.3 Visualization for Protocol Types, Flags, Services, and Class Distribution

Description

This series of bar graphs compares the distribution of protocol types, flags, services, and class labels (Normal vs. Anomaly) across your dataset. It helps understand how different features vary between normal and anomalous traffic.

- **Protocol Types**: Identifies if certain protocols (TCP, UDP, ICMP) are more commonly used in anomalies or normal traffic.
- **Flags**: Highlights the prevalence of specific flags (e.g., SYN, ACK, RST) that could indicate network attacks, such as SYN flooding.
- **Services**: Shows the frequency of service usage (HTTP, FTP, DNS) and identifies whether certain services are targeted during anomalous behavior.
- **Class Distribution**: Helps determine if your dataset is balanced or imbalanced between normal and anomaly classes, which is crucial for model performance.

Visualization

- **Bar Graph for Protocol Types by Class**
  - **X-axis**: Protocol types (e.g., TCP, UDP, ICMP).
  - **Y-axis**: Frequency of each protocol type in normal and anomalous classes.
  - **Purpose**: Identifies protocol patterns for different types of traffic.
  - **Usage**: Helps detect suspicious protocol behavior.
- **Bar Graph for Flags by Class**
  - **X-axis**: Network flags (e.g., SYN, ACK, RST).
  - **Y-axis**: Frequency of each flag in normal and anomalous data.
  - **Purpose**: Shows how flag usage differs between normal and anomalous traffic.
  - **Usage**: Detects attack patterns like SYN flooding or RST packet abuse.
- **Bar Graph for Services by Class**
  - **X-axis**: Services (e.g., HTTP, FTP, DNS).
  - **Y-axis**: Frequency of service usage in normal vs. anomalous data.
  - **Purpose**: Identifies service usage patterns for normal and anomalous traffic.
- **Bar Graph for Class Distribution**
  - **X-axis**: Class labels (Normal, Anomaly).
  - **Y-axis**: Number of instances in each class.
  - **Purpose**: Shows the overall distribution of normal and anomalous data.
  - **Usage**: Helps in understanding if your dataset is imbalanced, which may require handling techniques like oversampling or undersampling.

Usage

These bar graphs are crucial for visualizing how network traffic behaviors differ between normal and anomalous data, which aids in identifying attack patterns, anomalies, and class distribution issues. They also help in preparing your dataset for more effective anomaly detection and model training.

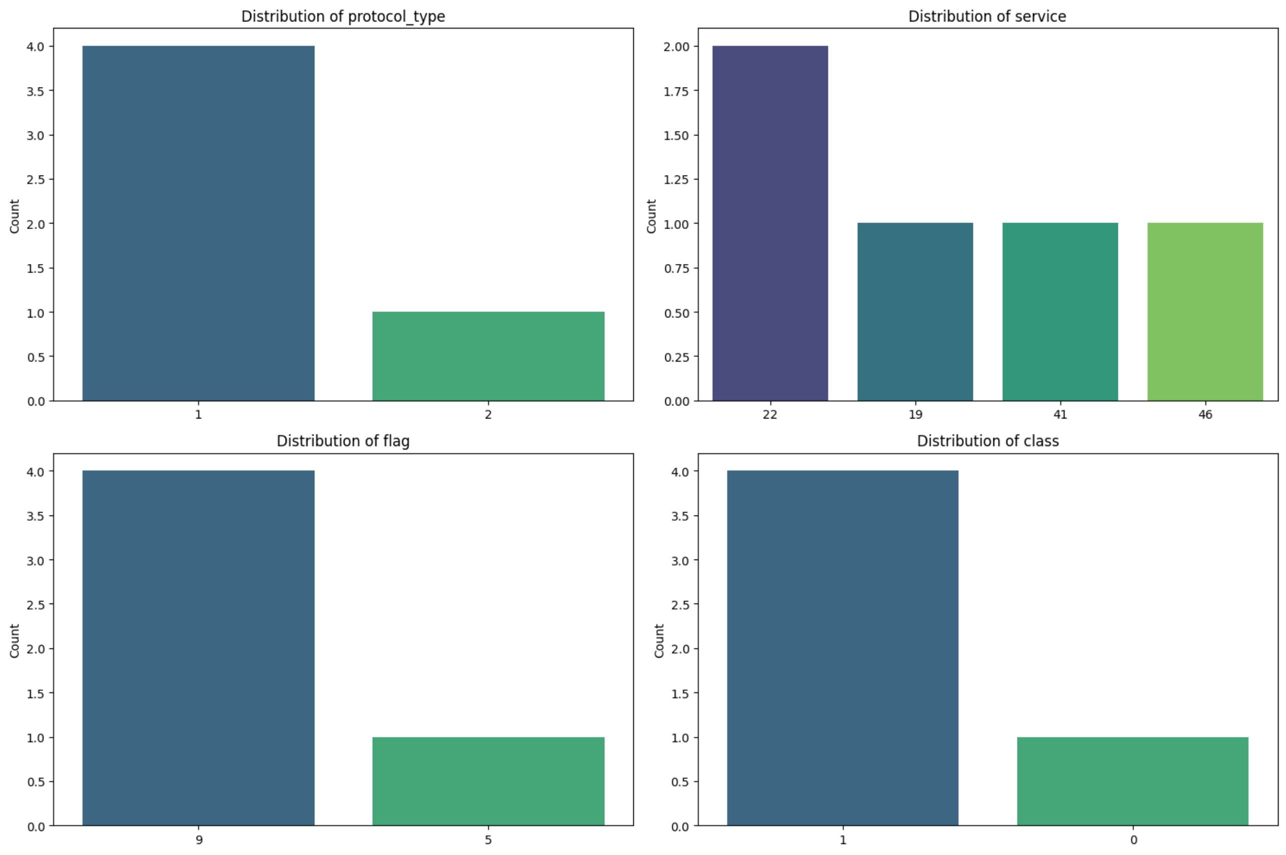


Fig:4.2.3 Bar Graph Protocol Types, Flags, Services, and Class Distribution

\

4.2.4 Line Plot Visualization for All Features in the Dataset

Description

Line plots track changes in data over time, helping to spot patterns and anomalies in features like bytes, protocols, flags, services, and class labels.

- **Source and Destination Bytes**: Shows changes in byte values, helping to spot spikes or drops that may signal problems.
- **Protocol Types**: Tracks how often different protocols are used, revealing unusual protocol patterns.
- **Flags**: Displays flag usage over time, helping detect attacks like SYN floods or resets.
- **Services**: Shows how services (e.g., HTTP, FTP) are used, helping to identify suspicious service activity.
- **Class Labels**: Adds markers for Normal or Anomaly to show when anomalies occur in other features.

Details for Each Feature

- **Source and Destination Bytes**: Spot unusual traffic spikes or drops.
- **Protocol Types**: Track protocol usage to identify abnormal patterns.
- **Flags**: Visualize flags to find attack patterns.
- **Services**: Track service usage to detect targeted attacks.
- **Class Labels**: See how feature changes match anomalies or normal behavior.

