"SPOTTING CYBER-ATTACKS USING MACHINE LEARNING"

PROJECT REPORT

Submitted in the partial fulfilment of the requirements

for award of the

Six Months Online Certificate Course

in

Cyber Security

Course Duration: [25-01-2024 to 24-07-2024]

By

GAJJALA SRINIVAS KUMAR

(Ht. No. 2406CYS109)

Under the Esteemed Guidance

Dr. UMA N DULHARE

Professor & Head

Computer Science and Artificial Intelligence Department,

Muffakham Jah College of Engineering and Technology,

Hyderabad

DIRECTORATE OF INNOVATIVE LEARNING & TEACHING(DILT)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

(Formerly SCDE_SCHOOL OF CONTINUING AND DISTANCE EDUCATION)

ABSTRACT

Cyber-crime is proliferating everywhere, exploiting every kind of vulnerability to the computing environment. Ethical Hackers pay more attention towards assessing vulnerabilities and recommending mitigation methodologies. The development of effective techniques has been an urgent demand in the field of the cyber security community. Most techniques used in today's IDS are not able to deal with the dynamic and complex nature of cyber-attacks on computer networks. Machine learning for cyber security has become an issue of great importance recently due to the effectiveness of machine learning in cyber security issues. Machine learning techniques have been applied for major challenges in cyber security issues like intrusion detection, malware classification and detection, spam detection and phishing detection. Although machine learning cannot automate a complete cyber security system, it helps to identify cyber security threats more efficiently than other software-oriented methodologies, and thus reduces the burden on security analysts. Hence, efficient adaptive methods like various techniques of machine learning can result in higher detection rates, lower false alarm rates and reasonable computation and communication costs. Our main goal is that the task of finding attacks is fundamentally different from these other applications, making it significantly harder for the intrusion detection community to employ machine learning effectively.

Keywords: Cyber-crime, Machine learning, Cyber-security, Intrusion detection system.

TABLE OF CONTENTS

CHAPTER-01

INTRODUCTION

1.1 Introduction

Research into identifying cyber- attacks using machine learning in smart IoT networks focuses on developing algorithms and methodologies to detect and mitigate various types of cyber threats within interconnected IoT systems. This area of study is crucial due to the proliferation of IoT devices and the increasing sophistication of cyber-attacks targeting these devices. The process typically involves collecting and analyzing data from IoT devices to identify patterns indicative of malicious activity. Machine learning algorithms are then trained on this data to recognize these patterns and distinguish between normal and malicious behavior. Various techniques such as anomaly detection, supervised learning, and reinforcement learning may be employed depending on the nature of the cyber threats and the characteristics of the IoT network. Key challenges in this field include the vast amount of data generated by IoT devices, the need for real-time detection to respond promptly to threats, and the limited computational resources available on IoT devices. Researchers are continually exploring novel approaches to address these challenges and improve the effectiveness of cyber -attack detection in smart IoT networks. Additionally, ensuring the privacy and security of the data collected from IoT devices is essential to maintaining user trust and compliance with regulation. Traditional security measures, such as firewalls and antivirus software, are often insufficient to protect IoT environments due to their distributed nature, diverse communication protocols, and resource-constrained devices. In this context, leveraging advanced technologies such as Machine Learning (ML) holds tremendous promise for enhancing cybersecurity and defending against evolving threats. Machine Learning, a subset of artificial intelligence, enables computers to learn from data and make predictions or decisions without being explicitly programmed. By analyzing vast amounts of data generated by IoT devices, ML algorithms can identify patterns, anomalies, and indicators of potential cyber- attacks. This proactive approach to threat detection empowers organizations to stay ahead of adversaries and mitigate risks before they escalate into full-blown security incidents. We explore the intersection of Machine Learning and IoT security, focusing specifically on the application of ML algorithms for spotting cyber- attacks within IoT environments. We examine the challenges posed by securing IoT ecosystems, discuss the limitations of traditional security approaches, and highlight the unique opportunities afforded by ML-based solutions. Furthermore, we

delve into the key outcomes and benefits of leveraging ML and IoT programs for cyber - attack detection, including improved threat detection, enhanced situational awareness, and faster incident response.The rate of attacks against networked systems has increased melodramatically, and the strategies used by the attackers are continuing to evolve. For example, the privacy of important information, security of stored data platforms, availability of knowledge, etc. Depending on these problems, cyber terrorism is one of the most important issues in today's world. Cyber terror, which caused a lot of problems to individuals and institutions, has reached a level that could threaten public and country security by various groups such as criminal organizations, professional persons, and cyber activists. Intrusion detection is one of the solutions to these attacks. A free and effective approach for designing Intrusion Detection Systems (IDS) is Machine Learning. In this study, were used to detect port scan attempts based on the dataset with a software-based application that is used to identify malicious behavior in the network . Based on the detection technique, intrusion detection is classified into anomaly-based and signature-based. IDS developers employ various techniques for intrusion detection. Information security is the process of protecting information from unauthorized access, usage, disclosure, destruction, modification, or damage. The terms "Information security", "computer security" and " information insurance" are often used interchangeably.

1.2 Problem statement

Machine learning has emerged as a promising solution for enhancing the security of IoT networks by enabling proactive threat identification and mitigation. However, several challenges hinder the effective application of machine learning techniques in this context. These challenges include: inherent vulnerabilities of IoT devices and networks make them prime targets for cyber- attacks. Traditional security mechanisms often fall short in effectively identifying and mitigating these threats due to the dynamic and heterogeneous nature of IoT environments. Therefore, there is a pressing need to develop advanced and adaptive approaches for the timely detection pose significant challenges.

1.2.1 Real-time Detection: Cyber- attacks in IoT networks often require real-time detection and response to prevent or minimize their impact. Machine learning algorithms must be capable of analyze streaming data and identifying anomalies in real-time.

1.2.2 Resource Constraints: Many IoT devices operate with limited computational resources, such as processing power, memory, and energy. Machine learning models deployed on these devices must be lightweight and energy-efficient

while maintaining high detection accuracy.

1.2.3 Adaptability to Emerging Threats: The threat landscape in IoT networks is constantly evolving, with adversaries employing sophisticated techniques to evade detection. Machine learning models must be adaptive and capable of learning from new attack patterns to effectively counter emerging threats.

1.2.4 Privacy and Data Security: IoT devices often collect sensitive data, raising concerns about privacy and data security. Machine learning algorithms must operate in a privacy- preserving manner, ensuring that sensitive information is not compromised during the detection process.

Addressing these challenges requires interdisciplinary research efforts aimed at developing innovative machine learning techniques tailored for cyber- attack identification in smart IoT networks. Additionally, there is a need for comprehensive evaluation frameworks and benchmark datasets to assess the effectiveness and scalability of machine learning-based security solutions in real-world IoT deployments. By overcoming these challenges, we can enhance the security and resilience of smart IoT networks, safeguarding critical infrastructure and ensuring the integrity of IoT-enabled services.

1.3 Scope of Research

The objective of a project focused on identifying cyber -attacks using machine learning in smart IoT networks can be summarized as follows:

1.3.1 **Enhanced Security**: The primary goal is to enhance the security posture of smart IoT networks by developing effective techniques to detect and mitigate cyber- attacks in real- time.

1.3.2 **Detection of Diverse Threats**: The project aims to identify various types of cyber threats targeting IoT devices, including malware infections, distributed denial-of-service (DDoS) attacks, intrusions, data exfiltration attempts, and other forms of malicious activity.

1.3.3 **Machine Learning Integration:** Leveraging the capabilities of machine learning algorithms, the project seeks to design intelligent systems capable of automatically recognizing patterns indicative of cyber -attacks within the vast and heterogeneous data generated by IoT devices.

1.3.4 **Real-time Response:** The project emphasizes the importance of real-time detection and response to cyber threats, enabling rapid mitigation actions to be taken to prevent or minimize the impact of attacks on IoT networks and their associated systems.

1.3.5 **Scalability and Efficiency:** Solutions developed within the project aim to be scalable to accommodate the increasing number of IoT devices in networks while maintaining efficiency in terms of computational resources and response times.

1.3.6 **Adaptability and Robustness:** The project intends to create detection mechanisms that are adaptable to evolving cyber threats and robust against evasion techniques employed by attackers, ensuring continued effectiveness over time.

1.3.7 **Privacy and Compliance:** Emphasis is placed on preserving the privacy of user data collected from IoT devices and ensuring compliance with relevant regulations and standards governing data protection and cybersecurity.

By achieving these objectives, the project aims to contribute to the overall security and reliability of smart IoT networks, fostering trust among users and stakeholders and facilitating the widespread adoption of IoT technologies in various.

1.4 Existing system

When an IDS detects suspicious activity, the violation is typically reported to a security information and event management (SIEM) system where real threats are ultimately determined amid benign traffic abnormalities or other false alarms. However, the longer it takes to distinguish a threat, the more Most techniques used in today's IDS are not able to deal with the dynamic and complex nature of cyber-attacks on computer networks. This is a huge concern as encryption is becoming more prevalent to keep our datasecure. One significant issue with an IDS is that they regularly alert you to false positives. In many cases false positives are more frequent than actual threats. If they don't take care to monitor the false positives, real attacks can slip damage can be done.

Disadvantage of Existing System :

1.4.1 One of the primary challenges associated with IDS is the generation of false positive alerts.

1.4.2 Implementing and managing IDS can be complex and resource-intensive, particularly in large and heterogeneous network environments.

1.4.3 IDS typically focus on monitoring network traffic and may have limited visibility into other layers of the technology stack, such as application-level protocols, host-based activities, or encrypted communications.

1.5 Proposed System

Machine Learning algorithms can help identify and respond to cyber attacks efficiently. Classification algorithms, such as Decision Trees, Autoencoders, and Deep Neural

Networks (DNN), can be used to categorize whether an attack.

Decision Trees: A supervised learning method that segments data based on patterns to predict the outcome class, distinguishing between normal traffic and various attack types.

Autoencoders: These unsupervised learning models identify patterns in data by encoding and decoding information, which can help distinguish anomalies or malicious behavior.

Deep Neural Networks (DNNs): Complex multi-layer models capable of learning intricate patterns and relationships in data, making them useful for accurate classification of cyber threats.

Advantages of proposed system

- Acquiring and labeling such data can be time-consuming and resource-intensive, particularly for rare or emerging cyber threats.
- This can lead to poor performance on unseen data and an increased susceptibility to false positives or false negatives.

The proposed system leverages machine learning to provide robust and real-time identification of cyber attacks in smart IoT networks. By integrating advanced data collection, preprocessing, feature extraction, and machine learning techniques, the system aims to enhance the security and resilience of IoT networks against evolving cyber threats.

1.6 Organization of Report

Creating an organizational report on the identification of cyber attacks using machine learning in smart IoT networks involves a detailed analysis of the current state of IoT security, machine learning techniques applied to cybersecurity, and the integration of these technologies to enhance security measures. Here's a structured outline to guide your report:

- Title
  - Identification of Cyber Attacks Using Machine Learning in Smart IoT Networks
- Executive Summary
  - The Objective is to provide an overview of how machine learning can be employed to identify and mitigate cyber attacks in smart IoT networks.It Summarize the effectiveness, challenges, and future prospects of using machine learning for IoT security.Highlights key actions and strategies for organizations to enhance IoT security using machine learning.
- Introduction
  - The Overview of smart IoT networks and their significance and the increasing

threat of cyber attacks in IoT networks.Its Purpose To explore the role of machine learning in detecting and preventing cyber attacks in IoT environments.

