Visvesvaraya Technological University, Belagavi – 590018

PROJECT REPORT
ON

Smart Defend : Strengthening IoT Security with Advanced Machine Learning Techniques

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE & ENGINEERING

Submitted by

Joyline Rencita Dsouza	4SO20CS073
Melanie Crystal Miranda	4SO20CS085
Natasha Elizabeth Correia	4SO20CS092

Under the Guidance of

Dr Saumya Y M
Associate Professor, Department of CSE

DEPT. OF COMPUTER SCIENCE AND ENGINEERING
ST JOSEPH ENGINEERING COLLEGE
An Autonomous Institution
(Affiliated to VTU Belagavi, Recognized by AICTE, Accredited by NBA)
Vamanjoor, Mangaluru - 575028, Karnataka
2023-24

ST JOSEPH ENGINEERING COLLEGE

An Autonomous Institution

(Affiliated to VTU Belagavi, Recognized by AICTE, Accredited by NBA)

Vamanjoor, Mangaluru - 575028, Karnataka

DEPT. OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled **"Smart Defend : Strengthening IoT Security with Advanced Machine Learning Techniques"** carried out by

Joyline Rencita Dsouza	**4SO20CS073**
Melanie Crystal Miranda	**4SO20CS085**
Natasha Elizabeth Correia	**4SO20CS092**

the bonafide students of VIII semester Computer Science & Engineering in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2023-2024. It is certified that all corrections/suggestions indicated during Internal Assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Dr Saumya Y M	**Dr Sridevi Saralaya**	**Dr Rio D'Souza**
Project Guide	HOD-CSE	Principal

External Viva:

Examiner's Name	Signature with Date
1.	....
2.	....

ST JOSEPH ENGINEERING COLLEGE

An Autonomous Institution

(Affiliated to VTU Belagavi, Recognized by AICTE, Accredited by NBA)

Vamanjoor, Mangaluru - 575028, Karnataka

DEPT. OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the entire work embodied in this Project Report Titled **“Smart Defend : Strengthening IoT Security with Advanced Machine Learning Techniques”** has been carried out by us at St Joseph Engineering College, Mangaluru under the supervision of **Dr Saumya Y M**, for the award of **Bachelor of Engineering in Computer Science & Engineering**. This report has not been submitted to this or any other University for the award of any other degree.

Joyline Rencita Dsouza	**4SO20CS073**
Melanie Crystal Miranda	**4SO20CS085**
Natasha Elizabeth Correia	**4SO20CS092**

Acknowledgement

We dedicate this page to acknowledge and thank those responsible for the shaping of the project. Without their guidance and help, the experience while constructing the dissertation would not have been so smooth and efficient.

We sincerely thank our Project guide **Dr Saumya Y M**, Associate Professor, Computer Science and Engineering for her guidance and valuable suggestions which helped us to complete this project. We also thank our Project coordinator **Dr Harivinod N**, Dept of CSE, for his consistant encouragement.

We owe a profound gratitude to **Dr Sridevi Saralaya**, Head of the Department, Computer Science and Engineering, whose kind support and guidance helped us to complete this work successfully

We are extremely thankful to our Principal, **Dr Rio D'Souza**, Director, Rev. **Fr Wilfred Prakash D'Souza**, and Assistant Director, **Rev Fr Kenneth Rayner Crasta** for their support and encouragement.

We would like to thank all faculty and staff of the Deprtment of Computer Science and Engineering who have always been with us extending their support, precious suggestions, guidance, and encouragement through the project.

We also extend our gratitude to our friends and family members for their continuous support.

Abstract

IoT is the fastest-growing technology and by the end of 2030, more than 100 billion IoT devices will be linked to internet. As a result, protecting devices from intrusions and maintaining user privacy has become challenging. Hence, we present an efficient model to identify security and privacy related vulnerabilities in IoT environments using several machine learning algorithms.

The major focus of this research is to build effective systems by extracting characteristics from raw data and transforming them into essential properties. The aim of this project is to create a system that focuses on privacy attack detection systems in IoT networks and learning the characteristics of all networks in a heterogeneous network platform. The built-in IDS/IPS feature of the Next Gen IDS provides integrated solution for threat detection and security administration.

The proposed solution along with data-set uses several Machine Learning approaches suitable for classification tasks, detecting malicious network traffic with help of machine learning algorithms, which will be able to handle novel records even if they can train the data. The dataset used was the UNSW-NB15 where it includes 45 features.

The attacks in this dataset are classified (multi-class) into nine attacks. The specification-based IDS technique exploits the benefits of both signature and anomaly-based detection techniques. In binary classification Random Forest Classifier gives accuracy of 92.72% and in multiclass classification Random Forest Classifier gives accuracy of 92.40% which is best among 7 algorithms used. Further work will be done on a comparative study of certain data mining techniques with selective feedback approaches that can accurately capture the actual behavior of intrusions and normal activities. A front-end is being developed for individuals and businesses.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Amidst the proliferation of IoT devices promising enhanced convenience and efficiency, the growing interconnected landscape brings forth a critical concern – the escalating vulnerability to cyber threats. The conventional security infrastructure for IoT networks, designed for a simpler era, has encountered substantial challenges in adapting to the evolving threat landscape. Issues such as delayed notifications of attacks, ineffective response mechanisms, and the inability to cope with sophisticated intrusion tactics have underscored the inadequacy of traditional security measures [2].

The surge in IoT adoption has exposed vulnerabilities that were previously unforeseen, rendering conventional security systems ill-equipped to safeguard against modern cyber threats. Delayed notifications of attacks have been a persistent issue, allowing malicious actors to exploit vulnerabilities and compromise IoT networks before countermeasures could be initiated [3]. These challenges necessitate a paradigm shift in approach, prompting the exploration of innovative solutions to fortify the security posture of IoT environments. This study delves into the shortcomings of existing security systems, emphasizing the pressing need for a more adaptive and sophisticated IoT intrusion detection system.

1.2 Problem statement

In the rapidly evolving landscape of Internet of Things (IoT) networks, ensuring the security and integrity of connected devices is of paramount importance. The increasing complexity and interconnectedness of IoT devices make them susceptible to a myriad of malicious activities. Traditional security measures often fall short in effectively detecting these threats. This project aims to address the critical challenge of enhancing the security of IoT networks by leveraging Machine Learning (ML) algorithms for anomaly detection and classification[2][3]. The primary goal is to develop a robust system capable of identifying deviations from normal network behavior, thus enabling the timely detection of malicious activities. Furthermore, the system will not only flag anomalies but will also provide a detailed classification of the detected anomalies, specifying the type of attack occurring within the IoT network. This project aspires to yield a comprehensive anomaly detection and classification system for IoT networks, encompassing key outcomes [1]. The implementation of advanced machine learning algorithms is poised to achieve a heightened level of accuracy in discerning normal and anomalous behaviors within the network. The system aims not only to detect anomalies but to delve deeper, providing detailed insights through the precise classification of detected anomalies into specific attack types. A pivotal focus is placed on real-time detection capabilities, ensuring the system's agility in promptly identifying and responding to potential security breaches, thereby minimizing the impact of attacks on the IoT network. The project also emphasizes user accessibility by integrating a user-friendly interface or dashboard, facilitating seamless interpretation of detected anomalies and attack classifications. The project is dedicated to the optimization of system reliability by prioritizing the minimization of false positives, ensuring that genuine security threats are effectively highlighted while mitigating unnecessary alerts. These outcomes are poised to establish a resilient and user-friendly security framework for IoT networks.

1.3 Scope

The project aims to develop an integrated cybersecurity system using Internet of Things (IoT) devices and machine learning algorithms. The system will focus on real-time threat detection, anomaly analysis, and responsive mitigation to enhance the overall security posture of networked environments. The project's relevance in a real-world context is significant, given the growing proliferation of Internet of Things (IoT) devices and the associated security challenges. Here are the key aspects highlighting its potential application domains:

1. **Anomaly detection in critical infrastructures:**
   The project's focus on anomaly detection and classification is highly relevant in critical infrastructures like smart grids, where any abnormal behavior could indicate a potential cyber threat or system malfunction.

2. **Healthcare IoT:**
   Internet of Things (IoT)-enabled devices have made remote monitoring in the healthcare sector possible, unleashing the potential to keep patients safe and healthy, and empowering physicians to deliver superlative care. It has also increased patient engagement and satisfaction as interactions with doctors have become easier and more efficient.

3. Industrial IoT (IIoT):
Industrial IoT is an ecosystem of devices, sensors, applications, and associated networking equipment that work together to collect, monitor, and analyze data from industrial operations. Analysis of such data helps increase visibility and enhances troubleshooting and maintenance capabilities.

4. Smart cities: Smart city initiatives rely heavily on IoT technologies for various applications, including traffic management, public safety, and environmental monitoring. The security of these interconnected systems is crucial to prevent cyber-attacks that could impact city services.

5. Supply chain security:
IoT helps businesses track good movement and conditions in real time. IoT-enabled smart contracts and blockchain can automate payment settlements. Either way, IoT reduces supply chain costs and improves supply chain efficiency in a nutshell.

6. Energy sector:
IoT energy management in the commercial sector involves the reduction of energy consumption. Iot offers systems that monitor consumption and reduce usage in an effective way. It helps both small and larger corporations. It helps in the optimization of power and at the same time improves the functionality.

7. Financial services:
IoT devices are increasingly used in financial services for tasks such as asset tracking and secure transactions. Ensuring the security of these devices is critical to safeguard financial data and prevent fraudulent activities.

8. Smart home security:
With the growing popularity of smart homes, where various devices are interconnected, the project's security framework is relevant in ensuring the protection of personal data and preventing unauthorized access to smart home systems.

9. Rising IoT adoption:
As the adoption of IoT devices continues to rise across various industries (such as healthcare, manufacturing, smart cities, and more), the need for robust security measures becomes paramount.

Chapter 2

Literature Survey

2.1 Detecting cybersecurity attacks in Internet of Things using artificial intelligence methods

In article [1] thoroughly compares Machine Learning (ML) and Deep Learning (DL) approaches within the realm of IoT cybersecurity. It strives to identify the most effective AI methods for detecting threats, emphasizing the importance of secure and intelligent IoT infrastructures. The study explores a spectrum of ML algorithms, including Naïve Bayes, Decision Tree, Random Forest, Support Vector Machine, and DL algorithms like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM). Notably, 15 studies focus on DL, highlighting its pivotal role.

Most studies rely on the KDD dataset, but the survey urges the need for diverse datasets, emphasizing real-world, real-time IoT systems. Identified threats in IoT span DoS, DDoS, malicious attacks, ransomware, and more, categorized into Probe, U2R, R2L, and DoS. Power grid disturbances, fog-based attack. Recommendations include the widespread use of Support Vector Machines (SVM) and Random Forest (RF) for their accuracy. DL methods, particularly based on Artificial Neural Networks, RNN, and LSTM, prove efficient in detecting IoT malware and attacks. Hybrid approaches and XGBoost, a boosting algorithm, are also highlighted.
The survey recognizes limitations in proposed frameworks, such as methodology and data analysis weaknesses, resulting in low accuracy. Some studies lack reporting on predictive features, and an emphasis on accuracy metrics may limit exploration of specific threats. Future research should focus on next-gen AI algorithms in IIoT, medical IoT, energy IoT, and CPS. Exploring diverse databases and enhancing intelligent frameworks is crucial. Researchers are encouraged to identify vulnerabilities beyond categorized attacks, anticipating the development of novel IDS/AI models for securing the IoT against evolving cybersecurity threats. Additionally, one paper recommends using two datasets to over-

come training time challenges and enrich feature sets, showcasing a practical approach to optimize AI models in IoT security.