Usage

Line plots help you see how features change over time, making it easy to spot trends and anomalies in network behavior. This is valuable for detecting attacks and fine-tuning detection models

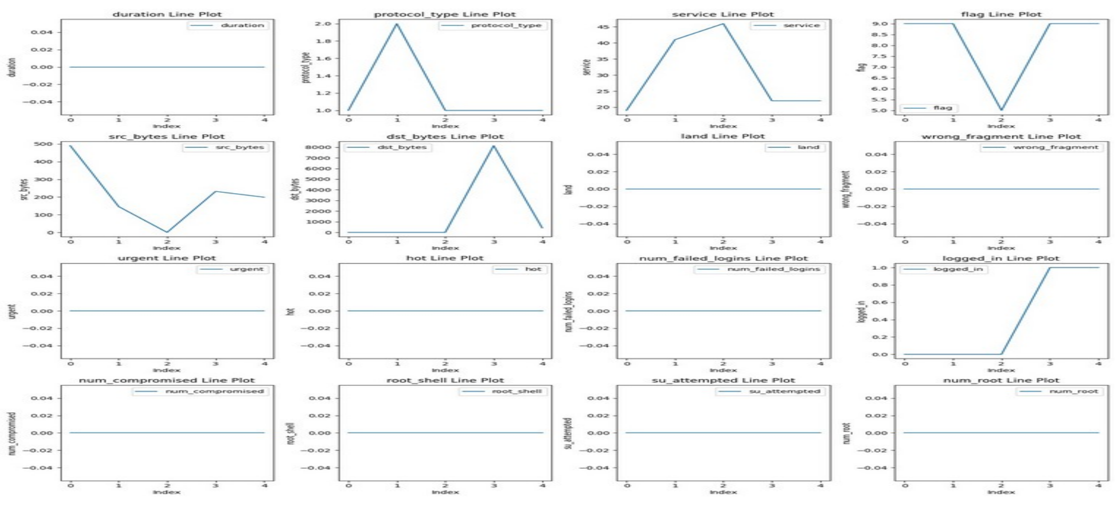


Fig:4.2.4 Line Plot Visualization for All Features in the Dataset

4.2.5 Pie charts for Class disturbution by feature categories

Description

Pie charts show the proportion of Normal and Anomaly classes within feature categories like Duration, Protocol Type, Service, and Flag. They can group numerical data (e.g., Duration ranges) into categories for easier visualization.

- **Shows class proportions** within categories (Normal vs. Anomaly).
- **Helpful for grouped data** like Duration ranges, Protocol Types, etc.
- **Quick comparison** of class distribution across features.

Visualization

- **Slices**: Represent Normal and Anomaly classes within each category.
- **Labels**: Show percentages for Normal and Anomaly classes.
- **Colors**: Different colors for Normal and Anomaly (e.g., blue for Normal, red for Anomaly).

Usage

- **Duration**: Show class distribution within different time ranges.
- **Protocol Type**: Compare Normal vs. Anomaly in protocols like TCP, UDP.
- **Service**: Visualize class distribution across services like HTTP, FTP.
- **Flag**: Display class distribution for flags like SYN, ACK.

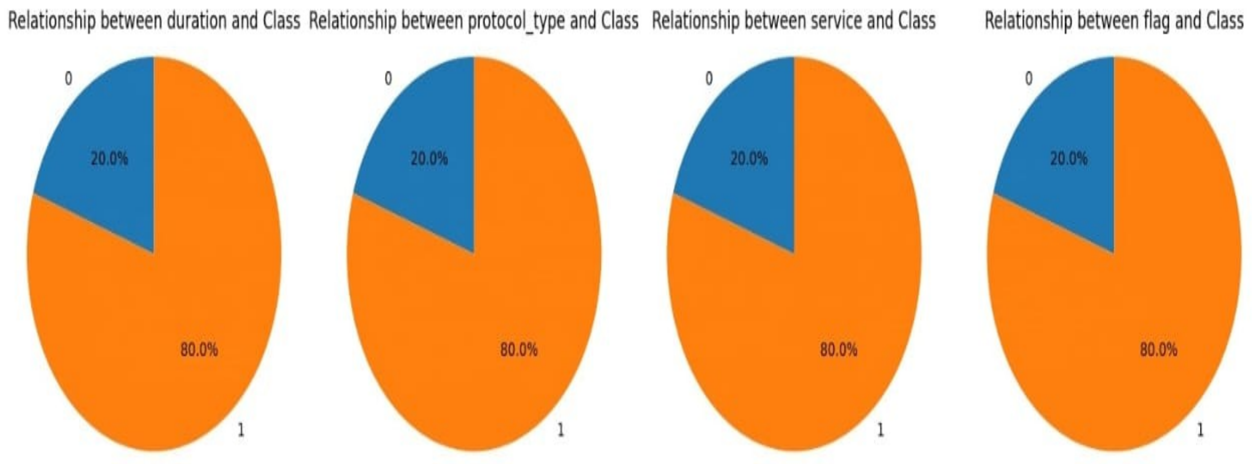


Fig:4.2.5 Pie Chart for Class Distribution by Feature Categories

4.2.6 Line Plot Visualization for Failed and Delivered Alerts In Cloud Metrics

Line Plot for Delivered Alerts

- **Purpose**: Tracks the trend of successful alerts delivered over time.
- **Insights**: Highlights periods with high or low delivery activity, helping to identify potential bottlenecks or system downtime that could affect alert delivery.

Line Plot for Failed Alerts

- **Purpose**: Monitors the number of failed alerts over time.
- **Insights**: Identifies patterns or spikes in failures, which could signal system issues, outages, or other anomalies that affect alert delivery.

These line plots help visualize alert performance, making it easier to detect anomalies or operational problems in the system. They provide valuable insights for system monitoring and troubleshooting.

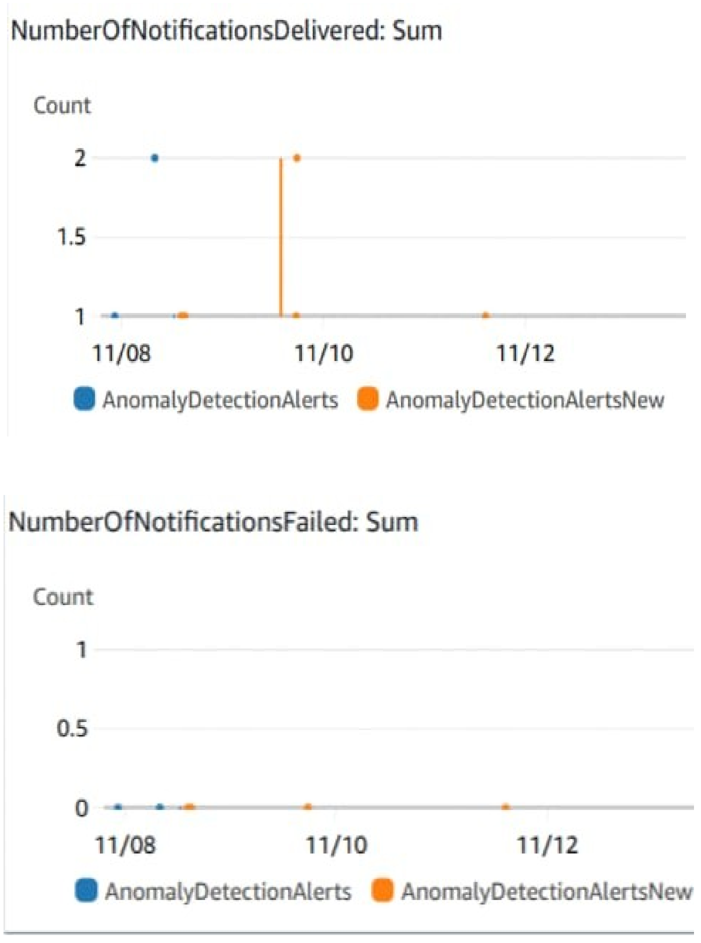


Fig:4.2.6 Line Plot Visualization for Failed and Delivered Alerts In Cloud Metrics

CHAPTER-5
IMPLEMENTATION

5.1 MODULE

1.Anomoly detection

5.2 MODULE DESCRIPTION

1. ANOMOLY DETECTION

This project implements a Network Anomaly Detection System that leverages a machine learning model and cloud notification services to detect and alert on anomalous activities in network traffic. Below is a breakdown of the project modules:

Inputs:

1. **Protocol Type**: Specifies the network protocol (e.g., TCP, UDP) to identify communication patterns and detect unusual protocol usage.
2. **Service**: Indicates the application-level service (e.g., HTTP, FTP) to monitor abnormal or unauthorized service access.
3. **Flag**: Denotes TCP connection status (e.g., SYN, ACK) to identify suspicious connection states or attacks.
4. **Destination Bytes**: Tracks data sent to the destination to detect anomalies like large transfers or failed attempts.
5. **Count**: Measures the number of connections to the same destination in a time window to detect potential scanning or flooding.
6. **Same Service Rate**: Percentage of connections to the same service, indicating consistent or unusual service usage.
7. **Different Service Rate**: Tracks the rate of connections to different services to identify diverse or erratic activity.
8. **Destination Hot Service Count**: Counts connections to frequently accessed services on the destination to flag hotspots or attacks.

