

# **MACHINE LEARNING DRIVEN CYBER INCIDENT DETECTION AND RESPONSE SYSTEM**



## **A DESIGN PROJECT REPORT**

*Submitted by*

**RAHUL R**

**SAMSON JEBASEELAN C**

**SURAJ M**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**MAY 2025**



# **MACHINE LEARNING DRIVEN CYBER INCIDENT DETECTION AND RESPONSE SYSTEM**



## **A DESIGN PROJECT REPORT**

*Submitted by*

**RAHUL R (811721104081)**

**SAMSON JEBASEELAN C (811721104086)**

**SURAJ M (811721104108)**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**MAY 2025**

**K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY**  
**(AUTONOMOUS)**  
**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report titled “**Machine Learning Driven Cyber Incident Detection and Response System**” is the bonafide work of **RAHUL R (811721104081)**, **SAMSON JEBASEELAN C (811721104086)**, **SURAJ M (811721104108)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

Dr. A Delphin Carolina Rani M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

PROFESSOR

Department of CSE

K Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

**SIGNATURE**

Mrs. R. Jasmine M.E.,

**SUPERVISOR**

ASSISTANT PROFESSOR

Department of CSE

K Ramakrishnan College of Technology

(Autonomous)

Samayapuram – 621 112

Submitted for the viva-voce examination held on .....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## DECLARATION

We jointly declare that the project report on “**MACHINE LEARNING DRIVEN CYBER INCIDENT DETECTION AND RESPONSE SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF ENGINEERING**.

**Signature**

---

RAHUL R

---

SAMSON JEBASEELAN C

---

SURAJ M

Place: Samayapuram

Date:

## ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and indebtedness to our institution, “**K. Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

We extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

We would like to thank Principal, **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, who gave the opportunity to frame the project to full satisfaction.

We thank **Dr. A. DELPHIN CAROLINA RANI M.E., Ph.D.**, Head of the Department of **COMPUTER SCIENCE AND ENGINEERING**, for providing her encouragement in pursuing this project.

We wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs. R. JASMINE M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

We render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **ABSTRACT**

This project focuses on evaluating and identifying the best machine learning model for detecting cyber incidents in real time, utilizing five popular algorithms: Random Forest, XGBoost, Decision Tree, Logistic Regression, and LightGBM. The primary goal is to compare these models based on key performance metrics, including accuracy, precision, recall, and F1 score, to determine the most effective approach for detecting cyber threats. Once the best-performing model is selected, it will be deployed to analyze large datasets such as network traffic, system logs, and user behavior to identify potential security incidents. The selected model will then be integrated into an automated response framework, enabling the system to swiftly mitigate detected threats without requiring manual intervention. This approach aims to enhance the accuracy, speed, and efficiency of cyber incident detection and response, providing a more proactive defense mechanism against evolving cybersecurity threats.

# TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>v</b>
	<b>LIST OF TABLES</b>	<b>viii</b>
	<b>LIST OF FIGURES</b>	<b>ix</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>x</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	OVERVIEW	1
1.2	OBJECTIVE	1
1.3	SCOPE	2
1.4	FEATURES	2
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>3</b>
2.1	CYBERSECURITY LANDSCAPE AND THREATS	3
2.2	TRADITIONAL INCIDENT DETECTION TECHNIQUES	3
2.3	MACHINE LEARNING IN CYBERSECURITY	4
2.4	COMPARATIVE STUDY OF EXISTING SYSTEMS	4
2.5	DEEP LEARNING FOR THREAD DETECTION	5
2.6	BEHAVIOUR BASED THREAD DETECTION	5
2.7	ANOMALY DETECTION IN NETWORK TRAFFIC	6
2.8	HYBRID INTELLIGENCE SYSTEM	6
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	<b>7</b>
3.1	OVERVIEW	7
3.2	REQUIREMENT SPECIFICATION	9
3.3	USECASE DIAGRAMS	10
3.3.1	USECASE SPECIFICATION	10
3.3.2	STRUCTURAL DESIGN	11
3.3.3	SYSTEM DESGIN ARCHITECTURE	11

<b>4</b>	<b>BACKEND DEVELOPMENT</b>	<b>12</b>
4.1	DATA COLLECTION & PREPROCESSING	12
4.2	MODEL TRAINING AND COMPARISON	12
4.3	PERFORMANCE EVALUATION	13
4.4	BEST MODEL SELECTION & INTEGRATION	13
<b>5</b>	<b>FRONTEND DEVELOPMENT</b>	<b>14</b>
5.1	DASHBOARD DESIGN	14
5.2	VISUALIZATION OF INCIDENTS AND RESPONSES	14
5.3	STREAMLIT FUNCTIONALITY	15
5.4	HTML & CSS INTERFACE	15
<b>6</b>	<b>TESTING AND DEPLOYMENT</b>	<b>16</b>
6.1	MODEL TESTING & VALIDATION	16
<b>7</b>	<b>FUTURE ENHANCEMENT</b>	<b>17</b>
7.1	FUTURE ENHANCEMENT	17
<b>8</b>	<b>CONCLUSION</b>	<b>18</b>
8.1	CONCLUSION	18

**APPENDICES I**

**APPENDICES II**

**REFERENCES**



## **LIST OF TABLES**

<b>TABLE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
3.1	REQUIREMENT SPECIFICATION	09
3.2	USECASE SPECIFICATION	10

## **LIST OF FIGURES**

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
3.1	STRUCTURAL DESIGN	11
3.2	SYSTEM DESIGN ARCHITECTURE	11
A2.1	STREAMLIT INTERFACE	37
A2.2	DATA ENTRY IN INCIDENT DETECTION	37
A2.3	PREDICTING THE INCIDENT THROUGH GIVEN DATASET	38

## **LIST OF ABBREVIATIONS**

### **ABBREVIATIONS**

**SIEM** — Security Information and Event Management

**IDS** — Intrusion Detection Systems

**PCA** — Principal Component Analysis

**KPI** — Key Performance Indicators

**RNN** — Recurrent Neural Networks

**SVM** — Support Vector Machine

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 OVERVIEW**

The overview of this project highlights a machine learning-based system designed for cyber incident detection and automated response. As modern threats become more complex and stealthy, traditional rule-based systems are insufficient. This project introduces a robust and intelligent solution that monitors user activity, network traffic, and system logs in real time to detect anomalies. Using machine learning algorithms like Random Forest, XGBoost, Decision Tree, Logistic Regression, and LightGBM, the system evaluates patterns and behaviors to identify cyber threats. Python plays a central role in implementing and managing the system, with libraries like Scikit-learn and Pandas enabling model training and data handling. Once trained, the best-performing model is deployed to process real-time data, allowing instant identification of potential threats. Integration with automated response mechanisms allows for prompt mitigation, reducing response time and human error. The system improves cyber resilience by enabling organizations to proactively detect and address attacks as they occur.

### **1.2 OBJECTIVE**

The primary objective of this project is to build an intelligent cybersecurity system that detects incidents using machine learning and responds automatically. It compares five models—Random Forest, XGBoost, Decision Tree, Logistic Regression, and LightGBM—to identify the most accurate and efficient one based on performance metrics like accuracy, precision, recall, and F1-score. The final selected model is trained on cybersecurity datasets and integrated into a real-time monitoring framework. This helps reduce detection delays, prevent data breaches, and minimize manual effort, providing a faster, data-driven, and more secure approach to incident handling.

### **1.3 SCOPE**

The scope of this project includes developing a complete end-to-end system for cyber incident detection and response using machine learning. It begins with the collection and preprocessing of cybersecurity data from sources like network logs and system events. The system then evaluates multiple algorithms and deploys the best-performing one for real-time detection. It is designed for use in varied domains such as enterprise IT, cloud systems, and critical infrastructure sectors. The modular structure allows future scalability, making it adaptable to growing datasets, evolving threats, and newer machine learning advancements in cybersecurity.