2.2 Botnet attack detection in IoT using machine learning

In paper [2] the literature survey discussed in the provided context extensively compares Machine Learning (ML) and Deep Learning (DL) approaches within the domain of detecting and categorizing botnet attacks in IoT environments. The study places a critical lens on the effectiveness of these approaches, employing datasets such as KDD99 and UNSW-NB15, to specifically address a range of attacks, including scanning activities, distributed denial of service (DDoS), and various malicious behaviors orchestrated by botnets.

In terms of the comparison between ML and DL models, the survey thoroughly scrutinizes their respective strengths. Noteworthy mentions include the exploration of ML classifiers like Decision Tree, XgBoost, and Logistic Regression, juxtaposed with DL models such as ResNet50 and Long Short-Term Memory (LSTM). This comparative analysis forms a foundational aspect of the survey, shedding light on the nuanced performance of these models in tackling the intricacies of botnet detection.
A pivotal aspect of the discussion revolves around the datasets employed, with particular emphasis placed on two primary datasets—KDD99 and UNSW-NB15. The survey underscores the significance of features simulated in the UNSW-NB15 dataset, showcasing their superior efficacy over the original KDD99 features. This meticulous consideration of datasets adds depth to the evaluation of ML and DL approaches. The survey comprehensively covers a diverse spectrum of attacks, ranging from Fuzzers and Analysis to Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Both ML and DL methodologies are deftly applied to classify these multifaceted attacks, addressing the inherent challenges posed by their complex and varied patterns.

Identifying limitations is a crucial fact of the survey, with a recognition of the need to handle imbalanced datasets and the inherent challenges presented by diverse attack patterns. Looking toward the future, the study proposes intriguing avenues for further exploration. This includes delving into additional ML classifiers, testing DL models like ResNet50 and LSTM, and integrating the developed model into both frontend and backend web applications.

2.3 A novel feature selection approach to classify intrusion attacks in network communications

The literature survey extensively compares Machine Learning (ML) and Deep Learning (DL) approaches in the context of intrusion detection [3]. Various feature selection techniques and classification methods have been explored to determine the effectiveness of these approaches. The study investigates the performance of traditional ML techniques and modern DL algorithms in addressing the challenges posed by cyber threats. The comparison aims to classify which approach is more suitable for different situations in intrusion detection.

Several datasets are utilized in the literature survey to evaluate the proposed methods. Notable datasets include KDD '99, UNSW-NB15, CIC-IDS2017, NSL-KDD, AWID, and IoTID20. These datasets serve as the basis for training and testing intrusion detection models.

The literature covers a range of cyber attacks, including remote-to-user (R2L), user-to-remote (U2R), denial of service (DoS), distributed DDoS, and probing. These attacks are classified and detected using various approaches, including signature-based, anomaly-based, and hybrid models like SABADT (Signature- and Anomaly-Based Attack Detection Technique). Intrusion Detection Systems (IDSs) play a crucial role in monitoring network traffic for indications of malicious activities.

This paper introduces a novel method to improve intrusion detection systems (IDS) against modern cyber attacks. The method involves a unique feature selection technique and a hybrid classification approach for faster and more accurate attack detection. Testing on KDD '99 and UNSW-NB15 datasets showed superior performance compared to traditional machine learning methods, achieving high accuracy in identifying attack types. The methodology also outperformed other studies in terms of feature usage and accuracy, making significant contributions to the field.

The proposed method exhibits superior speed and accuracy in classifying attacks but acknowledges room for improvement, intending to address lower detection rates for certain attack types, test on new unseen attacks, extend evaluation to diverse datasets, enhance algorithmic accuracy, and focus on real-world application considerations for scalability and practical deployment in future research endeavors.

2.4 Cyberattacks detection in IoT-based smart city applications using machine learning techniques

Article[4] focuses on smart city frameworks and the application of Intrusion Detection Systems (IDS) in enhancing cybersecurity within the Internet of Things (IoT) infrastructure. Various datasets, such as the UNSW-NB15 and CICIDS2017, are explored to evaluate and identify normal and attack network traffic. These datasets offer a comprehensive understanding of contemporary cyber threats, including DoS, DDoS, PortScan, SQL injection, Infiltration, Brute Force, and Bot attacks.

In the context of IDS techniques, the review delves into the selection of anomaly-based Network Intrusion Detection Systems (NIDS) over host-based IDS (HIDS) and signature-based NIDS. The rationale behind choosing anomaly-based NIDS lies in its adaptability to resource-constrained IoT devices and its capability to handle new attacks efficiently. The proposed model emphasizes the tracking of network traffic through fog nodes in close proximity to IoT sensors, thereby facilitating rapid cyber-attack detection.

Data pre-processing involves feature selection using information gain ratio, with the top 25 relevant features chosen for prediction. The selected features are then encoded, and a threshold is applied for optimal filtering. Machine learning techniques, including LR, SVM, DT, RF, KNN, and ANN, are employed to build the IDS scheme. The literature emphasizes the significance of ensemble methods (Bagging, Boosting, Stacking) to improve the accuracy of cyber-attack detection.

The review underlines the importance of smart city frameworks in delivering sustainable services, highlighting ongoing projects such as those in Hong Kong and Masdar City. Challenges related to vulnerable urban development plans and the need for sustainability in smart city services are discussed. IoT and related cyber-physical systems play a crucial role in managing sustainability programs, including intelligent transportation systems, smart buildings, and resource usage.

In conclusion, the literature review provides a comprehensive exploration of smart city frameworks, IDS techniques, machine learning algorithms, and ensemble methods in the context of IoT-based cyber-attack detection. The datasets, attacks, classifiers, and frameworks discussed collectively contribute to a holistic understanding of the challenges and opportunities in securing smart city infrastructures against evolving cyber threats.

2.5 IOT -based cyber security identification model through machine learning technique

The paper [5] discusses the development of a cognitive cybersecurity methodology using machine learning (ML) techniques, particularly focusing on threat classification and vulnerability rating severity. The research involves setting up a database that connects various internet vulnerability repositories and building a pipeline of ML methods to apply them to different groups of information categories.

Natural Language Processing (NLP) techniques are used to transform cybersecurity alerts into numerical representations, and ML algorithms are trained using the collected characteristics synchronously. The study also emphasizes the importance of model selection and ensemble construction to address cybersecurity challenges effectively.

Furthermore, the research evaluates the performance of the proposed ML pipeline, demonstrating the effectiveness of the ensemble of LSTM, NBSVM, and MLP in threat classification. The study also evaluates the model's performance in predicting Common Vulnerability Scoring System (CVSS) scores, showing promising results. The experimental analysis underscores the potential of the ensemble methods in risk categorization and intensity rating in the cybersecurity scenario.

In summary, the research contributes to the development of a cognitive cybersecurity methodology that leverages ML techniques to enhance threat classification and vulnerability rating severity, addressing the complexities of modern security repositories and providing valuable insights into cybersecurity analysis.

Additionally, the research emphasizes the positive prospects of the ensemble methods in risk categorization and intensity rating in the cybersecurity scenario. The study also discusses the potential of the proposed cognitive cybersecurity methodology in modifying investment choices at various managerial levels. Overall, the conclusion highlights the potential of ML methodologies in enhancing cybersecurity monitoring and evaluation, as well as addressing the complexities of modern security repositories.

2.6 Machine learning-enabled IoT security: open issues and challenges under advanced persistent threats

The document “Machine Learning-Enabled IoT Security: Open Issues and Challenges Under Advanced Persistent Threats” [6] discusses the security challenges and attack detection methods in Internet of Things (IoT) networks, with a focus on advanced persistent threats (APT). The authors review the security challenges in IoT networks, present well-known attacks and APT attacks, and discuss threat models in IoT systems.

They also summarize various intrusion detection systems (IDSs) for IoT networks, highlighting statistical insights regarding frequently applied machine learning (ML)-based methods against network intrusion. The document also addresses open issues and challenges for common network intrusion and APT attacks.

The authors highlight the vulnerabilities in IoT networks, such as DoS attacks, port scanning attacks, and web application attacks. They emphasize the importance of detecting and mitigating network intrusion to secure IoT systems, especially in the context of critical infrastructure and industrial control systems. The document also discusses the characteristics and stages of APT attacks, highlighting their advanced, persistent, and threatening nature.

Furthermore, the document reviews different ML-based approaches, including supervised ML methods such as Adaboost, Decision Tree, K-Nearest Neighbor (KNN), and Linear Regression (LR). The authors also discuss anomaly-based methods, hybrid methods, and collaborative IDSs for network intrusion detection in IoT systems. They emphasize the potential of ML algorithms in enhancing detection accuracy for various types of network attacks in IoT.

Overall, the document provides a comprehensive overview of the security challenges, attack detection methods, and the potential of ML-based approaches to address these challenges in IoT networks, particularly in the context of APT attacks.

2.7 Comparison of existing methods

Table 2.1 provides a comprehensive literature survey of various proposed approaches for addressing cybersecurity challenges in the Internet of Things (IoT). The table summarizes key aspects, including the project titles, problems addressed, techniques used, implementation and results, as well as limitations and future scope of each approach.

2.8 Proposed system

The increasing use of Internet of Things (IoT) devices has transformed how we interact with technology, from smart homes to industrial systems. However, this surge in IoT devices has also brought about a rise in security vulnerabilities, exposing networks to various cyber threats. Traditional security measures are struggling to cope with the

Table 2.1: Comparative analysis of different proposed approaches.

Project Title	Problems Addressed	Techniques Used	Implementation and Results	Limitations/ Future Scope
Detecting Cyber Attacks in IoT [1]	Vulnerabilities of IoT networks to malicious attacks across various domains	SVM, SVR, DT, RF, NB, LR, KNN, fuzzy algorithms	ML methods like RF and DNN enhance anomaly detection.	To achieve better performance, combine multiple models to obtain high accuracy.
Botnet Attack Detection in IoT [2]	Protect and prevent botnet attacks, focusing on securing IoT networks	Argus, and Bro-IDS Decision Tree Algorithm, XgBoost Classifier, Logistic Regression.	Decision Tree Algorithm, XgBoost Classifier, Logistic Regression.	Exploring larger datasets, testing diverse ML classifiers, and integrating ResNet50 and LSTM
Feature Selection in Network Communication [3]	Inadequacy of traditional intrusion detection systems in recognizing complex cyber attacks from normal network traffic.	Feature Selection Approach, Hybrid Signature- and Anomaly-Based Attack Detection Technique SABADT	UNSW-NB15 dataset shows high performance in feature selection as compared to KDD-99.	Seeks improvement in detecting specific attack types, aims to test on the new attacks.
Cyberattacks Detection in IoT-Based Smart City [4]	Securing smart cities is tricky due to surprise attacks, the need to catch cyber threats early, and the limited abilities of IoT devices, requiring special solutions.	Support Vector Machine(SVM), Random Forest (RF), Decision Tree(DT), k-nearest Neighbors Algorithm (KNN)	The results showed that ensemble methods particularly Stacking, outperformed individual classifiers in binary and multi-class classification.	Future research should prioritize improving ML algorithms to address the ever-changing nature of cyber threats while safeguarding data privacy
Machine Learning-Enabled IoT Security [5]	Cyber weaknesses in IoT include wireless medium defects and challenges in applying ML for APT attack detection.	CNN, Deep Auto Encoders(DAE), Deep Belief Networks(DBN), DT, FCN, KNN, LR, Long Short-Term Memory (LSTM), RNN	Developed an IoT intrusion detection system, assessing performance with diverse data, machine learning models.	Security limitations, privacy concerns, standardization gaps, and the absence of comprehensive intrusion detection datasets pose challenges.
IoT-based Cyber Security Identification Model [6]	The vulnerability and susceptibility of Cyber-Physical Systems (CPS) and the Internet of Things (IoT) to security threats and harmful actions from attackers or hackers.	KNN, NBSVM, LSTM-ANN, MLP, LR algorithms, ML stacking methodologies, NLP techniques and reconciles competing vulnerabilities.	Use of machine learning algorithms, NLP techniques	Major concern includes ethical, and data sources use open-access web datasets, incorrect ML model implemented

complexity of these emerging threats, necessitating advanced technologies like machine learning. This project is crucial because it addresses the security challenges faced by IoT networks by using machine learning algorithms for timely detection and classification of anomalies. The goal is to develop a robust system capable of not only flagging anomalies but also providing detailed insights into the type of attacks occurring, thereby enhancing the overall security of IoT networks in the digital age. The importance lies in safeguarding sensitive data and critical infrastructure from potential cyber threats, ensuring the reliability and safety of interconnected systems.