- IoT Security Landscape
  - Common cyber attacks targeting IoT devices (e.g., DDoS, ransomware, data breaches).It Specifies vulnerabilities in IoT devices and networks.Potential impact of these cyber attacks on organizations and individuals
- Machine Learning in Cybersecurity
  - Basic concepts and types of machine learning (supervised, unsupervised, reinforcement learning).General applications of machine learning in detecting and mitigating cyber threats.
- Integration of Machine Learning in IoT Security Techniques and Algorithms:
  - Supervised Learning Examples include anomaly detection using decision trees, and neural networks.
  - Unsupervised Learning: Techniques like clustering and anomaly detection using Auto Encoder.
- Data Collection and Preprocessing
  - Importance of data quality, feature selection, and data labeling in training machine learning models.
- Challenges and Limitations
  - The Issues related to data quality, model accuracy, and real-time processing.Integration with existing systems, scalability, and cost. Ensuring user privacy and ethical use of data.
- Future Enhancement
  - Advances in machine learning algorithms and their potential applications in IoT security.Areas for further research to improve the effectiveness of machine learning in IoT cybersecurity.The role of regulatory frameworks in promoting secure IoT environments.
- Conclusion
  - Recap of key points discussed in the report.The importance of continuous innovation and vigilance in IoT security.
- References
  - Cite of sources, studies, and articles referenced throughout the report.
  - By following this outline, you can create a comprehensive organizational report that addresses the critical aspects of using machine learning to identify and

mitigate cyber attacks in smart IoT networks. Ensure that each section is well-researched and supported by current data and examples to provide a thorough understanding of the subject.

CHAPTER-02

LITERATURE SURVEY

1. Jonatan Gomez and Dipankar Dasgupta. Evolving fuzzy classifiers for intrusion detection. In Proceedings of the 2002 IEEE Workshop on Information Assurance, West Point, NY, USA, 2002.

Authors: Jonatan Gomez and Dipankar Dasgupta

Description

This citation refers to a publication by Jonatan Gomez and Dipankar Dasgupta titled "Evolving fuzzy classifiers for intrusion detection." It was presented at the 2002 IEEE Workshop on Information Assurance held in West Point, New York, USA.

The paper discusses a novel approach to intrusion detection using fuzzy classifiers. Fuzzy classifiers incorporate the concept of fuzzy logic, which deals with reasoning that is approximate rather than fixed and exact. The authors explore how evolving these classifiers can enhance the adaptability and accuracy of intrusion detection systems.

In their work, Gomez and Dasgupta address a crucial challenge in cybersecurity: the dynamic and evolving nature of network intrusions. The paper provides a detailed methodology for training and evolving fuzzy classifiers using machine learning techniques, allowing the system to recognize new types of attacks while adapting to changes in attack patterns.

The research includes a comprehensive evaluation of the classifier's performance, demonstrating its efficacy in detecting known and emerging intrusions with a high degree of accuracy. The results presented in this paper highlight the potential of evolving fuzzy classifiers to improve intrusion detection mechanisms significantly.

2. Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. Journal of Computer Security, 6(3):151{180, August 1998.

Authors: Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji.

Description

This citation is for a paper by Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji, titled "Intrusion Detection Using Sequences of System Calls," published in the Journal of

Computer Security, volume 6, issue 3, pages 151-180, in August 1998.

The paper presents a pioneering method for detecting intrusions by monitoring sequences of system calls. The authors argue that sequences of system calls are indicative of a program's normal behavior, and deviations from these sequences can reveal malicious activities.

Their approach involves building profiles of normal system behavior by capturing and analyzing the sequences of system calls made by various programs. Once these profiles are created, the system can compare real-time sequences against the normal patterns, identifying deviations as potential security threats. By focusing on sequences instead of individual calls, the approach captures contextual behavior, leading to more accurate intrusion detection.

The authors describe their methodology in detail, including the design of the detection system, data collection, and analysis process. They also provide experimental results that demonstrate the effectiveness of their system in detecting various types of intrusions with minimal false positives. The paper is foundational in establishing the importance of system call sequences in security monitoring and laid groundwork for future research in behaviour-based intrusion detection systems

3. Peter Mell Karen Scarfone. Guide to intrusion detection and prevention systems (idps). National Institute of Standards and Technology, NIST SP - 800-94, 2007.

Authors: Peter Mell Karen Scarfone

Description

Intrusion Detection and Prevention Systems (IDPS)." Published by the National Institute of Standards and Technology (NIST) as Special Publication 800-94 in 2007, this document serves as a comprehensive guide to understanding and implementing intrusion detection and prevention systems.The guide begins by providing an overview of intrusion detection and prevention systems, describing how they monitor network and system activities for malicious behaviors. It outlines the fundamental differences between intrusion detection systems (IDS) and intrusion prevention systems (IPS), highlighting their respective capabilities.

Mell and Scarfone detail the various types of IDPS technologies, including:

1. Network-Based Systems: These monitor network traffic for suspicious activity.
2. Host-Based Systems: These operate on individual hosts to analyze system calls, file system changes, and other internal behaviors.
3. Wireless Systems: These detect suspicious activities within wireless networks.
4. Network Behavior Analysis Systems: These identify unusual traffic patterns in

network flows that could indicate intrusions.

4. Jungwon Kim, Peter J. Bentley, Uwe Aickelin , Julie Greensmith, Gianni Tedesco, and Jamie Twycross. Immune system approaches to intrusion detection { a review. Natural Computing, 6(4):413{466, December 2007.

Authors: Jungwon Kim, Peter J. Bentley, Uwe Aickelin , Julie Greensmith, Gianni Tedesco.

Description

This citation refers to a paper by Jungwon Kim, Peter J. Bentley, Uwe Aickelin, Julie Greensmith, Gianni Tedesco, and Jamie Twycross, titled "Immune System Approaches toIntrusion Detection: A Review," published in the journal Natural Computing, volume 6, issue 4, pages 413-466, in December 2007.

The paper presents a comprehensive review of intrusion detection techniques inspired by the human immune system. These techniques, collectively known as artificial immune systems (AIS), are part of the broader field of biologically inspired computing.

The authors describe the analogy between biological immune systems and computer security systems. Just as the immune system protects the body against harmful pathogens, an intrusion detection system (IDS) aims to protect networks and computers from malicious attacks. AIS-based intrusion detection systems use principles like pattern recognition, anomaly detection, and learning to identify security threats.

Key areas covered in this review include:

1. Immune System Models: A summary of the components and processes of the human immune system that are relevant to IDS, such as the concept of self/non-self discrimination, immunological memory, and the use of antibodies.
2. Applications in Intrusion Detection: An exploration of how different immune system concepts have been adapted to detect malicious activities in computer networks, including negative selection, clonal selection, and danger theory.
3. Challenges and Future Directions: A discussion of the challenges facing immune-based intrusion detection, such as scalability, adaptability, and reducing false positives. The paper also suggests future research areas to improve AIS-based systems' effectiveness.

5. A. Shabtai, E. Menahem and Y. Elovici. FSign: automatic, function-based signature generation for malware, systems, man, and cybernetics, Part C: applications and reviews. Transactions on IEEE, 41, 494–508, 2011.

Authors: A. Shabtai, E. Menahem and Y. Elovici.

Description

This citation refers to a paper authored by A. Shabtai, E. Menahem, and Y. Elovici, titled "FSign: Automatic, Function-Based Signature Generation for Malware." It was published in the IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, volume 41, on pages 494-508, in 2011.

The paper presents FSign, a novel system for automatically generating malware signatures based on the functions utilized by malicious software. The authors aimed to improve the speed and accuracy of malware detection by focusing on specific functions that characterize malware behaviors.

Key features and contributions include:

1. Function-Based Analysis: The authors emphasize the importance of identifying functions that are consistently used in malware programs. By understanding these functions' behavior, F Sign can recognize patterns indicative of malware.
2. Automatic Signature Generation: F Sign employs an automatic signature generation process to identify relevant functions within unknown samples, enabling it to produce effective signatures quickly and efficiently.
3. Evaluation and Testing: The paper provides comprehensive evaluations demonstrating that F Sign can detect a wide range of malware variants. The system's function-based approach offers more robust protection against obfuscation techniques used by modern malware authors.

CHAPTER-03

Methodology

3.1 System Architetcture:

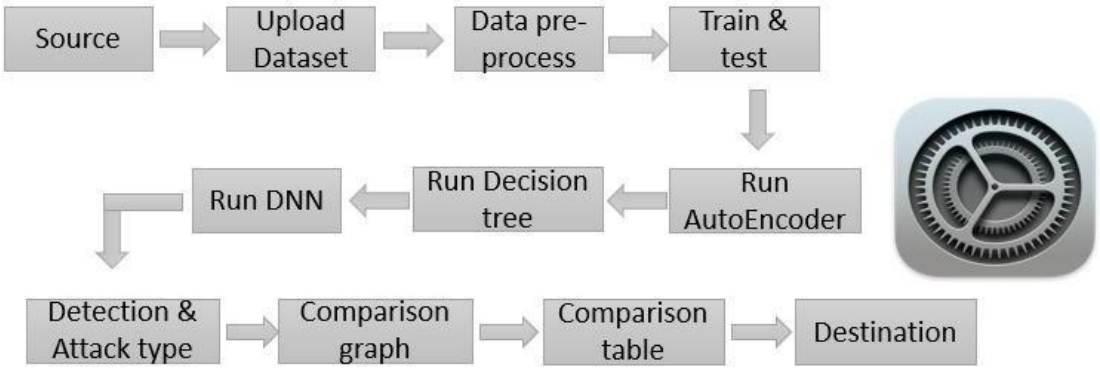


Figure 3.1 System Architecture

3.1.1 Source:

Source will provide the information to the user.

3.1.2 Upload Cyber Dataset

The Purpose is to uploads the Cyber dataset into the application. The main Functionality allows the user to select the dataset file and imports it into the system for further processing. It will Loads the dataset and provides an initial visualization showing the count of different attack types. Alerts the user to potential data imbalance.

3.1.3 Pre-process Dataset

The purpose of pre-process dataset is to cleanses and normalizes the dataset. The main functionality is to replaces missing values with zero and applies a Min-Max scaling algorithm to normalize the feature values. Splits the dataset into training and testing subsets, using 80% of the data for training and 20% for testing.It will returns the pre-processed dataset ready for model training.

3.1.4 Run Auto Encoder Algorithm

It works on to trains an auto encoder deep learning algorithm on the dataset.The main Function is to extracts features from the Auto Encoder model after training.It Provides the accuracy of the model and the learned feature representations.

3.1.5 Run Decision Tree with PCA

It Applies dimensionality reduction and classification using PCA and Decision Tree algorithms. It works to transforms the features extracted from the Auto Encoder using PCA to reduce dimensionality. Trains a Decision Tree classifier on the transformed features and predicts labels based on dataset signatures.It displays improved classification accuracy after dimensionality reduction.

3.1.6 Run DNN Algorithm

The purpose is to further enhances classification using a Deep Neural Network (DNN).It works to trains a DNN on the predicted labels from the Decision Tree classifier to detect and classify attacks.It Displays the enhanced accuracy and precision obtained with the DNN.

3.1.7 Detection & Attribute Attack Type

The purpose of this is to identifies and attributes attack types for unlabelled data.It works to uploads a new test dataset containing only signature data. Uses the trained DNN model to predict the attack type for each test data entry.Outputs the predicted attack types for each entry.

3.1.8 Comparison Graph

It purpose is to visualizes a comparison between different algorithms.It will plots a graph to compare precision, recall, accuracy, and F1 Score between Auto Encoder, Decision Tree with PCA, and DNN.It displays a bar graph comparing the performance of each algorithm.

3.1.9 Comparison Table

It will tabulates the performance metrics for each algorithm.It's functionality Generates a table that compares algorithms across various metrics such as accuracy, precision, recall, and F1 Score. It Displays a detailed comparison table with metric values for each algorithm.

3.2 Algorithms

3.2.1 Auto Encoder

Autoencoders are a type of neural network commonly used for unsupervised learning tasks, particularly in dimensionality reduction and data compression. In the context of anomaly detection, an autoencoder can learn to reconstruct normal patterns and identify deviations as potential cyber attacks.The autoencoder can reconstruct normal instances with low error rates. When presented with anomalous data (such as data generated by cyber attacks), the reconstruction error is typically higher. Therefore, instances with high reconstruction error are flagged as potential anomalies or cyber attacks.

3.2.2 Decision tree

Decision trees are another powerful machine learning technique commonly used in identifying cyber attacks. A decision tree model is trained using the labeled dataset, where each data point is labeled as either a normal or an attack instance. The decision tree algorithm recursively splits the data based on the features to create a tree-like structure. Decision trees are simple and interpretable models that partition the feature space based on attribute values. They are effective for detecting certain types of cyber attacks by recursively splitting the data into subsets.

