PREDICTION CYBER-ATTACK AND IDENTIFYING PERPETRATORS USING MACHINE LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

DINESH KUMAR S (312820205009)

SACHIN L (312820205031)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

AGNI COLLEGE OF TECHNOLOGY

THALAMBUR

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **PREDICTION CYBER-ATTACK AND IDENTIFYING PERPETRATORS USING MACHINE LEARNING** is the bonafide work of **D I N E S H K U M A R S (312819205009), SACHIN L (312819205031)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. S. GEERTHIK, M.E, Ph.D.	Mrs. G. Tharagai Rani, M. E,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Information Technology	Department of Information Technology
Agni College Of Technology,	Agni College Of Technology,
Thalambur, Chennai – 600130	Thalambur, Chennai - 600130

Submitted for the project viva held on …………………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to the management of "**AGNI COLLEGE OF TECHNOLOGY"** and would like to thank our respected Principal **Dr. SRINIVASAN ALAVANDAR** for his words of inspiration and for providing necessary facilities to carry out our project work successfully.

We are immensely thankful tour guide **Dr. S. GEERTHIK, M.E, Ph.D.,** Associate professor and Head of the Department, Information Technology for his words of wisdom and his constant source of inspiration.

I would like to offer our heartfelt thanks to our guide **Mrs. G. Tharagai Rani, M.E,** Associate professor Department of Information Technology who molded us accordingly and gave valuable suggestion for completing our project work successfully.

We extended our warmest thanks to all the faculty members of our department for their assistance and we also thank all our friends who helped us in bringing out our project in good shape and form.

Finally, we express our sincere benevolence to our beloved parents for theirperpetual encouragement and support in all endeavours.

ABSTRACT

The generation of technology has grown significantly in recent years, evolve. As a result, as technology advances, the number of crimes also rises. It will result in financial difficulties for the nation and its citizens. Cyberattacks account for the majority of reported attacks. It will target the privacy data of the people and the nation. As a result, it is challenging to determine what kind of attack would be employed, and the length of the inquiry would also rise. Numerous systems have been presented that use different machine-learning techniques to analyze and forecast threats. Consequently, we presented the machine learning and regression approach for identifying the assault. It determines the type of cyberattack that will be launched and how to foresee it. According to our research, the probability of a cyberattack declines as potential victims' income and educational levels rise. We believe that our suggested model will be helpful to law enforcement cybercrime units in identifying and stopping cyberattacks, increasing their efficacy in thwarting these threats. In conclusion, this study emphasizes how important machine learning is to improving the ability to detect cyberattacks and identify their perpetrators. By employing data-driven insights and advanced analytics, companies can enhance their capacity to anticipate, identify, and mitigate cyber threats, thereby fortifying their defences in a constantly changing digital landscape. Going forward, sustaining a proactive posture against cyber adversaries and safeguarding the security of our global community that is interconnected depends critically on continuous research and machine learning developments methods using algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

SVM	SUPPORT VECTOR MACHINE ALGORITHM
RF	RANDOM FOREST ALGORITHM
CNN	CONVOLUTIONAL NEURAL NETWORKS
APT	ADVANCED PERSISTENT THREATS
CSV	COMMA SEPARATED VALUES
IDS	INTRUSION DETECTION SYSTEM
FBV	FUNCTION BASED VIEW
CBV	CLASS BASED VIEW

CHAPTER 1

1.1 INTRODUCTION

In the consistently developing scene of advanced innovation, the danger of digital assaults poses a potential threat, presenting huge difficulties to people, organizations, and states around the world. These assaults, going from information breaks to malware diseases, can bring about serious results, including monetary misfortune, reputational harm, and, surprisingly, public safety dangers. With digital hoodlums turning out to be progressively modern in their strategies, there is a squeezing need for cutting-edge devices and procedures to anticipate and forestall such assaults.

AI, a subset of man-made consciousness, has arisen as an amazing asset in the battle against digital wrongdoing. Overwhelmingly of information, AI calculations can recognize examples, oddities, and patterns that might demonstrate potential digital dangers. Researchers and practitioners have been focusing on using machine learning to predict cyberattacks and identify their perpetrators, allowing for proactive risk mitigation and cybersecurity enhancement. It, first and foremost, requires the assortment and preprocessing of different information sources, including network traffic logs, framework occasion logs, and authentic assault information. Highlights designing assumes a critical part in this stage as it describes various parts of digital assaults and their culprits.

Predictive models can be created using a variety of machine-learning algorithms once the data has been prepared. These algorithms range from more conventional approaches like logistic regression and decision trees to more sophisticated ones like ensemble methods and neural networks. In addition, the predictive models' accuracy, robustness, and generalizability can only be assessed through model evaluation and validation. Digital assaults are generally intriguing contrasted with genuine client conduct, making it hard for AI model to gain from imbalanced datasets. One more basic part of anticipating digital assaults is the attribution of assaults to their culprits. Identifying the individuals, groups, or organizations responsible for particular cyber incidents is a frequently difficult and complex process known as cyber attribution.

AI can support this interaction by dissecting different advanced scientific antiques, including IP addresses, malware marks, and standards of conduct, to induce the probable beginning and thought processes of the assailants. Additionally, in order to improve the efficiency of cyber-attack prediction and attribution efforts, cybersecurity professionals, judicial bodies, and privately owned companies must work together and disseminate insights. By pooling assets, ability, and information, partners can work on the exactness and practicality of prescient models, accordingly fortifying aggregate protections against digital dangers.

1.2 OBJECTIVE

To develop a robust machine learning model capable of accurately predicting cyber attack perpetrators based on diverse data sources and behavioral patterns. By analyzing historical attack data, network traffic, user behaviors, and other relevant factors, the model aims to identify potential threat actors before they execute malicious activities. The ultimate goal is to enhance cybersecurity defenses by preemptively identifying and neutralizing potential threats, thereby reducing the risk of successful cyber attacks and minimizing the impact on organizations and individuals.

CHAPTER 2
LITERATURE SURVEY

[1] Project Title: Cyber-attack method and perpetrator prediction using machine learning Algorithms.

Authors: Abdulkadir Bilen and Ahmet Bedri Özer.

Year: 2021

In response to the escalating threat of cyber-attacks and cyber-crime, this study presents two machine-learning models for predicting cyber-attack methods and identifying perpetrators. By analyzing real data including crime type, perpetrator gender, damage, and attack methods, the study evaluates eight machine-learning methods. Support Vector Machine Linear emerges as the most effective in predicting cyber-attack methods with 95.02% accuracy, while Logistic Regression leads in identifying attackers with 65.42% accuracy. Notably, the study finds a correlation between higher education and income levels of victims and reduced probability of cyber-attacks. The proposed models offer promising tools for cyber-crime units to enhance detection and combat cyber-attacks effectively.

[2] Project Title: Computational system to classify cyber-crime offenses using machine learning.

Authors: R, Gadekallu TR, Abidi MH, Al- Ahmari A.

Year: 2020

With the exponential growth of internet usage, cybercrime has surged, projected to cost nearly $6 trillion annually by 2021. Cybercriminals exploit vulnerabilities in networked devices for financial gain and other motives. Traditional manual methods and existing technical approaches have struggled to adequately address this rising threat. This study introduces a flexible computational tool employing machine learning techniques to analyze and classify cybercrime rates at a state level within a country. By integrating security analytics with data analytic approaches, the study effectively analyzes and classifies cybercrimes from both structured and unstructured data sources in India. The key strength lies in achieving a remarkable 99% accuracy in accurately classifying offenses, providing a robust framework for cybercrime analysis and mitigation efforts.

[3] Project Title: Detection of advanced persistent threat using machine-learning correlation analysis, Future Generation Computer Systems.
Authors: I. Ghair, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, F. J. Aparicio-Navarro.
Year: 2018

This study addresses the critical challenge of detecting and predicting Advanced Persistent Threats (APT), a serious form of cyber attack with global implications. Introducing a novel machine learning-based system called MLAPT, the research proposes a three-phase approach: Threat detection, Alert correlation, and Attack prediction. Eight detection methods are developed to identify various APT techniques, followed by a correlation framework to link related alerts. Finally, a machine learning prediction module predicts the progression of APT attacks based on correlated alerts. MLAPT achieves an impressive 84.8% accuracy in predicting APT attacks in their early stages. This systematic approach offers significant contributions to APT detection and prediction, providing valuable tools for network security teams to proactively combat these sophisticated threats.

[4] Project Title: A Machine Learning Framework for Investigating Data Breaches Based on Semantic Analysis of Adversary's Attack Patterns in Threat Intelligence Repositories.
Authors: Zahid Anwar a, b, Asad Waqar Malik a, Sharifullah Khan a, umara

Noor a c, Shahzad Saleema

Year: 2019

This paper addresses the inefficiencies of manual cyber-attack investigation processes and the limitations of signature-based deep search solutions. It introduces a novel machine learning framework aimed at automating threat analysis by mapping adversary Tactics, Techniques, and Procedures (TTPs) to attack goals and detection mechanisms. The framework creates a semantic network by semantically relating threats and TTPs extracted from established threat sources. Utilizing probabilistic relationships within this network, the framework accurately identifies cyber threats with 92% accuracy and minimal false positives, even when faced with lost or spurious TTPs. Evaluation using threat artifacts demonstrates the framework's effectiveness, achieving an average detection time of 0.15 seconds for data breach incidents. This approach offers a promising solution to streamline cyber threat identification and improve response times in combating sophisticated cyber threats.

