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 degree

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE & ENGINEERING

Submitted by

Joyline Rencita Dsouza	**4SO20CS073**
Melanie Crystal Miranda	**4SO20CS085**
Natasha Elizabeth Correia	**4SO20CS092**
Nishayne Emelia Vaz	**4SO20CS099**

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**
Nishayne Emelia Vaz	**4SO20CS099**

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 for 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**
Nishayne Emelia Vaz	**4SO20CS099**

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 his guidance and valuable suggestions which helped us to complete this project. We also thank our Project coordinator **Dr Harivinod N**, Dept of CSE, for their 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

In an era where IoT devices are becoming ubiquitous, their demand is steadily increasing, promising enhanced convenience and efficiency across various sectors. However, this surge in IoT adoption is accompanied by a heightened vulnerability to cyber threats, ranging from DDoS attacks to unauthorized access. Traditional security measures struggle to grapple with the intricate nature and scale of emerging threats in the IoT landscape, necessitating innovative solutions. This study delves into the vulnerabilities inherent in IoT networks and the potential consequences of cyber attacks. By employing advanced ML techniques, the system aims not only to identify deviations from normal behavior but also to classify and specify the type of attacks, ensuring a comprehensive and effective defense against evolving security challenges in the realm of IoT. Through real-world case studies and examples, the research underscores the pressing need for intelligent security measures and proposes a novel approach that integrates machine learning to fortify the resilience of IoT networks. The ensuing project builds upon this imperative by aiming to develop a sophisticated system that employs machine learning for anomaly detection and classification, addressing the critical challenge of enhancing IoT network security. The implementation of advanced ML algorithms is poised to significantly elevate the accuracy of detecting and classifying abnormal

network behaviors, thereby establishing a robust and user-friendly security framework for the evolving IoT landscape.

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

Article [1] horoughly 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 overcome 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 [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

The literature review 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, NB-SVM, 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 un-

derscores 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 in-

trusion 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 / Summary

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.

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	Protect and prevent botnet attacks, focusing on securing IoT networks [2]	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

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 un-

derstand 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 contain-

ing 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, preprocessing, 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 ".csv" files. The ".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.

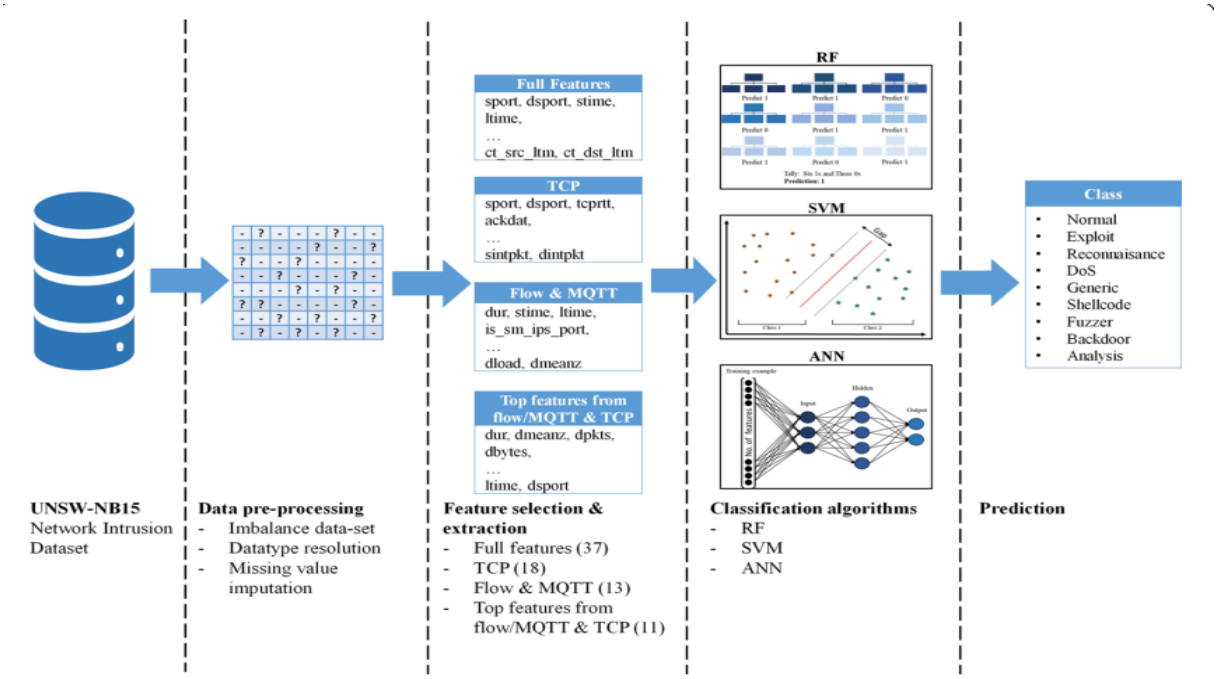

Figure 4.1: Proposed framework for detecting cyber-attacks in IoT networks

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 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