This project aims to make IoT networks more secure by using advanced machine learning (ML) algorithms for detecting and classifying anomalies. The key goal is to create a robust system that can quickly identify deviations from normal network behavior, helping detect malicious activities in a timely manner. Unlike traditional security measures, this project goes a step further by not only flagging anomalies but also providing detailed insights into the specific type of attack happening in the IoT network. The focus is on achieving high accuracy in discerning normal and anomalous behaviors, enabling real-time detection, and minimizing false alarms. By integrating a user-friendly interface, the project aims to make it easy for users to interpret detected anomalies and attack classifications. Overall, this initiative is set to contribute to a more resilient and user-friendly security framework for IoT networks, advancing the current state-of-the-art in IoT security.

The proposed project stands out in IoT security by doing more than just the basics. It looks at nine different kinds of threats, making sure it covers all the angles. This makes the defense for connected devices really precise and strong. What makes it even cooler is that it acts really fast when there's a problem. It's like having a smart system that spots issues quickly and knows how to handle them right away. It's not expensive, so you get top-notch security without spending a lot. This project is all about being smart, quick, and budget-friendly for keeping IoT devices safe.

Our approach stands out from existing works through two key features. Firstly, we meticulously selected algorithms with a proven track record of high accuracy in threat detection, ensuring a precise and reliable security mechanism. Secondly, unlike some counterparts that rely on older datasets, we leverage the latest and most up-to-date data. This commitment to using current information enhances our tool's adaptability and effectiveness, making it a more advanced and relevant solution in addressing contemporary security challenges. To train our tool, we use the most efficient model, like a hybrid, because it is better compared to others, ensuring top-notch performance in threat detection.

Chapter 3

Software Requirements Specification

3.1 Introduction

This section commences the exploration of the Software Requirements Specification (SRS), a pivotal phase in our project's advancement. Within the realm of software requirements, our focus is on elucidating fundamental aspects, precisely defining functionalities and constraints that will guide subsequent development phases. The objective is to establish transparent guidelines and specifications, providing a robust foundation for our software development endeavors.

3.2 Functional requirements

1. Develop smart algorithms using machine learning to quickly spot unusual activities in IoT networks and categorize them as specific cyber threats.
2. Ensure the system reacts promptly to identify and address potential security problems, making it challenging for cyber attacks to harm the IoT network.
3. Utilize advanced machine learning to distinguish between normal and abnormal behaviors in the IoT network, ensuring reliable detection of potential threats.
4. Design an easy-to-use system interface so that users can quickly understand and address any unusual activities or cyber attacks in the IoT system.
5. Provide detailed information by sorting out and classifying any unusual activities into specific types of cyber attacks, aiding in a better understanding of possible threats.
6. Focus on reducing false alarms to ensure the system predominantly highlights real security threats, enhancing the overall reliability of the security system.

7. Create a robust and user-friendly security system for IoT networks by integrating all these features.

3.3 Non-functional requirements

- **Performance:** The IoT cybersecurity system must respond to security incidents in real-time with minimal latency, ensuring swift and efficient threat detection and response.
- **Reliability:** The cybersecurity measures should exhibit a high accuracy rate in identifying and mitigating potential threats, building on the proven track record of successful threat detection algorithms.
- **Compatibility:** The IoT cybersecurity system must be compatible with commonly used operating systems within the Internet of Things (IoT) ecosystem, ensuring seamless integration and protection across diverse IoT devices and platforms.
- **Scalability:** The IoT cybersecurity system must be scalable to accommodate the growing complexity and diversity of the Internet of Things (IoT) ecosystem, ensuring seamless integration and protection across a wide range of IoT devices and platforms.

3.4 User Interface requirements

- **User Authentication:** The system must provide secure user authentication mechanisms, ensuring that only authorized personnel can access and manage the IoT security settings.
- **Real-time Monitoring:** Users should be able to monitor the status and activities of IoT devices in real-time through an intuitive and user-friendly interface.
- **Alerts and Notifications:** The system must promptly alert users about potential security threats or unusual activities, providing detailed notifications for immediate response.
- **Integration with Existing Systems:** The cybersecurity system should seamlessly integrate with existing IT infrastructure and security systems to ensure a cohesive and comprehensive security approach.
- **Scalability:** The system must be scalable to accommodate the growth of IoT devices within the network, ensuring consistent and effective security measures.
- **Secure Remote Access:** Users should be able to securely access and manage the IoT security system remotely, enhancing flexibility and convenience.

3.5 Software requirements

- **Operating System:** Windows 10
- **Python:** Python is a widely used programming language for image processing, machine learning, and data analysis. It provides numerous libraries and frameworks essential for building age estimation models.
- **Jupyter Notebook:** Jupyter Notebook is an interactive computing environment that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It's useful for experimentation and documentation.
- **OpenCV:** OpenCV (Open Source Computer Vision Library) is a critical library for image processing tasks such as image loading, pre-processing, feature extraction, and more.
- **NumPy:** NumPy is essential for numerical operations and data manipulation.
- **Scikit-learn:** Scikit-learn is a fundamental library for machine learning in Python, providing tools for data preprocessing, model development, and evaluation.
- **Matplotlib:** Matplotlib is used for creating data visualizations, plots, and charts to analyze the data and model performance.
- **Pandas:** Pandas facilitates data manipulation and analysis.

3.6 Hardware requirements

- **CPU:** A dual-core processor or higher is recommended to expedite model training.
- **RAM:** A minimum of 8 GB RAM is required for efficient data processing.
- **Hard Disk:** Approximately 10 GB of disk space is required for storing datasets and results.
- **Processor:** Intel's Core i7.
- **Storage:** Minimum of 256 GB or higher.
- **Input Devices:** Keyboard and Mouse.
- **Display:** 1920x1080 or higher.

Chapter 4

Methodology

4.1 Architecture design

Architecture design serves as the backbone of a project, outlining the systematic arrangement of components and their interconnections. In the realm of Intrusion Detection on IoT, this design is pivotal. It delineates the modular structure, specifying how different components interact and the flow of data within the system. With a focus on scalability, it ensures that the Intrusion Detection System can effectively adapt to the dynamic nature of IoT environments. By fostering modularity and clarity, the architecture design not only fortifies the system against potential threats but also facilitates streamlined development and future enhancements.

4.1.1 Proposed framework for attack classification

This Figure 4.1 illustrates framework for attack classification. The proposed methodology and data set make use of machine learning technologies used to identify fraudulent network traffic. We chose UNSW-NB15 data set from the outset and utilised its "UNSW-NB15.csv" files. The "UNSW-NB15.csv" files had a number of problems, including an unbalanced data collection, a data type mismatch with the classification system, and several missing or null values. All of these issues were fixed at the data pre-processing stage. After the data has been cleansed the creation of various clusters based on network levels is required. These clusters are subjected to classification algorithms in order to determine if the network traffic is legitimate or malicious.
Pre-processing is equivalent to data cleansing. It entails deleting characteristics that are redundant or do not provide a high IG as well as introducing derived features, or features that are derived from existing features in the data. We noticed that the data collection contains missing values that might signify severe bias, making it more difficult to handle and analyse the data and lowering its accuracy. Because of this, we employed linear regression, where the model is expected to infer values for a feature based on other data, and that observed model is then used to impute values in samples when the value of that

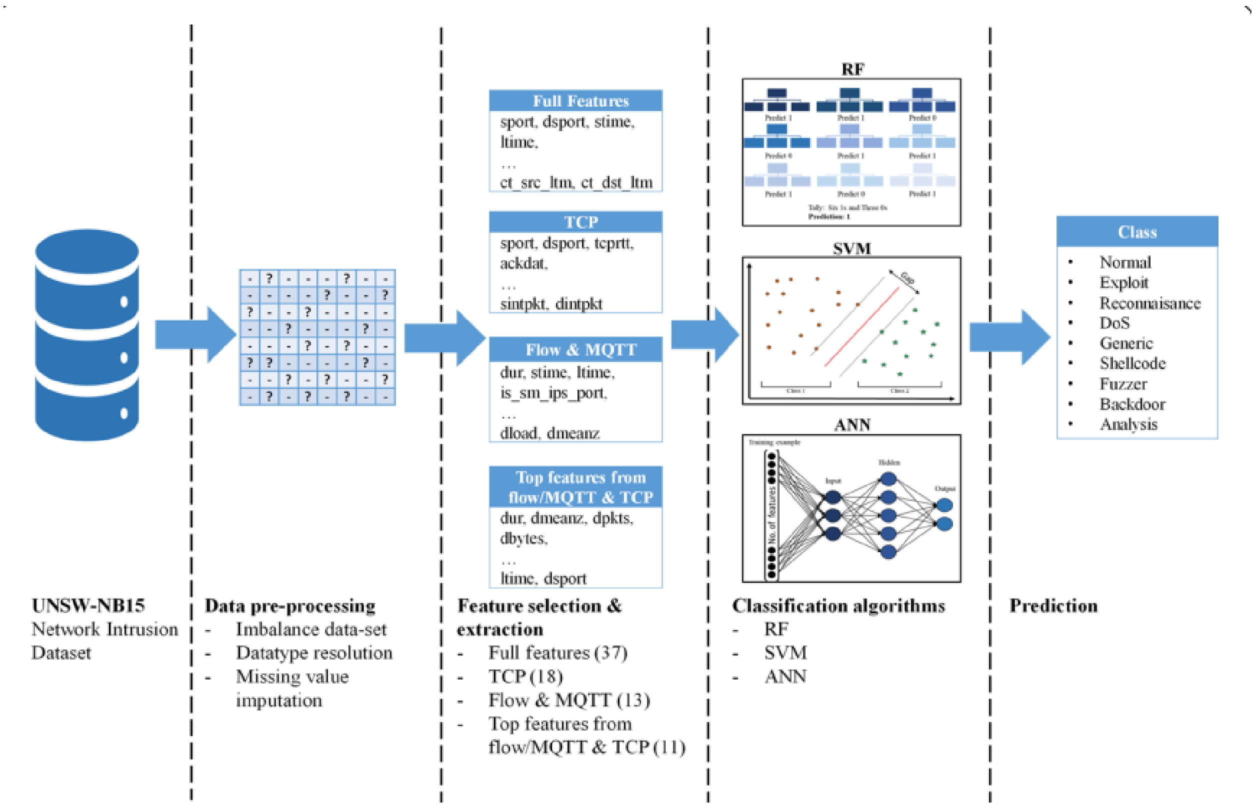

Full Features
sport, dsport, stime, ltime, … ct_src_ltm, ct_dst_ltm
TCP
sport, dsport, tcprtt, ackdat, … sintpkt, dintpkt
Flow & MQTT
dur, stime, ltime, is_sm_ips_port, … dload, dmeanz
Top features from flow/MQTT & TCP
dur, dmeanz, dpkts, dbytes, … ltime, dsport
RF
SVM
ANN
Class
Normal
Exploit
Reconnaisance
DoS
Generic
Shellcode
Fuzzer
Backdoor
Analysis
UNSW-NB15 Network Intrusion Dataset
Data pre-processing
- Imbalance data-set
- Datatype resolution
- Missing value imputation
Feature selection & extraction
- Full features (37)
- TCP (18)
- Flow & MQTT (13)
- Top features from flow/MQTT & TCP (11)
Classification algorithms
- RF
- SVM
- ANN
Prediction

Figure 4.1: Proposed framework for detecting cyber-attacks in IoT networks

feature is absent.