[5] Project Title: Cyber Threat Analysis And Prediction Using Machine Learning

Authors: Kulvinder Singh, Sudhan Jha

Year:2021

With the increase in cyber data attacks, the manual method of investigating cyber-attacks is more prone to errors and is time consuming. With the increase in advanced cyber threat attacks with the same patterns, timely investigation is not possible. There are many systems proposed which analyse and predict threats using various machine learning methods. In this model we applied machine learning algorithms to analyse and predict cyber-attacks.

CHAPTER 3
SYSTEM ANALYSIS

3.1 Existing system

Existing systems in cyber attack prediction leverage machine learning algorithms to analyze vast amounts of data and identify patterns indicative of potential threats. These systems typically utilize various techniques such as anomaly detection, classification, and clustering to forecast cyber attacks. Anomaly detection models identify deviations from normal behavior within network traffic, system logs, or user activity, flagging activities that diverge significantly from established norms. Classification algorithms categorize incoming data into predefined attack types based on historical data or known attack signatures, enabling rapid response to familiar threats. Clustering techniques group similar instances together to identify emerging attack trends or to detect previously unseen attack patterns. Additionally, ensemble methods combine multiple machine learning models to improve prediction accuracy and robustness. These systems often incorporate real-time monitoring capabilities to continuously update their models and adapt to evolving threats. Moreover, some systems integrate threat intelligence feeds and historical attack data to enhance prediction accuracy and prioritize alerts based on their severity. Overall, the synergy of machine learning algorithms with advanced data analysis techniques empowers these systems to proactively identify and mitigate cyber threats, thereby bolstering the resilience of organizations against malicious activities in the digital landscape.

3.2 Proposed system

In the realm of cyber attack prediction using machine learning, several proposed systems aim to enhance the detection and mitigation of cyber threats. Here are some examples: Anomaly Detection Systems: These systems leverage machine learning algorithms to detect abnormal patterns or behaviors in network traffic, system logs, or user activities. By training on historical data, these systems can identify deviations from normal behavior that may indicate potential cyber attacks. Proposed systems often use techniques like unsupervised learning (e.g., clustering, autoencoders) to detect anomalies without the need for labeled training data. Predictive Modeling Frameworks: Predictive modeling frameworks use machine learning to forecast the likelihood of future cyber attacks based on historical attack data, network configurations, and threat intelligence feeds. These systems employ algorithms such as logistic regression, decision trees, or ensemble methods to predict the probability of different types of attacks occurring within a given time frame. By continuously updating models with new data, these frameworks can adapt to evolving threats and provide proactive defense strategies. Threat Intelligence Platforms: Threat intelligence platforms integrate machine learning with external threat feeds and internal security data to identify emerging threats and vulnerabilities. These platforms analyze large volumes of data from diverse sources, including open-source intelligence, dark web forums, and security incident reports, to generate actionable insights for cybersecurity analysts. Machine learning algorithms are used to classify, prioritize, and correlate threat indicators, enabling organizations to make informed decisions about threat mitigation strategies. Behavioral Analysis Systems: Behavioral analysis systems monitor user and entity behavior within a network to detect suspicious activities indicative of cyber attacks. These systems use machine learning to model normal behavior patterns for users, devices, and applications and flag deviations from these patterns as potential threats. By analyzing patterns of access, resource usage, and communication, these systems can detect insider threats, compromised accounts, or unauthorized activities in real-time. Deep Learning Approaches: Deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are increasingly being explored for cyber attack prediction tasks. These models can learn complex patterns and relationships in large-scale datasets, allowing them to detect subtle indicators of cyber attacks that may be missed by traditional machine learning algorithms. Deep learning approaches show promise in areas like malware detection, network intrusion detection, and phishing detection. Attribution Analysis: ML techniques can be used to analyze various attributes associated with cyber-attacks, such as IP addresses, malware signatures, and attack patterns, to identify potential perpetrators or threat actors. This involves building models that can correlate different pieces of evidence to attribute cyber-attacks to specific individuals, groups, or organizations. Adversarial Machine Learning: This involves training ML models to detect and respond to adversarial attacks aimed at manipulating or evading detection systems. Adversarial training techniques, such as generative adversarial networks (GANs), can be used to enhance the robustness of ML-based cyber-attack detection systems. These are just a few examples of how machine learning techniques can be applied to predict cyber-attacks and identify perpetrators. The effectiveness of these approaches often depends on the quality and quantity of data available for training the ML models, as well as the expertise of cybersecurity professionals in interpreting and acting upon the insights provided by these models.

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

- Coding Language: HTML, CSS, Bootstrap, Django, Python, Flask, Data Analytics using Python.
- Data Base: Excell

4.2 HARDWARE REQUIREMENTS

- Operating system: Window 10/11
- Disk Storage: 10GB Free space
- Ram: 4GB

4.3 Development Requirements:

- Microsoft Visual Studio Code
- Django
- Flask
- Jupiter Notebook
- CSV File

CHAPTER 5
SYSTEM DESIGN

5.1 UML DIAGRAMS

5.1.1 Introduction

Unified Modeling Language (UML) is a standardized general- purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development. There are given as below:

- Sequence diagram
- Use case diagram
- Activity diagram
- Collaboration diagram
- Dataflow diagram

Characteristics of UM

- It is a generalized modeling language.
- It is different from software programming languages such as Python, C, C++, etc.
- It is a pictorial language which can be used to generate powerfulmodeling elements.
- It is related to object-oriented designs and analysis.

Five Rules for Better UML Diagrams

- ➢ To avoid large diagrams with too many items.
- ➢ Avoid any two lines in your diagram crossing each other.
- ➢ Lines in a diagram should go only horizontal or vertical with onlyright angles.
- ➢ Parent elements are higher than the child elements in generalization or realization hierarchies.
- ➢ Diagrams should be nice and clean

5.2 Sequence Diagram:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

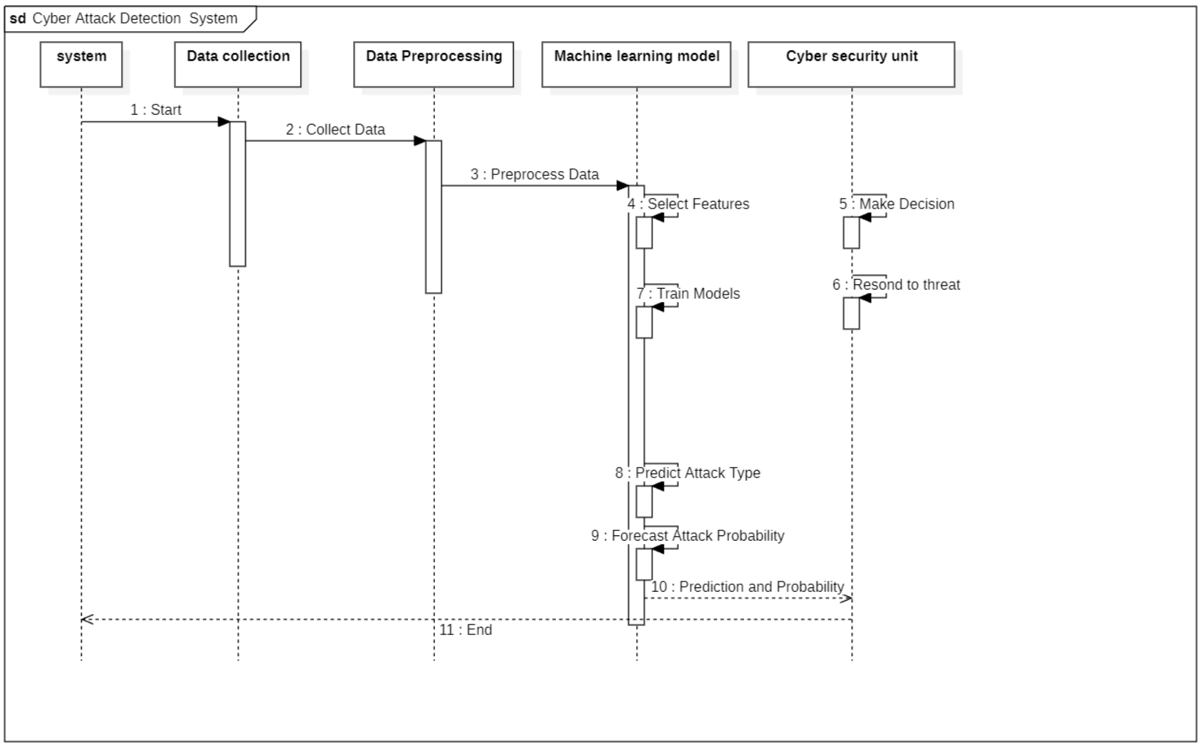


Fig 5.2 Sequence diagram

5.3 Use Case Diagram

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case

A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor

An actor is a person, organization or external system that plays a role in one or more interaction with the system.