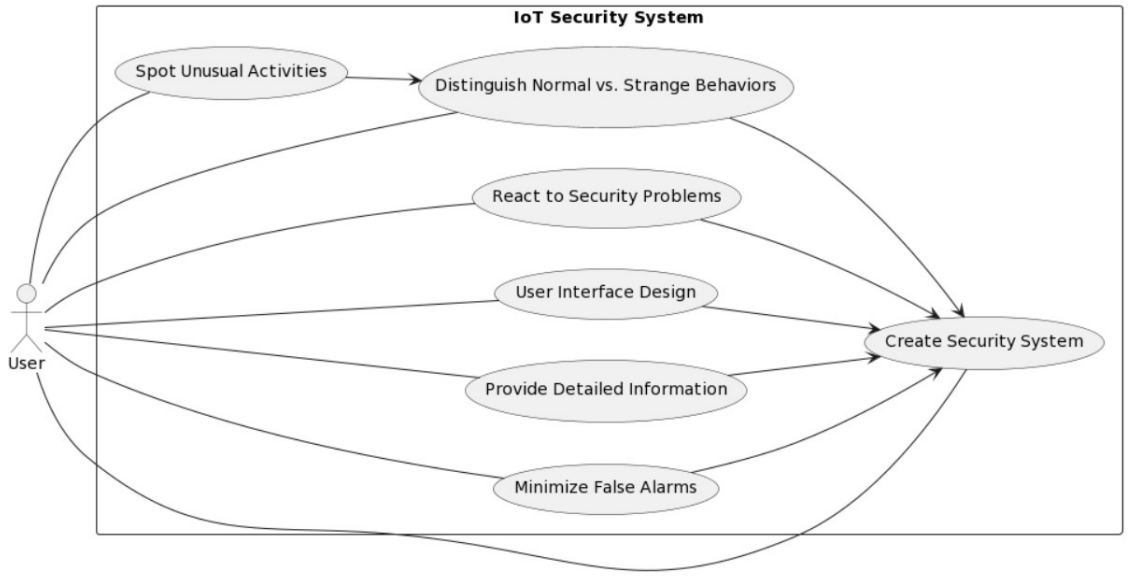

Figure 4.2: Use Case Diagram

Figure 4.2 illustrates the use case diagram for IoT Intrusion Detection System (IDS). The diagram depicts the interactions between the User and the various use cases within the IoT Security System. Key functionalities such as spotting unusual activities, reacting to security problems, distinguishing normal vs. strange behaviors, user interface design, providing detailed information, minimizing false alarms, and creating the security system are highlighted. The interconnected relationships showcase the flow of actions and dependencies in achieving a robust IoT security framework.