5.2 TECHNOLOGIES USED

1. Python

Python is a widely-used programming language known for its readability and simplicity. It supports multiple programming paradigms, includingprocedural, object-oriented, and functional programming.

Python’s extensive standard library and large community contribute to its versatility in various domains such as web development, data science, artificial intelligence, and automation. Its syntax is designed to be intuitive, making it an excellent choice for beginners and experienced programmers alike.

2. Streamlit

Streamlit is an open-source Python library that allows you to create beautiful, interactive web applications for data science and machine learning projects with minimal effort. It simplifies the process of building web apps by focusing on the simplicity of Python scripting. With Streamlit, you can turn data scripts into shareable web apps in just a few lines of code, making it easy to visualize and share data insights quickly.

3. Machine Learning

Machine learning, particularly Random Forest, plays a central role in the anomaly detection system. The model is trained to classify network traffic based on past data, learning to distinguish between normal and abnormal behavior. By analyzing various features like protocol types, service names, and traffic counts, the model can predict if incoming traffic exhibits signs of a network intrusion. This makes machine learning an essential tool for automating and scaling network security solutions.

4. AWS SNS (Simple Notification Service)

AWS SNS is used to send real-time notifications when an anomaly is detected. It is a fully managed service that allows you to publish messages to a large number of subscribers using various protocols, including email, SMS, and mobile push notifications. In this project, AWS SNS ensures that stakeholders are promptly alerted in case of security threats, facilitating quick responses to network anomalies by automatically triggering notifications when an attack or breach is detected.

5. Boto3 (AWS SDK for Python)

Boto3 is the Python SDK that allows seamless integration with AWS services like SNS. It enables Python applications to interact with AWS infrastructure, facilitating tasks such as publishing messages to SNS topics. With Boto3, the system can automatically send alerts to predefined recipients when anomalies are detected in the network traffic. This makes Boto3 a critical component for enabling cloud-based automation and notification systems in security applications.

6. Scikit-learn

Scikit-learn is a powerful machine learning library in Python that provides tools for data preprocessing, model selection, training, and evaluation. In this project, StandardScaler from Scikit-learn is used to normalize input data before feeding it into the anomaly detection model,ensuring that all features contribute equally to the prediction. The library also supports the training and prediction pipeline, making it essential for the overall machine learning workflow.

7. NumPy and Pandas

NumPy and Pandas are two cornerstone libraries in Python for handling large datasets and performing data manipulation. NumPy is for numerical computations and matrix operations,which are crucial when processing the network traffic data before feeding it into the machine learning model. Pandas, on theother hand, allows for easy handling of structured data in the form of DataFrames, providing efficient data manipulation and analysis capabilities for the input features

8. CSS Styling

CSS (Cascading Style Sheets) is used to enhance the visual presentation of the web application created with Streamlit. By incorporating custom CSS, you can personalize the app’s look and feel, ensuring that it is both functional and visually appealing. In this project, CSS is employed to add a professional background, modify the appearance of the sidebar, and customize the top bar, creating a user-friendly interface that enhances user engagement and experience.

9. Base64 Encoding

Base64 encoding is employed to convert binary files, such as images, into text format so that they can be embedded directly within HTML or CSS. This method is used to embed the background image in the Streamlit application, eliminating the need for external image hosting. By using Base64 encoding, the image is embedded directly into the app’s code, ensuring that the visual elements load efficiently and are available offline, creating a more seamless user experience.

5.3 SAMPLE CODE

1. ANOMOLY DETECTION

```
import streamlit as st

import pandas as pd

import numpy as np

import pickle

import boto3

import base64

from sklearn.preprocessing import

StandardScaler from streamlit_option_menu

import option_menu

# Initialize the SNS client (ensure AWS CLI is configured with appropriate permissions)

sns_client = boto3.client('sns', region_name='eu-north-1')  # Replace 'eu-north-1' with your region

sns_topic_arn = 'arn:aws:sns:eu-north-1:711387115919:AnomalyDetectionAlertsNew'   # Replace with
your SNS Topic ARN

# Convert the background image to base64 def
```

```
 get_base64_image(image_path):
  with open(image_path, "rb") as img_file:
    return base64.b64encode(img_file.read()).decode()
#Load the background image and convert it to base64
image_path = "/home/rgukt/Downloads/neural2.gif"  # Path to your high-definition background image
base64_image = get_base64_image(image_path)
# Set background and topbar colors
background_color = "#2E3B4E"  # Example background color for the
topbar text_color = "#FFFFFF"  # Example white text color for the topbar
# Apply CSS with the background image and topbar styling
background_css = f"""
<style>
.stApp {{
background-image: url("data:image/jpeg;base64,
{base64_image}"); background-size: cover;
background-position: center;
background-attachment: fixed;}}
.streamlit-topbar {{
background-color: {background_color};
color: {text_color};
font-size: 20px;
font-weight: bold;}}
.stTitle, .stSidebar {{ color: white; background-color: rgba(0, 0, 0, 0.6);}}
</style>
"""
st.markdown(background_css, unsafe_allow_html=True)
# Load trained model and scaler
```