3.2.3 DNN

A deep neural network (DNN) is an ANN with multiple hidden layers between the input and output layers. Similar to shallow ANNs, DNNs can model complex non-linear relationships. DNNs can be trained to recognize patterns of normal behavior within IoT networks. Deviations from these patterns can indicate potential cyber attacks. DNNs excel at anomaly detection tasks by learning to distinguish between normal and abnormal network behavior. Deep Neural Networks (DNNs) are increasingly being utilized in identifying cyber attacks within smart IoT networks due to their powerful ability to model complex patterns and detect subtle anomalies in large datasets. DNNs consist of multiple layers of neurons, including input layers, hidden layers, and output layers. Each layer transforms in IoT networks, this includes network traffic data, device logs, and sensor readings. Data preprocessing steps like normalization, noise reduction, and segmentation are critical to prepare the data for model training. By analyzing traffic patterns and identifying abnormal spikes, DNNs can predict and flag potential DoS attacks. DNN algorithms in identifying cyber-attacks in smart IoT networks leverages their capability

to learn and recognize intricate patterns from vast amounts of data, providing a robust mechanism for real-time threat detection and mitigation.

CHAPTER-04

System Requirement Specification

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

4.1 Functional Requirements

The functional requirement refers to the system needs in an exceedingly computer code engineering method.The key goal of determinant "functional requirements" in an exceedingly product style and implementation is to capture the desired behavior of a software package in terms of practicality and also the technology implementation of the business processes.

4.2 Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

The system should be able to interface with the existing system.

The system should be accurate.

The system should be better than the existing system.

4.3 Software Requriements

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

Operating system : Windows XP/7/10

Coding Language : Python 3.7

4.4 Hardware Requirements

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- System : Pentium IV 2.4 GHz.
- Hard Disk : 75 GB.
- Monitor: 15 VGA Color.
- Mouse : Logitech.
- RAM : 1 GB

4.5 Technology used

Identifying cyber attacks using machine learning in IOT networks,several technologies are commonly employed.

Here are some technologies used in this context:

4.5.1 Machine Learning

Various machine learning algorithms are applied for cyber attack detection including supervised algorithm such as decision tree. Learning technology such as deep neural network(DNNs) and auto encoders. Supervised learning algorithms may classify data into normal and malicious categories.Tools for preprocessing and analyzing IoT data play a crucial role.

These may include Python libraries such as Pandas, NumPy, and Scikit-learn for data manipulation, feature extraction, and model training. Additionally, specialized tools or platforms designed for handling large-scale IoT data, such as TensorFlow, may be utilized.

1.NUMPY

NumPy arrays are used to store and manipulate data, especially in the context of machine learning algorithms. For example, the dataset and its features are stored in NumPy arrays (X and Y) after preprocessing. NumPy provides mathematical functions for array computations. In this script, np.random.shuffle() is used to shuffle the dataset, and mathematical operations are performed for calculating accuracy, precision, recall, and F1-score metrics.NumPy arrays are involved in data normalization processes. For instance, MinMaxScaler from scikit-learn is used to scale features between a specified range, and NumPy arrays are used to store the normalized data.

2. Pandas

Pandas is used to read the dataset from CSV files into Data Frame objects. The dataset is then manipulated using various Data Frame operations such as filling missing values (fill na()), extracting values, and shuffling rows. Pandas is used to read the dataset from CSV files into Data Frame objects. The dataset is then manipulated using various Data Frame operations such as filling missing values (fill na()), extracting values, and shuffling rows. The pandas library to load a dataset from a CSV file. The dataset is displayed using dataset.head().Created a bar chart to visualize the various cyber-attacks found in the dataset.

3. MATAPLOTLIB

Matplotlib is used for data visualization. It is used to plot graphs showing the distribution of different types of cyber- attacks in the dataset. Plotting bar graphs to visualize the distribution of cyber-attacks plotting performance comparison graphs.

6. SCIKIT-LEARN

This library is used for machine learning tasks like preprocessing, model selection, evaluation, and many more. You've used sk-learn for tasks such as Data preprocessing: train_test_-plit, Min-Max-Scaler Model evaluation: accuracy-score, precision-score, recall-score, f1_scoreModel training: Decision Tree Classifier, MLP Classifier.

7. KERAS

It's a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. In your code, you've used Keras for. Building and training an autoencoder neural network.Saving and loading models (model_from_json, load_weights, save_weights).

8. Tkinker

A graphical user interface (GUI) application using Tkinker in Python for identifying cyber attacks in smart IoT networks. Import necessary libraries for GUI development, data manipulation, machine learning, and visualization. Create the main window of your application using Tkinker and set its title, geometry, and size functions for uploading datasets, preprocessing data, running machine learning algorithms (Auto Encoder, Decision Tree with PCA, DNN), detecting cyber attacks, and generating comparison graphs and tables. Each function performs specific tasks such as loading data, preprocessing, training machine learning models, and evaluating performance metrics. GUI components like buttons, labels, and text areas using Tkinker to provide user interaction and display information.

Users can upload datasets, preprocess data, run different machine learning algorithms, detect cyber attacks, and visualize performance results through the GUI.

integrate machine learning algorithms such as Auto Encoder, Decision Tree with PCA, and DNN into your application for cyber attack detection. The application preprocesses data, trains machine learning models, evaluates performance metrics, and provides visualizations to users. Users interact with the application through buttons and text areas to perform tasks such as uploading datasets, running algorithms, and viewing results. Application provides a user-friendly interface for identifying cyber attacks in smart IoT networks using machine learning techniques, allowing users to upload datasets, preprocess data, train models, detect attacks, and analyze performance metrics The main GUI window is created using tkinker Tk().The window title is set to “Identification of Cyber Attack in Network using Machine Learning Techniques”. The window dimensions are set to 1300x1200 pixels.

IoT networks and Protocol

Understanding IoT information i.e Bot-IoT Utilizing a bot for identifying cyber attacks in smart IoT networks involves creating an intelligent software agent that can autonomously monitor network traffic, analyze data from IoT devices, and detect suspicious activities indicative of cyberattacks. The bot continuously collects data from IoT devices and network traffic, monitoring for anomalies and patterns that may indicate malicious activity. This data could include sensor readings, device logs, network packets, and communication protocols.

The bot analyzes the preprocessed data to identify potential cyber threats. Based on the results of the analysis, the bot makes decisions regarding the presence of cyber attacks.If suspicious activity is detected, the bot generates alerts and notifies cybersecurity personnel or initiates automated response actions to mitigate the threat. By deploying a bot for IoT cybersecurity, organizations can enhance their ability to detect, respond to, and mitigate cyber threats in smart IoT networks, thereby strengthening overall cybersecurity defenses and safeguarding critical infrastructure and data.

CHAPTER-05

SYSTEM DESIGN

The purpose of the design phase is to arrange an answer of the matter such as by the necessity document. This part is that the opening moves in moving the matter domain to the answer domain. The design phase satisfies the requirements of the system. The design of a system is probably the foremost crucial issue warm heartedness the standard of the software package. It's a serious impact on the later part, notably testing and maintenance. The output of this part is that the style of the document. This document is analogous to a blueprint of answer and is employed later throughout implementation, testing and maintenance.

The design activity is commonly divided into 2 separate phases

System Design

System Design conjointly referred to as top-ranking style aims to spot the modules that ought to be within the system, the specifications of those modules, and the way them move with one another to supply the specified results. At the top of the system style all the main knowledge structures, file formats, output formats, and also the major modules within the system and their specifications square measure set. System design is that the method or art of process the design, components, modules, interfaces, and knowledge for a system to satisfy such as needs. Users will read it because the application of systems theory to development.

Detailed Design.

Detailed Design, the inner logic of every of the modules laid out in system design is determined. Throughout this part, the small print of the info of a module square measure sometimes laid out in a high-level style description language that is freelance of the target language within which the software package can eventually be enforced. In system design the main target is on distinguishing the modules, whereas throughout careful style the main target is on planning the logic for every of the modules.

5.1 Introduction to Uml:

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows. This view represents the system from the user's

perspective. The analysis representation describes a usage scenario from the end-users perspective and data and functionality are arrived from inside the system. This model view models the static structures.

5.2 UML diagrams

UML diagrams plays a crucial role in understanding, designing, and communicating the system architecture and functionality of cyber attack identification using machine learning in smart IoT networks. Depending on the specific requirements and complexity of the system, additional diagrams or variations of these diagrams may also be needed. UML diagrams can be adapted to model various aspects of cyber- attacks, providing a visual representation of the attack vectors, strategies, and defense mechanisms.

UML diagrams collectively provide a comprehensive visualization of the system's structure, interactions, and workflows, facilitating a clearer understanding of how the system identifies and responds to cyber attacks using machine learning in smart IoT networks. Creating a UML (Unified Modeling Language) diagram for a system that identifies cyber attacks using machine learning in smart IoT networks involves illustrating various components and their interactions. Below are descriptions of the key UML diagrams suitable for this system

5.2.1 Use Case Diagrams

Use-case diagrams graphically depict system behavior (use cases). These diagrams present a high -level view of how the system is used as viewed from an outsider’s (actor’s) perspective. A use-case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

- actors ("things" outside the system)
- use cases (system boundaries identifying what the system should do)
- Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Relationships in use case

1. Communication

The communication relationship of an actor in a use-case is shown by connecting the actor symbol to the use-case symbol with a solid path. The actor is said to communicate with the use-case.

2. Uses

A Uses relationship between the use-cases is shown by generalization arrow from the use-

case.

3. Extends

The extend relationship is used when we have one use-case that is similar to another use-case but does a bit more. In essence it is like subclass.

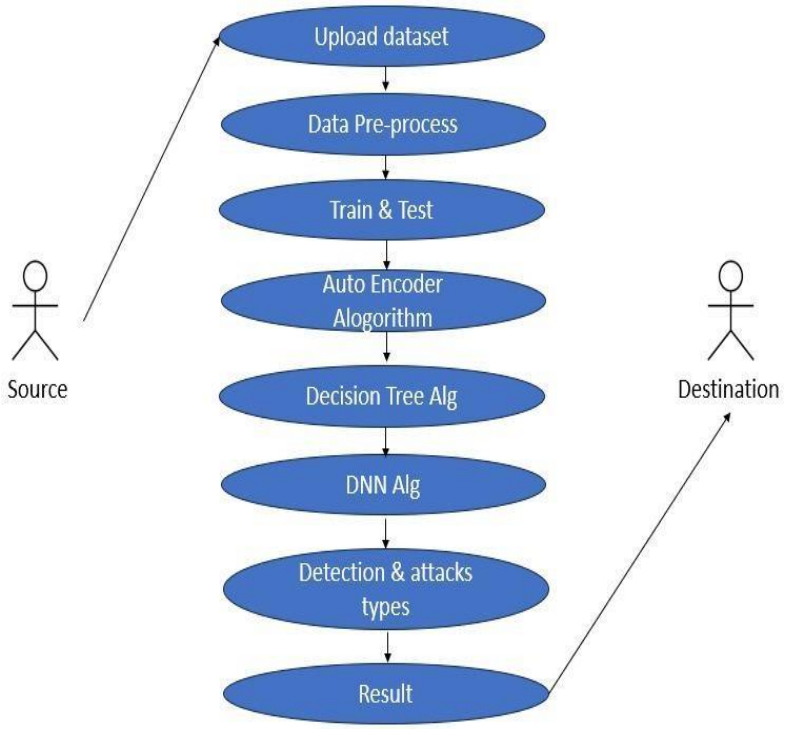


Figure 5.1 Use case Diagram Source

This is the starting point of the process.

Upload dataset The oval labeled "Upload dataset" suggests that data is being loaded into the system.

Data Pre-process After uploading the dataset, there's a step for data pre-processing. This likely involves cleaning, transforming, and preparing the data for further analysis.

Train & Test The next step involves training and testing machine learning models.

Three algorithms are shown in parallel

Auto Encoder Algorithm

Decision Tree Algorithm

DNN Algorithm

To identify the Accuracy ,Precision ,Recall and f1_Score

Detection & attacks types These trained models are then used for detecting and classifying different types of attacks. This could be related to cybersecurity or anomaly detection.