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

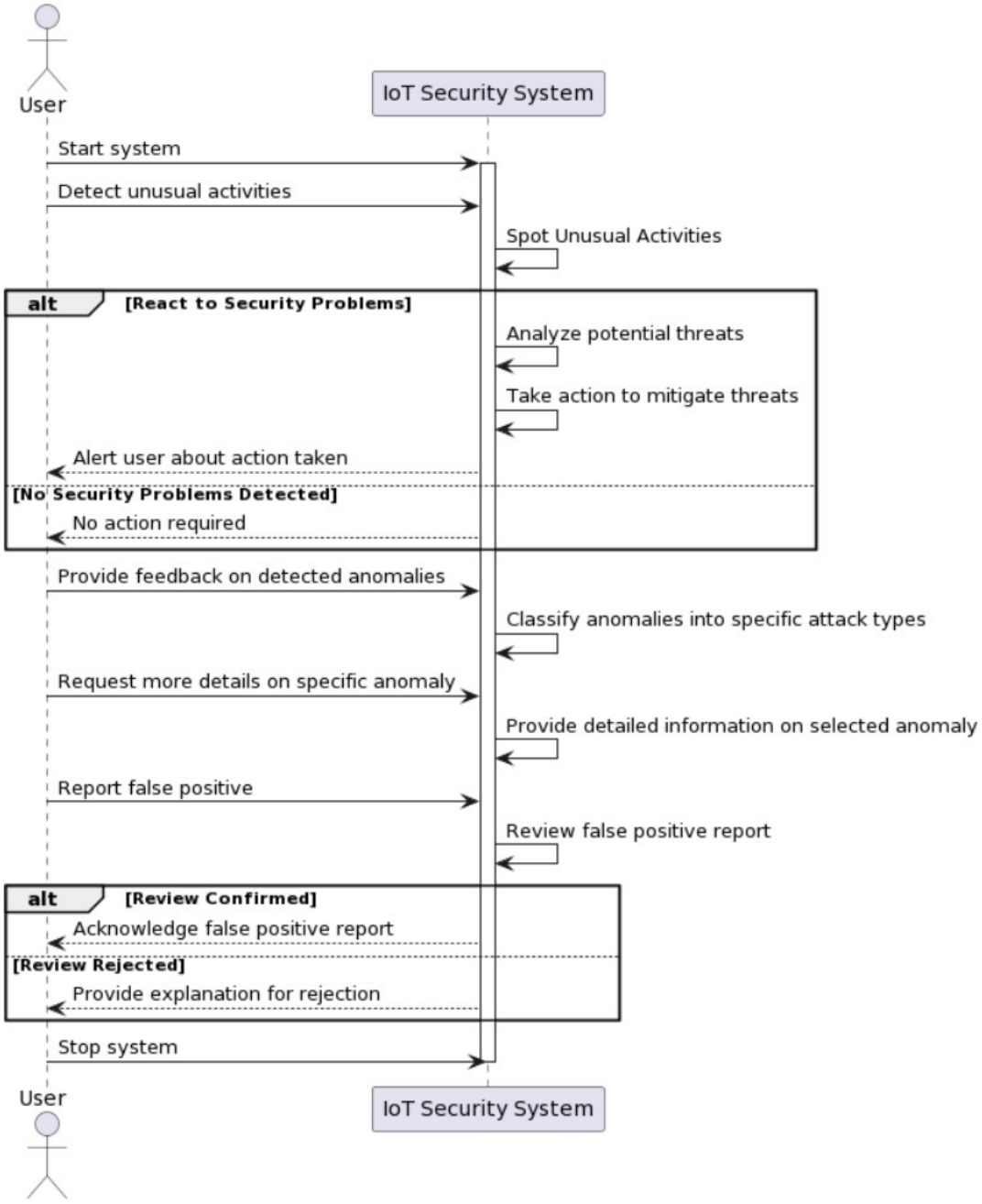

Figure 4.3: sequence diagram

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, 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.

4.2.2 Class Diagram

This class diagram, as illustrated in Figure 4.4, outlines the key components and relationships within an IoT security system. The “User” class represents users with attributes such as a username and password, along with a method for user login. The “SecuritySystem” class encapsulates the core functionalities of the system, including starting and stopping the system, detecting unusual activities, reacting to security problems, designing a user-friendly interface, providing detailed information, minimizing false alarms, and creating the security system itself.

The interaction between the “User” and “SecuritySystem” classes, as de-

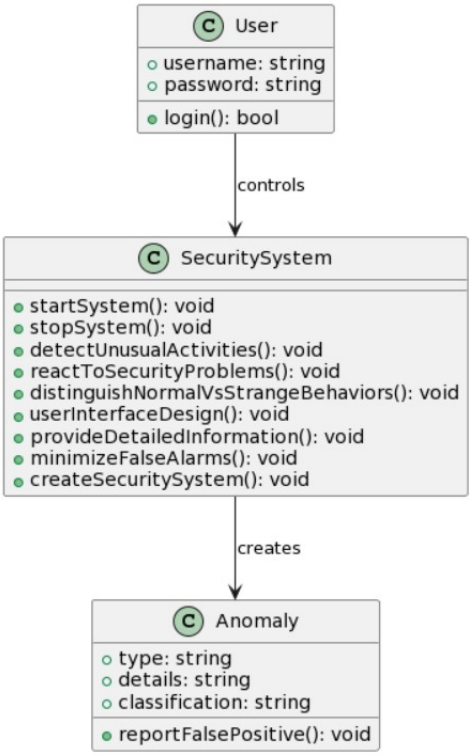

Figure 4.4: Class diagram

picted in Figure 4.4, signifies the user's control over the security system. On the anomaly side, the "Anomaly" class captures information about anomalies, encompassing type, details, and classification. Notably, Figure 4.4 emphasizes the "SecuritySystem" class's responsibility for creating instances of anomalies, showcasing its central role in anomaly management.