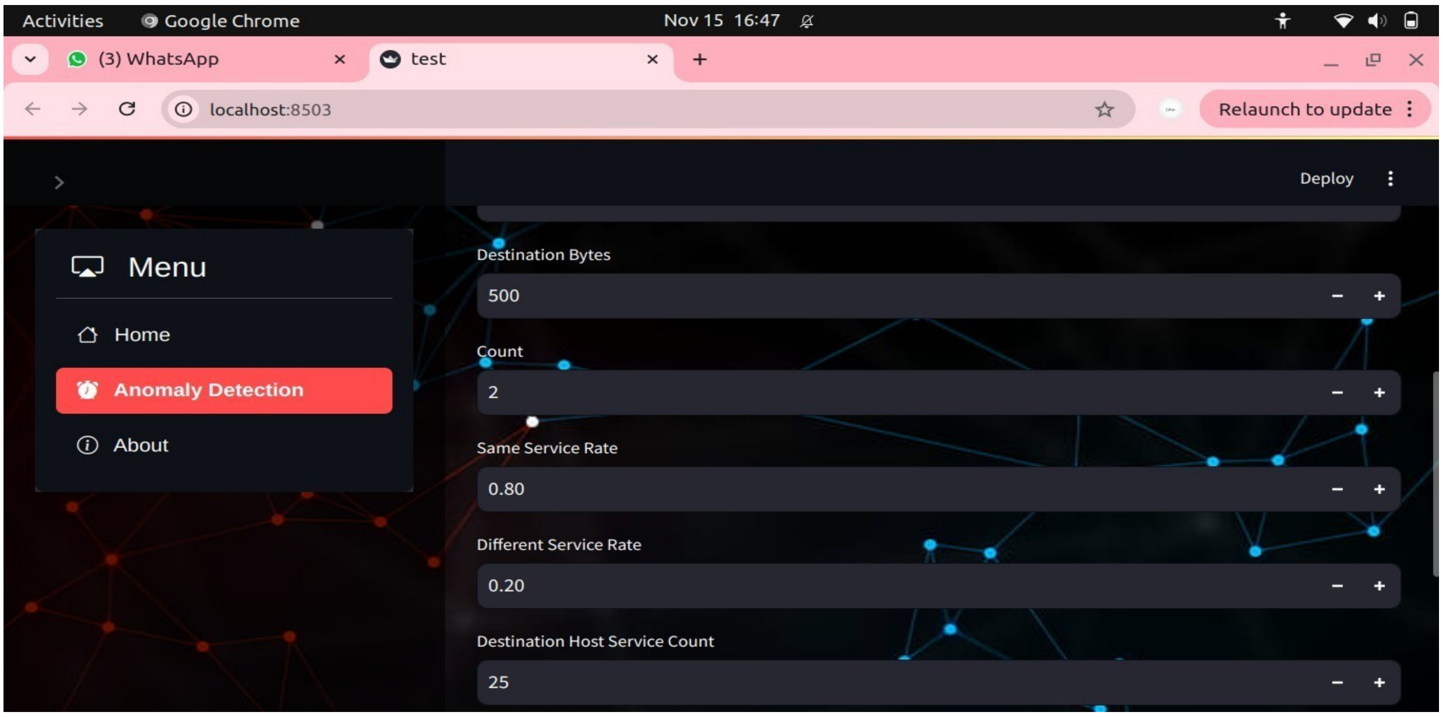
```
model = pickle.load(open("model.pkl", "rb")) scaler = StandardScaler()
scaler.fit(pd.DataFrame(np.random.randn(100, 10))) # Placeholder to fit
scaler # Main function to handle Streamlit UI and logic
def main():

st.title("Network Anomaly Detection System")

menu = ["Home", "Anomaly Detection", "About"]

# Create the menu in the sidebar

with st.sidebar:
        choice = option_menu("Menu", menu, icons=['house', 'alarm-fill', 'info-circle'], menu_icon="cast",
default_index=0)

if choice == "Home":

        st.markdown("<div class='home-text'><h4>Welcome To The Network Anomaly Detection
Alerting System Through      AWS SNS</h4></div>", unsafe_allow_html=True)

elif choice == "Anomaly Detection":

   st.subheader("Anomaly Detection with Random Forest")

   st.header("Enter Feature Values")

   # Define mappings for protocol_type, service names, and flag descriptions

   protocol_mapping = {"tcp": 0, "udp": 1, "icmp": 2}

   service_mapping = {

      "ftp": 0, "http": 1, "smtp": 2, "dns": 3, "telnet": 4,

      "ftp_data": 5, "ldap": 6, "pop3": 7, "irc": 8, "ntp": 9}

   flag_mapping = {

      "SF": 0, "S0": 1, "REJ": 2, "RSTO": 3, "RSTOS0": 4,

      "RSTR": 5, "SH": 6, "S1": 7, "S2": 8, "S3": 9 }

   # Collect user inputs for each feature

   protocol_type = st.selectbox("Protocol Type", list(protocol_mapping.keys()))service = st.selectbox("Service",
   list(service_mapping.keys())) flag = st.selectbox("Flag", list(flag_mapping.keys()))
   src_bytes = st.number_input("Source Bytes", min_value=0, max_value=int(1e7), value=0, step=1)
   dst_bytes = st.number_input("Destination Bytes", min_value=0, max_value=int(1e7), value=0, step=1)
   count = st.number_input("Count", min_value=0, max_value=255, value=0, step=1)
```

```
        same_srv_rate = st.number_input("Same Service Rate", min_value=0.0, max_value=1.0,
   value=0.0, step=0.01)


        diff_srv_rate = st.number_input("Different Service Rate", min_value=0.0, max_value=1.0, value=0.0,
 step=0.01)

        dst_host_srv_count = st.number_input("Destination Host Service Count", min_value=0, max_value=255,
 value=0, step=1)

        dst_host_same_srv_rate = st.number_input("Destination Host Same Service Rate",
 min_value=0.0, max_value=1.0, value=0.0, step=0.01)

   # Convert user selections to encoded values

   protocol_type_encoded = protocol_mapping[protocol_type]

   service_encoded = service_mapping[service]

   flag_encoded = flag_mapping[flag]

#Prepare input for model prediction

   input_data = np.array([

      protocol_type_encoded, service_encoded, flag_encoded, src_bytes, dst_bytes, count,

      same_srv_rate, diff_srv_rate, dst_host_srv_count, dst_host_same_srv_rate

   ]).reshape(1, -1)

   # Scale input

   data

   scaled_data =

   scaler.transform(input_data) # Predict and

   display results

   if st.button("Predict Anomaly"):

      prediction = model.predict(scaled_data)

      if prediction[0] == 1:

      st.write("Prediction: ANOMALY IS

      NOT DETECTED") else:
         st.write("Prediction: ANOMALY IS DETECTED")
        # Send an SNS alert if an anomaly is detected
                message = f"Anomaly detected in the network traffic:\nProtocol: {protocol_type}, Service: {service}, Flag: {flag}"
        sns_client.publish( TopicArn=sns_topic
          _arn, Message=message,
          Subject="Anomaly Detection Alert" )
         st.success("Anomaly detected. Alert sent to your email!")
elif choice == "About":
  st.subheader("About")
 st.write("""
            The Network Anomaly Detection Alerting System uses machine learning to detect unusual network traffic and triggers alerts via AWS SNS. It ensures that the real-time notifications through email or SMS, providing proactive security alerts. The system integrates with AWS for seamless, automated operation.
 """)
if _name_ == "_main_":
 main()
```

5.4 SAMPLE OUTPUT

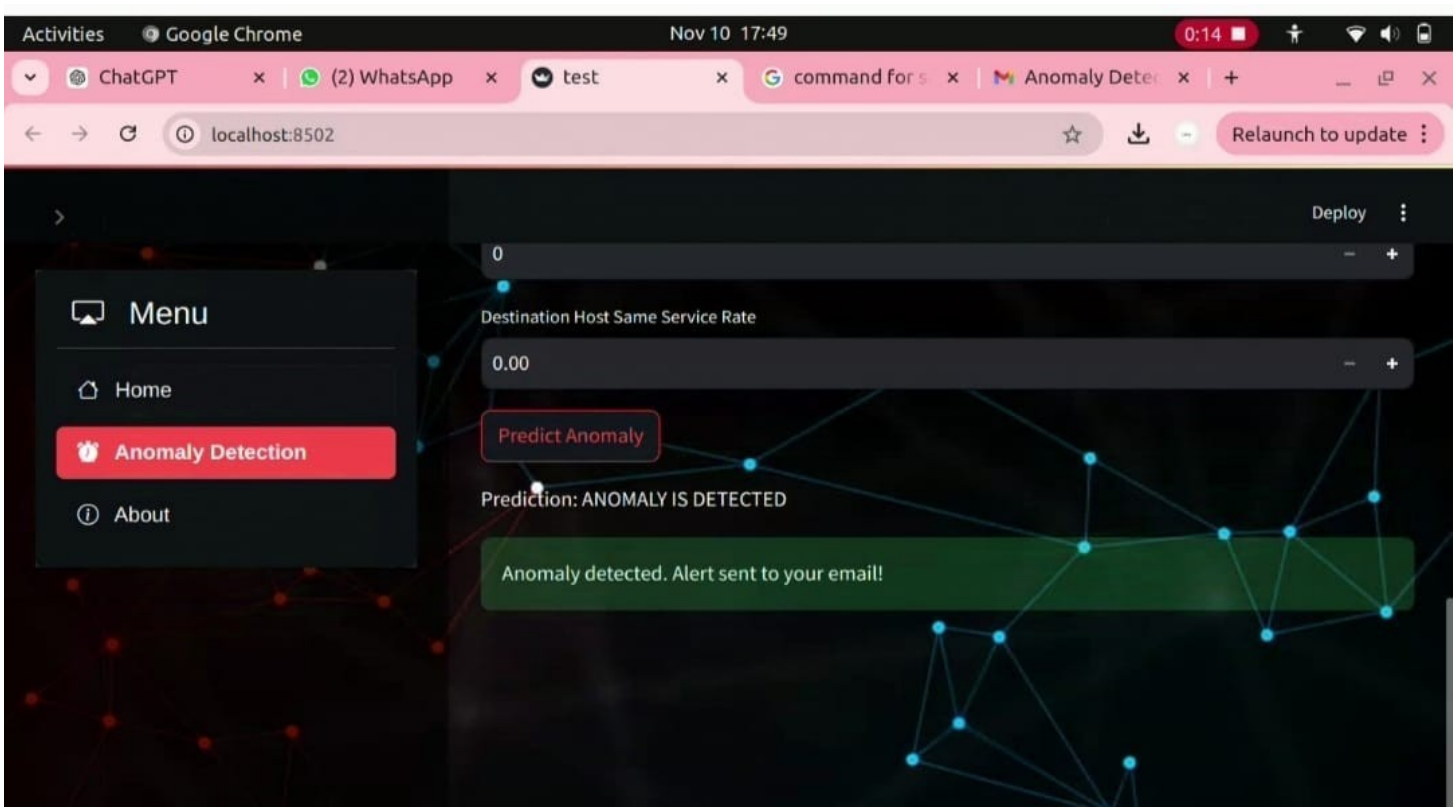
1.ANOMOLY DETECTION

INPUT IMAGE – ANOMOLY DETECTED



Destination Bytes
500
Count
2
Same Service Rate
0.80
Different Service Rate
0.20
Destination Host Service Count
25