Result Finally, the process leads to a result, which might be the classification of attacks

or some other outcome.

Destination The process ends here.

Overall, this use case detecting and identifying attacks using machine learning techniques.

5.2.1 Activity Diagram

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. It is a type of behavioral diagram and we can depict both sequential processing and concurrent processing of activities using an activity diagram an activity diagram focuses on the condition of flow and the sequence in which it happens.

1. **Initial State** The starting state before an activity takes place is depicted using the initial state.
2. **Action or Activity State** An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.
3. **Action Flow or Control flows** Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another activity state.
4. **Decision node and Branching** When we need to make a decision before deciding the flow of control, we use the decision node. The outgoing arrows from the decision node can be labelled with conditions or guard expressions. It always includes two or more output arrows.
5. **Fork** Fork nodes are used to support concurrent activities. When we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement. We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent activity state and outgoing arrows towards the newly created activities.
6. **Join** Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.

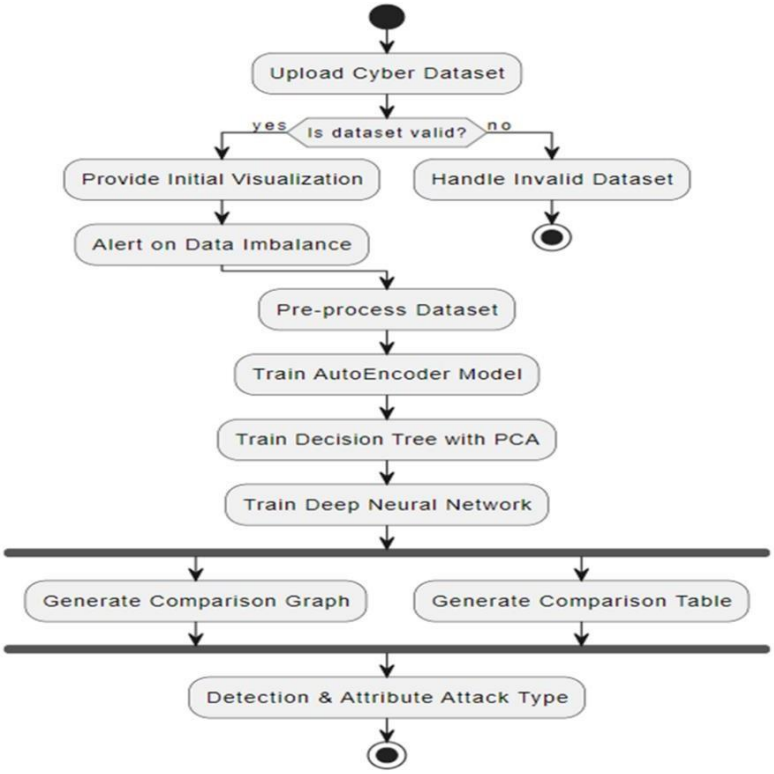


Figure 5.2 Activity Diagram

Data Set This is the starting point.

Select a dataset for your machine learning task.

Key actions include:

Select dataset():Choosing the appropriate dataset.

Import dataset(): Loading the data into your environment.

View dataset(): Exploring the dataset to understand its structure.

Data Preprocessing In this step, you clean and prepare the data for modeling.

Essential operations:

Missing data removal(): Handling missing values.

Encoding Categorical(): Converting categorical variables into numerical representations.

Feature Extraction

Extracting relevant features from the data.

Important steps:

Dataset Split Train and Test(): Dividing the dataset into training and testing subsets.

Feature Extraction(): Identifying relevant features for modeling.

Classification

Finally, you build a model and classify data.

Functions involved:

detection(): Detecting patterns or anomalies.

prediction(): Making predictions based on the model.

5.2.2 Sequence Diagram

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

1. Actors

An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.

2. Lifeline

A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.

The flow of data and interactions between components in a system designed to identify cyber attacks using machine learning in smart IoT networks. Each step ensures that the raw data from IoT devices is processed, analyzed, and acted upon efficiently to maintain network security.

Creating a sequence diagram for identifying cyber attacks using machine learning in smart IoT networks involves detailing the interactions between various components in the system.

Source

The process begins with

Upload dataset: This step involves providing the machine learning system with a dataset containing relevant information.

System: The data goes through two steps:

Clean and normalize

Here, the dataset is pre processed to remove noise, handle missing values, and ensure consistency.

Preprocessed data: The cleaned data is ready for further analysis.

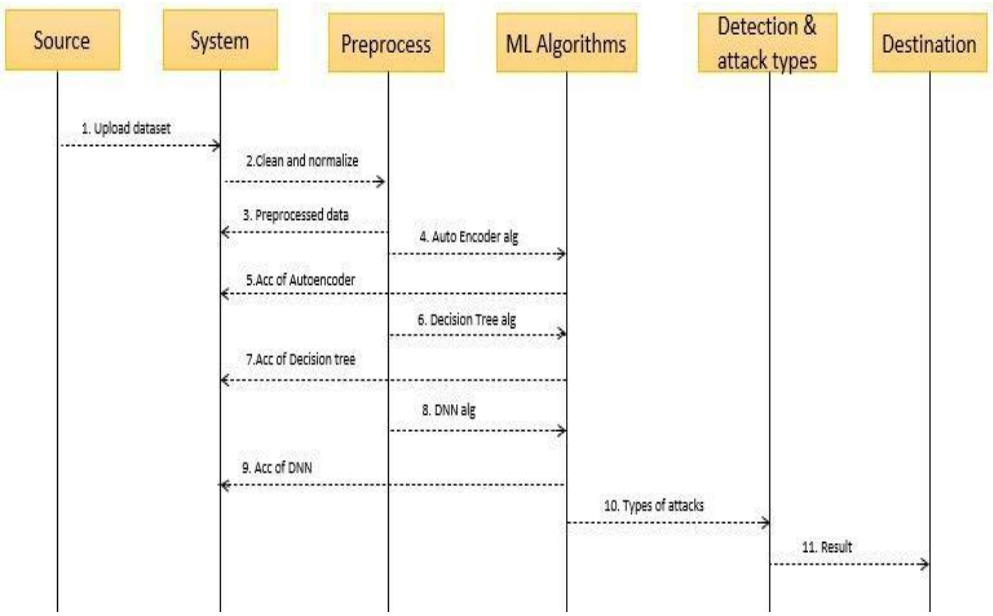


Figure 5.3 Sequence Diagram

ML Algorithm

The flowchart branches into two ML algorithms

Auto Encoder algorithm: Applied to the pre processed data.The accuracy of the Auto Encoder is evaluated .

Decision Tree algorithm: Decision trees are used for classification or regression tasks. The accuracy of the Decision Tree algorithm is assessed.

DNN algorithm: Deep Neural Networks (DNNs) are employed for more complex tasks. The accuracy of the DNN is also evaluated.

Detection & Attack Types

The final step is where the accuracy of the DNN model in detecting specific types of attacks is measured.

Destination

The process concludes with two steps:

Types of attacks This likely refers to identifying different attack categories.

Result: The overall outcome or findings based on the ML models and their accuracy.

This Sequence diagram outlines a systematic approach to using ML algorithms for detecting various cyber-attacks or anomalies in data.

5.2.4.Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their

attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

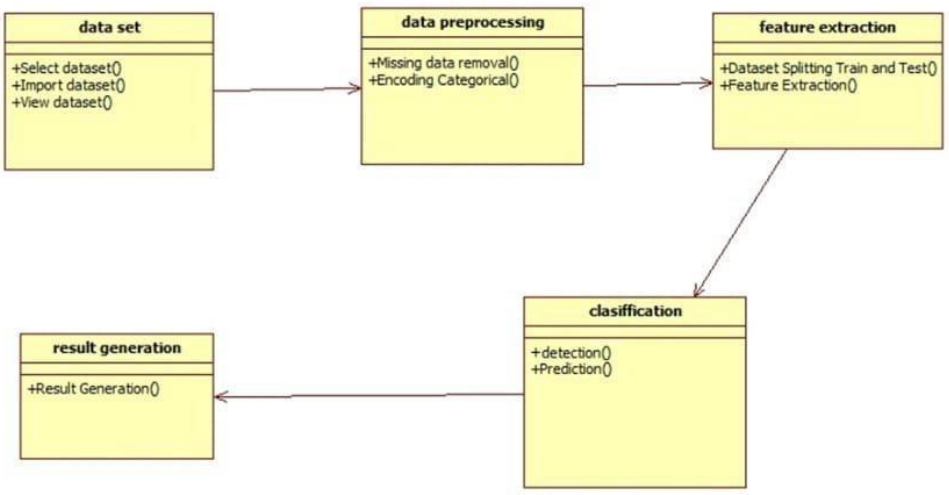


Figure 5.4 Class Diagram

Upload Cyber Attack Dataset

The process begins by uploading a dataset related to IOT releated . This dataset likely contains information about various incidents, their attributes, and outcomes.

Data Validation

The flowchart splits into two paths based on whether the dataset is valid or not.

If the dataset is valid, it proceeds to the next steps.If invalid, there might be missing values, inconsistencies, or other issues that need to be addressed.

Provide Initial Visualization

After validation, the dataset is visualized to gain insights.This step helps identify patterns, outliers, and potential areas of interest.

Alert on Data Imbalance

Imbalanced datasets (where some classes have significantly more samples than others) can affect model performance.An alert is raised if the dataset suffers from class imbalance.

Pre-process Dataset

Data preprocessing involves cleaning, transforming, and preparing the dataset for model training.Common steps include handling missing values, scaling features, and encoding categorical variables.

Model Training

The flowchart shows two models being trained:

Auto Encoder Model: An unsupervised neural network used for feature extraction and

anomaly detection.

Decision Tree with PCA: A supervised model that uses Principal Component Analysis (PCA) for dimensionality reduction.

Both models learn from the pre-processed data.

Comparison Graph and Table

The results from both models are compared.A graph (possibly showing performance metrics) and a table (listing relevant statistics) are generated.

Detection & Attribute Attack Type

Finally, the flowchart concludes by detecting and attributing the type of cyber attack based on the trained models' predictions.