The underlying class diagram serves as a foundational guide, illuminating the intricate relationships among users, the security system, and the identified anomalies. This blueprint lays the groundwork for a robust and intelligently controlled IoT security infrastructure.

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

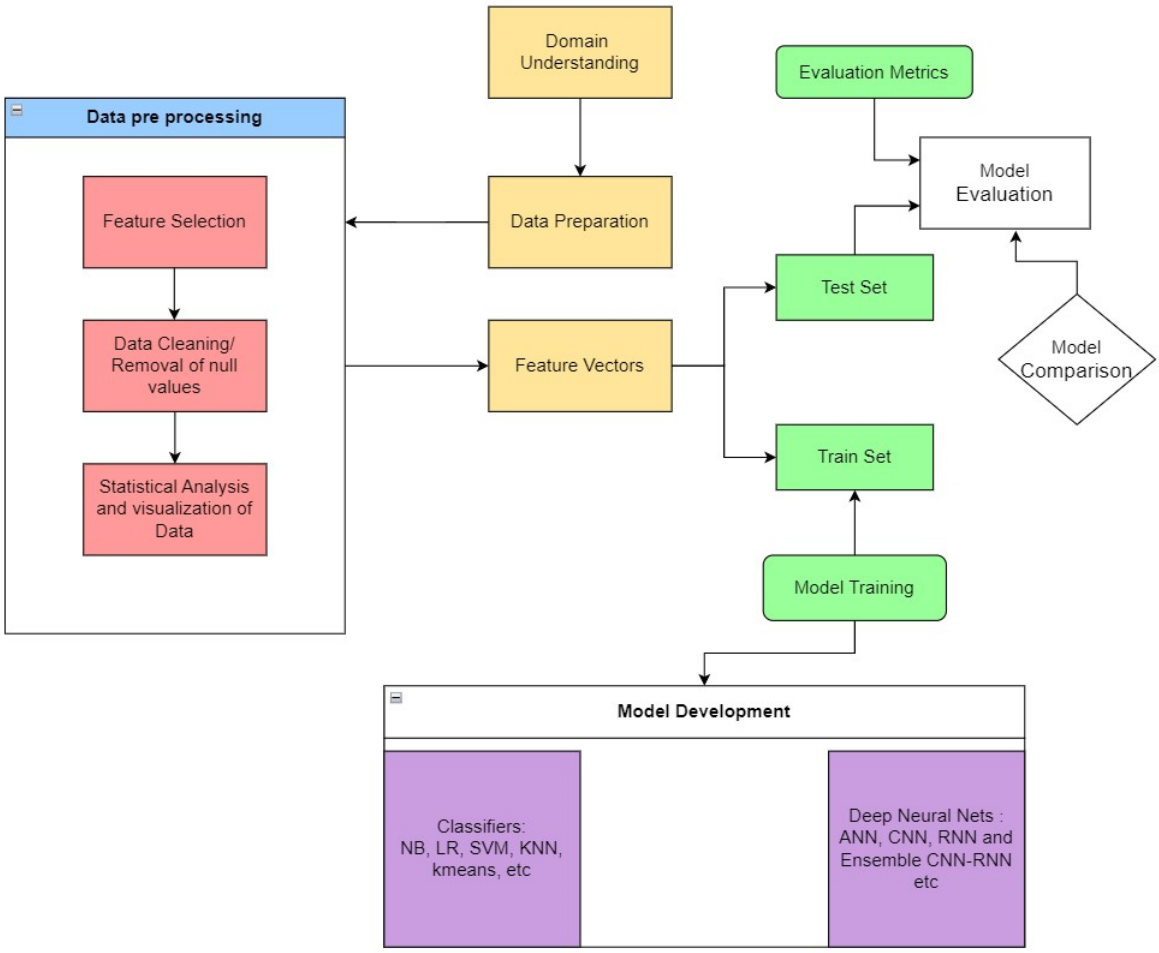

Figure 4.5: System Flow Diagram

In Figure 4.5, 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 more proficient and reliable for real-world applications.

4.3.2 Algorithms for Logic Implementation

The ML or DL algorithm is introduced in the following training phase utilising the training dataset. Deep learning may one day be used to identify IoT malware, as this system investigates. Depending on the amount of the dataset and the complexity of the suggested model, the algorithm's learning time will vary. Due to the deep and intricate structure of DL models, training time is often longer. Decision Tree, K-Nearest Neighbor (KNN), Artificial Neural Network (ANN), Support Vector Machine (SVM), K-Mean Clustering, Fast Learning Network, and Ensemble Methods are the most frequently used machine learning and deep Learning methods.