One of the fundamental ideas in ML, feature extraction has a significant impact on prediction accuracy. Our outcomes are significantly influenced by the data attributes used to train ML models. We decrease over-fitting, increase accuracy, and decrease training time by employing feature extraction. We use the feature importance approach with Random Forest in our model. During data type resolution, five characteristics were eliminated, and the last two are labels (binary and multi-class).

4.1.2 Use case diagram

Figure 4.2 presents the use case diagram for the IoT Intrusion Detection System (IDS), portraying the dynamic interactions between the User and the diverse functionalities encapsulated within the IoT Security System. It delineates pivotal operations including the identification of anomalous activities, timely responses to security threats, differentiation between normal and abnormal behaviors, design of an intuitive user interface, provision of comprehensive information, mitigation of false alarms, and the system's inception. Through interconnected relationships, the diagram elucidates the intricate flow of actions and dependencies necessary for establishing a resilient and effective IoT security

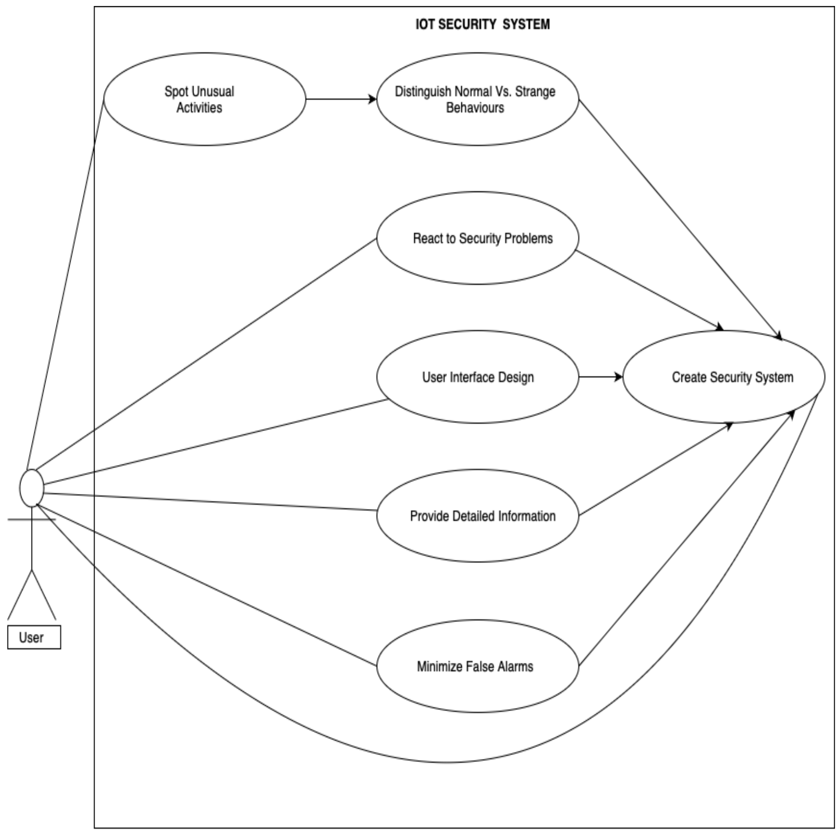


Figure 4.2: Use case diagram

framework. It serves as a visual representation of the system's capabilities, facilitating comprehensive comprehension and effective communication among stakeholders.

4.2 Functional design

The functional design phase is pivotal in translating project requirements into detailed specifications, guiding the development team in implementing the desired features. With a strong emphasis on a user-centric approach, the design ensures an intuitive and user-friendly system. The outlined functionalities, operations, and interactions serve as a precise roadmap for developers. Addressing interoperability, the design defines how the system integrates with external components for enhanced functionality. Additionally, a focus on scalability and flexibility anticipates future growth and ensures adaptability to evolving requirements, laying a robust foundation for project success.

4.2.1 Sequence diagram

In Figure 4.3, the sequence diagram portrays the interaction between the User and the IoT Security System. The process initiates with the User starting the system, activating the IoT Security System. The User then engages in detecting unusual activities, prompting the system to spot these anomalies. Depending on the presence of security problems,

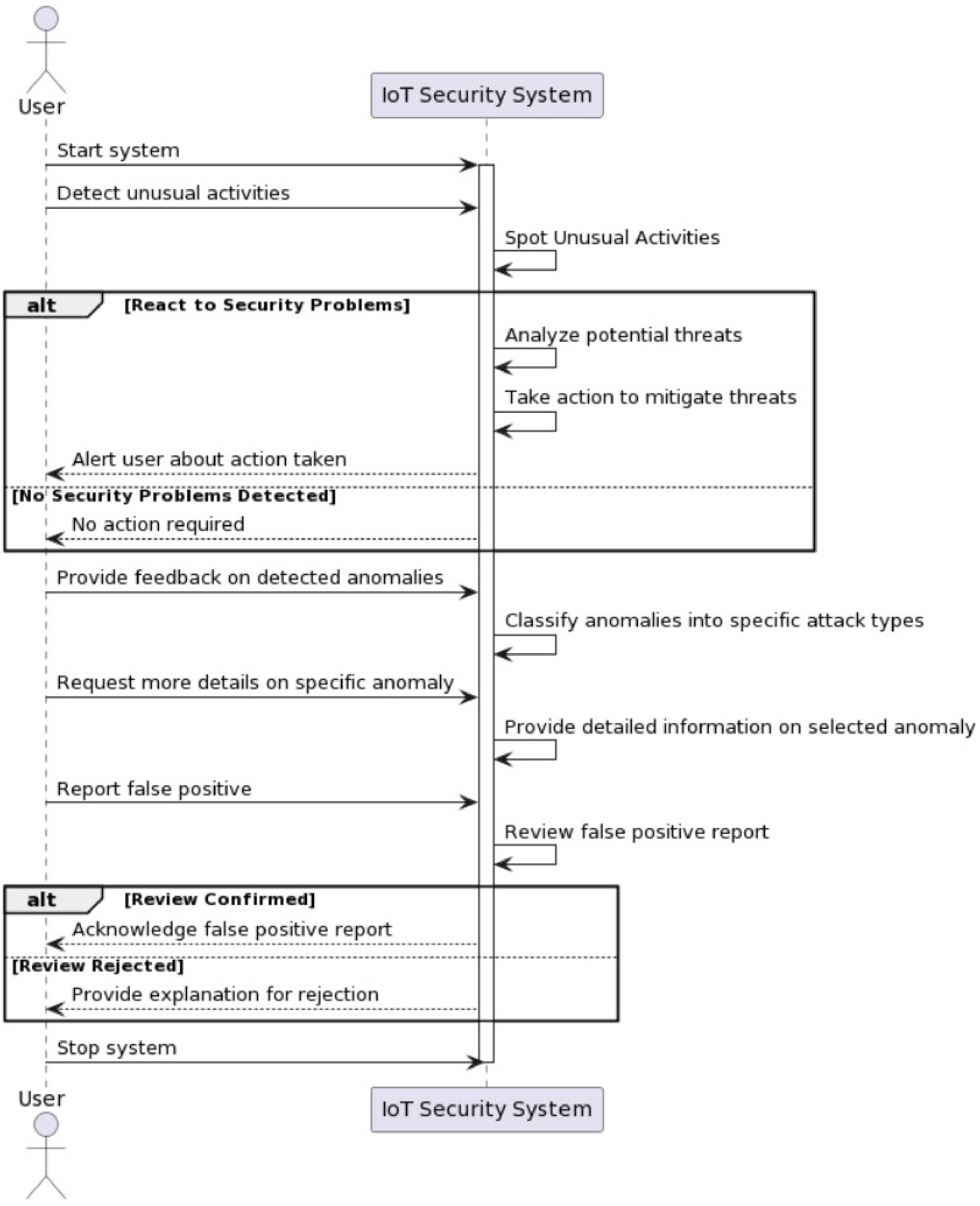

User
IoT Security System
Start system
Detect unusual activities
Spot Unusual Activities
alt
[React to Security Problems]
Analyze potential threats
Take action to mitigate threats
Alert user about action taken
[No Security Problems Detected]
No action required
Provide feedback on detected anomalies
Classify anomalies into specific attack types
Request more details on specific anomaly
Provide detailed information on selected anomaly
Report false positive
Review false positive report
alt
[Review Confirmed]
Acknowledge false positive report
[Review Rejected]
Provide explanation for rejection
Stop system
User
IoT Security System

Figure 4.3: Sequence diagram

the system proceeds to analyze potential threats and take appropriate actions to mitigate them, subsequently alerting the user about the actions taken. In the absence of security problems, the system informs the user that no action is required.

The User provides feedback on detected anomalies, enabling the system to classify them into specific attack types. The User may request more details on a specific anomaly, prompting the system to provide detailed information on the selected anomaly. Furthermore, the User can report false positives, initiating a review process by the system. In case the review confirms the false positive, the system acknowledges the report; otherwise, if the review rejects the false positive, the system provides an explanation for the rejection.

The User concludes the interaction by stopping the system, leading to the deactivation of the IoT Security System. This sequence diagram provides a comprehensive overview of the user-system interactions in the context of detecting and responding to security events in the IoT environment.

4.3 Control flow design

Control Flow Design guides the step-by-step order of actions in a program, making sure everything happens in a logical and organized way. It's like giving the program a clear set of instructions on what to do and when, improving how it works and making it easier to understand.

4.3.1 System flow diagram

In Figure 4.4, we illustrate the gradual development of our model. The process kicks off by cleaning up our data in what we call data preprocessing. Here, we select the important features, tidy up any messy bits, and analyze numbers and graphs to grasp the data better. This leads us to the creation of a feature vector, acting like a summary that captures the crucial aspects of our data.

Following this, we divide our data into two groups for training and testing the model. It's comparable to practicing and then taking a test to assess how well our model functions. The model undergoes training and development, where it learns from the practice data to enhance its capabilities.