CHAPTER-06

Implementation

6.1 Source Code

```
from tkinter import
    from tkinter
    from tkinter import
    import
    from tkinter import
    from tkinter.filedialog import
    import numpy as
    import matplotlib.pyplot
    import pandas as
    from sklearn.metrics import
    from sklearn.model_selection import
    import
    from sklearn.metrics import
    from sklearn.metrics import
    from sklearn.metrics import
    import
    import
    from sklearn.decomposition
    from sklearn.tree import
    from sklearn.preprocessing import
    import
    from keras import
    from keras.models import
    from keras.utils.np_utils import
    from keras.models import
    from sklearn.neural_network import
```

```
global filename, autoencoder, decision_tree, dnn,
    global
    global
    global accuracy, precision, recall, fscore,
    global X_train, X_test, y_train, y_test,
    labels = ['Normal', 'Naive Malicious Response Injection (NMRI)', 'Complex Malicious', 'Response Injection (CMRI)', 'Malicious State Command Injection (MSCI)',
        'Malicious Parameter Command Injection (MPCI)', 'Malicious Function Code Injection (MFCI)', 'Denial of Service (DoS)']
    main =
    main.title("Identification of Cyber Attack in Network using Machine Learning Techniques") #designing main screen
    main.geometry("1300x12
    #fucntion to upload
    def
      global filename,
      text.delete('1.0',
      filename = filedialog.askopenfilename(initialdir="Dataset") #upload
      text.insert(END,filename+"
      dataset = pd.read_csv(filename) #read dataset from
      text.insert(END,"Dataset
      text.insert(END,str(dataset.hea
      text.update_idletas
      unique, count = np.unique(dataset['result'],
      height =
      bars =
      print(heig
      print(bar
      y_pos =
      plt.bar(y_pos,
      plt.xticks(y_pos,

  plt.xticks(rotation=
```

```
    plt.title("Various Cyber-Attacks Found in Dataset") #plot graph with
    plt.show
def
    text.delete('1.0',
    global dataset,
    global X_train, X_test, y_train,
    #replace missing values
    dataset.fillna(0, inplace =
    scaler = MinMaxScaler() #min max scaling for datset
    with open('model/minmax.txt', 'rb')
        scaler =
    file.close
    dataset =
    X =
    Y =
    indices =
    np.random.shuffle(indices) #shuffle
    X =
    Y =
    Y =
    X =
    text.insert(END,"Dataset after features
    text.insert(END,str(X)+"\n
    text.insert(END,"Total records found in dataset :
    text.insert(END,"Total features found in dataset:
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
    text.insert(END,"Dataset Train and Test
    text.insert(END,"80% dataset records used to train ML algorithms :
"+str(X_train.shape[0])+"\n")
```

```
def calculateMetrics(algorithm, predict,
    a =
    p = precision_score(y_test,
    r = recall_score(y_test,
    f = f1_score(y_test,
    accuracy.append
    precision.append
    recall.append
    fscore.append
    text.insert(END,algorithm+" Accuracy  :
    text.insert(END,algorithm+" Precision :
    text.insert(END,algorithm+" Recall  :
    text.insert(END,algorithm+" FScore:
def
    text.delete('1.0',
    global X_train, X_test, y_train,
    global
    global accuracy, precision, recall,
    accuracy =
    precision =
    recall =
    fscore =
    if
        with open('model/encoder_model.json', "r") as
            loaded_model_json =
            autoencoder =
        json_file.clos
        autoencoder.load_weights("model/encoder_model_wei
  text.insert(END,"20% dataset records used to train ML algorithms :
"+str(X_test.shape[0])+"\n")
```

```
autoencoder._make_predict_func
    else
        encoding_dim = 256 # encoding dimesnion is 32 which means each row will be filtered 32 times to get important features from dataset
        input_size = keras.Input(shape=(X.shape[1],)) #we are taking
        encoded = layers.Dense(encoding_dim, activation='relu')(input_size) #creating dense layer to start filtering dataset with given 32 filter dimension
        decoded = layers.Dense(y_train.shape[1], activation='softmax')(encoded) #creating another layer with input size as 784 for encoding
        autoencoder = keras.Model(input_size, decoded) #creating decoded layer to get prediction result
        encoder = keras.Model(input_size, encoded)#creating encoder object with encoded and input images
        encoded_input = keras.Input(shape=(encoding_dim,))#creating another layer for same input dimension
        decoder_layer = autoencoder.layers[-1] #holding
        decoder = keras.Model(encoded_input, decoder_layer(encoded_input))#merging last layer with encoded input layer
        autoencoder.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])#compiling model
        hist = autoencoder.fit(X_train, y_train, epochs=300, batch_size=16, shuffle=True, validation_data=(X_test, y_test))#now start generating model with given Xtrain as input
        autoencoder.save_weights('model/encoder_model_weights.h5')#above line for creating model will take 100 iterations
        model_json = autoencoder.to_json()
        with open("model/encoder_model.json", "w") as
          json_file.write(model_j
        json_file.clo
    print(autoencoder.summary())#printing model
    predict =
    predict = np.argmax(predict,
    testY = np.argmax(y_test,
    calculateMetrics("AutoEncoder", predict,
def
```

```
global autoencoder, decision_tree,
    global X_train, X_test, y_train, y_test, X,
    encoder_model = Model(autoencoder.inputs, autoencoder.layers[-1].output)#creating autoencoder model
    vector = encoder_model.predict(X)  #extracting features using
    pca = PCA(n_components = 7) #applying PCA for features
    vector =
    Y1 = np.argmax(Y,
    X_train, X_test, y_train, y_test = train_test_split(vector, Y1,
    decision_tree = DecisionTreeClassifier() #defining
    decision_tree.fit(vector, Y1) #training with
    predict =
    text.insert(END,"Decision Tree Trained on New Features Extracted from AutoEncoder\n")
    calculateMetrics("Decision Tree", predict,
def
    global autoencoder, decision_tree, encoder_model,
    global X_train, X_test, y_train,
    attack_type =
    for i in
        temp =
        temp.append(vector
        attack = decision_tree.predict(np.asarray(temp)) #using decision tree we are predicting attack type
        attack_type.append(attac
    attack_type =
    X_train, X_test, y_train, y_test = train_test_split(vector, attack_type,
    dnn = MLPClassifier() #defining DNN
    dnn.fit(vector, attack_type) #train DNN with various
    predict = dnn.predict(X_test) #predict label forr
    text.insert(END,"Attack Prediction using
```

```
calculateMetrics("DNN", predict,
    def
      text.delete('1.0',
      global autoencoder, decision_tree, encoder_model,
      filename =
      dataset =
      dataset.fillna(0, inplace =
      values =
      temp =
      temp =
      test_vector = encoder_model.predict(temp)  #extracting features using
      test_vector =
      print(test_vector.sha
      predict =
      for i in
        if predict[i] ==
            text.insert(END,"New Test Data : "+str(values[i])+" ====> NO CYBER ATTACK DETECTED\n\n")
        else
            text.insert(END,"New Test Data : "+str(values[i])+" ====> CYBER ATTACK DETECTED Attribution Label : "+str(labels[predict[i]])+"\n\n")
    def


    df      =
pd.DataFrame([['AutoEncoder','Precision',precision[0]],['AutoEncoder','Recall',recall[0]
],['AutoEncoder','F1 Score',fscore[0]],['AutoEncoder','Accuracy',accuracy[0]],

['Decision Tree with PCA','Precision',precision[1]],['Decision Tree with
PCA','Recall',recall[1]],['Decision Tree with PCA','F1 Score',fscore[1]],['Decision Tree
with PCA','Accuracy',accuracy[1]],

            ['DNN','Precision',precision[2]],['DNN','Recall',recall[2]],['DNN','F1
Score',fscore[2]],['DNN','Accuracy',accuracy[2]],

            ],columns=['Algorithms','Performance
      df.pivot("Algorithms", "Performance Output",
```

```
plt.show
    def
    output = "<html><body><table align=center border=1><tr><th>Algorithm Name</th><th>Accuracy</th><th>Pr+ecision</th><th>Recall</th>"
    output+="<th>FSCORE</th>

output+="<tr><td>AutoEncoder</td><td>"+str(accuracy[0])+"</td><td>"+str(precision[0])+"</td><td>"+str(recall[0])+"</td><td>"+str(fscore[0])+"</td></tr>"

output+="<tr><td>Decision Tree with PCA</td><td>"+str(accuracy[1])+"</td><td>"+str(precision[1])+"</td><td>"+str(recall[1])+"</td><td>"+str(fscore[1])+"</td></tr>"

output+="<tr><td>DNN</td><td>"+str(accuracy[2])+"</td><td>"+str(precision[2])+"</td><td>"+str(recall[2])+"</td><td>"+str(fscore[2])+"</td></tr>"
    output+="</table></body></ht
    f = open("table.html",
    f.write(outp
    f.close
    webbrowser.open("table.html",ne
font = ('times', 16,
title = Label(main, text='Detection of Cyber Attack in Network using Machine Learning Techniques')
title.config(bg='greenyellow',
title.config(font=f
title.config(height=3,
title.place(x=0,y
font1 = ('times', 12,
text=Text(main,height=20,width
scroll=Scrollbar(te
text.configure(yscrollcommand=scro
text.place(x=50,y=1
text.config(font=fo
font1 = ('times', 13,
```

```
uploadButton = Button(main, text="Upload SWAT Water Dataset", command=uploadDataset)
    uploadButton.place(x=50,y=
    uploadButton.config(font=f
    processButton = Button(main, text="Preprocess Dataset",
    processButton.place(x=330,y=
    processButton.config(font=f
    autoButton = Button(main, text="Run AutoEncoder Algorithm", command=runAutoEncoder)
    autoButton.place(x=630,y=
    autoButton.config(font=fo
    dtButton = Button(main, text="Run Decision Tree with PCA", command=runDecisionTree)
    dtButton.place(x=920,y=
    dtButton.config(font=fo
    dnnButton = Button(main, text="Run DNN Algorithm",
    dnnButton.place(x=50,y=
    dnnButton.config(font=fo
    attributeButton = Button(main, text="Detection & Attribute Attack Type", command=attackAttributeDetection)
    attributeButton.place(x=330,y
    attributeButton.config(font=f
    graphButton = Button(main, text="Comparison Graph",
    graphButton.place(x=630,y=
    graphButton.config(font=f
    tableButton = Button(main, text="Comparison Table",
    tableButton.place(x=920,y=
    tableButton.config(font=fo
    main.config(bg='LightSkyB
    main.mainloop
```

Code Explanation

Import Statements

The code begins with several import statements. These are used modules and functions from Python libraries.For example, from tkinter all functions and classes from the tkinter library, which is commonly used graphical user interfaces (GUIs).Other import statements include numpy, pandas, sklearn, keras, and

Global Variables

The code defines several global variables such as filename, autoencoder, dnn, encoder_model, pca, X, Y, dataset, accuracy, precision, recall, X_train, X_test, y_train, y_test, and scaler.These variables are used to store data, models, and other

Tkinter GUI Setup

The main GUI window is created using tkinter.Tk().The title of the "Identification of Cyber Attack in Network using Machine Learning window dimensions are set to 1300x1200

Function Definitions

The code defines two

uploadDataset(): uploadDataset() allows the user to select a dataset file using The selected file is then read into a Pandas DataFrame (dataset).The loaded filename and the first few rows of the dataset.It also generates a bar the distribution of various cyber attacks found in

preprocessing(): It performs data preprocessing 0.Normalizes features using Min-Max scaling.Shuffles the dataset.Splits training and testing sets.Converts labels to

Visualization

The code includes a visualization step where it plots a bar chart showing of different cyber attacks in the

Function Definition:The calculateMetrics function takes three predict, and y_test.algorithm represents the name of the machine learning

evaluated.predict contains the predicted labels (output) from the
contains the true labels (ground truth) for the test
Calculating Metrics:Inside the function, the following metrics are
Accuracy: The percentage of correctly predicted instances out of the
Precision: The ability of the model to correctly predict positive instances
relative to all predicted positive instances (true positives + false
Recall: The ability of the model to correctly predict positive instances
relative to all actual positive instances (true positives + false
F1 Score: The harmonic mean of precision and recall, which balances both
metric is multiplied by 100 to express it as a
Appending Metrics to Lists:The calculated accuracy, precision, recall, and
appended to global lists (accuracy, precision, recall, and fscore).These
metrics for multiple algorithms or iterations.Updating the Text
inserts the calculated metrics into a text widget (text) using the insert
is: <algorithm> <Metric>: <Value> (e.g., “Decision Tree Accuracy:
argument ensures that the new text is added at the end of the

6.2 Read CSV file

The attacks found in dataset and dataset contains above labels as integer value of

its index for example NORMAL label index will be 0 and continues up to 8 class labels. Below screen showing dataset details.first row contains dataset column names and remaining rows contains dataset values and in last column we have attack type from label 0 to 7. We will used above dataset to train propose Auto Encoder, decision tree and DNN algorithms. It is significant to decrease thecount of features and just use the features needed totrain and test the algorithms to find a lightweight security solution appropriate for IoT systems.