OUTPUT

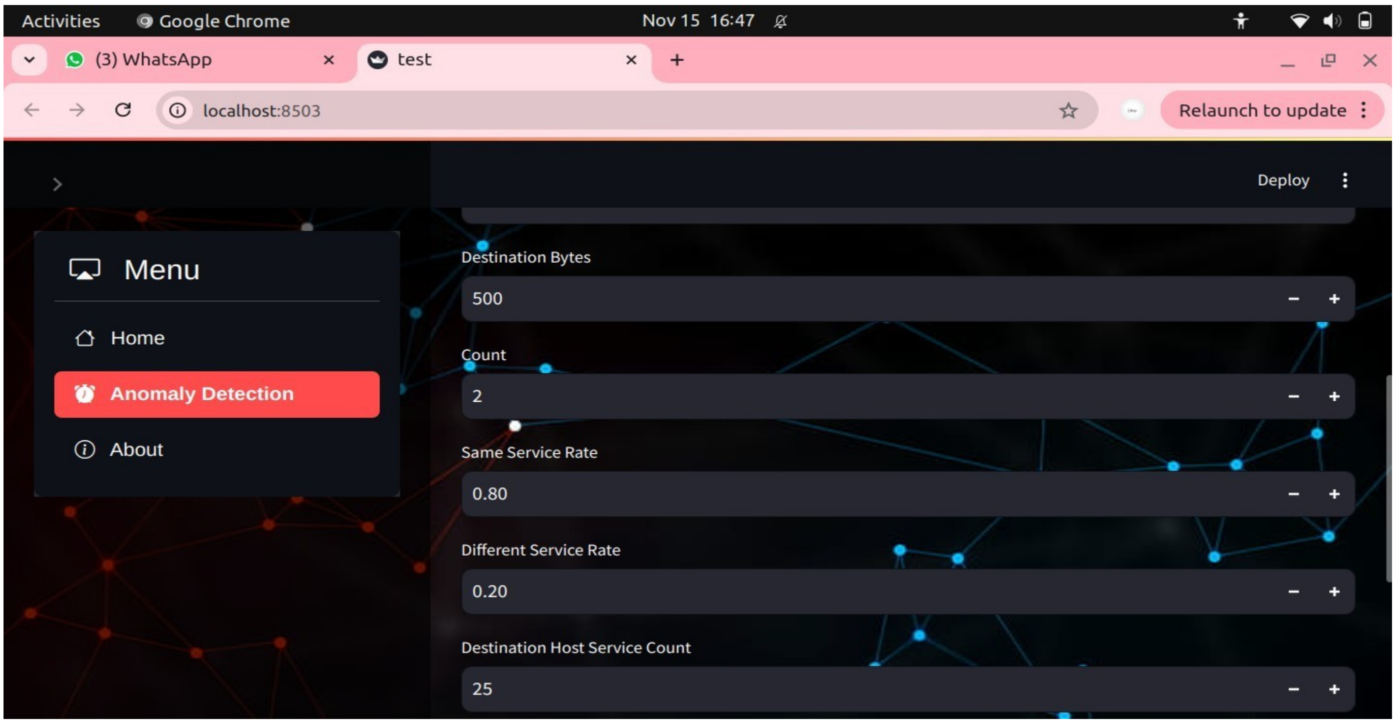


0
Destination Host Same Service Rate
0.00

Prediction: ANOMALY IS DETECTED

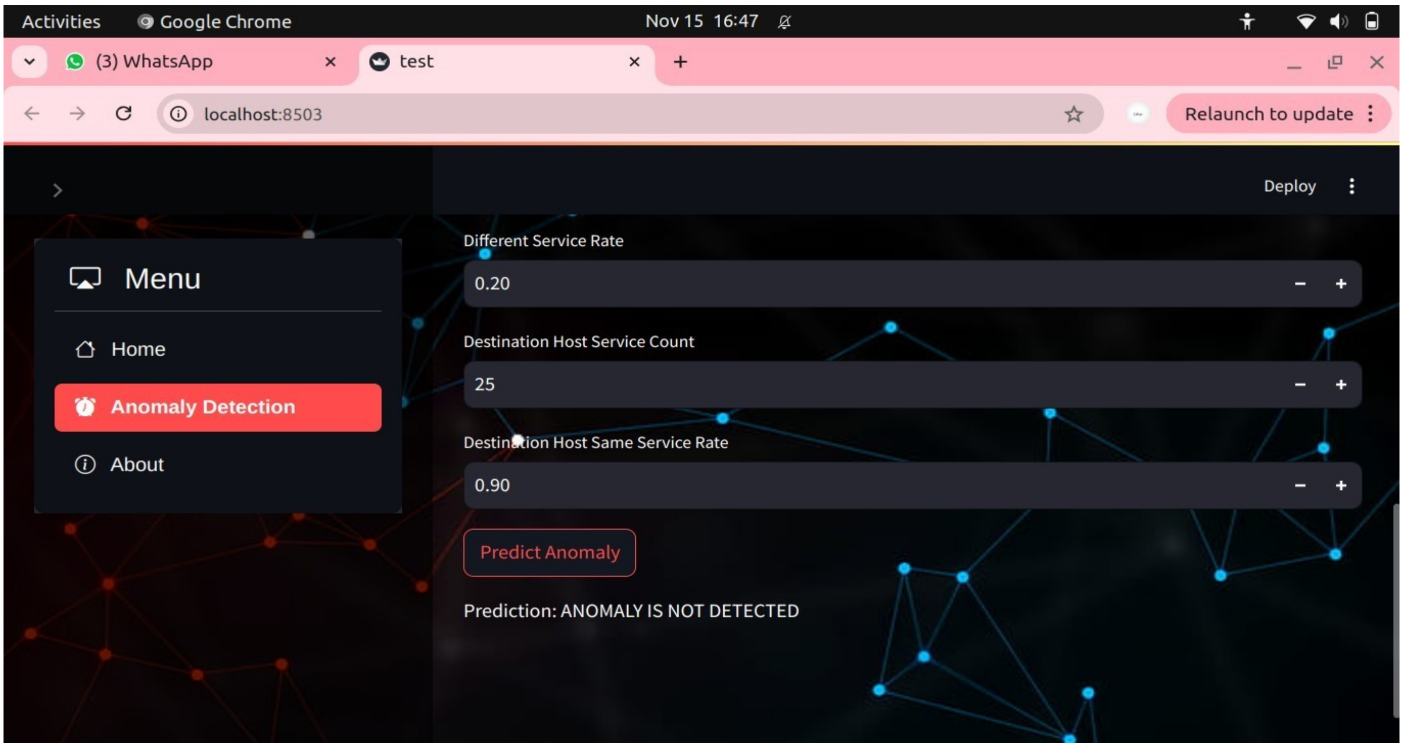
Anomaly detected. Alert sent to your email!