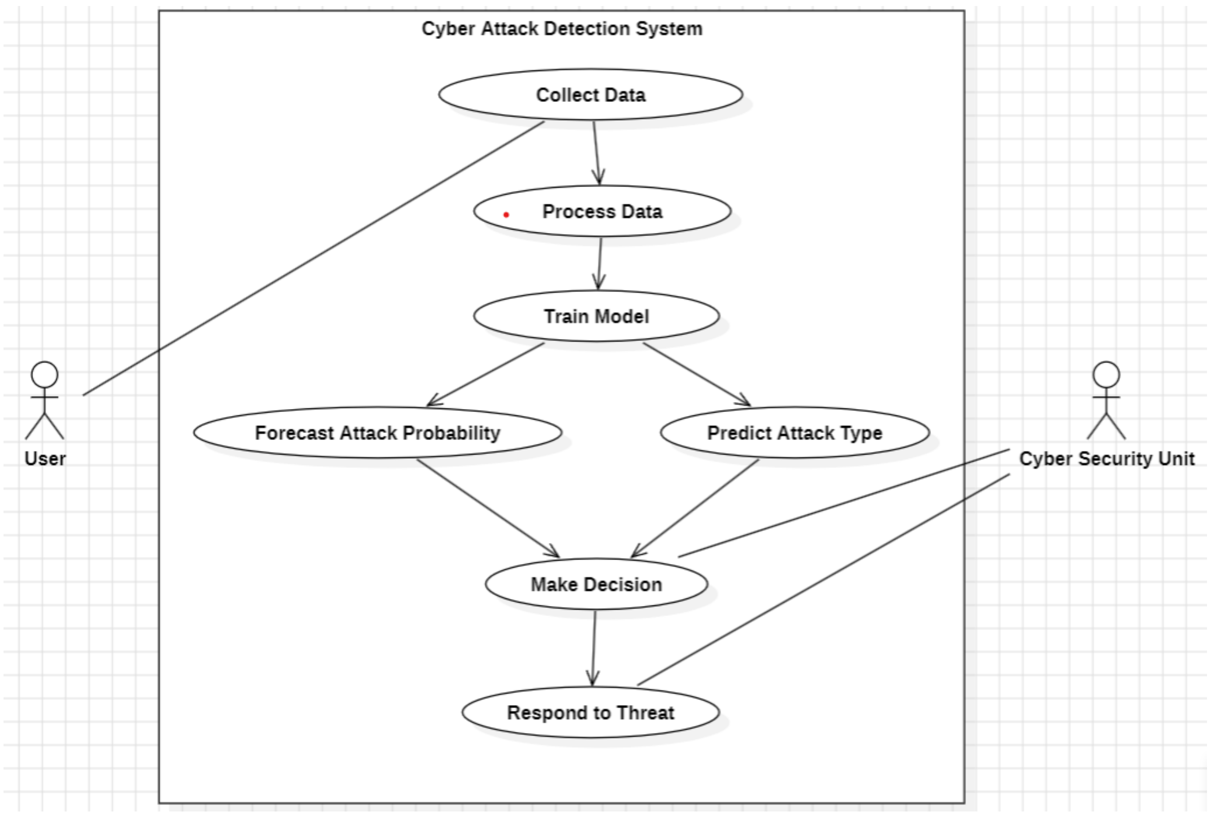


Fig. 5.3 Use case diagram

5.4 Activity Diagram

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.

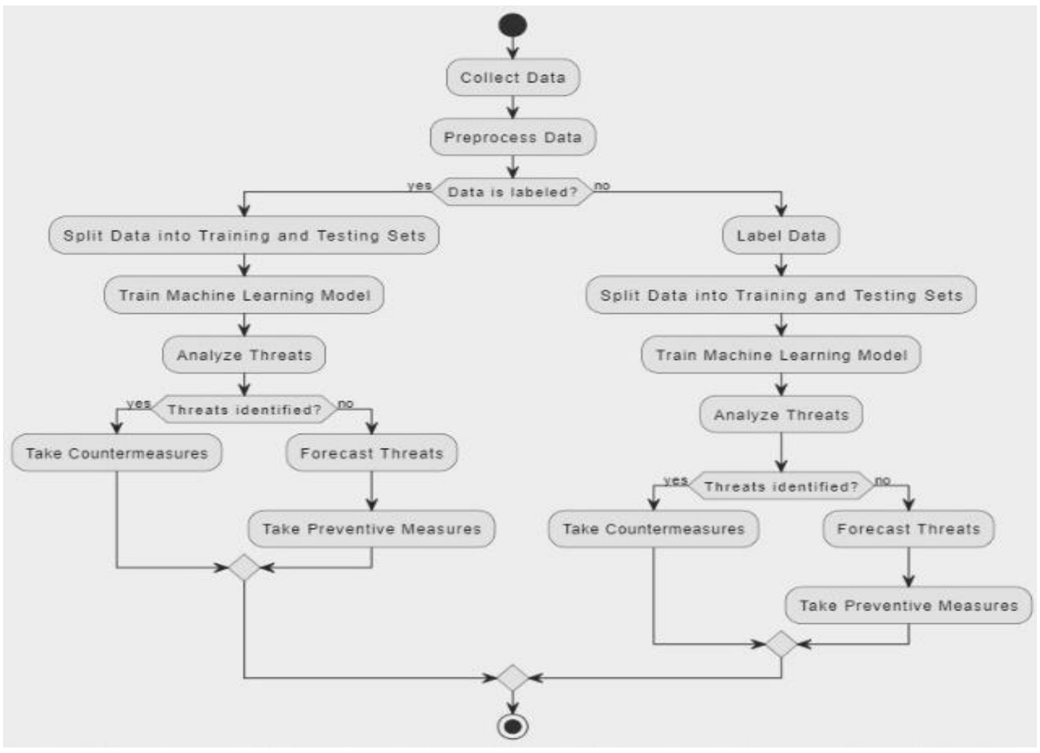


Fig 5.4. Activity diagram

5.5 Class Diagram

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

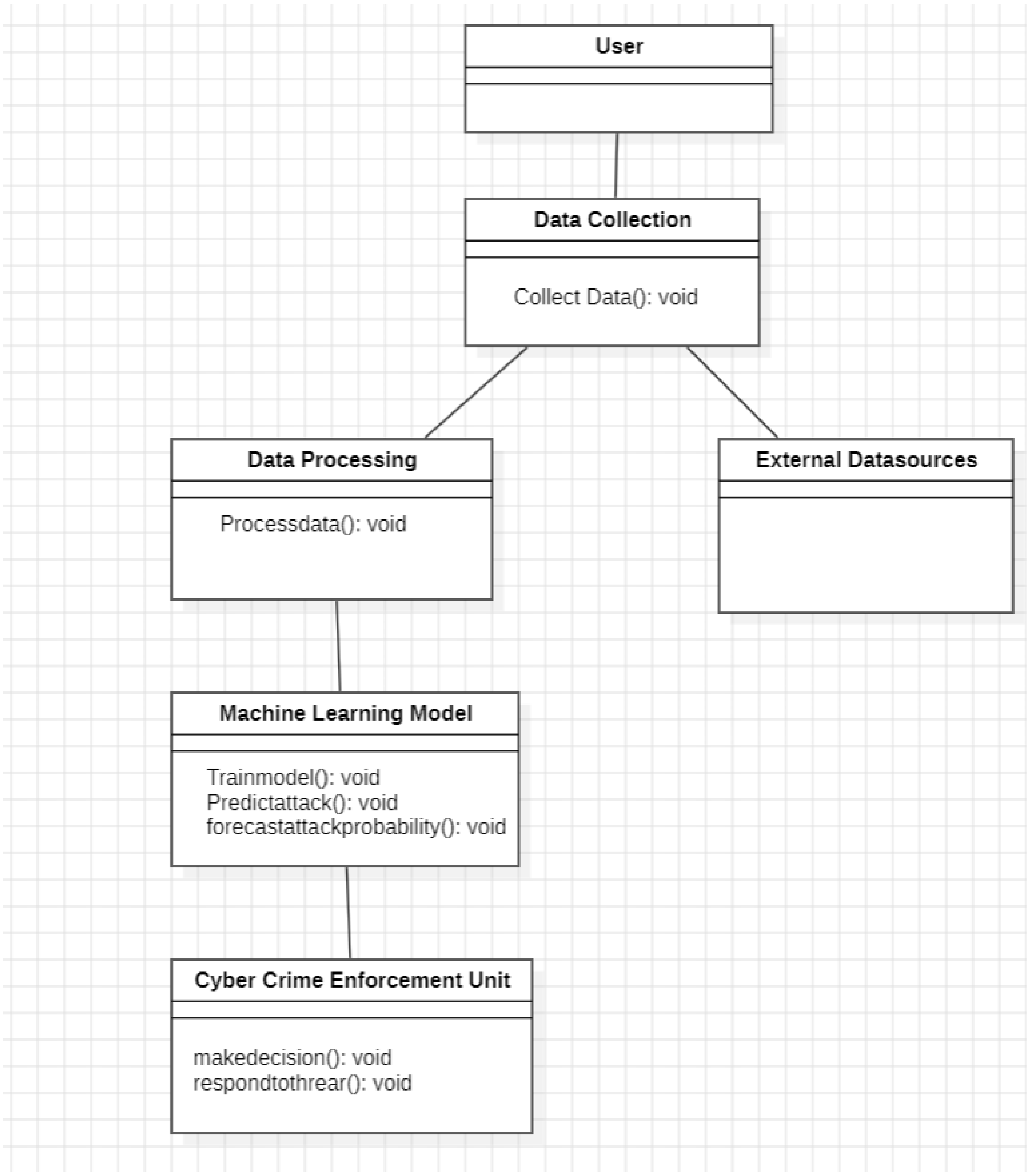


Fig 5.5 Class diagram

CHAPTER 6
SYSTEM IMPLEMENTATION

6.1 LIST OF MODULES

- User Management:
    - Signup
    - Login
    - Logout
- File Management:
    - Upload Files
    - Download Files
    - Delete Files
- Input Unit
- Output Unit

6.2 MODULE DESCRIPTION

6.2.1 User Management

Login

User should be able to login to their personal account.