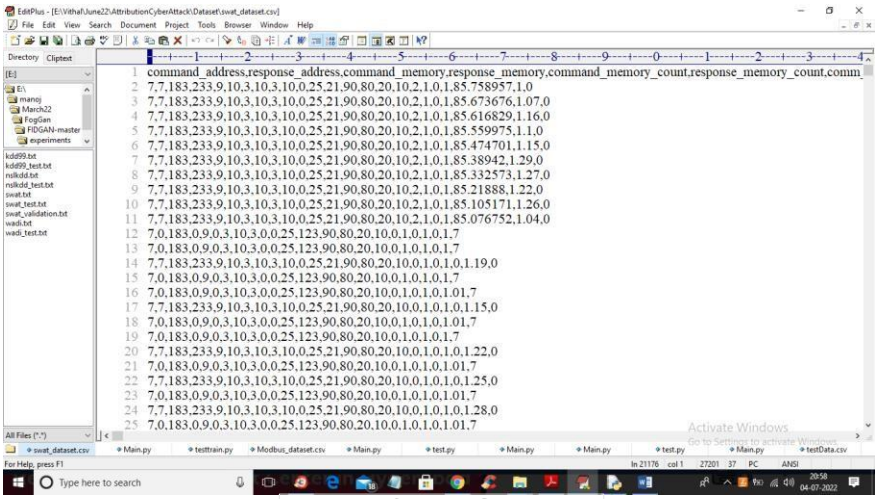


Figure 6.2.1 Csv

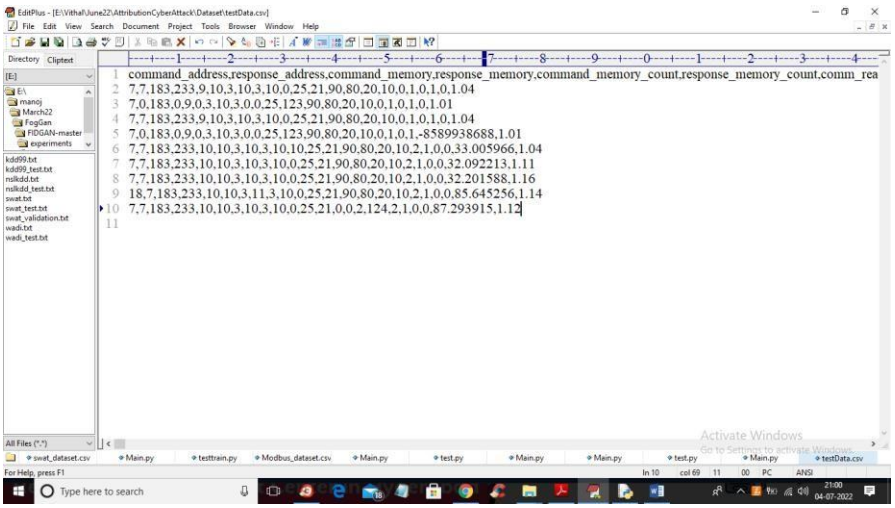
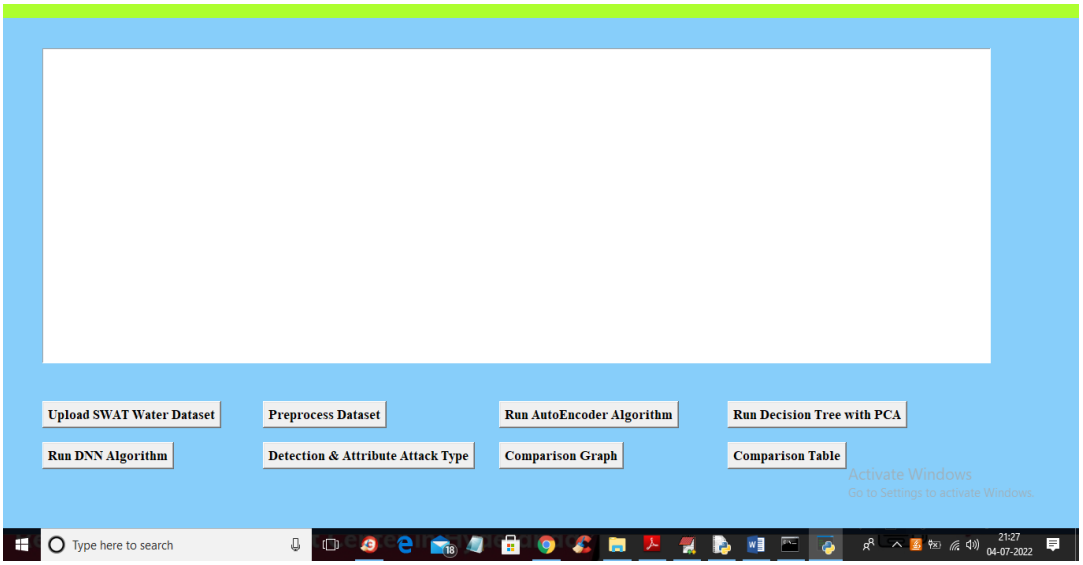


Figure 6.2.2 CSV

The whole dataset is used in this phase. Seven different methods of were implemented on the entire dataset, and we used feature sets that were each attack

6.3 Output:



1)Upload SWAT Water Dataset: Using this module we will upload dataset to application and then read dataset and then find different attacks found in dataset

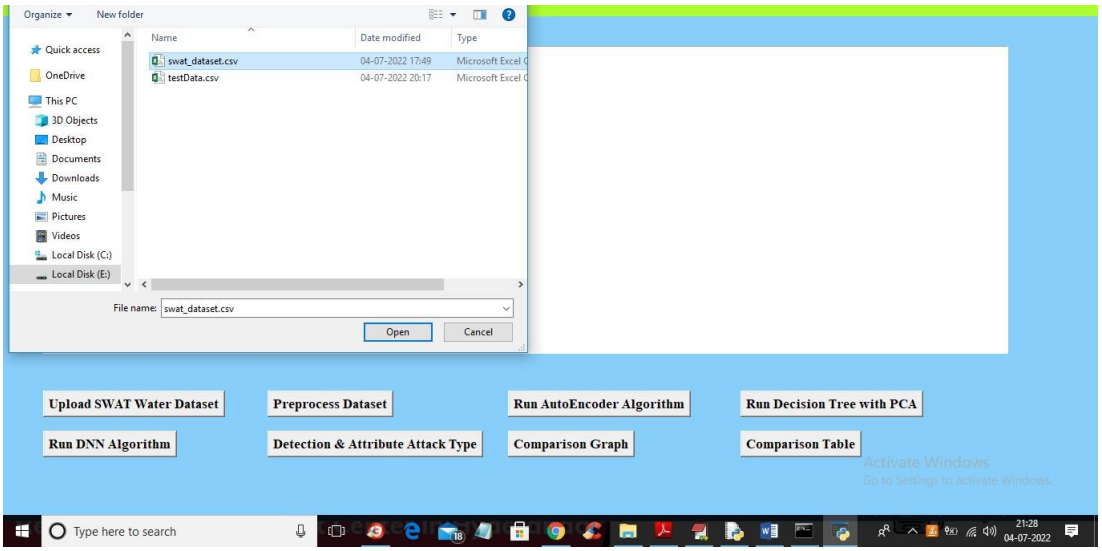


Figure 6.5 Swat Dataset

It reads the file and produces a visual document of the features extracted, and also offers a csv file of the dataset. This process was primarily designed to improve classifiers 'predictive capabilities by extracting new dataset features.

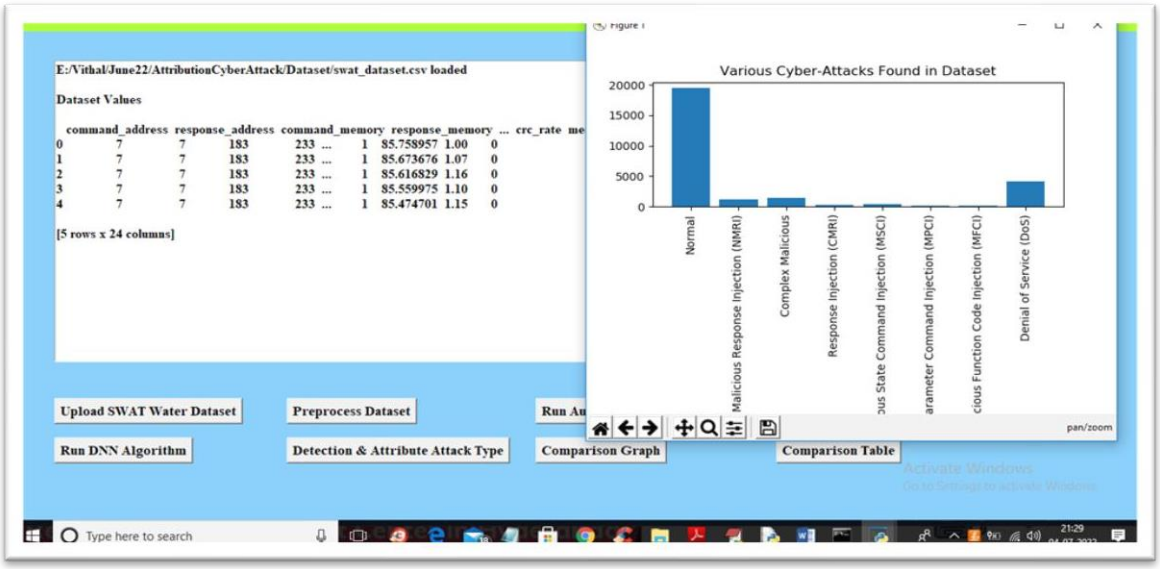


Figure 6.6 Upload swat dataset

2) Preprocess Dataset: using this module we will replace all missing values with 0 and then apply MIN-MAX scaling algorithm to normalized features values .

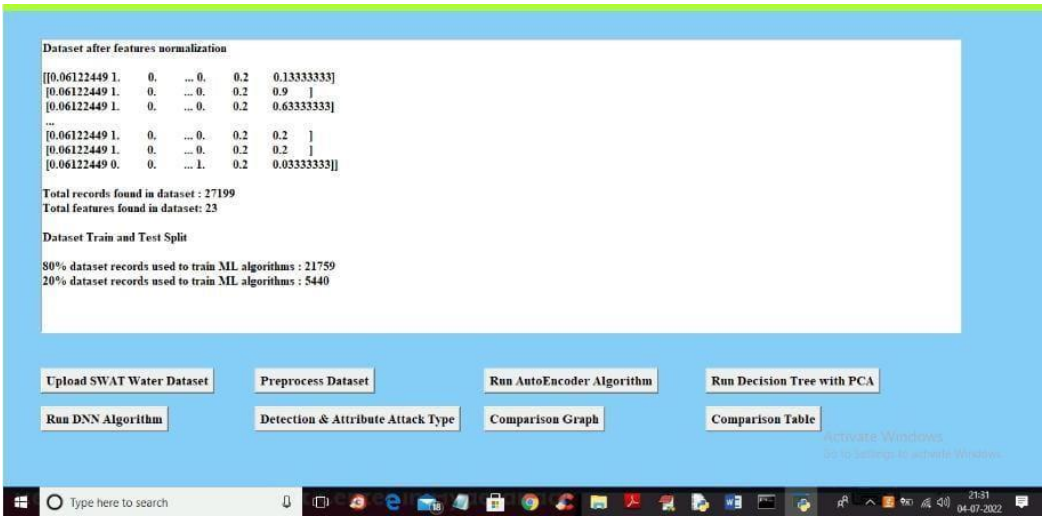


Figure 6.7: Preprocess the dataset

During the machine learning process, data are needed so that learning can take place. In addition to the data required for training, test data are needed to evaluate the performance of the algorithm min order to see how well it works. we considered 80% of the Bot-IoT dataset to be the training data and the remaining 20% to be the testing data.

Pre-processing data transformation operations are used to transform the dataset into a structure suitable for machine learning. This step also includes cleaning the dataset by

removing irrelevant or corrupted data that can affect the accuracy of the dataset, which makes it more efficient.

Applying machine learning algorithms on each attack in the dataset separately.

When evaluating the performance of machine-learning models, it is crucial to define performance measures that are suitable for the task to be solved. In order to evaluate our results.

3)Run Auto Encoder Algorithm: using this module we will tra ined Auto Encoder deep learning algorithm and then extract features from tat model.