Phase 1 (Data cleaning and Feature Extraction): This stage will be divided into two steps; first step is that the original dataset consists of many missing values in some feature columns which cannot be fed to machine learning or specifically to deep learning models. Splitting data into train and test, all the operation like cleaning, exploratory data analysis and Feature Extraction and deleting the concatenated data frame will be done on train data only.

The strongest qualities in this dataset, which has 45 attributes or features, may be suggested in order to better accurately find patterns. Unimportantly, the dataset includes several duplicate and unnecessary attributes.

In order to improve intrusion detection accuracy, feature selection is crucial. The UNSW-NB15 dataset is widely categorized into Testing and Training datasets with 175,341 and 82,332 records, respectively, by containing records of regular traffic and all attack kinds. 45 characteristics are included in the dataset.

Features: By generating new features from the current ones, feature extraction attempts to decrease the amount of features in a dataset (and then discarding the original features). The majority of the information in the original collection of characteristics should then be summarised by this new, smaller set of features. By combining the original set of features in this fashion, a condensed version of the original features may be produced. A variety of machine learning and deep learning algorithms are automatically applied to the feature selection process to determine accuracy, score, confusion matrix, etc.

```
features = pd.read_csv('datasets/UNSW/UNSW_NB15_features.csv')
```

Phase 2 (Pre-Processing and Normalization): The effectiveness of the preprocessing methods was assessed using scalar and normalizing functions. Different sets of attributes based on label and feature characteristics were used to apply each of these preprocessing models. Data normalization was notably beneficial for systems where measurements are often reported on wildly dissimilar levels. Min-max normalization aids in more reliable neural network construction. This normalization approach has the benefit of precisely retaining all data links, preventing bias from developing. As min-max is introduced, the rising function falls below the proper value range for classification, while the corresponding distributions of the associated features remain within the current value range. The Feature, Min Max converts every value of a column into a number between 0 and 1. The new value is calculated as the difference between the current value and the min value, divided by the range of the column values.

In scikit-learn we use the MinMaxScaler class. For example, we can apply the min max method as follows:

```
from sklearn.preprocessing import MinMaxScaler
X= np.array(df['feature']).reshape(-1,1)
scaler = MinMaxScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
df['min max'] = X_scaled.reshape(1,-1)[0]
```

One-hot encoding is an important step for preparing your dataset for use in machine learning. One-hot encoding turns your categorical data into a binary vector representation. Pandas get dummies makes this very easy. This means that for each unique value in a column, a new column is created. The values in this column are represented as 1s and Os, depending on whether the value matches the column header.

Phase 3 (Split Dataset and Classification): In this stage, we split the main dataset into a binary data and multi-class data with 50%, 60%, 70%, 80% training set and 50%, 40%, 30%, 20% test set respectively.

The dataset may be divided into subsets using train test split() from the data science toolkit scikit- learn to reduce the possibility of bias throughout the evaluation and validation process. The goal of supervised machine learning is to develop models that accurately connect the provided inputs (also known as independent variables or predictors) to the provided outputs (dependent variables, or responses).Binary classification refers to those classification tasks that have two class labels.

Term Related to classification:

1. Precision: Precision in binary classification (Yes/No) refers to a model's ability to correctly interpret positive observations. In other words,

how often does a positive value forecast turn out to be correct? We may manipulate this metric by only returning positive for the single observation in which we have the most confidence.

2. Recall: The recall is also known as sensitivity. In binary classification (Yes/No) recall is used to measure how "sensitive" the classifier is to detecting positive cases. To put it another way, how many real findings did we "catch" in our sample? We may manipulate this metric by classifying both results as positive.

3. F1 score: The F1 score can be thought of as a weighted average of precision and recall, with the best value being 1 and the worst being 0. Precision and recall also make an equal contribution to the F1 ranking.

4. Support: The support is the number of samples of the true response that lie in that class.

Multi-class classification refers to those classification tasks that have more than two class labels. Unlike binary classification, multi-class classification does not have the notion of normal and abnormal outcomes. Instead, examples are classified as belonging to one among a range of known classes like le1, le2, enc_label etc.

Phase 4 (Detection System): we proposed a privacy attack detection system in IoT devices based on machine learning techniques with certain accuracies for each algorithms.