Sign up

User should be able to create their personal account.

6.2.2 File Management

Upload files

Admin should be able to upload single or multiple files via selection or drag and drop in CSV File format.

Download files

Admin should be able to download any file of their personal storage.

Delete files

Admin should be able to delete any file stored in their personal storage space.

6.2.3 Input Unit

In input unit the required data would be entered for the prediction method.

6.2.4 Output Unit

In output the entered data would be analyzed and then display what kind of attack would happen and who is responsible that would be showed.

6.3 PROPOSED SYSTEM METHODOLOGY

6.3.1 ARCHITECTURE DIAGRAM

Diagram shows how the data get be collected and that collected data would be stored in database, that collected data would be preprocessed and then train the data and finally test the data it gives desired result of given input.

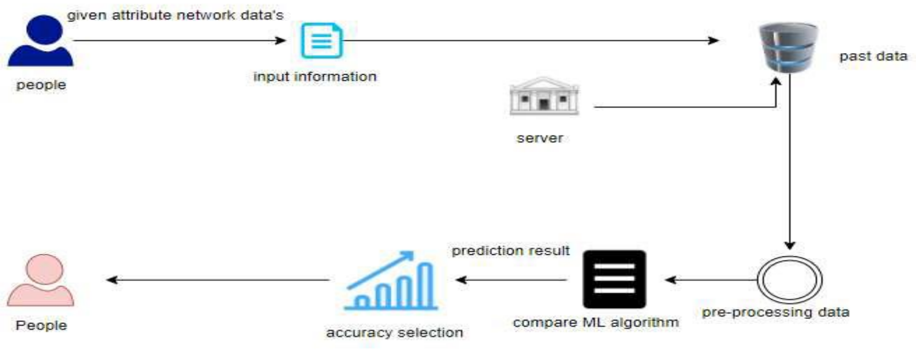


Fig.6.3.1 Architecture diagram

6.3.2 Flow Chart

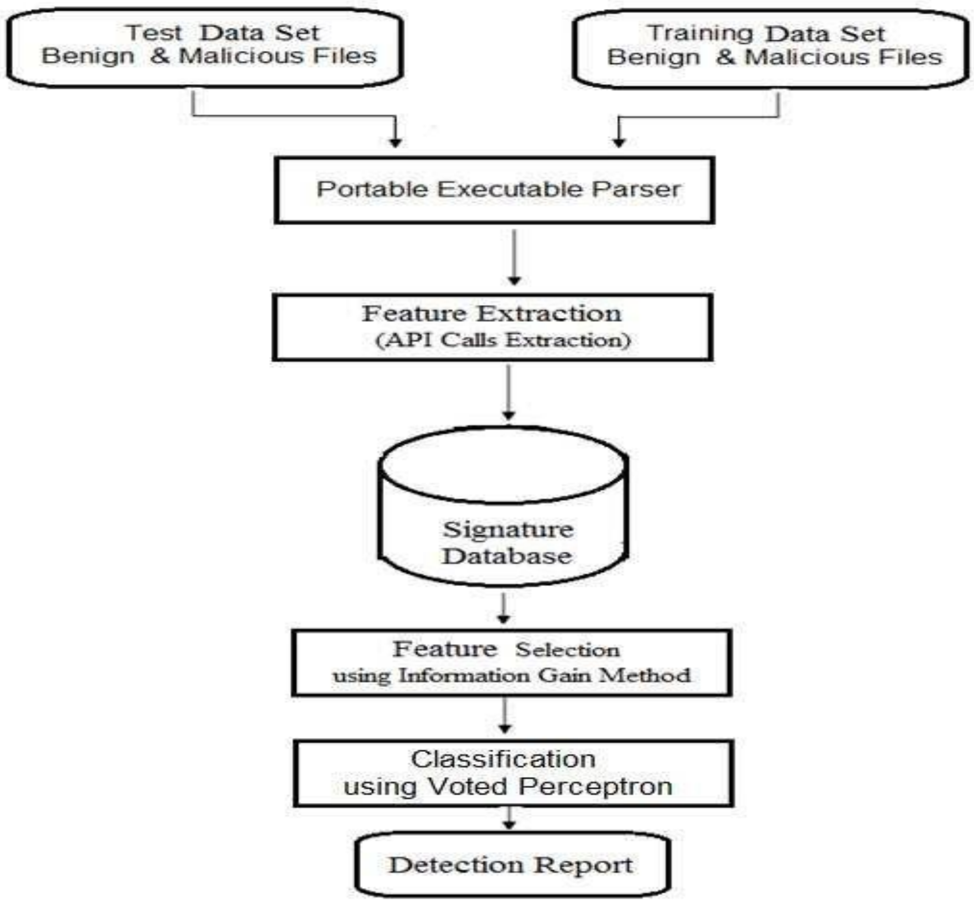


Fig.6.3.2 Flow Chart diagram

6.3.3 Machine Learning

Machine learning in cyber attack prediction involves using algorithms and statistical models to analyze large volumes of data and identify patterns or anomalies indicative of potential cyber threats. These algorithms learn from historical data to make predictions about future cyber attacks, helping cybersecurity analysts and systems detect and respond to threats more effectively.

One common approach in machine learning for cyber attack prediction is anomaly detection. Anomalies are deviations from normal behavior or patterns in data that may indicate malicious activity. Machine learning algorithms, such as clustering, classification, and regression, can be trained on historical data to recognize patterns of normal behavior and flag deviations as potential threats. For example, unsupervised learning techniques like clustering can identify groups of data points that are unusual or outliers, while supervised learning algorithms like classification can label instances as benign or malicious based on training data.

Another approach is predictive modeling, where machine learning algorithms use historical data to forecast future cyber attacks. This involves training algorithms on features extracted from past attack data, such as attack vectors, vulnerabilities exploited, and attack signatures. These models can then predict the likelihood of future attacks based on current network activity and threat intelligence feeds.

Machine learning techniques can also enhance traditional cybersecurity measures such as intrusion detection systems (IDS) and antivirus software. By analyzing network traffic and system logs in real-time, machine learning algorithms can identify suspicious behavior and generate alerts for further investigation by cybersecurity analysts.

However, machine learning in cyber attack prediction also faces challenges such as data imbalance, where benign instances outnumber malicious ones, and adversarial attacks, where attackers deliberately manipulate data to evade detection. Overcoming these challenges requires robust data preprocessing, feature engineering, and continuous model monitoring and updating.

Overall, machine learning plays a crucial role in cyber attack prediction by enabling proactive threat detection and response, helping organizations stay ahead of evolving cyber threats in an increasingly complex digital landscape.

There are many use cases for machine learning, including:

1. Predictive Maintenance: Machine learning models can analyze sensor data from machinery to predict when equipment is likely to fail, enabling proactive maintenance and reducing downtime.

2. Fraud Detection: Machine learning algorithms can detect fraudulent activities in financial transactions by identifying patterns and anomalies in transaction data, helping financial institutions prevent fraud.

3. Healthcare Diagnosis: Machine learning can analyze medical data such as patient records, lab results, and imaging scans to assist in diagnosing diseases, predicting patient outcomes, and recommending treatment plans.

4. Recommendation Systems: Machine learning powers recommendation engines used by platforms like Amazon, Netflix, and Spotify to personalize content recommendations based on user preferences and behavior.

5. Natural Language Processing (NLP): NLP techniques enable machines to understand and generate human language, powering applications such as sentiment analysis, chatbots, language translation, and text summarization.

6. Image Recognition: Machine learning algorithms can classify and recognize objects, faces, and patterns in images, leading to applications like facial recognition, autonomous vehicles, and medical imaging analysis.

7. Supply Chain Optimization: Machine learning can optimize supply chain operations by analyzing demand forecasts, inventory levels, and transportation routes to minimize costs, reduce delays, and improve efficiency.

8. Customer Churn Prediction: Machine learning models can predict customer churn by analyzing customer behavior, usage patterns, and demographics, enabling businesses to take proactive measures to retain customers.

9. Sentiment Analysis: Machine learning can analyze text data from social media, customer reviews, and surveys to determine sentiment and opinion trends, helping businesses understand customer feedback and market trends.