### **1.4 FEATURES**

The system includes features like advanced data preprocessing, where raw cybersecurity logs and network traffic are cleaned, filtered, and structured for model input. Multiple models are trained and evaluated to determine which offers the best threat detection capability. The Flask framework enables the backend to handle real-time data flow and integrate with the automated response system. Automated threat mitigation actions are triggered without manual input, reducing response latency. Visualization of model metrics and detected events is achieved using Python tools for better insight. The modular design supports dynamic model switching, making the system adaptable and upgrade-friendly. These features collectively create a responsive, scalable, and intelligent system that enhances security operations.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 TITLE: CYBERSECURITY LANDSCAPE AND THREATS**

**AUTHOR:** Yichen Li, Xu Zhang, Shilin He, et al.

**YEAR:** 2024

The cybersecurity landscape has evolved significantly with the increasing sophistication of cyber threats. Traditional security measures are no longer sufficient to address the complexity and speed of modern cyber-attacks. Yichen Li et al. introduce a framework leveraging AI and machine learning (ML) for cloud incident detection, aiming to improve detection timeliness and accuracy. The rapid growth of cloud technologies and digital infrastructures has led to an increase in cyber threats, making proactive threat detection critical. With the introduction of automated incident detection through AI, security systems can now process vast amounts of data in real time, identifying potential security breaches before they escalate into significant issues.

#### **2.2 TITLE: TRADITIONAL INCIDENT DETECTION TECHNIQUES**

**AUTHOR:** Jie Chen, Yu Gan, Wei Xu

**YEAR:** 2023

Traditional incident detection techniques primarily rely on predefined rules and signature-based systems to identify threats, such as intrusion detection systems (IDS) and security information and event management (SIEM) platforms. Jie Chen and colleagues discuss the Cloud Incident framework, which utilizes operational data from cloud environments, such as logs and metrics, to detect incidents. While traditional methods have been effective in detecting known threats, they often struggle with emerging and sophisticated attacks that do not match predefined signatures.

### **2.3 TITLE: MACHINE LEARNING IN CYBERSECURITY**

**AUTHOR: Zhichao Cao, et al.**

**YEAR:2023**

Machine learning has become a critical component in the fight against cyber threats, providing a means to analyze large datasets and detect anomalies that traditional methods may overlook. Zhichao Cao and team provide an extensive survey on the role of machine learning in system log analysis for cybersecurity. Their review highlights various machine learning techniques, such as supervised and unsupervised learning, that enable more accurate threat detection by analyzing system logs for unusual patterns or behaviors. Machine learning models, particularly those using deep learning and clustering, can automatically identify new, previously unknown types of attacks.

### **2.4 TITLE: COMPARATIVE STUDY OF EXISTING SYSTEMS**

**AUTHOR: Dongjin Song, Xuan Zhang, et al.**

**YEAR:2022**

A comparative study of existing systems for cyber incident detection reveals a wide range of approaches, from rule-based systems to machine learning-driven frameworks. Dongjin Song and colleagues focus on improving anomaly detection in logs, specifically addressing the challenges posed by unstable and changing log data. Their work compares various anomaly detection techniques, emphasizing the importance of adaptability in the face of fluctuating data formats and structures. While many existing systems rely on static, predefined rules or require consistent data structures, machine learning models offer flexibility by automatically adjusting to new log patterns. However, these models also introduce complexity, making them harder to implement and interpret.

## **2.5 TITLE: DEEP LEARNING FOR THREAD DETECTION**

**AUTHOR: Mohit Verma, Li Wang, et al.**

**YEAR: 2022**

Mohit Verma and his team present an AI-based framework for real-time log analysis that supports proactive threat hunting in enterprise systems. Their approach leverages Natural Language Processing (NLP) to preprocess and normalize unstructured logs before feeding them into a deep learning classification model. The framework has shown high accuracy in detecting lateral movement and privilege escalation tactics used by advanced persistent threats (APTs). This study underscores the role of automation and AI in reducing the time to detection and response in complex network environments.

## **2.6 TITLE: BEHAVIOUR BASED THREAD DETECTION**

**AUTHOR: Y Priya Natarajan, Dev Mehta**

**YEAR: 2022**

Zero-day attacks represent a significant challenge in cybersecurity due to their novelty and absence from signature databases. Priya Natarajan and Dev Mehta survey contemporary strategies for zero-day detection, emphasizing heuristic analysis, sandboxing, and AI-based predictive models. Their research compares performance metrics across several datasets and tools, revealing that ensemble learning models, particularly random forests combined with anomaly scoring, yield the highest detection rates for zero-day exploits. The paper advocates for integrating multiple detection approaches for layered security. Their approach leverages Natural Language Processing (NLP) to preprocess and normalize unstructured logs before feeding them into a deep learning classification model.



## **2.7 TITLE: ANOMALY DETECTION IN NETWORK TRAFFIC**

**AUTHOR: Ravi Sharma, Elena Petrova.**

**YEAR: 2020**

Ravi Sharma and Elena Petrova explore the use of unsupervised machine learning models to detect anomalies in high-volume network traffic. Their study implements clustering algorithms like DBSCAN and Isolation Forests to identify abnormal patterns without relying on labeled training data. The approach proves effective in uncovering previously unseen threats, including stealthy malware and slow-moving data exfiltration attempts. This paper highlights the scalability of unsupervised models in dynamic environments and stresses the importance of real-time adaptability for modern cybersecurity systems.

## **2.8 TITLE: HYBRID INTELLIGENT SYSTEM**

**AUTHOR: Neha Kulkarni, Harish Kumar**

**YEAR: 2019**

Neha Kulkarni and Harish Kumar propose a hybrid intelligent framework that combines rule-based reasoning with deep learning techniques for improved cyber incident detection. The system uses a rule engine for filtering known threats and a Convolutional Neural Network (CNN) to analyze complex patterns in system logs and network packets. Experimental evaluation on benchmark datasets shows enhanced detection accuracy and reduced false alarm rates. The study advocates for hybrid architectures as a bridge between deterministic logic and adaptive learning, offering robust defense mechanisms against sophisticated cyberattacks.

## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**

#### **3.1 OVERVIEW**

The field of cybersecurity is rapidly evolving, driven by an increase in cyber threats and the complexity of digital infrastructures. Traditional methods of threat detection, often rule-based or signature-based, struggle to keep up with the dynamic and sophisticated nature of modern cyber-attacks. As a result, organizations are increasingly adopting machine learning (ML) techniques to improve the detection of security incidents and enhance their ability to respond in real time. This project seeks to develop a robust, adaptive system for cybersecurity incident detection by leveraging the power of machine learning algorithms such as Random Forest, XGBoost, Decision Tree, Logistic Regression, and LightGBM.

The use of big data and machine learning allows for the analysis of vast datasets, including system logs, network traffic, and user behavior, to uncover hidden patterns that may indicate potential threats. Tools like Python and libraries such as Scikit-learn, Pandas, and Matplotlib enable data preprocessing, feature engineering, and model training. Techniques like Principal Component Analysis (PCA) can be employed to reduce the dimensionality of complex datasets, ensuring that the most relevant features are used in the threat detection process.

The adoption of machine learning in cybersecurity presents several key advantages. It enables the creation of adaptive systems that can continuously learn and improve, staying ahead of evolving threats. Furthermore, by automating the detection process, businesses can reduce response times, mitigate risks more effectively, and allocate resources more efficiently.

In today's digital era, the growing dependency on interconnected systems and cloud-based infrastructures has made organizations increasingly vulnerable to a broad range of cyber threats. Attackers now use advanced and stealthy techniques, such as polymorphic malware, lateral movement, and privilege escalation, which often bypass conventional defense mechanisms. This rapid evolution in attack sophistication necessitates a shift from static, rule-based detection systems to more intelligent and dynamic approaches. Machine learning (ML) has emerged as a promising solution, offering the capability to analyze complex data, learn from patterns, and identify anomalies indicative of malicious activities.