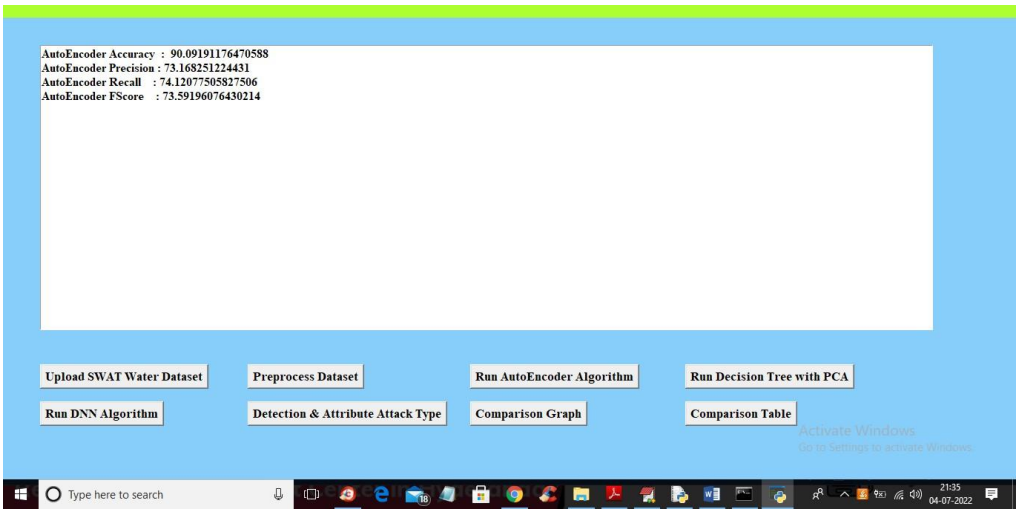


Figure 6.8 Auto Encoder

4).Run Decision Tree with PCA: extracted features from Auto Encoder will get transform.

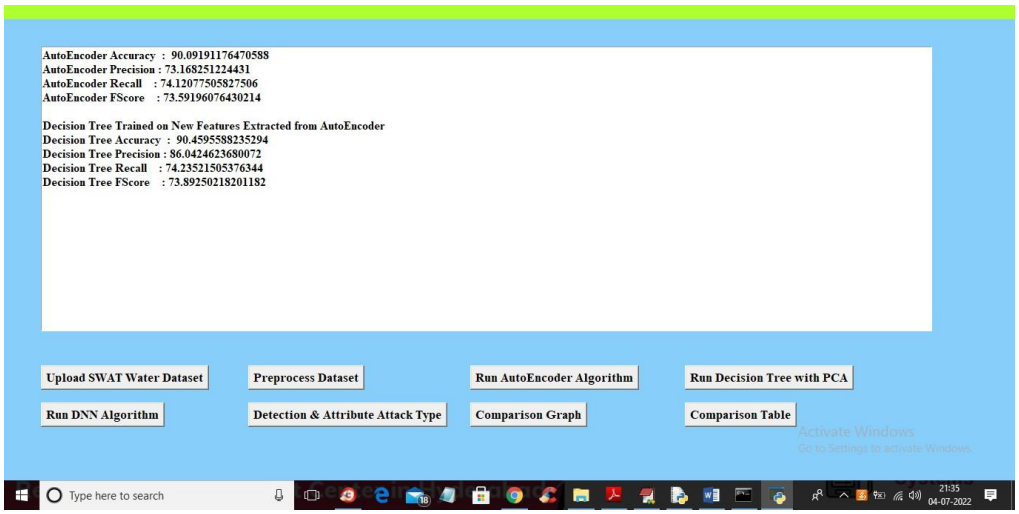


Figure 6.9 Decision tree

5) Run DNN Algorithm: predicted decision tree label will further train with DNN (deepneural network) algorithm to detect and attribute attacks.

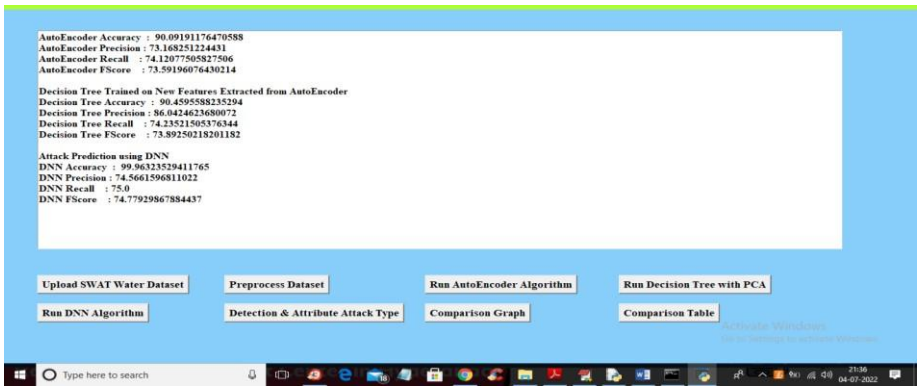


Figure 6.10: DNN algorithm

6) Detection & Attribute Attack Type: using this module we will upload unknown or un-label TEST DATA and then DNN will predict attack type.

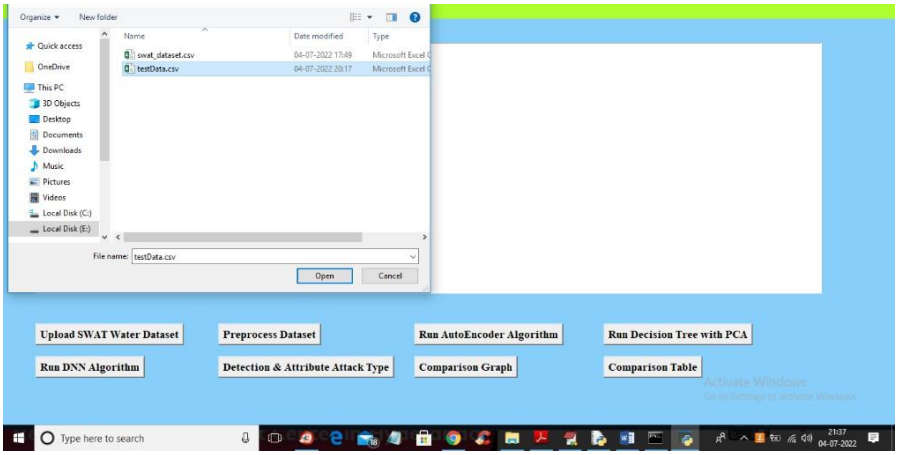


Figure 6.11 Test dataset

7) Comparison Graph: Using this module we will plot comparison graph between all algorithms.

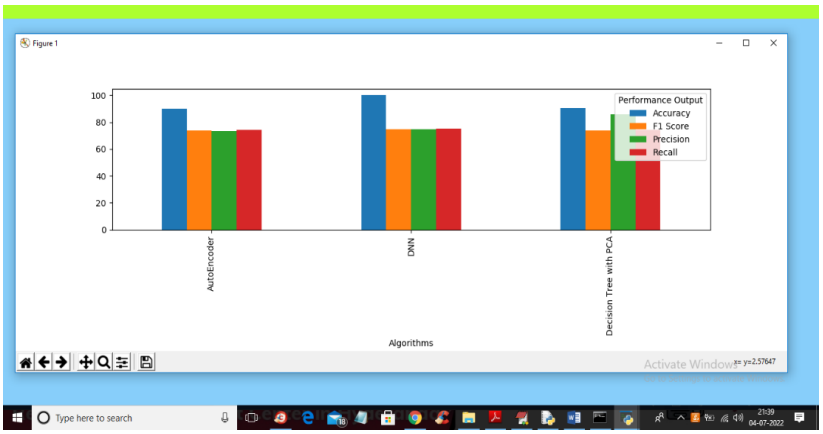
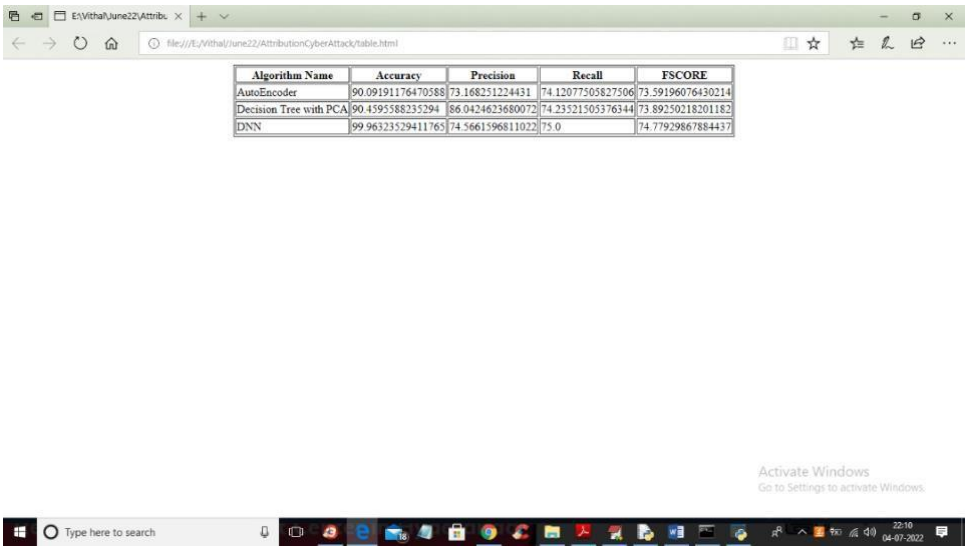


Figure 6.12 Comparison Graph

8)Comparison Table: using this module we will display comparison table of all algorithms which contains metrics like accuracy, precision, recall and FSCORE.



Algorithm Name	Accuracy	Precision	Recall	FSCORE
AutoEncoder	90.09191176470588	73.168251224431	74.12077505827506	73.59196076430214
Decision Tree with PCA	90.4595588235294	86.0424623680072	74.23521505376344	73.89250218201182
DNN	99.96323529411765	74.56615968811022	75.0	74.77929867884437

Activate Windows
Go to Settings to activate Windows.

Figure 6.13Comparison Table

Detected various attacks and now click on ‘Comparison Graph’ button to get below graph.graph x-axis represents algorithms names and y-axis represents different metric values such as precision, recall, accuracy and F_SCORE with different colour bars and in all algorithms DNN got high accuracy and now close above graph and then click on ‘Comparison Table’ to get below comparison table of all algorithms.

CHAPTER-07

Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. The following is the description of the testing strategies, which were carried out during the testing period.

7.1 System testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

7.2 Types of testings:

7.2.1 Module Testing:

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately.

7.2.2 Intergration testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

7.2.3 Acceptance testing

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

7.2.4 Behavioral Testing

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.

7.2.5 Unit testing

Unit testing is a critical component in the development of machine learning-based systems for identifying cyber attacks in smart IoT networks. It involves validating individual parts or units of the code to ensure that each one functions correctly on its own. In the context of cyber attack detection, unit testing focuses on various components such as data preprocessing scripts, feature extraction functions, individual machine learning model components, and utility functions.

Data preprocessing is a foundational step in preparing raw IoT network data for machine learning models. Unit tests for data preprocessing functions ensure that operations like data cleaning, normalization, and transformation are performed accurately.

Feature extraction functions are another vital area for unit testing. These functions are responsible for converting raw data into meaningful features that machine learning

models can utilize. Unit tests can validate that these functions correctly derive features such as IP addresses, port numbers, protocol types, and payload sizes from the raw network traffic data. Ensuring the accuracy of these features is essential for the model's performance in detecting anomalies and cyber attacks.

When it comes to the machine learning models themselves, unit tests focus on individual components and algorithms used within these models.Unit testing in the identification of cyber attacks using machine learning in smart IoT networks is essential for validating the functionality of individual components. . This forms a robust foundation for building and maintaining effective cyber attack detection systems that can safeguard smart IoT networks against evolving threats.

7.3 Test cases

Testing a system for identifying cyber attacks using machine learning in smart IoT networks requires a variety of test cases to ensure the system performs accurately and efficiently under different scenarios.

Here are detailed test cases that cover various aspects of the system:

Table 7.1: Test cases

Test Case Id	Test case name	Test case description	Test steps			Test case status	Test priority
			Step	Expected	Actual		
01	Upload the tasks dataset	Verify the file is loaded or not	If dataset is not uploaded	It cannot display the file uploaded	File is uploaded which display a task waiting time	High	High
02	Upload patients dataset	Verify either dataset loaded or not	If dataset is not upload	It cannot display dataset reading process complete	It can display dataset reading process completed	Low	High
03	Preprocessing	Whether preprocessing on the dataset applied or not	If not applie d	It cannot display the	It can display the necessa ry data for further process	Medium	High

				necessary data for further Process			
04	Prediction Random Forest	Whether Prediction algorithm applied on the data or not	If not applied	Random tree is not generated	Random tree is generated	High	High
05	Recommendation	Whether predicted data is displayed or not	If not displayed	It cannot view prediction containing patient Data	It can view prediction containing patient Data	High	High
06	Noisy Records Chart	Whether the graph is displayed or not	If graph is not displayed	It does not show the variations in between clean and noisy Records	It shows the variations in between clean and noisy records	Low	Medium

The system detects and mitigates attacks without compromising its functionality.