10. Energy Forecasting: Machine learning algorithms can forecast energy consumption and generation based on historical data and external factors like weather forecasts, aiding in energy management and resource planning.

These are just a few examples of how machine learning is being applied across various domains to automate tasks, gain insights from data, and improve decision-making processes.

6.3.4 Data Collection

Data collection in cyber attack prediction using machine learning involves gathering diverse datasets from various sources to train predictive models capable of identifying and preempting cyber threats. These datasets typically include historical attack data, network traffic logs, system event logs, threat intelligence feeds, and contextual information about the organization's.

The first step in data collection is identifying relevant sources of data and aggregating them into a centralized repository. This may involve integrating with security information and event management (SIEM) systems, network intrusion detection systems (NIDS), endpoint detection and response (EDR) solutions, and external threat intelligence platforms.

Next, the collected data undergoes preprocessing, which includes cleaning, normalization, and feature engineering to prepare it for analysis. Feature engineering involves selecting and transforming raw data into meaningful features that capture important aspects of cyber threats, such as network traffic patterns, system activity, and known indicators of compromise (IOCs).

Once the data is prepared, machine learning algorithms are trained on labeled datasets to learn patterns and relationships between features and cyber attacks. Supervised learning algorithms, such as decision trees, random forests, support vector machines (SVM), and neural networks, are commonly used to classify instances as benign or malicious based on historical data.

In addition to supervised learning, unsupervised learning techniques like clustering and anomaly detection can uncover unknown patterns and anomalies indicative of potential threats in unlabeled data. Reinforcement learning may also be employed to adaptively respond to evolving threats by optimizing decision-making processes in real-time.

Overall, data collection in cyber attack prediction using machine learning is a critical step in building effective cybersecurity systems that can proactively detect and mitigate cyber threats before they cause harm to organizations.

6.3.5 Data Preprocessing

Data preprocessing in cyber attack prediction using machine learning involves preparing and cleaning the raw data to make it suitable for analysis and model training. This process is essential for improving the quality of input data and enhancing the performance of machine learning algorithms in detecting and predicting cyber threats.

Firstly, data preprocessing involves data cleaning, which includes handling missing values, removing duplicates, and correcting inconsistencies in the data. Missing values can be imputed using techniques like mean, median, or mode substitution, while duplicates and inconsistencies are resolved to ensure data integrity.

Next, data preprocessing involves feature selection and extraction, where relevant features are identified and extracted from the raw data. This step aims to reduce dimensionality and focus on the most informative features for predicting cyber attacks. Feature engineering techniques may include transforming categorical variables into numerical representations, scaling numerical features, and creating new features through mathematical transformations or domain knowledge.

After feature selection, data preprocessing involves data normalization or standardization to ensure that all features have similar scales and distributions. This step helps prevent certain features from dominating others during model training and improves the convergence of machine learning algorithms.

Furthermore, data preprocessing may involve data encoding to convert categorical variables into a numerical format suitable for machine learning algorithms. Techniques like one-hot encoding or label encoding are commonly used for this purpose.

Additionally, data preprocessing may include data splitting, where the dataset is divided into training, validation, and test sets to evaluate model performance accurately. This ensures that the model is trained on a subset of the data, validated on another subset, and tested on a separate unseen subset to assess its generalization ability.

Overall, data preprocessing is a critical step in cyber attack prediction using machine learning, as it enhances the quality of input data and improves the effectiveness of predictive models in identifying and mitigating cyber threats.

6.3.6 Training and Testing

Amazon In cyber attack prediction using machine learning, the process typically involves training and testing machine learning models on historical data to develop predictive capabilities. Here's how training and testing work in this context:

1. Training: During the training phase, the machine learning model learns patterns and relationships in the historical data to understand normal behavior and identify indicators of cyber attacks. This involves feeding the model with labeled data, where each instance is associated with a target variable indicating whether it represents a benign or malicious activity. The model then uses various algorithms and techniques to learn from these labeled examples and adjust its internal parameters to minimize prediction errors.

2. Testing: Once the model is trained, it needs to be evaluated on unseen data to assess its performance and generalization ability. This is done during the testing phase, where the model is presented with a separate dataset that it hasn't seen before. The testing dataset contains instances similar to those encountered during training but with unknown labels. The model predicts the labels for these instances, and the predictions are compared with the ground truth labels to measure the model's accuracy, precision, recall, and other performance metrics.

The goal of testing is to ensure that the model can effectively generalize to new, unseen data and make accurate predictions in real-world scenarios. It helps identify potential issues such as overfitting (where the model memorizes the training data but fails to generalize) or underfitting (where the model fails to capture the underlying patterns in the data). By iteratively refining the model based on testing results, cybersecurity analysts can improve its performance and reliability for predicting cyber attacks.

6.3.7 ALGORITHM EXPLANATION

6.3.7.1 SUPPORT VECTOR MACHINE

In Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

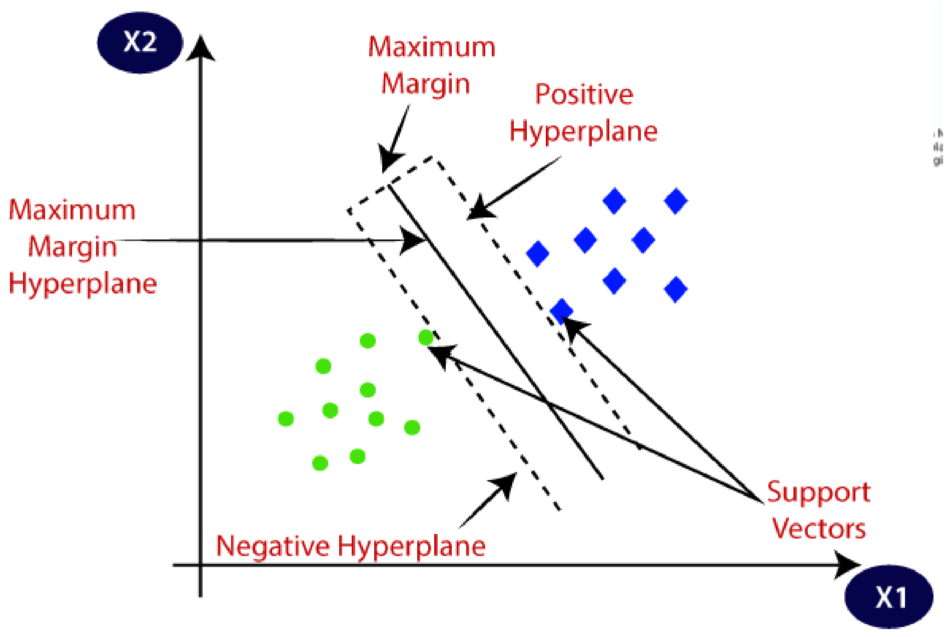


Fig: 6.3.7.1. Support Vector Machine Hyperplane

6.3.7.2 RANDOM FOREST ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

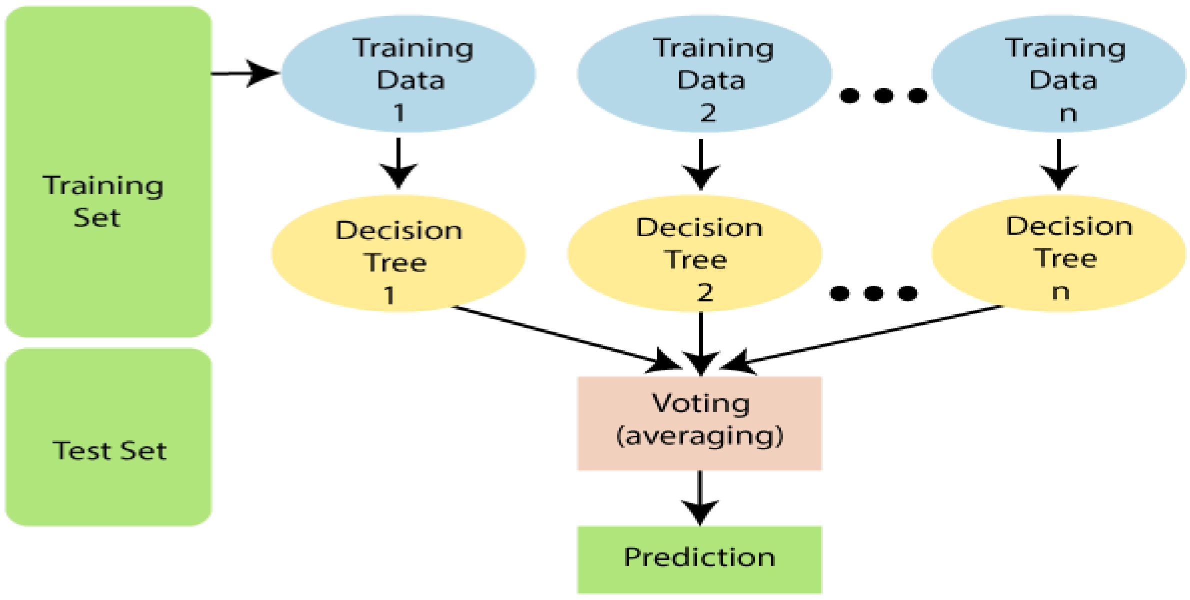


Fig: 6.3.7.2. Random Forest Algorithm

6.3.7.3 GRADIENT BOOST ALGORITHM

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

In contrast to AdaBoost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of the predecessor as labels. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees). The below diagram explains how gradient-boosted trees are trained for regression problems.