The primary objective of this project is to develop a machine learning-based framework that can detect and respond to cyber incidents more effectively and with greater accuracy than traditional systems. By leveraging both historical and real-time data, the system aims to provide timely alerts and automate responses to potential threats. This is especially valuable in large-scale environments where manual monitoring is impractical due to the volume and velocity of data. The system will integrate various ML algorithms such as Random Forest, Decision Tree, Logistic Regression, LightGBM, and XGBoost to classify and predict malicious behavior. Each algorithm brings unique strengths to the detection process, with ensemble methods offering improved performance through model combination.

Furthermore, the system utilizes Python due to its extensive support for machine learning and data science through libraries like Scikit-learn, NumPy, and Pandas. Data visualization tools such as Matplotlib and Seaborn will assist in interpreting the results and understanding the detection process. Dimensionality reduction techniques like PCA (Principal Component Analysis) will play a key role in optimizing model performance by eliminating irrelevant or redundant features from high-dimensional datasets.

### 3.2 REQUIREMENT SPECIFICATION

ID	NAME	FUNCTIONAL	NON-FUNCTIONAL	DESCRIPTION	PRIORITY	ACTOR
1	User registration	Yes		Functionality for user to create account	High	User
2	User login	Yes		Allow users to securely log into the system	High	User
3	Incident reporting	Yes		Allow users or systems to report suspected security incidents	High	User, System
4	Automated threat detection	Yes		System automatically detects threats based on predefined rules	High	System
5	Email alert notifications	Yes		Send email alerts when a potential incident is detected	High	System
6	Real-time monitoring	Yes		Continuously monitor network traffic and system logs	High	System
7	User activity logging	Yes		Log all user activities for auditing and forensics	Medium	System
8	Incident prioritization	Yes		Classify and prioritize incidents based on severity	High	System, Analyst
9	Performance		Yes	Application performance must be better	High	Admin
10	Usability		Yes	Easy for newbies	High	Admin
11	Reliability		Yes	Trusted by users	High	Admin

Table 3.1 Requirement Specification

### 3.3 USECASE DIAGRAMS

A use case diagram for the machine learning-driven cyber incident detection and response system illustrates the interactions between the system and its users, focusing on the key functionalities. The primary actors include the Security Analyst, System Administrator, Incident Detection System, and Data Sources such as Network Traffic, System Logs, and User Behavior. Important use cases include Importing Data, which involves ingesting data from various sources for analysis, and Preprocessing Data, which addresses cleaning and normalizing the raw data. The Train Machine Learning Model use case applies algorithms like Random Forest, XGBoost, and LightGBM.

#### 3.3.1 USECASE SPECIFICATION

Use Case ID	Use Case Description	Actors	Trigger	Preconditions	Postconditions
UC- 01	Classify incoming network traffic	Security Analyst	New traffic log is processed.	Feature extraction and preprocessing are completed.	Traffic is labeled and flagged if necessary
UC- 02	Generate incident severity scores	Detection System	Suspicious activity is detected.	Models are calibrated to output severity levels.	Incidents for prioritization
UC- 03	Retrain models with newly labeled incident data	Data Engineer	A significant number of new incidents are	Incident labels and data are collected.	Updated models with improved detection accuracy.
UC- 04	Send real-	Security Team.	An incident is classified above a risk threshold.	Alerting system is configured.	Alerts are delivered to the appropriate response team.

Table 3.2 Usecase Specification

### 3.3.2 STRUCTURAL DESIGN

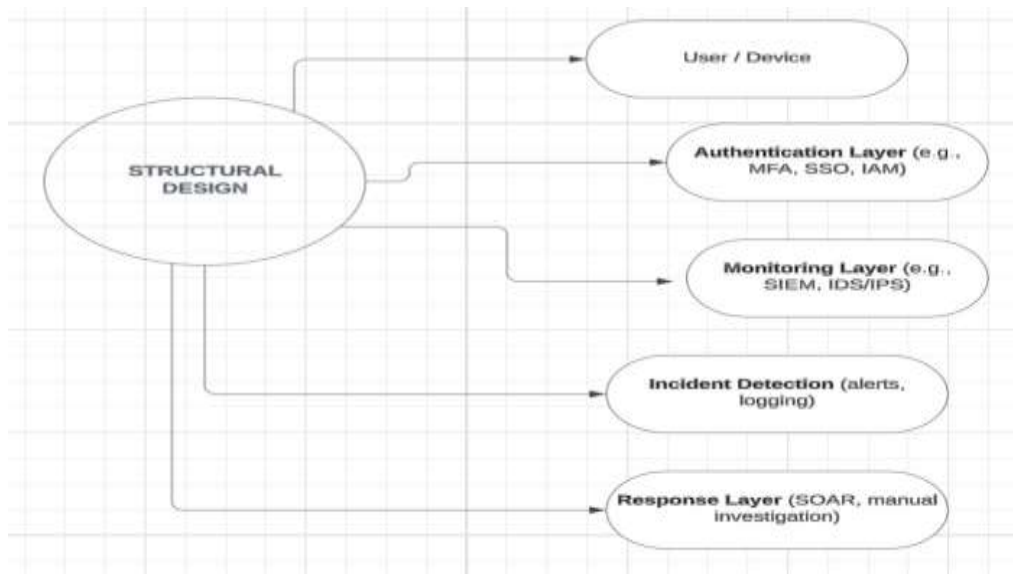


Fig. 3.1 Structural Design

### 3.3.3 SYSTEM DESIGN ARCHITECTURE

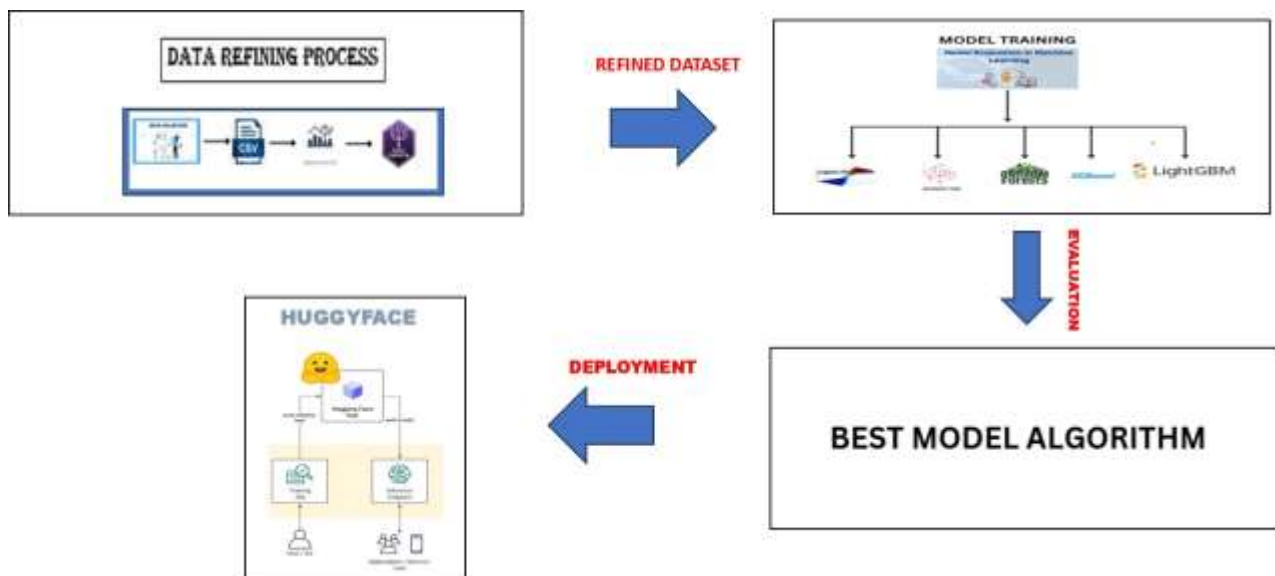


Fig. 3.2 System design architecture

## **CHAPTER 4**

### **BACKEND DEVELOPMENT**

#### **4.1 DATA COLLECTION & PREPROCESSING**

Data collection and preprocessing in cyber incident detection involve gathering and preparing data from diverse sources such as system logs, network traffic, and user behavior. The first step is data collection, where relevant data from various logs and monitoring systems is gathered. Once the data is collected, preprocessing begins, including cleaning, where missing or corrupted data is identified and removed. This is followed by data transformation, including normalization or feature scaling to ensure consistency. Feature engineering is then performed to create relevant attributes such as network packet counts or login frequency, which are critical in identifying potential incidents. The preprocessed data is now ready for model training, enabling accurate detection of cyber threats based on historical and real-time data.