INPUT IMAGE – NO ANOMOLY DETECTED



Destination Bytes
500
Count
2
Same Service Rate
0.80
Different Service Rate
0.20
Destination Host Service Count
25

OUTPUT

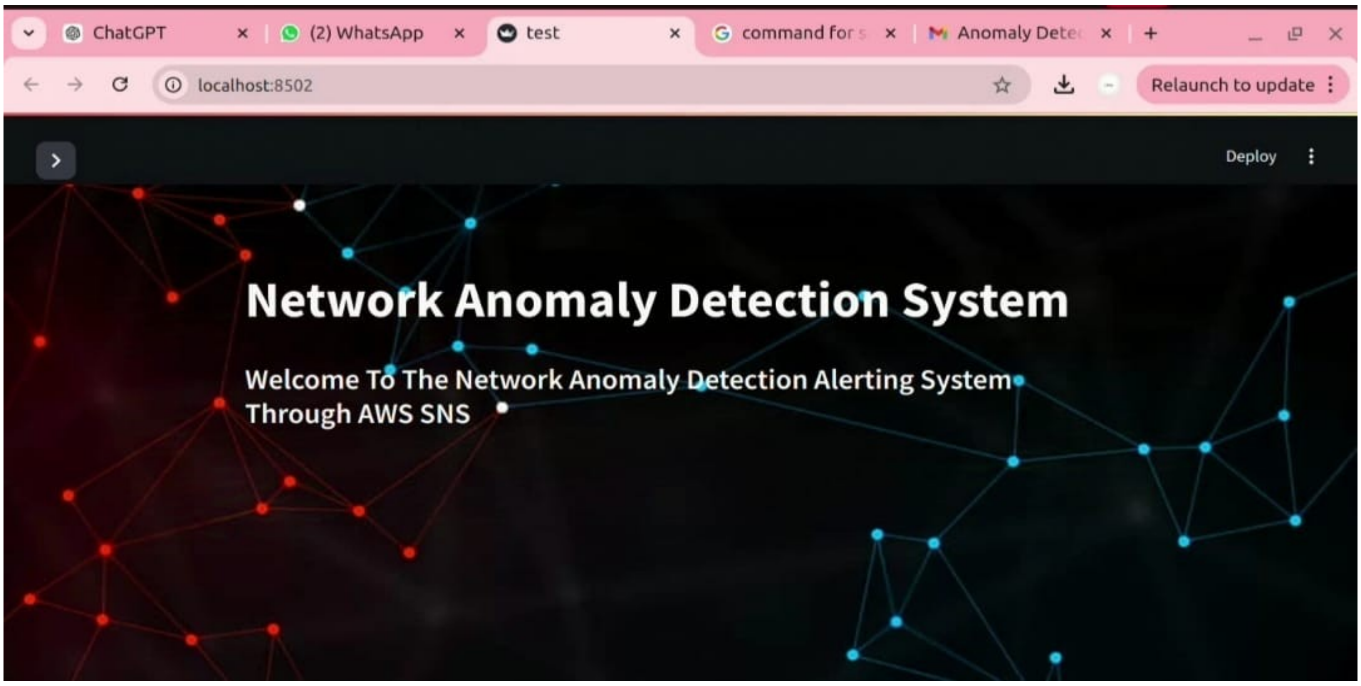


Different Service Rate
0.20
Destination Host Service Count
25
Destination Host Same Service Rate
0.90