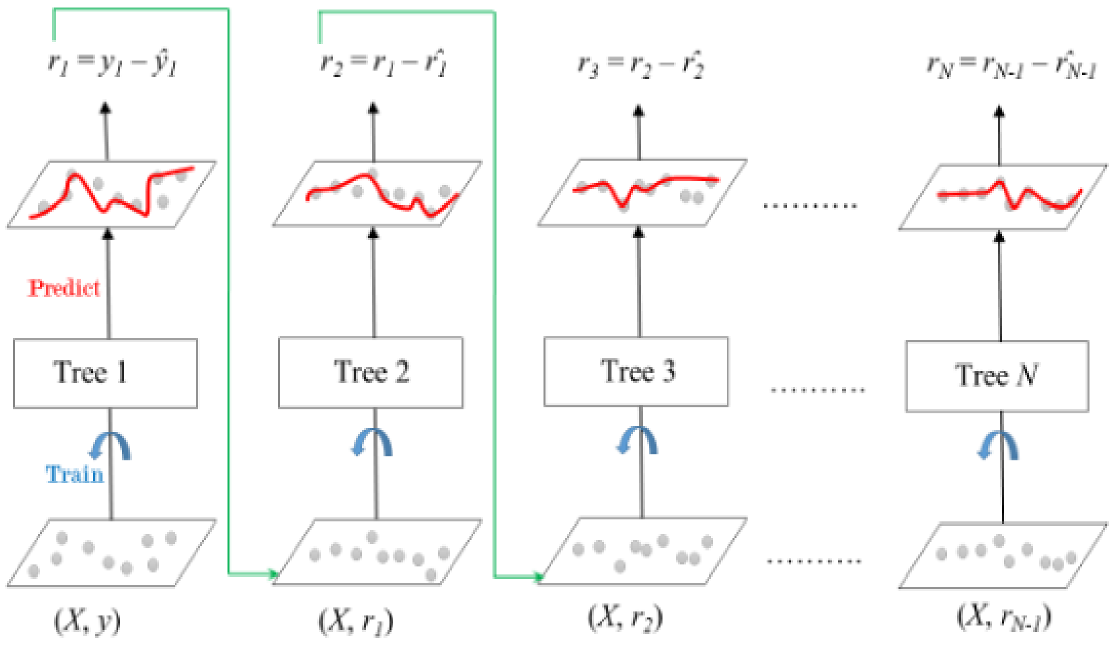


Fig. 6.3.7.3 Gradient Boost Algorithm

The ensemble consists of M trees. Tree1 is trained using the feature matrix X and the labels y. The predictions labeled y1(hat) are used to determine the training set residual errors r1. Tree2 is then trained using the feature matrix X and the residual errors r1 of Tree1 as labels. The predicted results r1(hat) are then used to determine the residual r2. The process is repeated until all the M trees forming the ensemble are trained. There is an important parameter used in this technique known as Shrinkage. Shrinkage refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate (eta) which ranges between 0 to 1. Since all trees are trained now, predictions can be made. Each tree predicts a label and the final prediction is given by the formula,

y(pred) = y1 + (eta * r1) + (eta * r2) + ... + (eta * rN)

CHAPTER 7

CODING AND TESTING

7.1 CODING

Once the design aspect of the system is finalized the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in each programming language. Therefore, a good coding style must be taken whenever changes are required it easily screwed into the system.

7.2 CODING STANDARDS

Coding standards are guidelines to programming that focus on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system implements.

The blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows

- ➢ Program should be simple, clear and easy to understand.
- ➢ Naming conventions
- ➢ Value conventions
- ➢ Script and comment procedure
- ➢ Message box formal
- ➢ Exception and error handling

7.2.1 NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy** understanding of the intended message by the user. So, it is customary to follow the conventions.These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capitalletter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with eachsubsequent letter of the new words in uppercase and the rest of letters in lowercase.

7.2.2 VALUE CONVENTIONS

Value conventions ensure values for variables at any point of time. This involves the following:

7.2.2.1 Proper default values for the variables.

7.2.2.2 Proper validation of values in the field.

7.2.2.3 Proper documentation of flag values.

7.2.3 SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditionaland looping statements are to be properly aligned to facilitate easy understanding.

Comments are included to minimize the number of surprises that could occurwhen going through the code.

7.2.4 MESSAGE BOX FORMAT

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted indisplaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

CHAPTER 8
SOFTWARE DESCRIPTION

8.1 DJANGO

8.1.1 Introduction to Django

Python-based web framework Django allows you to create efficient web applications quickly. It is also called batteries included web framework Django because It provides built-in features for everything including Django Admin Interface, default database – SQLlite3, etc.

When you’re building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django Python gives you ready-made components to use for rapid development. There any many more benefits of using the Django framework. Let’s look at some other reasons why you should learn Python Frameworks in Django.

Django Views

In Django, views are the backbone of handling user requests and rendering responses. There are two primary paradigms for implementing views: Function Based Views (FBVs) and Class Based Views (CBVs). Function Based Views offer simplicity and directness, allowing developers to define views as Python functions. Within this paradigm, common functionalities like creating, listing, displaying details, updating, and deleting objects are implemented as separate functions.

While both paradigms have their merits, the choice between FBVs and CBVs ultimately depends on factors such as project requirements, development preferences, and scalability concerns.

Django URLs

In Django, URL patterns serve as a crucial mechanism for directing incoming requests to the appropriate views within your web application. With the flexibility of regular expressions, Django’s URL dispatcher allows you to define patterns that match specific URL patterns and route them to corresponding views. When dealing with GET parameters passed through URLs in Django, accessing these parameters within views is straightforward, enabling efficient handling of user inputs and customization of responses.

8.2 Flask

Flask is a lightweight and flexible web framework for Python. It's designed to make building web applications quick and easy, with a focus on simplicity and minimalism. Flask is known for its simplicity, which means it doesn't come with built-in features like database abstraction layers, form validation, or authentication. Instead, Flask allows developers to choose the tools and libraries that best suit their needs, making it highly customizable.

At its core, Flask provides tools for routing HTTP requests to Python functions, rendering templates to generate HTML pages, and accessing request data such as form inputs or query parameters. This makes it easy to create dynamic web pages and handle user interactions. Flask also supports extensions, which are third-party libraries that add additional functionality to the framework. These extensions cover a wide range of features, including database integration, authentication, and API development.

One of the key concepts in Flask is the idea of "views," which are Python functions that handle HTTP requests and return responses. Views are mapped to specific URLs using a decorator syntax, making it easy to define the structure of your web application. Flask also includes a built-in development server, which makes it easy to test your application locally before deploying it to a production server.

Flask is often compared to other web frameworks like Django, which is more opinionated and comes with more built-in features. While Django is great for building complex, feature-rich applications, Flask is better suited for smaller projects or applications where flexibility and simplicity are more important. Flask's lightweight nature also makes it a popular choice for building RESTful APIs, where performance and scalability are key concerns.

Overall, Flask is a powerful tool for building web applications in Python, offering simplicity, flexibility, and extensibility for developers of all skill levels. Whether you're building a simple website, a RESTful API, or a complex web application, Flask provides the tools you need to get the job done efficiently.

Some of the advantages of using Flask for building applications include:

1. Simplicity: Flask follows a minimalist design philosophy, making it simple and easy to learn. Its small core and intuitive API allow developers to get started quickly without being overwhelmed by unnecessary complexity.

2. Flexibility: Flask gives developers the freedom to choose the tools and libraries they prefer for various tasks such as database integration, authentication, and form validation.
3. Lightweight: Flask has a small codebase and minimal dependencies, resulting in lightweight applications with lower resource usage. This makes Flask suitable for projects where efficiency and performance are crucial, especially in scenarios with limited hardware resources or high traffic loads.
4. Modularity: Flask is highly modular, allowing developers to organize their codebase into reusable components or extensions. This promotes good coding practices such as separation of concerns and modularity, making the codebase easier to maintain and extend over time
5. Extensibility: Flask's architecture is designed to be extensible, with a vibrant ecosystem of third-party extensions available for various functionalities. These extensions cover a wide range of features, including database integration, authentication, caching, and more, allowing developers to easily add new features to their applications without reinventing the wheel.
6. Community and Documentation: Flask has a large and active community of developers who contribute to its ecosystem by creating tutorials, plugins, and answering questions on forums like Stack Overflow. Additionally, Flask has comprehensive documentation that covers all aspects of the framework, making it easy for developers to find answers to their questions and troubleshoot issues.

Overall, Flask offers a balanced combination of simplicity, flexibility, and extensibility, making it a popular choice for building web applications of varying sizes and complexities.

8.3 Data Analytics Using Python

Data analytics Data analytics in Python involves the use of Python programming language and its associated libraries to analyze and interpret data. Python is a popular choice for data analysis due to its simplicity, versatility, and extensive ecosystem of libraries specifically designed for handling data.