#### **4.2 MODEL TRAINING AND COMPARISON**

Model training and comparison in cyber incident detection involve selecting multiple Machine learning algorithms, training them on preprocessed data, and evaluating their performance. Popular algorithms such as Random Forest, XGBoost, and LightGBM are applied to the prepared dataset to detect patterns indicative of cyber threats. During training, the model learns to differentiate between normal and anomalous behavior. After training, various models are compared based on performance metrics like accuracy, precision, recall, and F1 score to determine which algorithm provides the most reliable threat detection. The comparison process helps identify the strengths and weaknesses of each model and ensures that the best-performing algorithm is selected for further analysis.

### **4.3 PERFORMANCE EVALUATION**

Performance evaluation in cyber incident detection involves assessing how well trained models detect and classify cyber incidents. Metrics such as accuracy, precision, recall, and F1 score are used to measure the model's ability to correctly identify incidents and minimize false positives or negatives. Cross-validation is also employed to assess the model's performance across different subsets of the data, ensuring it generalizes well to unseen data. The evaluation process helps to fine-tune the model by identifying areas of improvement, ensuring that the model can reliably detect security threats in real-time scenarios and provide actionable insights.

### **4.4 BEST MODEL SELECTION & INTEGRATION**

Best model selection and integration involve choosing the optimal machine learning model for real-time cyber incident detection and integrating it into a system for continuous monitoring. After evaluating the performance of various models, the one with the highest performance metrics is selected. This model is then fine-tuned, adjusting hyperparameters to further enhance its accuracy and reliability. Once the best model is identified, it is integrated into a detection system that continuously analyzes incoming data, such as logs and network traffic, to identify potential threats. The integrated model works in real-time, enabling automated responses to detected incidents, ensuring swift mitigation of cybersecurity threats.



## **CHAPTER 5**

### **FRONTEND DEVELOPMENT**

#### **5.1 DASHBOARD DESIGN**

The dashboard in a Machine Learning Driven Cyber Incident Detection and Response System serves as the central interface for visualizing and managing real-time security alerts and system behavior. It is designed to provide a user-friendly overview of key performance indicators (KPIs), including the number of detected anomalies, attack types, system status, and the response actions taken. A well-organized dashboard allows cybersecurity analysts to easily monitor the status of the network, assess threat levels, and make quick decisions. The layout typically includes widgets or panels for displaying real-time metrics, graphs showing historical trends of incidents, a log panel for event tracking, and an incident response timeline. With proper use of design principles such as color-coding for threat severity and interactive filters, the dashboard transforms complex ML output into actionable insights.

#### **5.2 VISUALIZATION OF INCIDENTS AND RESPONSES**

Visualization plays a key role in making complex machine learning outcomes more interpretable for security professionals. In this system, incidents detected by the ML model are visually represented using charts, graphs, and heat maps to show attack vectors, frequencies, time of occurrence, and affected nodes. Tools like Chart.js, D3.js, or Plotly can be used to render interactive visualizations that update dynamically as new data is processed. These visualizations help in identifying attack patterns, geographical attack sources, and behavior anomalies in a concise manner. Furthermore, response actions such as IP blocking, alert generation, or firewall updates can also be visualized to provide transparency and traceability. The ability to drill down into specific incidents allows analysts to investigate the context and take appropriate countermeasures.

### **5.3 STREAMLIT FUNCTIONALITY**

Streamlit is an excellent choice for the front end of an incident detection project because it allows you to quickly build an interactive and user-friendly web interface directly in Python without needing deep web development skills. With Streamlit, you can easily upload logs, display authentication data, visualize incident detection results, and monitor real-time alerts, all through simple Python scripts. Its ability to create dynamic charts, tables, and dashboards makes it ideal for cybersecurity use cases, enabling security analysts to interact with machine learning models, view incident predictions, and take necessary actions within a clean and responsive layout.

### **5.4 HTML & CSS INTERFACE**

HTML and CSS form the foundational structure and design of the system's user interface. HTML provides the semantic layout for all components on the dashboard such as navigation bars, content containers, tables, buttons, and input forms. Each section of the dashboard is structured using HTML tags to maintain clarity and accessibility. CSS complements this by styling these elements to ensure a professional and visually consistent design. It defines the color schemes (e.g., red for high-severity incidents), font styles, spacing, and responsive layouts to support usage on desktops, tablets, and mobile devices. Media queries are employed to make the design adaptable to different screen sizes, while CSS frameworks like Bootstrap or Tailwind CSS can be used to accelerate UI development. Together, HTML and CSS ensure that the system interface is not only functional but also visually engaging and intuitive to use.

## CHAPTER 6

### TESTING AND DEPLOYMENT

#### 6.1 MODEL TESTING & VALIDATION

Model testing and validation are crucial steps in developing a reliable machine learning-driven system for cyber incident detection and response. This process begins with the proper partitioning of the dataset into training, validation, and testing sets. Popular datasets such as NSL-KDD, CICIDS2017, and UNSW-NB15 are used, which simulate real-world network traffic and various cyber-attack scenarios. During training, the model learns to differentiate between normal and malicious behavior patterns based on features like protocol types, source/destination IPs, payload sizes, and connection durations. Techniques such as k-fold cross-validation are employed to assess the model's generalizability and avoid overfitting, which ensures that the model performs well not only on the training data but also on unseen data.

To evaluate the model's effectiveness, several performance metrics are used, including **Accuracy**, **Precision**, **Recall**, **F1-Score**, and **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)**. Precision and recall are particularly important in cyber incident detection systems, as false positives can lead to unnecessary alert fatigue, while false negatives can allow real threats to bypass detection. Models such as Random Forest, Support Vector Machines (SVM), Decision Trees, and Neural Networks are tested and compared. In some cases, anomaly detection models like Isolation Forest and Autoencoders are also validated to catch zero-day attacks or rare behavior patterns. Hyperparameter tuning is performed using grid search or random search techniques to fine-tune each model and achieve the best balance between detection rate and computational efficiency.

## **CHAPTER 7**

### **FUTURE ENHANCEMENT**

#### **7.1 FUTURE ENHANCEMENT**

The future enhancement of a Machine Learning Driven Cyber Incident Detection and Response System lies in expanding its capabilities beyond basic anomaly detection into more intelligent, adaptive, and real-time threat mitigation. A major enhancement could involve incorporating deep learning techniques, such as Recurrent Neural Networks (RNNs) and Transformer models, to analyze network traffic patterns over time. These models can detect subtle and evolving attack patterns that traditional rule-based or shallow machine learning models might miss. Additionally, integrating behavioral analytics into the system can allow for profiling user and device behavior, improving the accuracy of insider threat detection and reducing false positives.