Prediction: ANOMALY IS NOT DETECTED

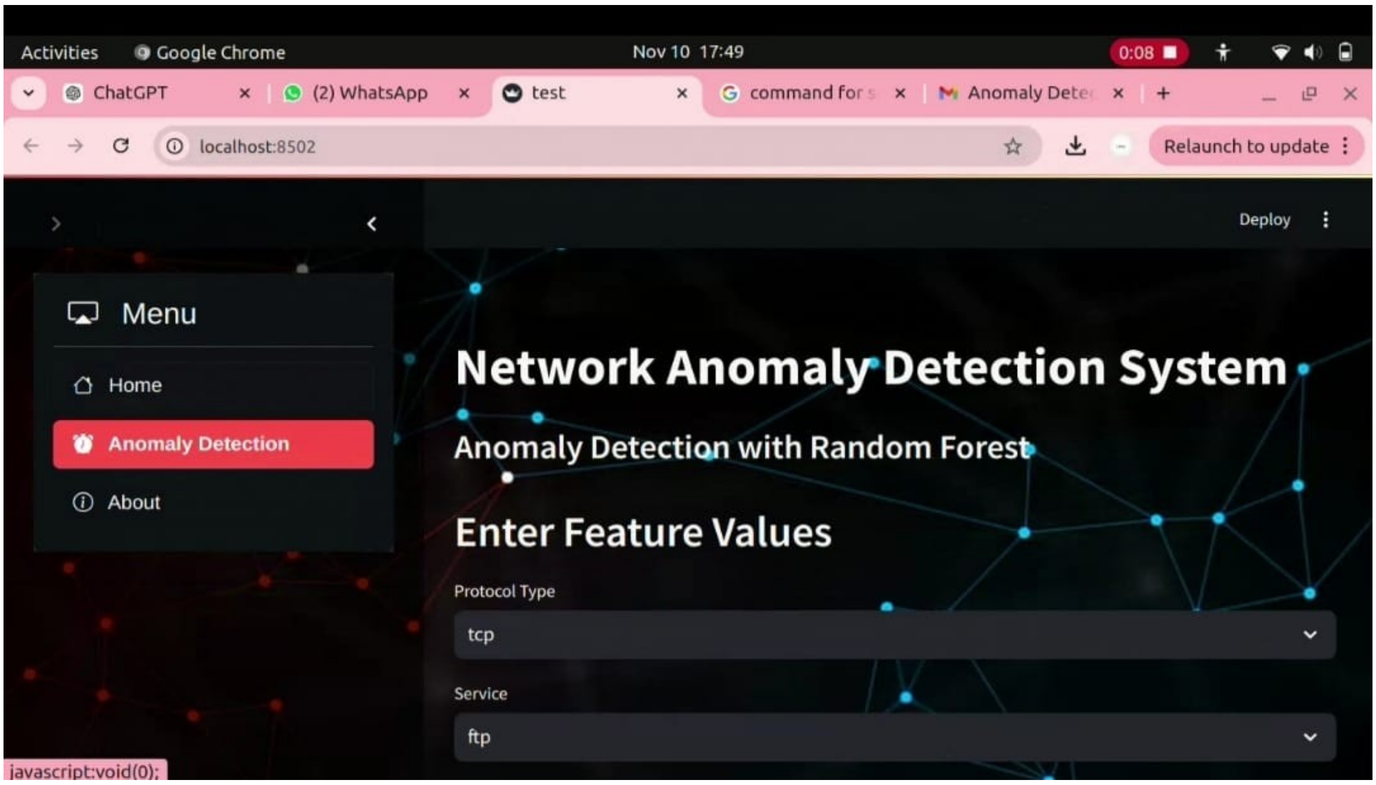
CHAPTER-6

SCREENSHOTS



Network Anomaly Detection System

Welcome To The Network Anomaly Detection Alerting System Through AWS SNS



Network Anomaly Detection System

Anomaly Detection with Random Forest

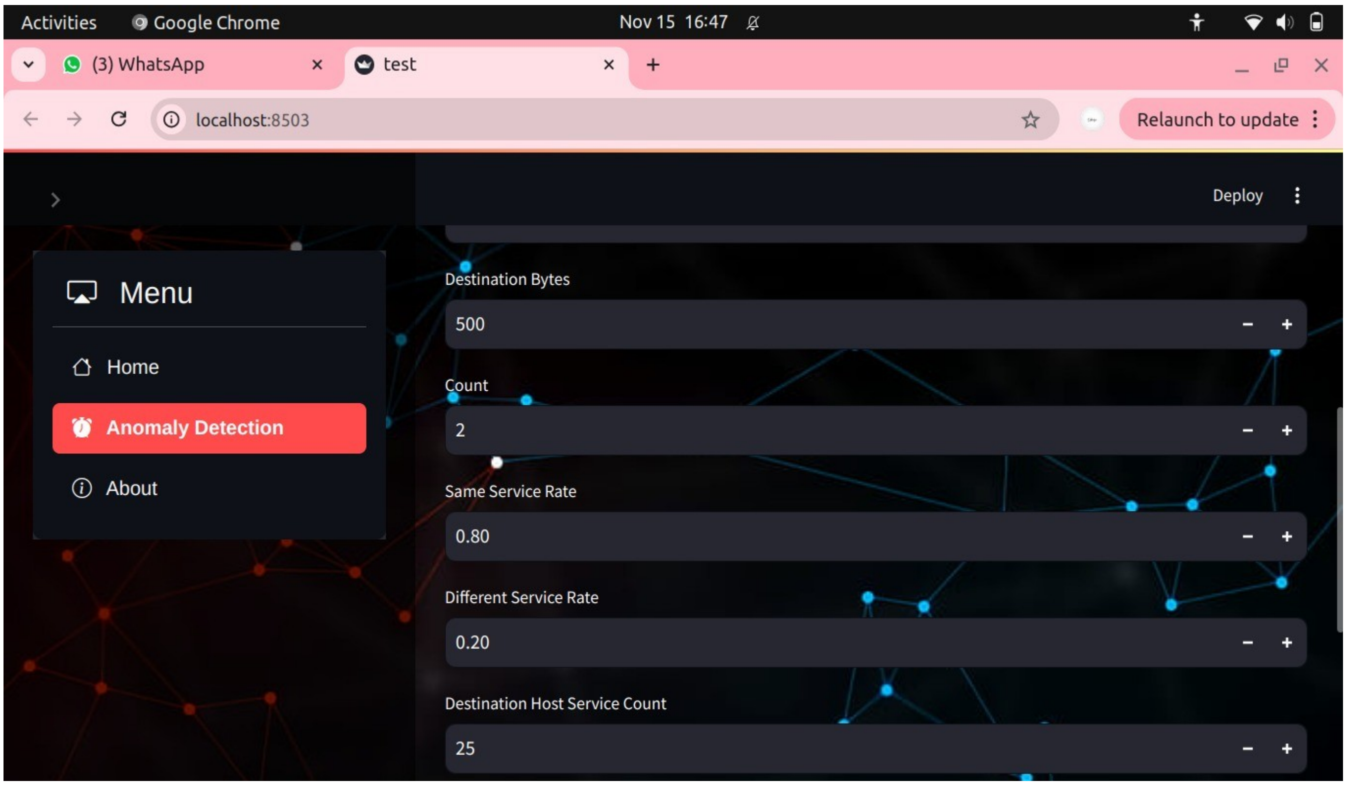
Enter Feature Values

Protocol Type

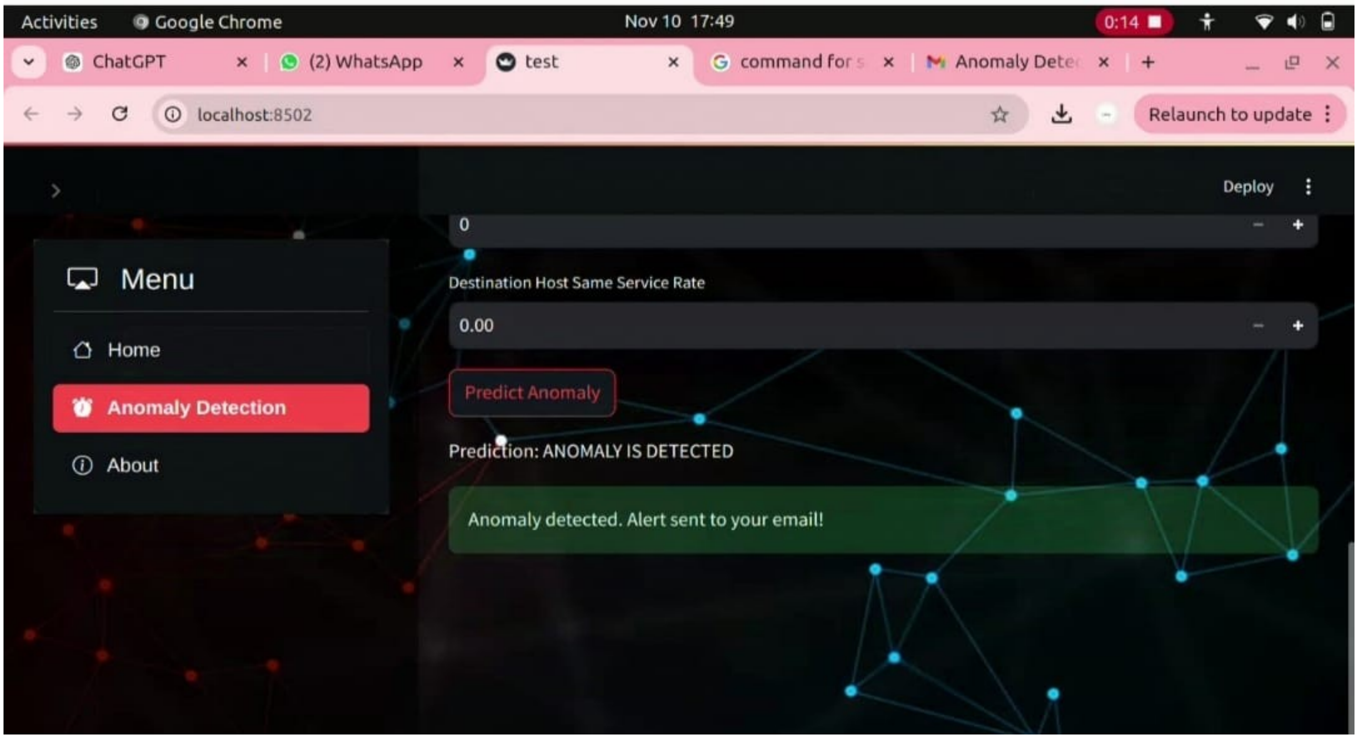
tcp

Service

ftp



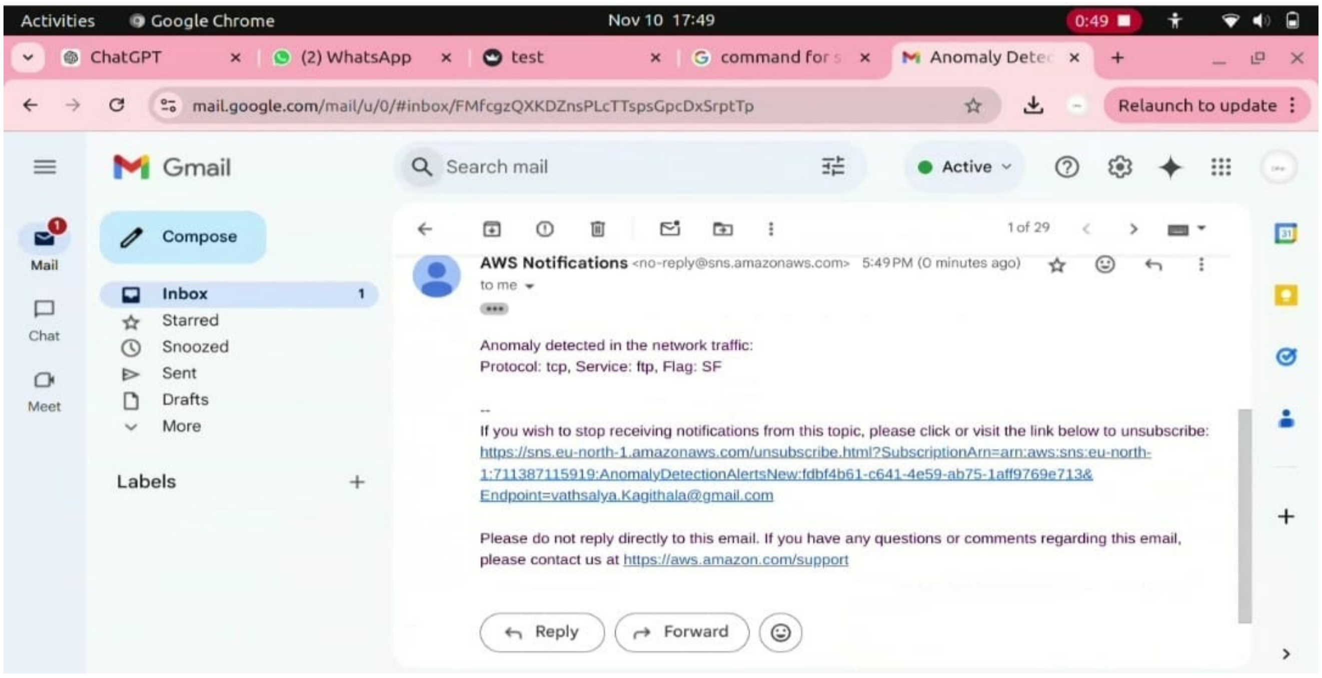
Destination Bytes
500
Count
2
Same Service Rate
0.80
Different Service Rate
0.20
Destination Host Service Count
25