The key contributions in this study are presented as follows:

1. A feature selection method is proposed based on the correlation matrix calculated between the features. Features that were highly correlated with one another were removed as they added no significant difference and only increased the complexity of the model.

2. Optimal features are selected from the UNSW NB 15 dataset.

Each Machine Learning Model includes - Real and Predicted data along with plot graphs for Binary classification for each of the following:

- **Linear Regression:** Linear Regression fits a linear model with coefficients w = (w1,..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

  lr_bin = LinearRegression(normalize=False)
  lr_bin.fit(X_train, y_train)

- **Logistic regression:** Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on.

  log_bin = Logistic Regression(max_iter=5000)
  log_bin.fit(X_train,y_train)
  y_predlog_bin.predict(X_test)

- **Linear Support Vector Machine:** Similar to SVC with parameter kernel="linear", but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples. This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

  lsvm_bin = SVC(kernel="linear",gamma='auto')
  lsvm_bin.fit(X_train,y_train)
  y_pred = lsvm_bin.predict(X_test)

- **K Nearest Neighbor Classifier:** K-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

knn_bin=KNeighborsClassifier(n_neighbors=5)
knn_bin.fit(X_train,y_train)
y_pred = knn_bin.predict(X_test)

- **Random Forest Classifier:** A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap True (default), otherwise the whole dataset is used to build each tree.

  rf_bin = RandomForestClassifier(random_state=123)
  rf_bin.fit(X train,y_train)
  y_pred = rf_bin.predict(X_test)

- **Decision Tree Classifier:** Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

  dt_bin = Decision TreeClassifier(random_state=123) dt_bin.fit(X_train,y_train)
  y_pred = dt_bin.predict(X_test)

- **Multi-Layer Perceptron:** A feedforward artificial neural network that produces a set of outputs from a set of inputs is called a multilayer perceptron (MLP). A directed graph connecting the input and output layers of an MLP is made up of many layers of input nodes. Backpropagation is used by MLP to train the network.

  mlp_multi= MLPClassifier(random_state=123, solver='adam',
  max_iter=100)mlp_multi.fit(X_train,y_train)

- **Artificial neural network:** Artificial Neural Networks (ANN) are a subset of supervised machine learning in which our dataset contains both the input and the matching output. Finding a means to transfer this input to the appropriate output is our goal. ANN may be used to solve classification and regression issues.

classifier Sequential()
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu', input_dim = 15))
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation= 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics= ['accuracy'])
classifier.fit(X_train, y_train, batch_size = 10, epochs = 100,verbose = 0)

- **K-Fold Cross Validation:** A resampling technique called cross-validation is used to assess machine learning models on a small data sample. The process contains a single parameter, k, that specifies how many groups should be created from a given data sample.

  from sklearn.model selection import cross_val_score
  cv_scores_linear.append(cross_val_score(LinearRegression(), X_test, y_test, cv=kf)*100)

In summary, our machine learning workflow for IoT intrusion detection leverages a range of algorithms such as Logistic Regression, Support Vector Machines, Random Forests, and Neural Networks. This comprehensive approach ensures effective detection of anomalies and potential security threats. From data preprocessing to model evaluation, the systematic execution of this workflow enhances the accuracy and reliability of our intrusion detection system, contributing to the overall security and resilience of IoT environments.