This visual representation serves as a guide to understand the entire journey of refining our model. From initial data cleanup to the intricate stages of training and testing, the process ensures a step-by-step improvement of our model's effectiveness. It encapsulates the dynamic evolution of our model, highlighting the essential steps taken to make it

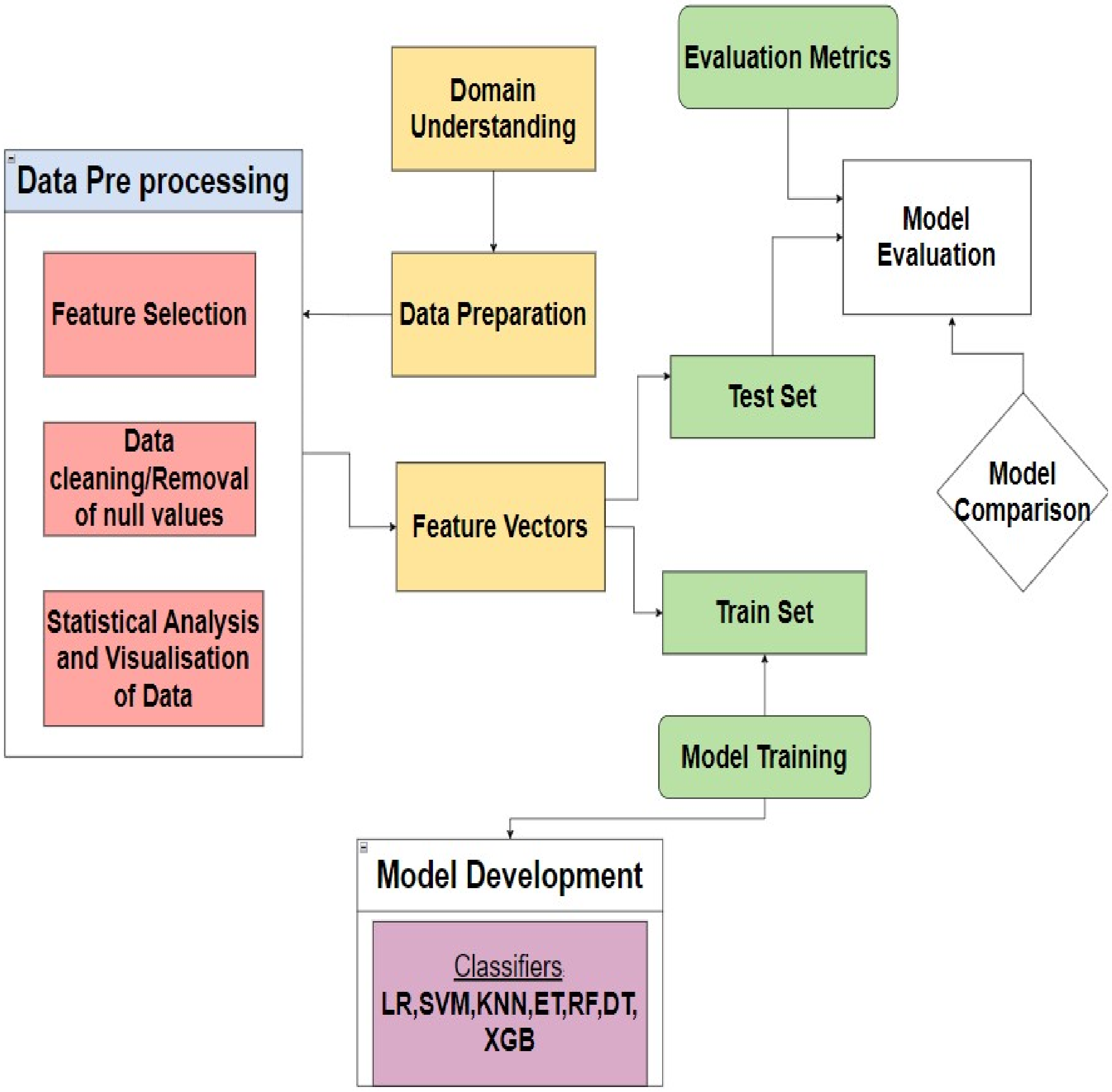


Figure 4.4: System Flow Diagram

more proficient and reliable for real-world applications.

4.3.2 Activity diagram for use cases

The activity diagram in Figure 4.6 intricately details the operational dynamics of the IoT security system. Initiated by the user, the system embarks on a comprehensive sequence of activities designed to fortify its security capabilities. Leveraging advanced machine learning algorithms, the system adeptly detects unusual activities, showcasing its proactive stance in identifying potential security threats. This real-time responsiveness underscores the system's agility and efficiency, ensuring timely and effective mitigation strategies.

User experience is a pivotal focus in this operational sequence. Dedicated steps, such as distinguishing between normal and abnormal behaviors and crafting an intuitive user interface, reflect a commitment to user-centric design principles. By emphasizing accessibility and usability, the IoT security system aims to provide a seamless and engaging experience for its users.

Beyond user-centric considerations, the diagram illustrates the system's intelligence in classifying anomalies. This classification not only contributes to a more informed response strategy but also enhances the system's adaptive capabilities. Additionally, the diagram sheds light on the system's optimization measures, addressing the imperative of minimizing false alarms. By fine-tuning its operations, the system aims to achieve a more accurate and reliable threat detection mechanism, ultimately contributing to its overall effectiveness.

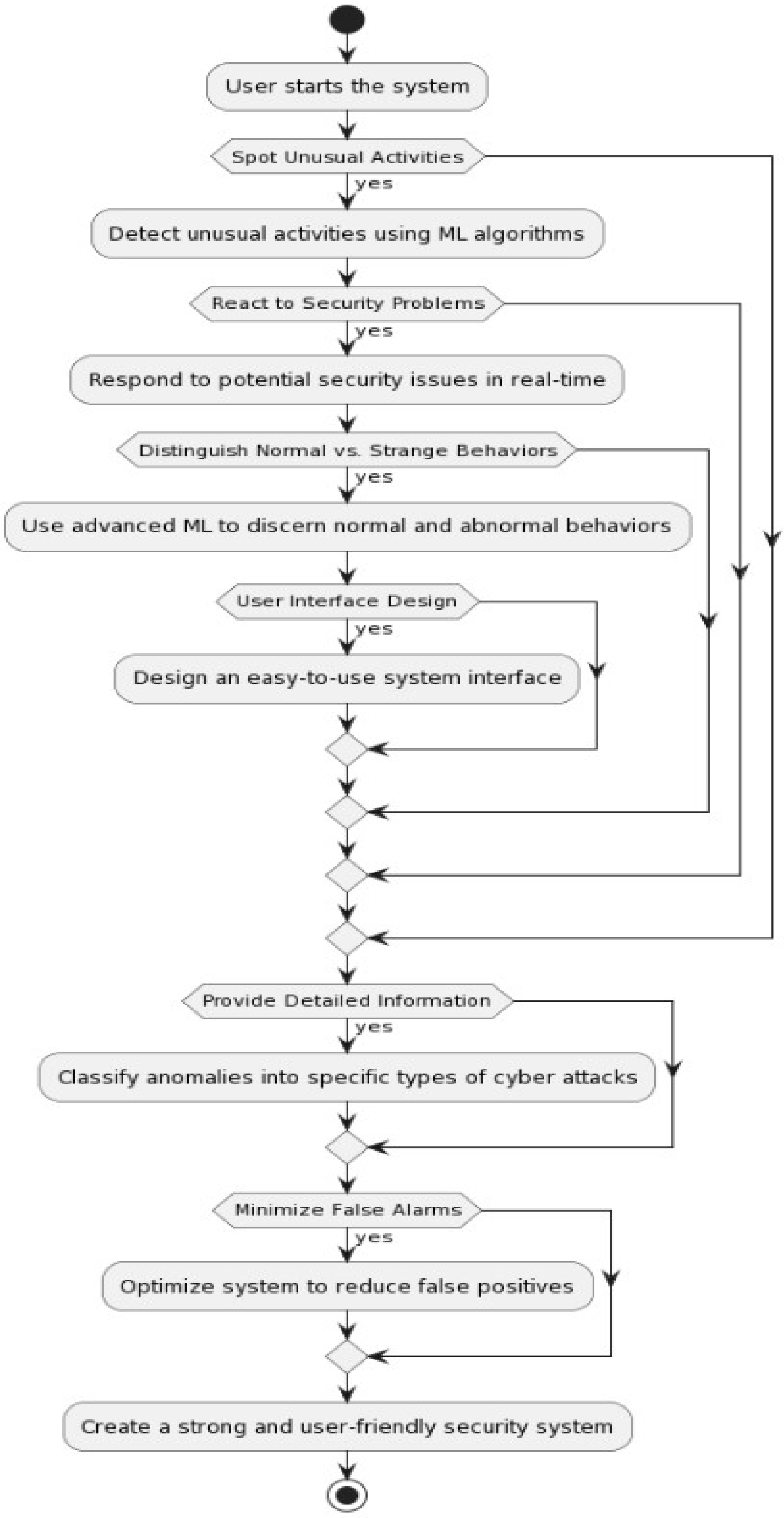

User starts the system
Spot Unusual Activities
yes
Detect unusual activities using ML algorithms
React to Security Problems
yes
Respond to potential security issues in real-time
Distinguish Normal vs. Strange Behaviors
yes
Use advanced ML to discern normal and abnormal behaviors
User Interface Design
yes
Design an easy-to-use system interface
Provide Detailed Information
yes
Classify anomalies into specific types of cyber attacks
Minimize False Alarms
yes
Optimize system to reduce false positives
Create a strong and user-friendly security system

Figure 4.5: Activity Diagram

Chapter 5

Implementation

The basic idea behind machine learning techniques is that they primarily need learning datasets before applying the developed models t s to actual data. The data sets are regarded as being very significant and outdated for identifying current attack types. In this research, the dataset UNSW- NBIS is used to create a customised integrated classification-based model for identifying fraudulent network activity.

The UNSW-NB15 dataset has been introduced as the benchmark dataset in the proposed system. This chapter suggests a variety of machine learning and deep learning methods to improve the network's attack detection rate. The IDS model begins with an analysis of the UNSW-NB15 dataset, followed by data preparation, integrated model proposal, and testing on a test dataset. The performance of the proposed model is then assessed using the test dataset.

5.1 Experimental setup

The cv2 functions provide methods to load the pre-trained models, apply them to images or video frames, and draw bounding boxes around the detected faces. By leveraging cv2's face detection capabilities, you can automate tasks such as facial recognition, emotion analysis, or face tracking in various applications like surveillance, biometrics, or augmented reality.

5.1.1 Prerequsites

The following Python libraries are essential for our project implementation.

- **Sklearn:** Distributed under the 3-Clause BSD license, scikit-learn is a Python module for machine learning built on top of SciPy.

- **Pandas:** Pandas is a Python package that offers quick, adaptable, and expressive data structures intended to make it simple and natural to deal with "relational" or "labelled" data. It seeks to serve as the essential, high-level building block for using Python for actual, useful data analysis.

- **Numpy:** The Python software for scientific computing, Numpy offers a robust N-dimensional array object, complex (broadcasting) functions, tools for integrating C/C++ and Fortran code, and practical linear algebra, Fourier transform, and random number capabilities.
- **Matplotlib:** This extensive Python package allows you to build interactive, animated, and static visualizations.
- **Pickle:** For serializing and de-serializing Python object structures, the pickle module supports binary protocols. Pickling is the process of turning a Python object hierarchy into a byte stream, while unpickling is the process of turning a binary file or object's byte stream back into a Python object hierarchy.
- **Seaborn:** Based on matplotlib, Seaborn is a Python visualization package. It offers an advanced drawing interface for beautiful statistics visuals.
- **Keras:** Based on the machine learning API built in Python.