0
Destination Host Same Service Rate
0.00

Prediction: ANOMALY IS DETECTED

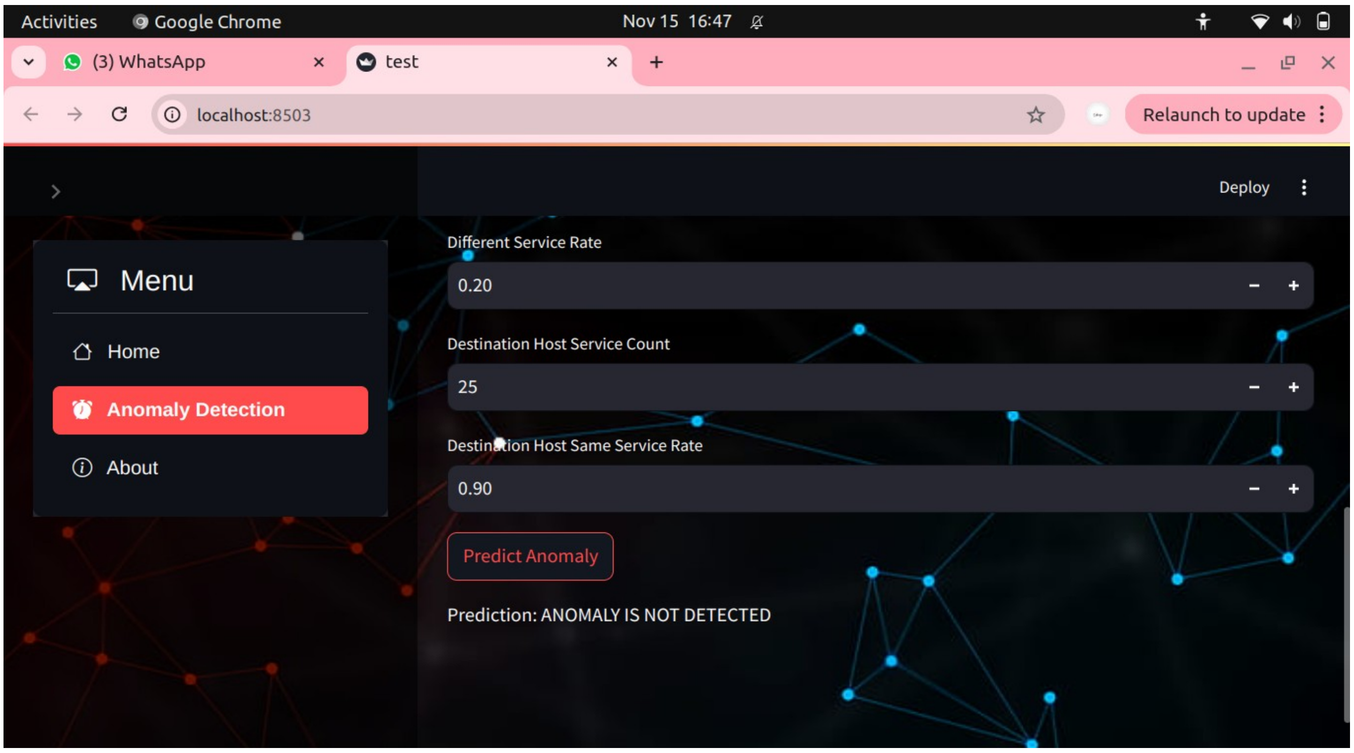
Anomaly detected. Alert sent to your email!


AWS Notifications <no-reply@sns.amazonaws.com> 5:49 PM (0 minutes ago)
to me

Anomaly detected in the network traffic:
Protocol: tcp, Service: ftp, Flag: SF

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.eu-north-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:eu-north-1:711387115919:AnomalyDetectionAlertsNew:fdbf4b61-c641-4e59-ab75-1aff9769e713&Endpoint=vathsalya.Kagithala@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support


Different Service Rate: 0.20
Destination Host Service Count: 25
Destination Host Same Service Rate: 0.90
Predict Anomaly
Prediction: ANOMALY IS NOT DETECTED

CHAPTER-7

CONCLUSION

The project on network anomaly detection using AWS Simple Notification Service (SNS) and the Random Forest algorithm successfully demonstrated the potential of integrating machine learning with cloud-based alerting systems. By analyzing network traffic data, the Random Forest algorithm effectively identified anomalies and differentiated between normal and malicious activities. Its robustness and high accuracy made it a reliable choice for this application, ensuring minimal false positives and false negatives.

The use of AWS SNS for real-time alerting showcased the scalability and reliability of cloud infrastructure in delivering timely notifications. This integration allowed for immediate communication with stakeholders, reducing response time to potential threats. The combination of machine learning and cloud services created an efficient and cost-effective system that streamlined anomaly detection and response processes.

Real-time monitoring was a significant strength of the system, as it enabled continuous surveillance of network traffic and immediate action when anomalies were detected. This capability is crucial in mitigating risks associated with cyber threats. The notification mechanism further empowered IT teams by providing actionable insights, allowing them to focus on threat mitigation rather than manually analyzing large volumes of data.

In conclusion, this project successfully established a scalable and effective system for network anomaly detection. By leveraging machine learning and cloud services, it set a strong foundation for enhancing cybersecurity frameworks and ensuring swift responses to potential threats.

CHAPTER-8

FUTURE ENHANCEMENT

1. Improve the Machine Learning Model: Fine-tune the Random Forest algorithm and consider adding advanced models like XGBoost or deep learning to make the system more accurate and efficient in detecting complex threats.

2. Automated Threat Response: Use AWS services like Lambda to automatically take actions, such as blocking suspicious IPs or isolating affected devices, when an anomaly is detected.

3. Real-Time Learning: Enable the system to learn from new data continuously so it can adapt to changing threats without needing frequent manual updates.

4. Better Monitoring and Reports: Add dashboards and tools for visualizing network activity, making it easier for teams to understand issues and trends quickly.

5. Multi-Cloud Support: Expand the system to work with other cloud platforms like Azure and Google Cloud, making it more flexible for organizations with mixed infrastructures.

REFERENCES

[1] https://ieeexplore.ieee.org/document/9003507

[2] https://ieeexplore.ieee.org/document/9033532/

[3] https://ieeexplore.ieee.org/document/9777557/

[4] https://discuss.streamlit.io/t/fav-icon-title-customization/10662

[5] https://docs.aws.amazon.com/sns/latest/dg/sns-getting-started.html

[6] https://www.kaggle.com/datasets

[7] https://www.astrj.com/pdf-149934-76147?filename=Network%20Intrusion.pdf

[8] https://www.kaggle.com/datasets

[9] https://chatgpt.com/c/68571920-0efc-4067-a129-8a701718190b

[10]https://www.javatpoint.com/machine-learning-random-forest-algorithm