These test cases cover a comprehensive range of scenarios to ensure the reliability,

accuracy, and robustness of the system for identifying cyber attacks using machine learning in smart IoT networks. Conducting these tests helps validate that the system performs well under various conditions and meets security and performance requirements.

CHAPTER-08
RESULT

Machine learning (ML) in smart IoT networks has been increasingly utilized for the identification of cyber attacks due to its capability to analyze vast amounts of data and detect patterns indicative of malicious activities.

some malicious function command injection found :

Navie Malicious Response Injection:

Naive malicious response injection refers to a simplistic or straightforward method of injecting malicious content into a response generated by a web application or service. This type of attack typically targets vulnerabilities in the application's output mechanisms, such as HTML, JavaScript, or other markup languages, with the goal of executing unauthorized actions or compromising the security of the system or its users.

Complex Malicious:

"Complex injection" typically refers to more sophisticated and intricate methods of injecting malicious code or commands into vulnerable systems or applications , which rely on straightforward exploitation of known vulnerabilities, complex injection attacks often involve advanced evasion techniques or manipulation of complex data structures to bypass security controls and achieve their objectives.

Response Injection:

"Response injection" refers to a type of cyber attack where an attacker injects malicious content into the response generated by a web application or service which involve injecting malicious content into the input fields or parameters of a request, response injection attacks occur when the application fails to properly sanitize or validate user-generated content before including it in the response sent back to the client.

Malicious State Command Injection:

"Malicious State Command Injection" refers to a type of cyber attack where an attacker exploits vulnerabilities in the management of application or system state to inject and execute malicious commands. This attack is a variation of command injection, where the attacker manipulates the state of an application or system to execute unauthorized commands, rather than directly injecting commands into input fields or parameter

Malicious Function Command Injection:

Function command injection is a type of cyber attack where an attacker injects malicious

code into a function or command within an application, often with the goal of executing unauthorized commands or manipulating the behavior of the application or system.

Dos :

A Denial-of-Service (DoS) attack is a malicious attempt to disrupt the normal functioning of a targeted server, service, or network by overwhelming it with a flood of illegitimate traffic or resource requests.

Normal:

It refers to legitimate activities and behaviors within the IoT network that are considered typical or expected under normal operating conditions.

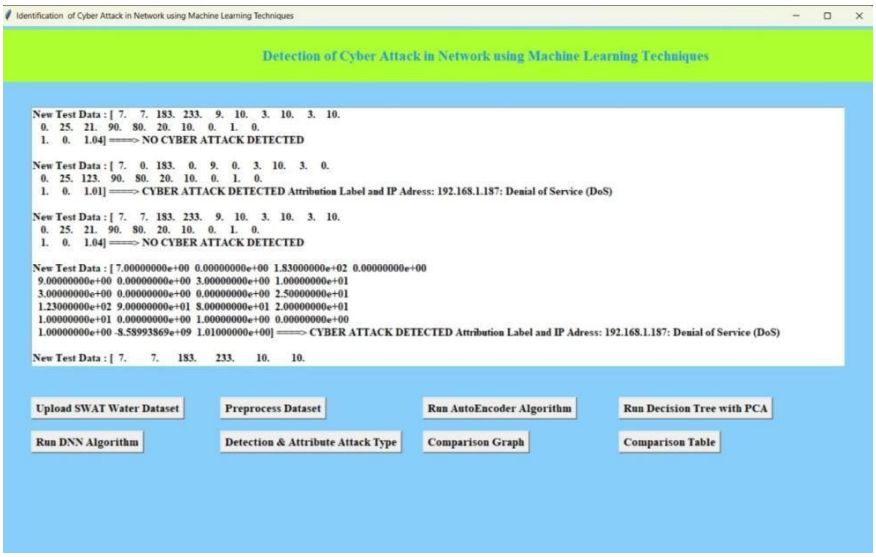


Figure 8.1: Result.

The above paragraph specify that the attacks are identified with in the specified conditions.

In the results of the algorithms, the following values are examined in order detect or identify the accuracy, precision, recall, and f_Score.

CHAPTER-09
CONCLUSION

Identifying cyber attacks in smart IoT networks through machine learning presents a promising approach to bolstering network security in an increasingly interconnected world. Through our exploration of this methodology, several key conclusions emerge.

Firstly, the integration of machine learning techniques offers a dynamic and adaptive means of discerning normal network behavior from potentially malicious activities. By analyzing vast amounts of data generated by IoT devices, machine learning models can learn intricate patterns and anomalies, enabling them to detect and respond to emerging threats in real-time.Furthermore, the effiency of machine learning in cyber attack identification hinges on the quality and diversity of the data available for training. Robust datasets that encompass a wide range of network activities and attack scenarios are essential for enhancing the accuracy and reliability of detection algorithms. Additionally, ongoing data collection and model refinement are crucial for adapting to evolving cyber threats and maintaining a high level of detection efficacy over time. Moreover, the deployment of machine learning-based detection systems in smart IoT networks necessitates careful consideration of scalability, resource constraints, and privacy concerns. Implementing lightweight and efficient algorithms capable of running on resource-constrained IoT devices is essential for minimizing computational overhead and ensuring seamless integration into existing network infrastructure. The application of machine learning in the identification of cyber attacks within smart IoT networks offers a promising approach to enhancing security posture and mitigating risks associated with increasingly complex and interconnected IoT environments.

By leveraging machine learning algorithms trained on historical data to recognize patterns of normal behavior, organizations can detect deviations indicative of cyber attacks, including intrusion attempts, data exfiltration, malware infections, and denial-of-service (DoS) attacks. Furthermore, machine learning enables proactive threat detection and response by continuously monitoring and analyzing real-time data streams from IoT devices, enabling organizations to swiftly identify and mitigate security incidents before they escalate. However, effective implementation of machine learning-based security solutions requires addressing challenges such as data privacy, model interpretability, and

adversarial attacks, while also ensuring seamless integration with existing security infrastructure and compliance with regulatory requirements. Moving forward, ongoing research and development efforts are necessary to advance the capabilities of machine learning in smart IoT security and stay ahead of evolving cyber threats.

Cyber attacks pose a significant and growing threat in our interconnected digital world. As technology advances, so do the capabilities and tactics of malicious actors, ranging from individual hackers to sophisticated state-sponsored groups. The consequences of cyber attacks can be severe, ranging from financial losses and data breaches to disruption of critical infrastructure and even endangering lives.To mitigate the risks posed by cyber attacks, it's essential for individuals, organizations, and governments to prioritize cybersecurity measures. This includes implementing robust security protocols, regularly updating software and systems, conducting thorough risk assessments, and educating users about best practices for online safety.

CHAPTER-10

Future Enhancement

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. The power of artificial intelligence, organizations can enhance their ability to detect and mitigate cyber threats, thereby safeguarding the integrity, confidentiality, and availability of IoT-driven services and applications in the digital age. However, continual research, collaboration, and innovation are imperative to stay ahead of adversaries and effectively counter emerging cyber threats in the ever-evolving landscape of IoT security.

Some of the future enhancements that can be done to this system are:

Multi-Modal Data Fusion: Incorporate multiple sources of data, such as network traffic logs, device logs, sensor readings, and metadata, to improve the accuracy and robustness of cyber attack detection.

Anomaly Detection: Develop anomaly detection algorithms to identify abnormal behavior patterns in IoT networks, enabling proactive detection of emerging cyber threats and zero-day attacks.

Explainable AI: Enhance the interpretability and transparency of machine learning models by integrating explainable AI techniques to provide insights into the decision-making process and facilitate human understanding and trust in the system.

Dynamic Adaptation: Develop adaptive algorithms that can dynamically adjust their behavior and parameters based on changing environmental conditions, evolving cyber threats, and feedback from cybersecurity analysts to ensure continuous improvement and optimization.

Privacy-Preserving Techniques: Integrate privacy-preserving methods such as federated learning, differential privacy, and secure multiparty computation to protect sensitive data while still enabling collaborative model training across distributed IoT networks.

Edge Computing Integration: Explore the integration of edge computing technologies to perform data preprocessing, model inference, and decision-making closer to IoT

devices, reducing latency, bandwidth usage, and dependence on centralized cloud resources.

In conclusion, while significant strides have been made in identifying cyber attacks using machine learning in smart IoT networks, there remains substantial potential for future enhancements. Continued advancements in machine learning algorithms, particularly in the realms of deep learning and reinforcement learning, promise to increase the accuracy and speed of anomaly detection. Integrating more sophisticated, context-aware models that can understand the intricacies of IoT environments will be crucial.Finally, developing more robust and adaptive systems capable of evolving with emerging threats will be essential in maintaining resilient IoT networks in the face of an ever-changing cyber threat landscape.

REFERENCES

[1]The White House, “Making college affordable,” https:// www:whitehouse:gov/issues/education/higher-education/ making-college-affordable, 2016.

[2] Complete College America, “Four-year myth: Making college more affordable,” http://completecollege:org/wp-content/uploads/2014/ 11/4-Year-Myth:pdf, 2014.

[3] H. Cen, K. Koedinger, and B. Junker, “Learning factors analysis–a general method for cognitive model evaluation and improvement,” in International Conference on Intelligent Tutoring Systems. Springer, 2006, pp. 164–175.

[4] M. Feng, N. Heffernan, and K. Koedinger, “Addressing the assessment challenge with an online system that tutors as it assesses,” User Modeling and User-Adapted Interaction, vol. 19, no. 3, pp. 243–266, 2009.

[5] H.-F. Yu, H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. McKenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, Y.-H. Wei et al., “Feature engineering and classifier ensemble for kdd cup 2010,” in Proceedings of the KDD Cup 2010 Workshop, 2010, pp. 1–16.

[6]Y. Meier, J. Xu, O. Atan, and M. van der Schaar, “Personalized grade prediction: A data mining approach,” in Data Mining (ICDM), 2015 IEEE International Conference on. IEEE, 2015, pp. 907–912.

[7] C. G. Brinton and M. Chiang, “Mooc performance prediction via clickstream data and social learning networks,” in 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2015, pp. 2299– 2307.

[8] KDD Cup, “Educational data minding challenge,” https://pslcdatashop: web:cmu:edu/KDDCup/, 2010.

[9] Y. Jiang, R. S. Baker, L. Paquette, M. San Pedro, and N. T. Heffernan, “Learning, moment-by-moment and over the long term,” in International Conference on Artificial Intelligence in Education. Springer, 2015, pp. 654–657.

[10] C. Marquez-Vera, C. Romero, and S. Ventura, “Predicting school failure using data mining,” in Educational Data Mining 2011, 2010.

[11] Y.-h. Wang and H.-C. Liao, “Data mining for adaptive learning in a tesl-based e- learning system,” Expert Systems with Applications, vol. 38, no. 6, pp. 6480–6485, 2011.

[12] N. Thai-Nghe, L. Drumond, T. Horvath, L. Schmidt-Thieme ´ et al., “Multi-relational factorization models for predicting student performance,” in Proc. of the KDD Workshop on Knowledge Discovery in Educational Data. Citeseer, 2011.

Annexure – I

List of Figures

Annexure –II

List of Symbols/Acronyms

S No.	Abbreviation	Full form
1.	ML	Machine Learning
2.	IoT	Internet of Things
3.	Bot-IoT	Robot Network
4.	SIEM	Security Information And Event Management
5.	IDS	Intrusion Detection System
6.	DNN	Deep Neural Network
7.	MLP	Multi-layer Perceptron
8.	ANN	Artifical Neural Network
9.	IDPS	Intrusion detection and prevention system
10.	NIST	National Institute of Standards and Technology
11.	PCA	Principal Component Analysis
12.	csv	Comma-separated values
13.	GUI	Graphical User Interface
14.	UML	Unified Modeling Language
15.	DoS	Denial-of-Service

Annexure – III

List of tables