4.3.3 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 se-

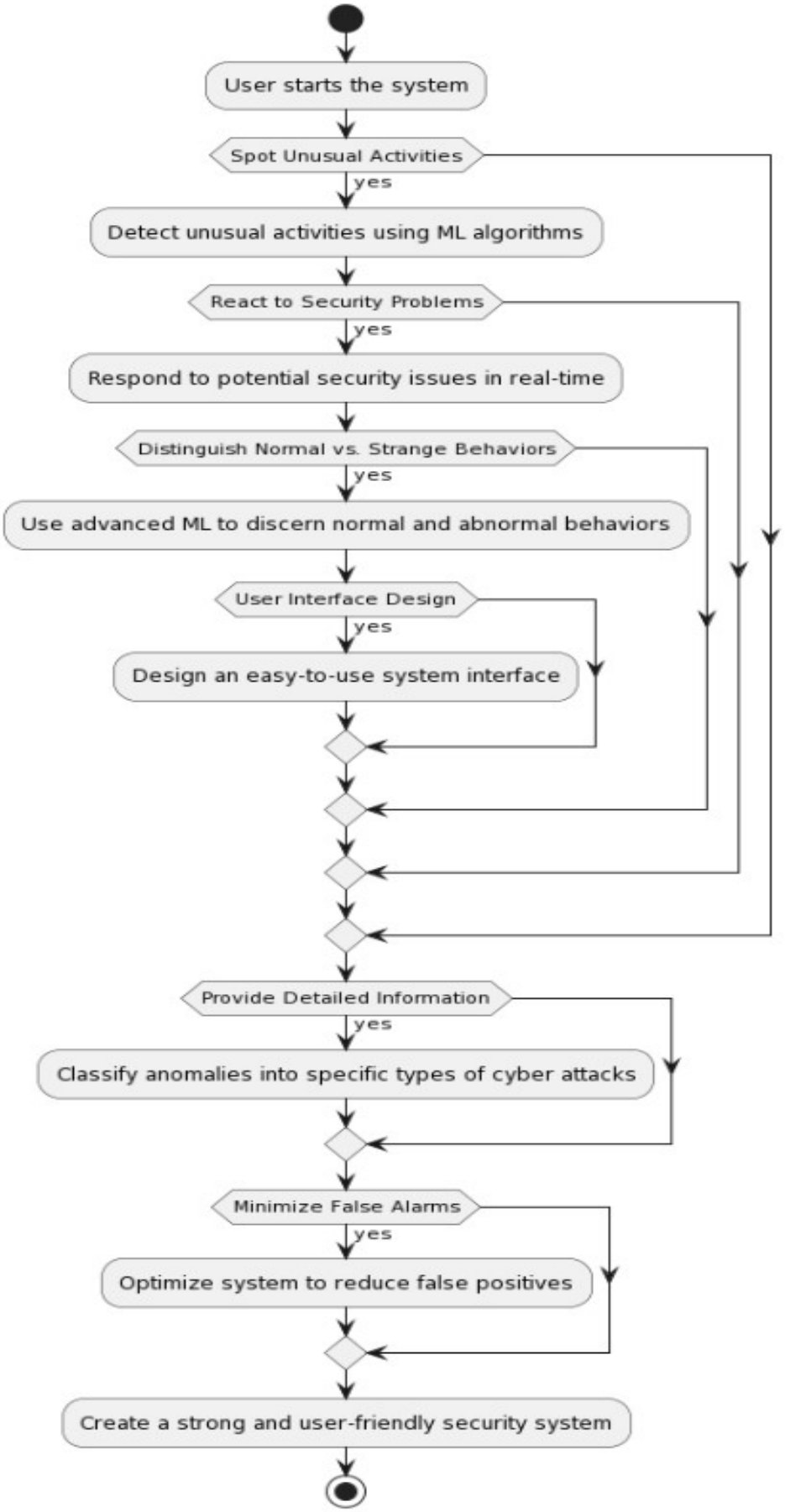

Figure 4.6: Activity Diagram

curity 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.

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 Dataset Description

The dataset was created by applying IXIA Perfect Storm tool. It includes nine categories of the modern attack types and involves realistic activities of normal traffic. The UNSW-NB 15 dataset contains 175,341 number of instances in the training Dataset, 82,332 number of instances in testing dataset, 49 features that comprised of 9 attack categories known as the Analysis, Fuzzers, Backdoors, DoS Exploits, Reconnaissance, Generic,

Shellcode, and Worms.

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm	ct_srv_dst	is_sm
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	1	0	0	0	1	1	
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	2	0	0	0	1	6	
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	3	0	0	0	2	6	
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	3	1	1	0	2	1	
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	40	0	0	0	2	39	
5	6	0.380537	tcp	-	FIN	10	6	534	268	39.417980	...	1	40	0	0	0	2	39	
6	7	0.637109	tcp	-	FIN	10	8	534	354	26.683033	...	1	40	0	0	0	1	39	
7	8	0.521584	tcp	-	FIN	10	8	534	354	32.593026	...	1	40	0	0	0	3	39	
8	9	0.542905	tcp	-	FIN	10	8	534	354	31.313031	...	1	40	0	0	0	3	39	
9	10	0.258687	tcp	-	FIN	10	6	534	268	57.985135	...	1	40	0	0	0	3	39	

Figure 5.1: Dataset

5.2 Dataset Features

5.3 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.

5.3.1 Prerequsites

- **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.

5.3.2 Environment for running the project

- **Jupyter Notebook:** Jupyter Notebook is a server-client program that enables web browser-based editing and execution of notebook builds.
- **Google Colaboratory:** Colab notebooks, which also include images, HTML, LaTeX, and more, combine rich text with executable code in a single document.