Another promising future enhancement involves building a real-time automated response framework using reinforcement learning. Such a system could learn the most effective incident response actions over time by simulating various attack scenarios and outcomes. This would enable the system to proactively respond to threats—such as isolating compromised nodes or blocking malicious IPs—without manual intervention. Integration with Security Information and Event Management (SIEM) systems and threat intelligence feeds can also allow for context-aware decision-making, where the system adapts its response based on attack severity, past incidents, and global threat data. These enhancements aim to evolve the system into a self-defending cybersecurity solution, minimizing damage, reducing response time, and ensuring robust protection in modern digital environments.

## **CHAPTER 8**

### **CONCLUSION**

#### **8.1 CONCLUSION**

In this study, we evaluated five machine learning models — LightGBM, Decision Tree, Random Forest, XGBoost, and Linear Regression — for the task of cybersecurity incident detection. Each model demonstrated different capabilities in recognizing threats, with ensemble methods like Random Forest and boosting algorithms like XGBoost and LightGBM showing particularly strong performance. This comparison revealed that machine learning offers significant potential for improving the speed and accuracy of incident detection in cybersecurity systems.

Our findings suggest that ensemble and boosting models outperform simpler models like Decision Tree and Linear Regression, particularly in handling complex patterns within the data. LightGBM and XGBoost, in particular, provided higher detection accuracy and better generalization, while Random Forest proved valuable for reducing overfitting. However, even simpler models like Decision Tree and Linear Regression played an important role in offering interpretability and fast decision-making in certain scenarios.

In conclusion, applying multiple models provides a well-rounded and resilient approach to cybersecurity incident detection. By leveraging the strengths of LightGBM, Decision Tree, Random Forest, XGBoost, and Linear Regression together, organizations can achieve more reliable detection and a quicker response to emerging threats. Future research should focus on developing hybrid systems that integrate these models and explore real-time deployment for even stronger cybersecurity defenses.

## APPENDICES I

### ➤ SAMPLE CODINGFOR INCIDENT DETECTION

#### TO TRAIN THE DATASET

```
import pandas as pd
import numpy as np
Train = pd.read_csv("/content/drive/MyDrive/new_train_sample.csv")
Test = pd.read_csv("/content/drive/MyDrive/GUIDE_Test.csv",
    low_memory=False)
print("Train")
print(Train.shape)
print("Test")
print(Test.shape)
print("Train")
print(Train.info())
print("Test")
print(Test.info())
print("Train")
print(Train.describe())
print("Test")
print(Test.describe())
print("Train")
print(Train['Category'].unique())
print("Test")
print(Test['Category'].unique())
print("Train")
print(Train.isnull().sum())
print("Test")
print(Test.isnull().sum())
print("Train")
```

```

print(Train.duplicated().sum())
print("Test")
print(Test.duplicated().sum())
print("Train")
print(Train.drop_duplicates(inplace=True))
print("Test")
print(Test.drop_duplicates(inplace=True))
print("Train")
print(Train.duplicated().sum())
print("Test")
print(Test.duplicated().sum())
print("Train")
print(Train.shape)
print("Test")
print(Test.shape)
print("Train")
print(Train.isnull().sum())
print("Test")
print(Test.isnull().sum())
missing_percentage = Train.isnull().mean() * 100
missing_percentage
columns_to_drop = missing_percentage[missing_percentage > 50].index
columns_to_drop
data_cleaned = Train.drop(columns=columns_to_drop)
data_cleaned
data_cleaned.shape
data_cleaned.isnull().sum()
data_cleaned.IncidentGrade.unique()
data_cleaned.dropna(subset=['IncidentGrade'], inplace=True)
array(['BenignPositive', 'TruePositive', 'FalsePositive'], dtype=object)

```

```

print(data_cleaned.shape)
data_cleaned
data_cleaned.nunique()
print(data_cleaned.shape)
print(data_cleaned.Id.nunique())
data_cleaned['Timestamp'] = pd.to_datetime(data_cleaned['Timestamp'])
data_cleaned['Timestamp'].head()

# Extract day, month, and hour from the Timestamp
data_cleaned['Day'] = data_cleaned['Timestamp'].dt.day
data_cleaned['Month'] = data_cleaned['Timestamp'].dt.month
data_cleaned['Hour'] = data_cleaned['Timestamp'].dt.hour
data_cleaned['Year'] = data_cleaned['Timestamp'].dt.year

# Drop the original Timestamp column
data_cleaned.drop('Timestamp', axis=1, inplace=True)
print(data_cleaned.head())
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='IncidentGrade', data=Train)
plt.title('Distribution of Incident Grades')
plt.show()

# Select numerical columns
numerical_df = data_cleaned.select_dtypes(include=['number'])

# Compute the correlation matrix
corr_matrix = numerical_df.corr().abs()

```



```

# Plot the heatmap
plt.figure(figsize=(20,17))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', vmin=-1,
            vmax=1, linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()

# Setting a correlation threshold
threshold = 0.8

# Find pairs of columns with high correlations
high_corr_var = np.where(corr_matrix > threshold)
high_corr_pairs = [(numerical_df.columns[x], numerical_df.columns[y])
                   for x, y in zip(*high_corr_var)
                   if x != y and x < y]

# Create a list of columns to remove
columns_to_remove = set()
for col1, col2 in high_corr_pairs:
    columns_to_remove.add(col2) # or col1, based on your strategy

# Convert to list
columns_to_remove = list(columns_to_remove)

#print(columns_to_remove)

# Drop the redundant columns
df_cleaned_corr = data_cleaned.drop(columns=columns_to_remove)

```

```

# Verify the cleaned DataFrame
print("Columns removed:", columns_to_remove)
print(df_cleaned_corr.head())

from sklearn.feature_selection import f_classif
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from scipy.stats import pearsonr

categorical_cols = df_cleaned_corr.select_dtypes(include=['object']).columns

# Initialize a dictionary to hold the label encoders
label_encoders = {}

# Apply LabelEncoder to each categorical column
for column in categorical_cols:
    le = LabelEncoder()
    df_cleaned_corr[column] = le.fit_transform(df_cleaned_corr[column])
    label_encoders[column] = le

# The DataFrame now has label-encoded categorical columns and any
    numerical columns
print(df_cleaned_corr)
import joblib

# Save label encoders for categorical features
for column, le in label_encoders.items():
    joblib.dump(le, f'/content/drive/MyDrive/{column}_label_encoder.pkl')
df_cleaned_corr.to_csv('/content/drive/MyDrive/train_data_processed.csv',

```

```

    index=False)
df_cleaned_corr
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='IncidentGrade', data=Test)
plt.title('Distribution of Incident Grades')
plt.show()

Test['Timestamp'] = pd.to_datetime(Test['Timestamp'])

# Extract day, month, and hour from the Timestamp
Test['Day'] = Test['Timestamp'].dt.day
Test['Month'] = Test['Timestamp'].dt.month
Test['Hour'] = Test['Timestamp'].dt.hour

# Drop the original Timestamp column
Test.drop('Timestamp', axis=1, inplace=True)

# Check the new DataFrame
print(Test.head())

Test.drop(['AccountObjectId', 'ApplicationName', 'FolderPath', 'State',
          'FileName', 'OSVersion', 'AccountName', 'City', 'RegistryValueData'], axis=1,
          inplace= True)

Test.info()

Test = Test[['Id','OrgId','IncidentId',
            'AlertId','DetectorId','AlertTitle','Category','IncidentGrade','EntityType','EvidenceRole','DeviceId','Sha256','IpAddress','Url','AccountSid','AccountUpn','DeviceName','NetworkMessageId','RegistryKey','RegistryValueName','ApplicationId','OAuthApplicationId','ResourceIdName','OSFamily','CountryCode','Day','Month','Hour']]

Test.head()