5.2 Datasets

The dataset "UNSW NB15.csv" encompasses 49 features alongside a class label, constituting a portion of the larger original dataset known as UNSW NB1S. Descriptions of these characteristics are provided within the "UNSW-NB15 features.csv" file. For binary classification tasks, the dataset "bin data.csv" is utilized, while for multi-class classification tasks, "multidata.csv" serves as the dataset for computation. Table 5.1 shows the small sample of mthe dataset.

Table 5.1: Dataset

dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload
0	0.000011	2	0	496	0	90909.0902	254	1.8036e+08
1	0.000008	2	0	1762	0	125000.0003	254	8.8100e+08
2	0.000005	2	0	1068	0	200000.0051	254	8.5440e+08
3	0.000006	2	0	900	0	166666.6608	254	6.0000e+08
4	0.000010	2	0	2126	0	100000.0025	254	8.5040e+08
5	0.000012	3	0	748	0	80000.0019	254	6.7920e+08
6	0.000007	1	0	992	0	200000.0051	254	1.4080e+09
7	0.000009	1	0	394	0	90909.0902	254	3.4848e+08

5.3 Dataset Description

The UNSW-NB15 dataset was crafted using the IXIA PerfectStorm tool, aiming to simulate real-world cyber-attack scenarios. With detailed features categorized into nine modern attack types, it provides a comprehensive representation of both normal network activities and malicious intrusions. The training dataset comprises 175,341 instances, while the testing dataset contains 82,332 instances, facilitating robust model training and evaluation.

Table 5.2: Description of features (Part 1)

No.	Name	Type	Description
1	srcip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol
6	state	nominal	Indicates the state and its dependent protocol
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	Service, such as http, ftp, smtp
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count
18	Dpkts	integer	Destination to source packet count
19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number
23	smeansz	integer	Mean of the flow packet size transmitted by the source
24	dmeansz	integer	Mean of the flow packet size transmitted by the destination
25	trans_depth	integer	Represents the pipelined depth into the connection
26	res_bdy_len	integer	Actual uncompressed content size of the data transferred

Table 5.3: Description of features (Part 2)

No.	Name	Type	Description
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	Record start time
30	Ltime	Timestamp	Record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time
34	synack	Float	TCP connection setup time, the time between the SYN and the SYN-ACK packets
35	ackdat	Float	TCP connection setup time, the time between the SYN-ACK and the ACK packets
36	is_sm_ips_ports	Binary	If source and destination IP addresses match
37	ct_state_ttl	Integer	No. for each state according to specific rules
38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service
39	is_ftp_login	Binary	If the ftp session is accessed by user and password
40	ct_ftp_cmd	integer	No. of flows that has a command in ftp session
41	ct_srv_src	integer	No. of connections that contain the same service (source)
42	ct_srv_dst	integer	No. of connections that contain the same service (destination)
43	ct_dst_ltm	integer	No. of connections of the same destination address
44	ct_src_ltm	integer	No. of connections of the same source address
45	ct_src_dport_ltm	integer	No. of connections of the same source address and destination port
46	ct_dst_sport_ltm	integer	No. of connections of the same destination address and source port
47	ct_dst_src_ltm	integer	No. of connections of the same source and destination
48	attack_cat	nominal	The name of each attack category
49	Label	binary	0 for normal and 1 for attack records

The dataset comprises 49 features representing 9 attack categories, including Analysis, Fuzzers, Backdoors, DoS, Exploits, Reconnaissance, Generic, Shellcode, and Worms.

Table 5.4: Descriptions of attacks

Attack	**Description**
Analysis	Analysis Used to penetrate web applications through emails ming spam, web scripts e.g. using HTML. files, and port scans. A technique used to bypass authentication process of system which allow remote access and lead to an unauthorized access to a computer or device, which gives attacker an opportunity to issue commands remotely.
Backdoor	An attack, through which attacker tries to bring down the services of a computer network (or server) or by making resources unavailable for authorised user requests.
DoS	An attack, through which attacker tries to bring down the services of a computer network (or server) or by making resources unavailable for authorised user requests.
Exploit	Sequence of steps taken by an attacker in order to take advantage of any vulnerability, glitch or bug present in a system or network.
Fuzzers	Activity through which attacker tries to find security vulnerability is system, program, network, or operating system by flooding it with random data in order to crash it.
Generic	It is a type of activity in which an attacker does not bother about the cryptographic implementation of any primitives and runs the attack.
Reconnaissance	Process of gathering information related to computer system or network for evading security controls.
Shellcode	An attacker writes code and injects it to any application which triggers command shell in order to take control of compromised machine.
Worm	Attackers try to replicate their functional copies and use system vulnerability or any social engineering techniques to enter the system.

Table 5.4 gives the highlights of all 9 types of attacks present in the UNSW_NB-15 dataset.

5.4 Data Preprocessing

- The dataset had 175341 rows and 49 characteristics features.
- The dataset has 82333 rows and 49 characteristics features after eliminating null values.

- In the pre-processing step, "Label Encoding" was applied to convert categorical labels into numerical values.
- This technique assigns a unique numerical value to each category or subcategory for better compatibility with machine learning algorithms.
- For the main attack category, 'Normal' was mapped to 0 and 'attack' was mapped to 1
- Similarly, for the attack subcategory, 'Normal' was mapped to 0, 'Analysis' was mapped to 1, 'Backdoor' was mapped to 2, 'DoS' was mapped to 3, 'Exploits' was mapped to 4, 'Fuzzers' was mapped to 5, 'Generic' was mapped to 6, 'Reconnaissance' was mapped to 7, 'Shellcode' was mapped to 8 and 'Worms' was mapped to 9.

5.5 Dataset Splitting

- Binary data is split in 50%, 60%, 70%, 80% for training and 50%, 40%, 30%, 20% for testing respectively.
  X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.70,random_state =50)
- Randomly splitting the multi_data in 50%, 60%, 70%, 80% for testing and 50%, 40%, 30%, 20% for training respectively.
  X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.80,random_state =50)

5.6 Classification Models

5.6.1 Machine Learning Models

Several machine learning models are applied to analyze the performance of the proposed feature engineering approach. Seven well-known machine learning models, which are reported to show good performance for tasks similar to genetic disorder prediction, are utilized. The main focused algorithms are listed below:

- **Logistic Regression (LR):** LR is a statistical learning method for classifications, often used for multilabel classification tasks. It predicts dependent categorical variables using independent variables and maps predicted outputs to probabilities using the Sigmoid function.

- **K-Nearest Neighbors (KNN):** KNN is a non-parametric supervised learning technique that predicts the class of data based on the similarity of nearest neighbors. It groups data points based on their proximity, with classification performed using distance metrics such as Euclidean distance.

- **Decision Tree Classifier (DTC)**: DTC is a supervised learning algorithm used for classification tasks. It constructs a tree-like structure where inner nodes represent data attributes and leaf nodes contain outcome labels. The algorithm aims to minimize generalization error by selecting optimal decision trees based on measures like information gain and Gini index.

- **Random Forest Classifier (RFC):** RFC is an ensemble learning technique based on multiple decision trees. It aggregates predictions from these trees using majority voting, reducing overfitting and improving classification performance compared to individual classifiers.

- **Extra Trees Classifier (ETC):** ETC is another ensemble-based, bagged decision tree technique similar to RFC. It reduces model variance by employing a random split selection of values and a meta estimator that fits randomized decision trees on sample datasets, leading to improved accuracy and reduced overfitting.

- **XGBoost (XGB):** XGB is a flexible and efficient classification algorithm based on boosting techniques. It employs parallel gradient boosting tree technique to solve classification problems and uses regularization to reduce overfitting, leading to improved predictive performance.

- **Support Vector Classifier (SVC):** SVC is a supervised learning algorithm primarily used for classification tasks. It finds the optimal hyperplane that separates input data points into different classes by maximizing the margin between them, with predictions made using a hypothesis function based on support vectors.

Chapter 6

Results and Discussion

6.1 Binary Classification Results:

A classification-based machine learning model's classification report is one of the performance assessment measures. The model's accuracy, recall, F1 score, and support are shown. It also helps us understand the overall effectiveness of our trained model. Python includes libraries to plot data, and Matplotlib is the most well-known of them. An effective tool that produces a far more illustratable plot is Seaborn, which also makes use of Matplotlib as its foundation.

After the evaluation of accuracies for binary classification, maximum accuracy was found with 70% training set and 30% testing set and the results are as follows:

1. Logistic Regression:
   Evaluation metrics for Logistic Regression:
   Mean Absolute Error: 0.2277281836400073
   Mean Squared Error: 0.2277281836400073
   Root Mean Squared Error: 0.4772087422082786
   Accuracy: 77.22718163599927

Table 6.1: Classification report for logistic regression

Class	**Precision**	**Recall**	**F1-score**	**Support**
abnormal	0.82	0.63	0.71	7418
normal	0.75	0.89	0.81	9049
Accuracy			0.77	16467
Macro avg	0.78	0.76	0.76	16467
Weighted avg	0.78	0.77	0.77	16467

Table 6.1 shows the binary classification report for the Logistic Regression

2. K-Nearest Neighbors Classifier:
   Evaluation metrics for K-Nearest Neighbors Classifier:
   Mean Absolute Error: 0.11738628772696909
   Mean Squared Error: 0.11738628772696909
   Root Mean Squared Error: 0.3426168234733506
   Accuracy: 88.26137122730309

Table 6.2: Classification report for k-nearest neighbors classifier

Class	Precision	Recall	F1-score	Support
abnormal	0.85	0.90	0.87	7418
normal	0.91	0.87	0.89	9049
Accuracy			0.88	16467
Macro avg	0.88	0.88	0.88	16467
Weighted avg	0.88	0.88	0.88	16467

   Table 6.2 shows the binary classification report for the K-Nearest Neighbors Classifier

3. Random Forest Classifier:
   Evaluation metrics for Random Forest Classifier:
   Mean Absolute Error: 0.02081496325985304
   Mean Squared Error: 0.030814963259804
   Root Mean Squared Error: 0.2388658956295369
   Accuracy: 92.89531212047606

Table 6.3: Classification report for random forest classifier

abnormal	0.89	0.93	0.94	7618
normal	0.96	0.91	0.93	9059
Accuracy			0.92	18467
Macro avg	0.94	0.92	0.93	46467
Weighted avg	0.88	0.88	0.88	46467

   Table 6.3 shows the binary classification report for the random forest classifier

4. Extra Trees Classifier:
   Evaluation metrics for Extra Trees Classifier:
   Mean Absolute Error: 0.04081496325985304
   Mean Squared Error: 0.0508755963259804
   Root Mean Squared Error: 0.02387542956295369
   Accuracy: 92.5431139179014
   Table 6.4 shows the binary classification report for Extra Trees Classifier

Table 6.4: Classification report for extra trees classifier

Class	Precision	Recall	F1-score	Support
abnormal	0.90	0.91	0.89	7513
normal	0.95	0.95	0.95	9086
Accuracy			0.94	15267
Macro avg	0.93	0.94	0.89	15267
Weighted avg	0.93	0.94	0.89	15267

5. XGBoost Classifier:
   Evaluation metrics for XGBoost Classifier:
   Mean Absolute Error: 0.012121496325985304
   Mean Squared Error: 0.01250814963259804
   Root Mean Squared Error: 0.3218658956295369
   Accuracy: 92.62812727714355