The foundational library for data analytics in Python is Pandas. Pandas provides data structures such as DataFrame and Series, which allow for easy manipulation and analysis of structured data. With Pandas, you can load data from various sources, clean and preprocess it, perform exploratory data analysis (EDA), and generate insights through statistical analysis.

In addition to Pandas, Python offers libraries like NumPy for numerical computing and mathematical operations. NumPy provides support for multi-dimensional arrays and matrices, which are essential for many data analysis tasks such as linear algebra operations and numerical computations.

For visualizing data and gaining insights through graphical representations, Matplotlib and Seaborn are widely used plotting libraries in Python. These libraries enable the creation of various types of plots, including scatter plots, histograms, bar charts, and heatmaps, allowing analysts to explore data visually and communicate findings effectively.

Furthermore, Python provides tools for advanced analytics and machine learning through libraries such as Scikit-learn, TensorFlow, and PyTorch. Scikit-learn offers a wide range of machine learning algorithms for tasks such as classification, regression, clustering, and dimensionality reduction.

CHAPTER 9

CODING

9.1 CODE

App.py

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import joblib

app = Flask(__name__)
model1 = joblib.load('Attack.pkl')
model2 = joblib.load('Perpetrator.pkl')

# @app.route('/')
# def login():
#     return render_template('main.html')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    int_features = [(x) for x in request.form.values()]
    final_features = int_features[0:9]
    final_features = [np.array(final_features)]
    predictionC = model1.predict(final_features)
    output = str(predictionC[0])

    if output == '0':
        a = 'Card Copying / Generating Devices'
    elif output == '1':
        a = 'Creating a Fake Shopping Site'
    elif output == '2':
        a = 'Hacking Tools or Malware'
    elif output == '3':
        a = 'Phishing Attack'
    elif output == '4':
        a = 'Receiving Public Data on Social Media'
    elif output == '5':
        a = 'Social Engineering'

    int_features.append(output)
    final_featuresY = [np.array(int_features)]
    predictionY = model2.predict(final_featuresY)
    outputY = predictionY[0]
    if outputY==0:
        outputY='Known'
    else:
        outputY="UnKnown"

    return render_template('index.html', prediction_text1=f"Attack Method is
{a}", prediction_text2=f"Perpetrator is {outputY}")


if __name__ == "__main__":
    app.run(host="localhost", port=5000)
```

index.html

```
    <!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>CYBER ATTACK AND PERPETRATOR PREDICTION</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/bootstrap.min.css') }}">
    <style>
        .back {
            background-image: url("{{ url_for('static', filename='image/ca1.jpg')
}}");
            background-repeat:no-repeat;
            background-attachment:fixed;
            background-position:center;
            background-size:cover;
        }
        .white {
            color:white;
        }
        .nspace {
            margin:15px 15px 30px 30px;
            background: white;
            padding:7px 10px;
            width:500px
        }
        .space {
            margin:10px 30px;
            padding:10px 10px;
            background: white;
            width:500px
        }
        .gap {
            padding:10px 20px;
        }
        /* Added styles for horizontal display */
        .image-container {
            display: flex;
            justify-content: space-around; /* Adjust as needed */
            align-items: center;
            flex-wrap: wrap;
            margin-top: 20px; /* Adjust as needed */
        }
        .image-container img {
            width: 300px;
            height: 300px;
            margin-right: 10px; /* Adjust as needed */
        }
    </style>
</head>
<body>
    <div>
        <div class="jumbotron">
            <h1 style="text-align:center">CYBER ATTACK AND
PERPETRATOR PREDICTION</h1>
        </div>
        <div class="back">
            <form class="form-group" action="{{ url_for('predict')}}"
method="post">
                        <div class="row">
                    <div class="gap col-md-6 ">
                      <label class="white"  for="">CRIME</label>
<select class="nspace form-control" name="CRIME" id="CRIME">
  <option value=0>Hacking into the Information System and Capturing
Data</option>
  <option value=1>Misuse of Debit Cards or Credit Cards</option>
  <option value=2>Through Informatics Theft</option>
</select>


<label class="white"  for="">GENDER</label>
<select class="nspace form-control" name="GENDER" id="GENDER">
  <option value=0>Female</option>
  <option value=1>Male</option>
</select>


<label class="white"  for="">AGE</label>
<select class="nspace form-control" name="AGE" id="AGE">
  <option value=0>27 and Under</option>
  <option value=1>51 and Above</option>
  <option value=2>Between 28 and 37</option>
  <option value=3>Between 38 and 50</option>
</select>


<label class="white"  for="">INCOME</label>
<select class="nspace form-control" name="INCOME" id="INCOME">
  <option value=0>High</option>
  <option value=1>Low</option>
  <option value=2>Middle</option>
</select>


<label class="white"  for="">JOB</label>
<select class="nspace form-control" name="JOB" id="JOB">
  <option value=0>Education</option>
  <option value=1>Finance Sector</option>
  <option value=2>Health Sector Manager</option>
  <option value=3>Housewife</option>
  <option value=4>IT Sector</option>
  <option value=5>Justice And Security</option>
  <option value=3>Others</option>
  <option value=4>Retired</option>
  <option value=5>Student</option>
  <option value=5>Technical</option>
</select>
                    </div>
                    <div class="gap col-md-6">
                    <label class="white"  for="">MARITAL STATUS</label>
<select class="nspace form-control" name="MARITAL STATUS"
id="MARITAL STATUS">
  <option value=0>Married</option>
  <option value=1>Single</option>
</select>


<label class="white"  for="">EDUCATION</label>
<select class="nspace form-control" name="EDUCATION"
id="EDUCATION">
  <option value=0>Graduate</option>
  <option value=1>High School</option>
  <option value=2>Primary Education</option>
  <option value=3>Under Graduate</option>
</select>


<label class="white"  for="">HARM</label>
<select class="nspace form-control" name="HARM" id="HARM">
  <option value=0>Do Moral Harm</option>
  <option value=1>Fraud</option>
  <option value=2>Internet Shopping Out of Knowledge</option>
  <option value=3>Internet Shopping by Intoducing Himself as a Bank
Clerk</option>
  <option value=4>Threating / Blackmailing</option>
  <option value=5>Withdrawals Without Knowledge</option>
  <option value=6>Withdrawing Money by Introducing Himself as a Bank
Clerk</option>
</select>


<label class="white"  for="">ATTACK</label>
<select class="nspace form-control" name="ATTACK" id="ATTACK">
  <option value=0>Copying Bank / ATM Cards</option>
  <option value=1>Hacking Social Media Accounts</option>
  <option value=2>Obtaining Electronic Bank Accounts</option>
  <option value=3>Obtaining and Using Data in Digital Environment</option>
  <option value=4>Selling Counterfeit Products</option>
</select>
                </div>
            </div>
            <div style="padding:2% 35%">
```

```
                <button type="submit" class="btn btn-success btn-block"
style="width:350px;padding:20px">Predict</button>
            </div>
        </form>
    </div>
    <br>
    <br>
    <div style="background: skyblue; padding: 2% 10%">
        <h6><span>{{ prediction_text1 }}</span><span style="float:right">{{
prediction_text2 }}</span></h6>
        <div class="image-container">
            {% if prediction_text1 == 'Attack Method is Card Copying /
Generating Devices' %}
                <img src="{{ url_for('static', filename='image/cc.jpg') }}"
alt="Card Copying Image">
            {% elif prediction_text1 == 'Attack Method is Creating a Fake
Shopping Site' %}
                <img src="{{ url_for('static', filename='image/fss.jpg') }}"
alt="Fake Shopping Site Image">
            {% elif prediction_text1 == 'Attack Method is Hacking Tools or
Malware' %}
                <img src="{{ url_for('static', filename='image/mal.jpg') }}"
alt="Hacking Tools Image">
            {% elif prediction_text1 == 'Attack Method is Phishing Attack' %}
                <img src="{{ url_for('static', filename='image/p.jpeg') }}"
alt="Phishing Attack Image">
            {% elif prediction_text1 == 'Attack Method is Receiving Public Data
on Social Media' %}
                <img src="{{ url_for('static', filename='image/sm.jpg') }}"
alt="Social Media Data Image">
            {% elif prediction_text1 == 'Attack Method is Social Engineering' %}
                <img src="{{ url_for('static', filename='image/se.jpg') }}"
alt="Social Engineering Image">
            {% endif %}
            {% if prediction_text2 == 'Perpetrator is Known' %}
                <img src="{{ url_for('static', filename='image/perpetrator.jpg') }}"
alt="Known Perpetrator Image">
            {% elif prediction_text2 == 'Perpetrator is UnKnown' %}
                <img src="{{ url_for('static',
filename='image/unknown_perpetrator.jpg') }}" alt="Unknown Perpetrator
Image">
            {% endif %}
        </div>
        </div>
    </div>
</body>
</html>
```

Login.css

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: "Poppins", sans-serif;
}

body{
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background: url('https://i.postimg.cc/fbxnjjsQ/ss.jpg') no-repeat;
    background-size: cover;
    background-position: center;
}

.wrapper{
    width: 420px;
    background: transparent;
    border: 2px solid rgb(255, 255, 255, .2);
    backdrop-filter: blur(20px);
    box-shadow: 0 0 10px rgba(0, 0, 0, .2);
    color: #faf2f2;
    border-radius: 10px;
    padding: 30px 40px;
}

.wrapper h1{
    font-size: 36px;
    text-align: center;
}

.wrapper .input-box{
    position: relative;
    width: 100%;
    height: 50px;
    margin: 30px 0px;
}

.input-box input{
    width: 100%;
    height: 100%;
    background: transparent;
    border: none;
    outline: none;
    border: 2px solid rgba(255, 255, 255, .2);
    border-radius: 40px;
    font-size: 16px;
    color: #faf2f2;
    padding: 20px 45px 20px 20px;
}

.input-box input::placeholder{
    color: #faf2f2;
}

.input-box i{
    position: absolute;
    right: 20px;
    top: 50%;
    transform: translateY(-50%);
    font-size: 20p;
}

.wrapper .remember-forgot{
    display: flex;
    justify-content: space-between;
    font-size: 14.5px;
    margin: -15px 0 15px;
}

.remember-forgot label input{
    accent-color: #fff;
    margin-right: 3px;
}

.remember-forgot a{
    color: #fff;
    text-decoration: none;
}

.remember-forgot a:hover{
    text-decoration: underline;
}

.wrapper .btn{
    width: 100%;
    height: 45px;
    background: #fff;
    border: none;
    outline: none;
    border-radius: 40px;
    box-shadow: 0 0 10px rgba(0, 0, 0, -1);
    cursor: pointer;
    font-size: 16px;
    color: #333;
    font-weight: 600;
}

.wrapper .register-link{
    font-size: 14.5px;
    text-align: center;
    margin: 20px 0 15px;
}

.register-link p a{
    color: #fff;
    text-decoration: none;
    font-weight: 600;
}

.register-link p a:hover{
    text-decoration: underline;
}
```