```

```

# Load label encoder
loaded_label_encoders = {}

categorical_cols = ['Category', 'IncidentGrade', 'EntityType', 'EvidenceRole'] #
    list your categorical columns
for column in categorical_cols:
    loaded_label_encoders[column] =
        joblib.load(f'/content/drive/MyDrive/{column}_label_encoder.pkl')

    # Apply label encoding to categorical features
for column, le in loaded_label_encoders.items():
    if column in Test.columns:
        Test[column] = le.transform(Test[column])
Test
Test.to_csv('/content/drive/MyDrive/test_data_processed.csv', index=False)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score,
    confusion_matrix, ConfusionMatrixDisplay
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.preprocessing import LabelEncoder
import lightgbm as lgb
import matplotlib.pyplot as plt
import pandas as pd

df_train = pd.read_csv("/content/drive/MyDrive/train_data_processed.csv")
df_test = pd.read_csv("/content/drive/MyDrive/test_data_processed.csv")
df_train
df_test

```

```

#Splitting data
X= df_train.drop('IncidentGrade',axis=1)
y= df_train['IncidentGrade']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

#Selecting top features using anova
from sklearn.feature_selection import f_classif, SelectKBest
selector = SelectKBest(score_func=f_classif, k=15) # Adjust k as needed
X_new = selector.fit_transform(X_train, y_train)
selected_features = X_train.columns[selector.get_support()]
print("Selected Features:", selected_features)
X_new=X[['OrgId', 'IncidentId', 'AlertId', 'DetectorId', 'AlertTitle',
    'Category', 'EntityType', 'EvidenceRole', 'Sha256', 'IpAddress',
    'AccountSid', 'DeviceName', 'NetworkMessageId', 'CountryCode', 'Day']]
X_new

#Training train data with selected features
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2,
    random_state=42)

# Output the shapes of the training and validation sets
print("Training set shape:", X_train.shape)
print("Validation set shape:", X_test.shape)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

```

## **# LOGISTIC REGRESSION MODEL**

```
model_logistic = LogisticRegression(random_state=42)

# Print message for evaluation
print(f"\nEvaluating Logistic Regression Model...")

# Fit the logistic regression model
model_logistic.fit(X_train, y_train)

# Predict on test data
y_pred_logistic = model_logistic.predict(X_test)

# Print accuracy and classification report
print("Accuracy:", accuracy_score(y_test, y_pred_logistic))
print("Classification Report:")
print(classification_report(y_test, y_pred_logistic, zero_division=0))

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_logistic)

# Display the confusion matrix with custom labels
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=['BenignPositive', 'FalsePositive',
                                              'TruePositive'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - Logistic Regression Model')
plt.show()
```

## **# RANDOM FOREST MODEL**

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

# Initialize RandomForest with optimizations

```
model_rf = RandomForestClassifier(
    n_estimators=50,      # Reduce the number of trees
    max_depth=10,        # Limit the depth of the trees
    max_features='sqrt',  # Use the square root of features
    n_jobs=-1,           # Enable parallel processing
    random_state=42
)
```

# Print message for evaluation

```
print(f"\nEvaluating RandomForest...")
```

# Fit the model

```
model_rf.fit(X_train, y_train)
```

# Predict on test data

```
y_pred_rf = model_rf.predict(X_test)
```

# Print accuracy and classification report

```
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_rf)

# Display the confusion matrix with custom labels
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=['BenignPositive', 'FalsePositive',
                                             'TruePositive'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - RandomForest Model')
plt.show()
```



## # XGBOOST MODEL

```
model_xgb = XGBClassifier(random_state=42)
```

## #XGBOOST

```
print(f"\nEvaluating XGBoost...")
```

## # Fit the model

```
model_xgb.fit(X_train, y_train)
```

## # Predict on test data

```
y_pred_xgb = model_xgb.predict(X_test)
```

## # Print accuracy and classification report

```
print("Accuracy:", accuracy_score(y_test, y_pred_xgb))
```

```
print("Classification Report:")
```

```
print(classification_report(y_test, y_pred_xgb))
```

## # Generate the confusion matrix

```
cm = confusion_matrix(y_test, y_pred_xgb)
```

## # Display the confusion matrix with custom labels

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,  
                               display_labels=['BenignPositive', 'FalsePositive',  
                                               'TruePositive'])
```

```
disp.plot(cmap=plt.cm.Blues)
```

```
plt.title('Confusion Matrix - XGBoost Model')
```

```
plt.show()
```

## # LGBOOST MODEL

```
model_lgb = LGBMClassifier(random_state = 42)
```

```
#LGBBoost
```

```
print(f"\nEvaluating LGBBoost...")
```

```
# Fit the model
```

```
model_lgb.fit(X_train, y_train)
```

```
# Predict on test data
```

```
y_pred_lgb = model_lgb.predict(X_test)
```

```
# Print accuracy and classification report
```

```
print("Accuracy:", accuracy_score(y_test, y_pred_lgb))
```

```
print("Classification Report:")
```

```
print(classification_report(y_test, y_pred_lgb))
```

```
# Generate the confusion matrix
```

```
cm = confusion_matrix(y_test, y_pred_lgb)
```

```
# Display the confusion matrix with custom labels
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
```

```
                             display_labels=['BenignPositive', 'FalsePositive',
```

```
                             'TruePositive'])
```

```
disp.plot(cmap=plt.cm.Blues)
```

```
plt.title('Confusion Matrix - LightGBM Model')
```

```
plt.show()
```

```

# DECISION TREE MODEL

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt


# Initialize Decision Tree with optimizations
model_dt = DecisionTreeClassifier(
    max_depth=10,          # Limit the depth of the tree
    random_state=42        # For reproducibility
)

# Print message for evaluation
print(f"\nEvaluating Decision Tree...")


# Fit the Decision Tree model
model_dt.fit(X_train, y_train)


# Predict on test data
y_pred_dt = model_dt.predict(X_test)


# Print accuracy and classification report
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Classification Report:")
print(classification_report(y_test, y_pred_dt))


# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_dt)

```

```

# Display the confusion matrix with custom labels
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=['BenignPositive', 'FalsePositive',
                                              'TruePositive'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - Decision Tree Model')
plt.show()

import joblib

# Save the trained XGBoost model to a file
joblib.dump(model_xgb, '/content/drive/MyDrive/xgboost_model.pkl')

# Error Analysis
errors = (y_test != y_pred_logistic)
error_analysis = pd.DataFrame({'True': y_test[errors],
                              'Predicted': y_pred_logistic[errors]})
print("Error Analysis (Misclassifications):")
print(error_analysis)

# Error Analysis
errors = (y_test != y_pred_rf)
error_analysis = pd.DataFrame({'True': y_test[errors], 'Predicted':
                              y_pred_rf[errors]})
print("Error Analysis (Misclassifications):")
print(error_analysis)

```

```

# Error Analysis
errors = (y_test != y_pred_lgb)
error_analysis = pd.DataFrame({'True': y_test[errors], 'Predicted':
    y_pred_lgb[errors]})
print("Error Analysis (Misclassifications):")
print(error_analysis)

# Error Analysis for Decision Tree
errors_dt = (y_test != y_pred_dt) # Identify misclassifications
error_analysis_dt = pd.DataFrame({'True': y_test[errors_dt], 'Predicted':
    y_pred_dt[errors_dt]})

# Display misclassifications
print("Error Analysis (Misclassifications) - Decision Tree:")
print(error_analysis_dt)

X2 = df_test[['OrgId', 'IncidentId', 'AlertId', 'DetectorId', 'AlertTitle',
    'Category', 'EntityType', 'EvidenceRole', 'Sha256', 'IpAddress',
    'AccountSid', 'DeviceName', 'NetworkMessageId', 'CountryCode', 'Day']]
y2= df_test['IncidentGrade']
import joblib

# Load the model
loaded_model_rf = joblib.load('/content/drive/MyDrive/xgboost_model.pkl')

```

```

# Check the model type
print(type(loader_model_rf))
y_pred_new = loader_model_rf.predict(X2)
print("Accuracy:", accuracy_score(y2, y_pred_new))
print("Classification Report:")
print(classification_report(y2, y_pred_new))

# Generate the confusion matrix
cm = confusion_matrix(y2, y_pred_new)

# Initialize Decision Tree with optimizations
model_dt = DecisionTreeClassifier(
    max_depth=10,          # Limit the depth of the tree
    random_state=42        # For reproducibility
)

# Print message for evaluation
print(f"\nEvaluating Decision Tree...")

# Fit the Decision Tree model
model_dt.fit(X_train, y_train)

# Predict on test data
y_pred_dt = model_dt.predict(X_test)

# Print accuracy and classification report
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Classification Report:")
print(classification_report(y_test, y_pred_dt))

```

```

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_dt)
model_lgb = LGBMClassifier(random_state = 42)

#LGBBoost
print(f"\nEvaluating LGBBoost...")