Table 6.5: Classification report for xgboost classifier

Class	Precision	Recall	F1-score	Support
abnormal	0.85	0.92	0.93	7418
normal	0.91	0.93	0.96	9049
Accuracy			0.92	16467
Macro avg	0.80	0.91	0.93	16467
Weighted avg	0.89	0.84	0.94	16467

   Table 6.5 shows the binary classification report for XGBoost Classifier

6. Decision Tree Classifier:
   Evaluation metrics for Decision Tree Classifier:
   Mean Absolute Error: 0.542478897188316
   Mean Squared Error: 0.542478897188316
   Root Mean Squared Error: 0.7365316674714781
   Accuracy: 92.0937575904785

Table 6.6: Classification report for decision tree classifier

Class	Precision	Recall	F1-score	Support
abnormal	0.45	0.99	0.62	7418
normal	0.68	0.02	0.05	9049
Accuracy			0.46	16467
Macro avg	0.57	0.51	0.33	16467
Weighted avg	0.58	0.46	0.31	16467

   Table 6.6 shows the binary classification report for the Decision Tree Classifier

7. Support Vector Machine:
   Evaluation metrics for Support Vector Machine:
   Mean Absolute Error: 0.13
   Mean Squared Error: 0.13
   Root Mean Squared Error: 0.36055512754639896
   Accuracy: 89.12545454545

Table 6.7: Classification report for support vector machine

Class	Precision	Recall	F1-score	Support
abnormal	0.83	0.91	0.87	93
normal	0.92	0.83	0.87	107
Accuracy			0.87	200
Macro avg	0.87	0.87	0.87	200
Weighted avg	0.87	0.87	0.87	200

Table 6.7 shows the binary classification report for Support Vector Machine

Table 6.8: Accuracy of all models implemented for binary class

Algorithm	Accuracy (%)
Logistic Regression	76.52
Support Vector Machine	87.37
K-Nearest Neighbors	85.85
Random Forest Classifier	92.73
Decision Tree Classifier	91.85
Extra Trees Classifier	92.51
XGBoost Classifier	91.73

The Table 6.8 presents the accuracy percentages of various machine learning algorithms applied to binary classification tasks. Random Forest Classifier achieved the highest accuracy of 92.73%, followed closely by Extra Trees Classifier at 92.51%. Logistic Regression exhibited comparatively lower accuracy at 76.52%.

6.2 Multiclass Classification Results:

The classification report serves as a crucial performance assessment tool for multiclass machine learning models, offering insights into metrics like accuracy, recall, F1 score, and support for each class. It provides a comprehensive overview of the model's effectiveness across multiple classes, aiding in understanding its overall performance. Python offers powerful libraries for data visualization, including Matplotlib, widely used for plotting data. However, Seaborn stands out for its enhanced plotting capabilities, leveraging Matplotlib as its foundation, to generate more visually appealing and informative plots.

After the evaluation of accuracies for multi-class classification, maximum accuracy was found with 70% training set and 30% testing set and the results are as follows:

1. Logistic Regression:
   Evaluation metrics for Logistic Regression:
   Mean Absolute Error: 2.1839436448654888
   Mean Squared Error: 8.795226817270905
   Root Mean Squared Error: 2.9656747659294846
   Accuracy: 36.604323536555746

   Table 6.9: Classification report for logistic regression

Class	Precision	Recall	F1-score	Support
Normal	1.00	0.10	0.18	7418
Analysis	0.00	0.00	0.00	131
Backdoor	0.00	0.00	0.00	117
DoS	0.00	0.00	0.00	786
Exploits	0.19	0.74	0.30	2275
Fuzzers	0.00	0.00	0.00	1212
Generic	0.54	0.97	0.70	3723
Reconnaissance	0.00	0.00	0.00	723
Shellcode	0.00	0.00	0.00	75
Worms	0.00	0.00	0.00	7
Accuracy			0.37	16467
Macro avg	0.17	0.18	0.12	16467
Weighted avg	0.60	0.37	0.28	16467

   Table 6.9 presents the classification report for Logistic Regression, indicating mixed performance across classes with varying precision, recall, and F1-scores. While achieving high precision for certain classes like “Normal” and “Generic”, the model exhibits low recall for others such as ”Analysis” and ”Fuzzers.” The overall accuracy of 37% highlights the need for improvements to enhance classification effectiveness.

2. Random Forest Classifier:
   Evaluation metrics for Random Forest Classifier:
   Mean Absolute Error: 0.19493532519584625
   Mean Squared Error: 0.45375599684216916
   Root Mean Squared Error: 0.6736141305244191
   Accuracy: 82.35365557444742

Table 6.10: Classification report for random forest classifier

Class	Precision	Recall	F1-score	Support
Normal	1.00	1.00	1.00	7418
Analysis	0.08	0.02	0.03	131
Backdoor	0.08	0.01	0.02	117
DoS	0.42	0.52	0.47	786
Exploits	0.69	0.71	0.70	2275
Fuzzers	0.70	0.77	0.73	1212
Generic	0.99	0.97	0.98	3723
Reconnaissance	0.91	0.79	0.85	723
Shellcode	0.39	0.35	0.37	75
Worms	0.36	0.57	0.44	7
Accuracy			0.89	16467
Macro avg	0.56	0.57	0.56	16467
Weighted avg	0.88	0.89	0.89	16467

Table 6.10 depicts the classification report for the Random Forest Classifier, showcasing robust performance across various classes with high precision, recall, and F1-scores. Notably, the model achieves exceptional scores for classes like "Normal" and "Generic," indicating accurate classification.

3. K-Nearest Neighbors:
   Evaluation metrics for K-Nearest Neighbors:
   Mean Absolute Error: 0.494321977287909
   Mean Squared Error: 0.879546972733346
   Root Mean Squared Error: 0.9216765512639293
   Accuracy: 88.65610007894577

   Table 6.11 presents the classification report for the K-Nearest Neighborsr, indicating significant performance disparities across classes with varying precision, recall, and F1-scores. While achieving a relatively high recall for the "Normal" class, the model demonstrates poor performance for other classes, as evidenced by low precision and F1-scores. With an overall accuracy of 76%, the model's effectiveness in accurately classifying instances is limited, suggesting a need for improvements or alternative methodologies to enhance classification accuracy.

Table 6.11: Classification report for k-nearest neighbors

Class	Precision	Recall	F1-score	Support
Normal	0.45	0.95	0.61	7418
Analysis	0.00	0.00	0.00	131
Backdoor	0.00	0.00	0.00	117
DoS	0.00	0.00	0.00	786
Exploits	0.66	0.05	0.09	2275
Fuzzers	0.24	0.05	0.08	1212
Generic	0.01	0.00	0.00	3723
Reconnaissance	0.00	0.00	0.00	723
Shellcode	0.00	0.00	0.00	75
Worms	0.00	0.00	0.00	7
Accuracy			0.44	16467
Macro avg	0.14	0.10	0.08	16467
Weighted avg	0.31	0.44	0.29	16467

4. Support Vector Machine:
   Evaluation metrics for Support Vector Machine:
   Mean Absolute Error: 1.494321977287909
   Mean Squared Error: 7.133964899495962
   Root Mean Squared Error: 2.6709483146433146
   Accuracy: 63.02908847999028

Table 6.12: Classification report for support vector machine

Class	Precision	Recall	F1-score	Support
Normal	0.65	0.91	0.76	7418
Analysis	0.00	0.00	0.00	131
Backdoor	0.00	0.00	0.00	117
DoS	0.00	0.00	0.00	786
Exploits	0.32	0.04	0.07	2275
Fuzzers	0.00	0.00	0.00	1212
Generic	0.61	0.96	0.74	3723
Reconnaissance	0.00	0.00	0.00	723
Shellcode	0.00	0.00	0.00	75
Worms	0.00	0.00	0.00	7
Accuracy			0.63	16467
Macro avg	0.16	0.19	0.16	16467
Weighted avg	0.47	0.63	0.52	16467

The Table 6.12 showcases the classification report for the Support Vector Machine (SVM), revealing varied performance across classes with notable precision, recall, and F1-scores discrepancies. While achieving relatively high precision for "Normal" and "Generic" classes, the model demonstrates low recall and F1-scores for several others, indicating challenges in accurately classifying instances. The overall accuracy of 63% suggests moderate performance, implying potential room for improvement in classification accuracy and effectiveness.

5. Decision Tree Classifier:
   Evaluation metrics for Decision Tree Classifier:
   Mean Absolute Error: 0.2095099289488067
   Mean Squared Error: 0.48363393453573816
   Root Mean Squared Error: 0.6954379444175721
   Accuracy: 80.90842846733058

Table 6.13: Classification report for decision tree classifier

Class	Precision	Recall	F1-score	Support
Normal	1.00	1.00	1.00	7418
Analysis	0.08	0.02	0.03	131
Backdoor	0.13	0.02	0.03	117
DoS	0.40	0.54	0.46	786
Exploits	0.67	0.67	0.67	2275
Fuzzers	0.71	0.73	0.72	1212
Generic	0.98	0.98	0.98	3723
Reconnaissance	0.88	0.77	0.82	723
Shellcode	0.31	0.37	0.34	75
Worms	0.36	0.57	0.44	7
Accuracy			0.88	16467
Macro avg	0.55	0.57	0.55	16467
Weighted avg	0.88	0.88	0.88	16467

Table 6.13 displays the classification report for the Decision Tree Classifier, showcasing strong performance across various classes with high precision, recall, and F1-scores. Notably, the model achieves exceptional scores for classes like "Normal" and "Generic," indicating accurate classification. With an overall accuracy of 80.9%, the model demonstrates effective classification capabilities, suggesting its suitability for the task at hand.

6. Extra Trees Classifier:
   Evaluation metrics for Extra Trees Classifier:
   Mean Absolute Error: 0.19548187283658225
   Mean Squared Error: 0.4527236290763345
   Root Mean Squared Error: 0.6728474040050496
   Accuracy: 81.97716784066068

Table 6.14 provides the classification report for the Extra Trees Classifier, showcasing robust performance across various classes with high precision, recall, and F1-scores. Notably, the model achieves exceptional scores for classes like "Normal" and "Generic," indicating accurate classification. With an overall accuracy of 82%, the model demonstrates effective classification capabilities, suggesting its suitability for the task at hand.

Table 6.14: Classification report for extra trees classifier

Class	**Precision**	**Recall**	**F1-score**	**Support**
Normal	1.00	1.00	1.00	7418
Analysis	0.08	0.02	0.03	131
Backdoor	0.07	0.01	0.02	117
DoS	0.42	0.53	0.47	786
Exploits	0.69	0.71	0.70	2275
Fuzzers	0.71	0.76	0.74	1212
Generic	0.99	0.98	0.98	3723
Reconnaissance	0.91	0.79	0.84	723
Shellcode	0.38	0.41	0.40	75
Worms	0.40	0.57	0.47	7
Accuracy			0.89	16467
Macro avg	0.57	0.58	0.56	16467
Weighted avg	0.88	0.89	0.88	16467

7. XGBoost Classifier:
   Evaluation metrics for XGBoost Classifier:
   Mean Absolute Error: 0.18497601263132324
   Mean Squared Error: 0.42084168336673344
   Root Mean Squared Error: 0.6487231176447572
   Accuracy: 82.11076026232693

   Table 6.15 illustrates the classification report for the XGBoost Classifier, demonstrating robust performance across various classes with high precision, recall, and F1-scores. Notably, the model achieves exceptional scores for classes like "Normal" and "Generic," indicating accurate classification. With an overall accuracy of 82%, the model exhibits effective classification capabilities, highlighting its suitability for the task at hand.