5.3.3 Instructions

- **Programming Environment and Dataset:** Ensure that the system has the necessary programming environment and dataset.
- **Import the Notebook and Data:** Choose between Jupyter Notebook or Google Colaboratory. Click on the .ipynb file to import the notebook and data.
- **Executing the Program:**
  - To execute the complete program at once, press `Ctrl + F9`.
  - Press the run command after selecting the cell you wish to run.

5.4 Datasets

49 features with the class label can be found in UNSW NB15 features.csv, which is part of the original dataset UNSW NB1S. The UNSW-NB15 features.csv file contains a description of these characteristics. The csv dataset used for binary classifier is bin data.csv and for multi class classifier, multidataċsv are used for computation.

5.5 Data Preprocessing

- The dataset had 175341 rows and 45 characteristics features.
- The dataset has 81173 rows and 45 characteristics features after eliminating null values.
- The datatype information from the features dataset is used to transform the data type of the attributes.

5.6 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)

5.7 Classification Models

5.7.1 Machine Learning Models

- Linear Regression
- Logistic Regression

- Linear Support Vector Machine
- K-Nearest-Neighbor-Classifier
- Random Forest Classifier
- Decision Tree Classifier

5.7.2 Deep Learning Models

- Multi-Layer Perceptron

Chapter 6

System Testing

Testing is a procedure of executing the program with unequivocal intension of [1] discovering mistakes, assuming any, which makes the program, fall flat. This stage is an essential piece of improvement.

It plays out an exceptionally basic part for quality affirmation and for guaranteeing unwavering quality of programming. It is the way toward finding the mistakes and missing operation and furthermore an entire confirmation to decide if the targets are met the client prerequisites are fulfilled.

The objective of testing is to reveal prerequisites, outline or coding blunders in the projects. Therefore, unique levels of testing are utilized in programming frameworks. The testing results are utilized amid upkeep.

This area manages the points of interest in the various classes of the test which should be directed to approve capacities, imperatives and execution. This can be accomplished fundamentally by using the methods for testing, which assumes a crucial part in the improvement of a product.

The structure of the program is not being considered in useful testing. Test cases are exclusively chosen on the premise of the prerequisites or particulars of a program or module of program but the internals of the module or the program are not considered for determination of experiments.

The program to be tried is executed with an arrangement of experiments and the yield of the program for the experiments is assessed to decide whether the program is executing not surprisingly. The accomplishment of testing in uncovering mistakes in projects depends basically

Figure 6.1: testcases

on the experiments. There are two fundamental ways to deal with testing Black Box or functional Testing and White Box or structural testing.

Table 6.1: Work Flow

Sl No	**Work**	**Duration(in Weeks)**
1	Audio Extraction	1
2	Audio Enhancement using LAVSE	4
3	Audio Separation using Speechbrain	3
4	Noice Reduction using spectral subtraction	2
5	Image segmentation	3
6	Speaker Identification	5

Chapter 7

Results and Discussion

7.1 Face detection

Figure 7.1: Face detection

Above figure 8.1 shows initial face detection process using opencv and dlib. It convert the image to grayscale, apply the model using cv2.detectMultiScale(), and draw bounding boxes around the detected faces using cv2.rectangle(). Display or save the result using cv2.imshow() or cv2.imwrite().

7.2 Speaker recognition

Figure 7.2: Speaker recognition 1,person 1

Figure 7.3: Speaker recognition 1,person 2

Figure 7.4: Speaker recognition 2,person 1

Figure 7.5: Speaker recognition 2,person 2

Figure 7.6: Speaker recognition 3,person 1

Figure 7.7: Speaker recognition 3,person 2

Above figures from 8.2 to 8.7 shows speaker recognition process using opencv and dlib.Speaker detection using cv2 and dlib involves utilizing dlib's pre-trained models along with cv2 functions to detect and locate human faces. By combining face detection with additional techniques such as audio analysis or lip movement tracking, speaker detection can be achieved in various applications like video conferencing or surveillance.

Chapter 8

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] Moustafa, Nour, et al. "An Ensemble Intrusion Detection Technique based on proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things." IEEE Internet of Things Journal (2018).

[4] da Costa, Kelton AP, et al. "Internet of Things: A survey on machine learning-based intrusion detection approaches." Computer Networks 151 (2019): 147-157.

[5] Moustafa, Nour, et al. "A New Threat Intelligence Scheme for Safeguarding Industry 4.0 Systems." IEEE Access (2018).

[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 , ACM Computing Surveys, vol. 55, no. 5, pp. 1–37, Dec. 2022.