# Fit the model
model_lgb.fit(X_train, y_train)

# Predict on test data
y_pred_lgb = model_lgb.predict(X_test)

# Print accuracy and classification report
print("Accuracy:", accuracy_score(y_test, y_pred_lgb))
print("Classification Report:")
print(classification_report(y_test, y_pred_lgb))

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_lgb)

# Display the confusion matrix with custom labels
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=['BenignPositive', 'FalsePositive',
                              'TruePositive'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix - LightGBM Model')
plt.show()

```

APPENDICES II

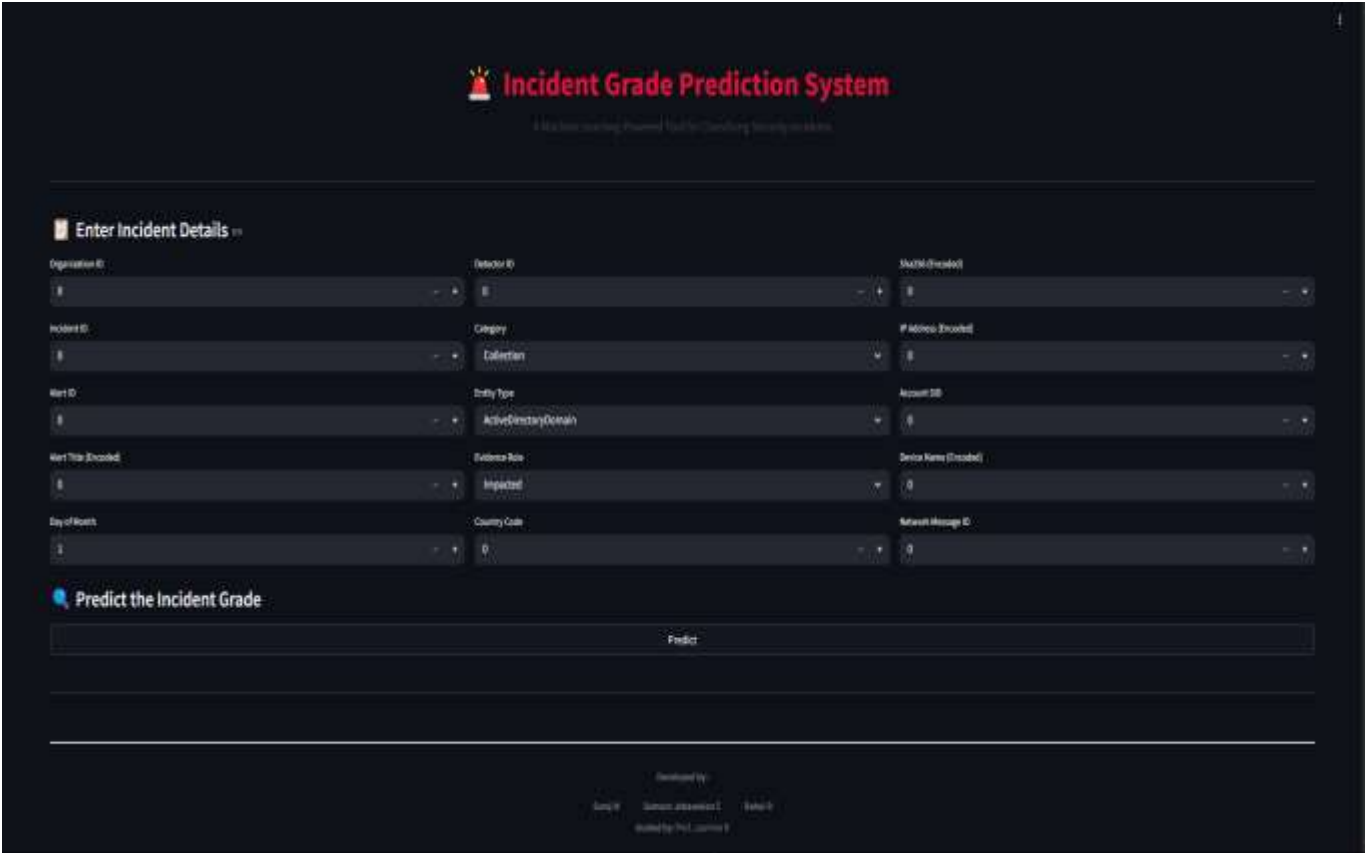


Fig. A2.1 Streamlit Interface

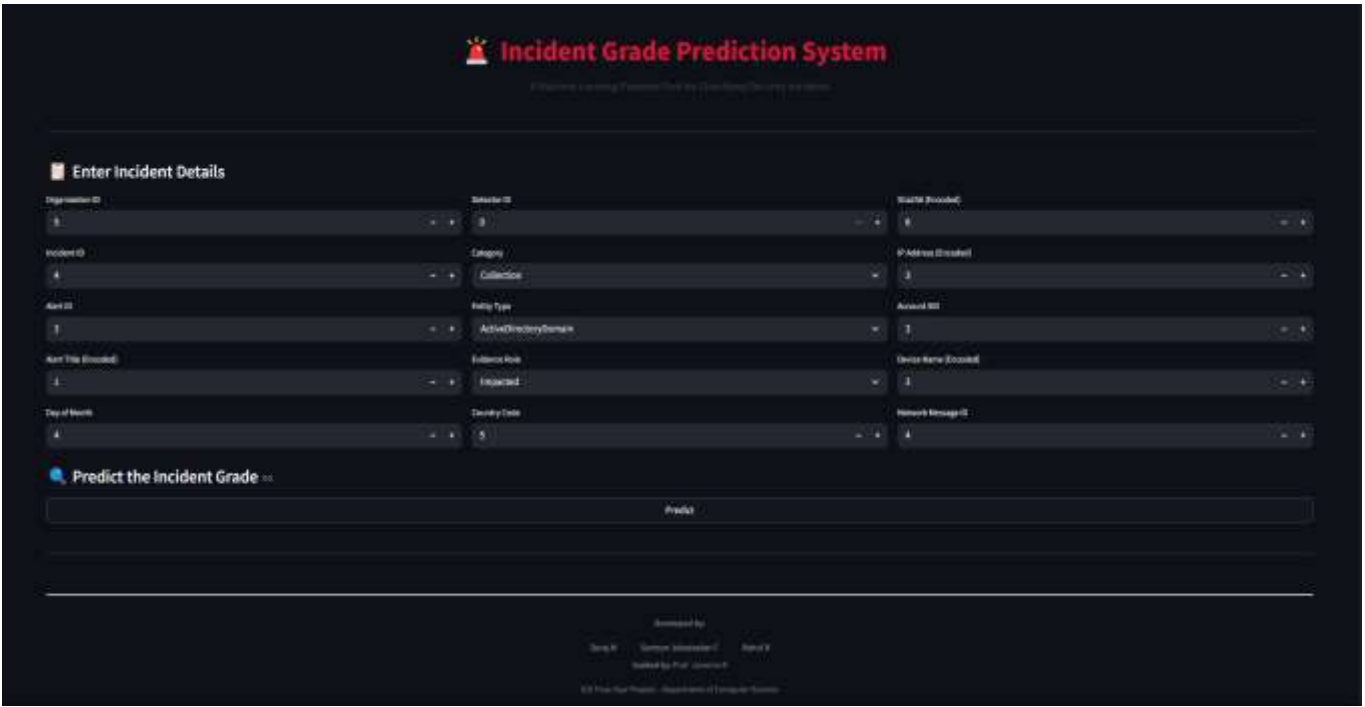



Fig. A2.2 Data Entry in Incident Detection





# Incident Grade Prediction System


Intelligent Learning Process | Incident Classification | Security Incident

## Enter Incident Details

Organization ID	Detector ID	Device ID (Encoded)
5	8	8
Incident ID	Category	IP Address (Encoded)
4	Collection	3
Alert ID	Alert Type	Account ID
3	ActiveDirectoryDomain	3
Alert Title (Encoded)	Incident Note	Device Name (Encoded)
2	Impacted	3
Day of Month	Country Code	Network Message ID
4	5	4

## Predict the Incident Grade

Predict


Predicted Incident Grade: TruePositive

### Model Details

- Model Used: XGBoost Classifier
- Type: Multi-class Classification
- Input Features: (5 (numeric and encoded categorical values))
- Label: Incident Grade (High, Medium, Low, etc.)

Fig. A2.3 Predicting the Incident Through Given Dataset

## REFERENCES:

- [1] Y. Li, X. Zhang, S. He, et al., “An Intelligent Framework for Timely, Accurate, and Comprehensive Cloud Incident Detection,” *ACM SIGOPS Operating Systems Review*, vol. 56, no. 1, pp. 1–7, Jun. 2022.
- [2] X. Zhang, Y. Xu, Q. Lin, et al., “Onion: Identifying Incident-Indicating Logs for Cloud Systems,” in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2021, pp. 1253–1263.
- [3] C. Luo, P. Zhao, B. Qiao, et al., “NTAM: Neighborhood-Temporal Attention Model for Disk Failure Prediction in Cloud Platforms,” in *Proceedings of the Web Conference 2021*, pp. 1181–1191.
- [4] Y. Wang, G. Li, Z. Wang, et al., “Fast Outage Analysis of Large-Scale Production Clouds with Service Correlation Mining,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 885–896.
- [5] W. Wang, J. Chen, L. Yang, et al., “How Long Will it Take to Mitigate this Incident for Online Service Systems?,” in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 36–46.
- [6] D. Song, X. Zhang, et al., “Robust Log-Based Anomaly Detection on Unstable Log Data,” in *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2019, pp. 807–817.
- [7] W. Meng, Y. Liu, Y. Zhu, et al., “LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs,” in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 4739–4745.
- [8] R. Li, et al., “OmniAnomaly: Detecting Anomalies in Multivariate Time Series with Dynamic Bayesian Networks,” in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 4690–4696.
- [9] J. Zhu, Z. Xu, L. Zhou, et al., “AIOps: Real-world Challenges and Research Innovations,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pp. 4–5.
- [10] X. Zhang, J. Kim, Q. Lin, et al., “Cross-Dataset Time Series Anomaly Detection for Cloud Systems,” in *2019 USENIX Annual Technical Conference (USENIX ATC)*, pp. 1063–1076.

- [11] M. Du, F. Li, G. Zheng, and V. Srikumar, “DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1285–1298.