Table 6.15: Classification report for xgboost classifier

Class	Precision	Recall	F1-score	Support
Normal	1.00	1.00	1.00	7418
Analysis	0.00	0.00	0.00	131
Backdoor	0.11	0.01	0.02	117
DoS	0.44	0.55	0.49	786
Exploits	0.71	0.73	0.72	2275
Fuzzers	0.69	0.77	0.73	1212
Generic	0.99	0.97	0.98	3723
Reconnaissance	0.93	0.79	0.86	723
Shellcode	0.49	0.39	0.43	75
Worms	0.60	0.43	0.50	7
Accuracy			0.89	16467
Macro avg	0.60	0.56	0.57	16467
Weighted avg	0.89	0.89	0.89	16467

Table 6.16 presents the classification report for multi-class classification, showcasing varying precision, recall, and F1-scores across different classes. Notably, the model achieves high precision and recall for classes like "Analysis," "DoS," "Exploits," and "Generic," indicating accurate classification. However, classes like "Backdoor," "Fuzzers," "Reconnaissance," and "Shellcode" have poor performance, as evidenced by low precision, recall, and F1-scores. With an overall accuracy of 82%, the model demonstrates effective classification capabilities, though improvements may be needed for certain classes to enhance overall performance.

Table 6.16: Classification report for multi-class

Class	Precision	Recall	F1-score	Support
Analysis	1.00	1.00	1.00	166
Backdoor	0.00	0.00	0.00	32
DoS	0.95	0.96	0.96	521
Exploits	1.00	0.99	1.00	4900
Fuzzers	0.00	0.00	0.00	508
Generic	0.95	0.99	0.97	11839
Normal	0.91	1.00	0.96	5855
Reconnaissance	0.00	0.00	0.00	502
Shellcode	0.00	0.00	0.00	29
Accuracy			0.95	24352
Macro avg	0.54	0.55	0.54	24352
Weighted avg	0.91	0.95	0.93	24352

Table 6.17 summarizes the accuracy of all models implemented for multi-class classification. The Linear Support Vector Machine, K-Neighbors Classifier, Random

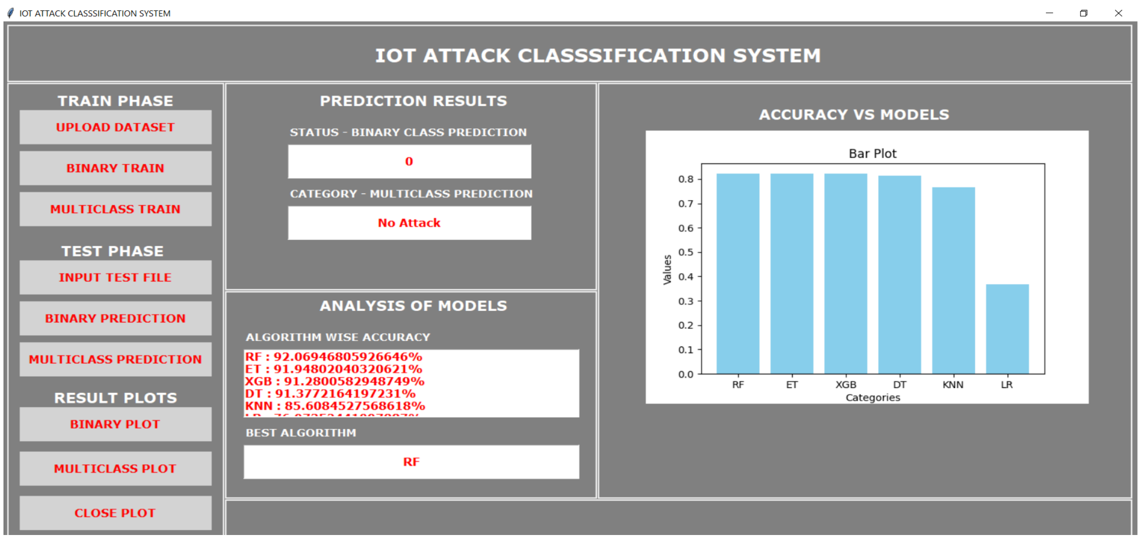
Forest Classifier, and Decision Tree Classifier all achieve high accuracies ranging from 63.029% to 82.356%. Notably, the Random Forest Classifier outperforms others with an accuracy of 82.356%. However, the Logistic Regression exhibit significantly lower accuracy of 36.604%.

Table 6.17: Accuracy of all models implemented for multi-class

Algorithm	Accuracy (%)
Logistic Regression	36.604
Linear Support Vector Machine	63.029
K-Nearest Neighbors	76.876
Random Forest Classifier	82.356
Decision Tree Classifier	81.002
Extra Trees Classifier	81.977
XGBoost Classifier	82.131

6.3 User-Interface

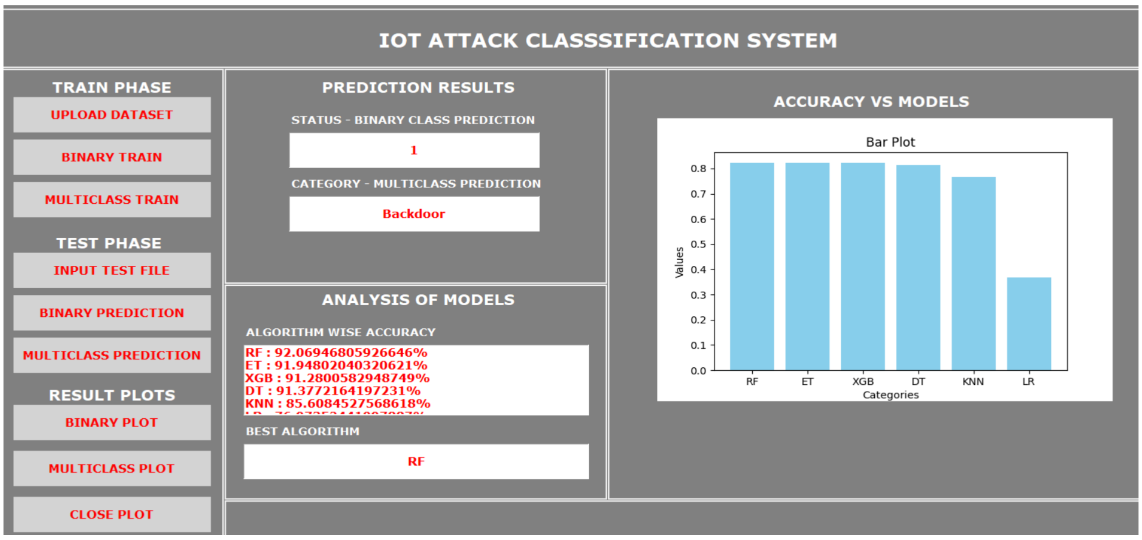
The developed web page functions as an interactive platform for IoT attack classification. Users can upload datasets for training purposes, after which the system evaluates the performance of multiple machine learning models. This evaluation process is showcased in the "Analysis of Models" section, where the accuracy of each model is displayed, offering insights into their effectiveness in classifying IoT attacks. Here Random Forest Classifier has the highest accuracy in terms of classifying the attacks, indicating its potential as a robust solution for IoT security challenges.



STATUS - BINARY CLASS PREDICTION
0
CATEGORY - MULTICLASS PREDICTION
No Attack
ALGORITHM WISE ACCURACY
RF : 92.06946805926646%
ET : 91.94802040320621%
XGB : 91.2800582948749%
DT : 91.3772164197231%
KNN : 85.6084527568618%
BEST ALGORITHM
RF

Figure 6.1: Shows the System is in the Normal State

Figure 6.1 vividly illustrates the system's operational state, highlighting its readiness to detect potential threats. During the testing phase, users can input specific test cases to assess whether an attack has occurred. If the system detects no malicious activity, the status is reported as 0, along with a corresponding confirmation of "no attack" in the category field. Conversely, if an attack is identified, the status changes to 1, indicating the presence of malicious activity, and the associated attack category is provided for further analysis. This seamless interaction with the system empowers users to swiftly identify and respond to potential security breaches, enhancing the overall resilience of IoT infrastructures.



STATUS - BINARY CLASS PREDICTION: 1

CATEGORY - MULTICLASS PREDICTION: Backdoor

ALGORITHM WISE ACCURACY

```
RF : 92.06946805926646%
ET : 91.94802040320621%
XGB : 91.2800582948749%
DT : 91.37721641972 31%
KNN : 85.6084527568618%
```

BEST ALGORITHM: RF

Figure 6.2: Indicates the Malicious Attack has Occurred

Figure 6.2 shows that the malicious attack has occurred and the attack category belongs to Backdoor.

Additionally, the web page features a graphical representation in the form of a bar plot, illustrating the comparative accuracy of different machine learning models. This visualization offers a clear and intuitive understanding of each model's performance, allowing users to identify the most effective approach for IoT attack classification.

Chapter 7

Conclusion and Future work

In this study, we built a robust model to analyze the UNSW-NB15 dataset related to IoT, covering various attacks. We applied both traditional machine learning algorithms and a Deep Learning approach to compare their effectiveness. The evaluation used practical metrics like accuracy, precision, recall, F-measure, and support for both binary and multi-class classifications, the latter involving nine parameters. Our focus is on detecting malicious attacks in network traffic using these techniques, demonstrating adaptability to novel records even during training. While our models showed results comparable to traditional methods, the Multi-Layer Perceptron model achieved superior performance, especially with balanced data. Additionally, the Random Forest classifier performed well with binary data.

For future work, we propose a comparison of data mining techniques with selective feedback approaches to better capture the nuanced behavior of intrusions and normal activities. The vision includes developing a user-friendly interface for individuals and businesses. Further improvements may involve exploring different types of neural networks, such as LSTM, and diverse architectures within the neural network landscape.

References

[1] Mujaheed Abdullahi, Yahia Baashar, Hitham Alhussian, Ayed Alwadain, Norshakirah Aziz, Luiz Fernando Capretz and Said Jadid Abdulkadir, https://doi:10.3390/electronics11020198, "Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review". Electronics, vol. 11, no. 2, p. 198, Jan. 2022.

[2] HasanAlkahtani and TheyaznHHAldhyani. 2021. "Botnet Attack Detection by Using CNN-LSTM Modelfor Internet of Things Applications". Security and Communication Networks 2021 (2021)

[3] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4815–4830, Jun. 2019, doi: 10.1109/jiot.2018.2871719.

[4] K. A. P. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque, "Internet of Things: A survey on machine learning-based intrusion detection approaches," Computer Networks, vol. 151, pp. 147–157, Mar. 2019, doi: 10.1016/j.comnet.2019.01.023.

[5] N. Moustafa, E. Adi, B. Turnbull, and J. Hu, "A New Threat Intelligence Scheme for Safeguarding Industry 4.0 Systems," IEEE Access, vol. 6, pp. 32910–32924, 2018, doi: 10.1109/access.2018.2844794.

[6] Zhiyan Chen, Jinxin Liu, Yu Shen, Murat Simsek, Burak Kantarci, Hussein T. Mouftah, and Petar Djukic, "Machine Learning-Enabled IoT Security: Open Issues and Challenges Under Advanced Persistent Threats," ACM Computing Surveys, vol. 55, no. 5, pp. 1–37, Dec. 2022, doi: 10.1145/3530812.