- [12] Q. Fu, et al., “Drain: An Online Log Parsing Approach with Fixed Depth Tree,” in *2017 IEEE International Conference on Web Services (ICWS)*, pp. 33–40.
- [13] X. Zhang, et al., “Microscope: Pinpointing the Root Cause of Cloud System Failures,” in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pp. 60–70.
- [14] P. He, J. Zhu, P. Xu, Z. Zheng, and M. R. Lyu, “A Directed Acyclic Graph Approach to Online Log Parsing,” *arXiv preprint arXiv:1806.04356*, 2018.
- [15] P. Ding, H. Gao, H. Bu, H. Ma, and H. Si, “Multivariate-Time-Series-Driven Real-time Anomaly Detection Based on Bayesian Network,” *Sensors*, vol. 18, no. 3367, 2018.
- [16] R. Agrawal, et al., “Experience Report: Log Mining Using Natural Language Processing and Clustering,” in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 96–101.
- [17] S. Qin, Y. Xu, S. Zhou, et al., “Prediction-Guided Design for Software Systems,” in *Proceedings of the 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pp. 1243–1254.
- [18] P. Zhao, C. Luo, B. Qiao, et al., “F3: Fault Forecasting Framework for Cloud Systems,” in *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 1243–1254.
- [19] Y. Xu, K. Sui, R. Yao, et al., “Improving Service Availability of Cloud Systems by Predicting Disk Error,” in *2018 USENIX Annual Technical Conference (USENIX ATC)*, pp. 481–494.
- [20] H. Guo, S. Yuan, and X. Wu, “LogBERT: Log Anomaly Detection via BERT,” *arXiv preprint arXiv:2103.04475*, 2021.



**SRM TRP**  
ENGINEERING COLLEGE  
Affiliated to ANNA UNIVERSITY  
**TIRUCHIRAPPALLI**



## *Certificate of Participation*

**RAHUL R**

This is to certify that Dr. / Prof. /Mr. /Ms. ....  
of **K.Ramakrishnan College Of Technology** ..... has presented a paper titled  
**Machine Learning Driven Cyber Incident Detection And Response System**  
in the 4<sup>th</sup> National Conference on **“Recent Advances in Communicative Electronics (NCRACE 2025)”** organized by the **Department of Electronics and Communication Engineering, SRM TRP Engineering College**, Tiruchirappalli, during April 10-11, 2025.

*B. Ramasubramanian.*  
**Convener**

*[Signature]*  
**Principal**





INSTITUTION'S  
INNOVATION  
COUNCIL  
(Ministry of Education Initiative)



**SRM TRP**  
ENGINEERING COLLEGE  
Affiliated to ANNA UNIVERSITY  
**TIRUCHIRAPPALLI**



## *Certificate of Participation*

**SAMSON JEBASEELAN C**

This is to certify that Dr. / Prof. /Mr. /Ms. ....

of **K.Ramakrishnan College Of Technology** ..... has presented a paper titled  
**Machine Learning Driven Cyber Incident Detection And Response System**

in the 4<sup>th</sup> National Conference on **"Recent Advances in Communicative Electronics (NCRACE 2025)"** organized by the **Department of Electronics and Communication Engineering, SRM TRP Engineering College**, Tiruchirappalli, during April 10-11, 2025.

*B. Ramasubramanian*

**Convener**

**Principal**



**SRM TRP**  
ENGINEERING COLLEGE  
Affiliated to ANNA UNIVERSITY  
**TIRUCHIRAPPALLI**



## *Certificate of Participation*

**SURAJ M**

This is to certify that Dr. / Prof. / Mr. / Ms. ....  
of **K.Ramakrishnan College Of Technology** ..... has presented a paper titled  
**Machine Learning Driven Cyber Incident Detection And Response System**  
in the 4<sup>th</sup> National Conference on **"Recent Advances in Communicative Electronics  
(NCRACE 2025)"** organized by the **Department of Electronics and Communication  
Engineering, SRM TRP Engineering College**, Tiruchirappalli, during April 10-11, 2025.

*B. Ramasubramanian*

**Convener**

**Principal**