m1.ipynb

```
import pandas as p
import numpy as n
import matplotlib.pyplot as plt
import seaborn as s
import warnings
warnings.filterwarnings('ignore')
data=p.read_csv('data.csv')
data.columns
df=data.dropna()
from sklearn.preprocessing import LabelEncoder
l = LabelEncoder()
columns = ['S.No','Crime', 'Gender', 'Age', 'Income', 'Job', 'Maritalstatus',
      'Education', 'Harm', 'Attack', 'AttackMethod', 'Perpetrator']
for column in columns:
    df[column] = l.fit_transform(df[column]).astype(int)

#preprocessing, split test and dataset, split response variable
X = df[['Crime', 'Gender', 'Age', 'Income', 'Job', 'Maritalstatus', 'Education',
'Harm', 'Attack']]

#Response variable
y = df['AttackMethod']
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT         : ", Counter(y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.30,
random_state=1, stratify=y_ros)
print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))

from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

lr = LogisticRegression()

lr.fit(X_train,y_train)
```

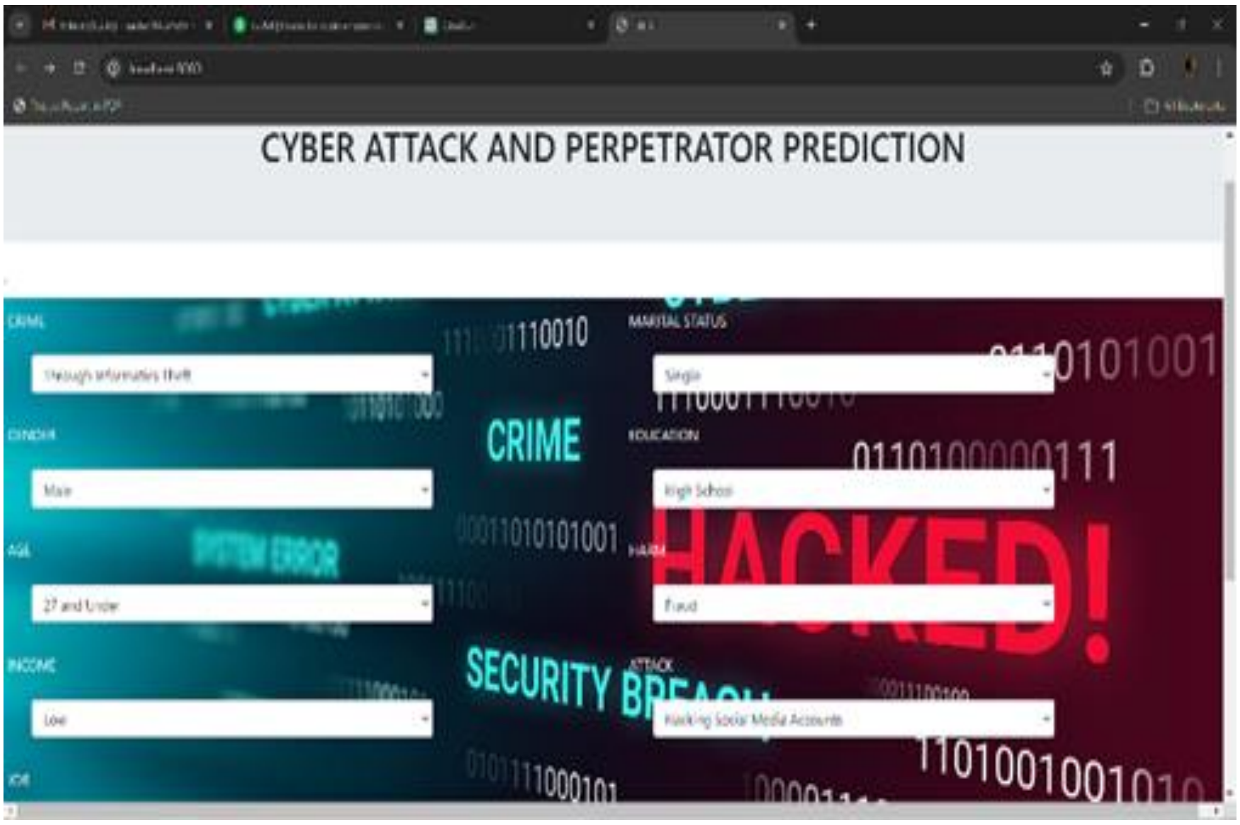
```
predictLR = lr.predict(X_test)

print("")
print('Classification report of Logistic Regression Result is:')
print("")
print(classification_report(y_test,predictLR))
print("")

cm=confusion_matrix(y_test,predictLR)
print('Confusion Matrix result of Logistic Regression is:\n',cm)
print("")

accuracy = cross_val_score(lr, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

print("")
print("Accuracy Result of Logistic Regression is:",accuracy.mean() * 100)
LogR=accuracy.mean() * 100

def Bar_Chart():
    import matplotlib.pyplot as plt
    data=[LogR]
    alg="Logistic Regression"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("m"))
    plt.title("Accuracy Result of Logistic Regression",fontsize=15)
    plt.legend(b,data,fontsize=9)
Bar_Chart()
def plot_confusion_matrix(cm, title='Confusion matrix-Logistic Regression',
cmap=plt.cm.autumn):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
cm1=confusion_matrix(y_test, predictLR)
print('Confusion matrix-Logistic Regression:')
print(cm)
plot_confusion_matrix(cm)
df.columns

#preprocessing, split test and dataset, split response variable
X = df[['Crime', 'Gender', 'Age', 'Income', 'Job', 'Maritalstatus', 'Education',
'Harm', 'Attack', 'AttackMethod']]
```

```
#Response variable
y = df['Perpetrator']
import imblearn
from imblearn.over_sampling import RandomOverSampler from collections
import Counter
ros =RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print("OUR DATASET COUNT         : ", Counter(y))

print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.30,
random_state=1, stratify=y_ros)

print("Number of training dataset : ", len(X_train))
print("Number of test dataset     : ", len(X_test))
print("Total number of dataset    : ", len(X_train)+len(X_test))

from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

lr = LogisticRegression()

lr.fit(X_train,y_train)

predictLR = lr.predict(X_test)
print("")

print('Classification report of Logistic Regression Result is:')
print("")

print(classification_report(y_test,predictLR))
print("")

cm=confusion_matrix(y_test,predictLR)
print('Confusion Matrix result of Logistic Regression is:\n',cm)
print("")

accuracy = cross_val_score(lr, x_ros, y_ros, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
print("")
print("Accuracy Result of Logistic Regression is:",accuracy.mean() * 100)
LogR=accuracy.mean() * 100
def Bar_Chart():
   import matplotlib.pyplot as plt
   data=[LogR]

   alg="Logistic Regression"
   plt.figure(figsize=(5,5))
   b=plt.bar(alg,data,color=("m"))
   plt.title("Accuracy Result of Logistic Regression",fontsize=15)
   plt.legend(b,data,fontsize=9)

Bar_Chart()
def plot_confusion_matrix(cm, title='Confusion matrix-Logistic Regression',
cmap=plt.cm.autumn):
   plt.imshow(cm, interpolation='nearest', cmap=cmap)
   plt.title(title)
   plt.colorbar()

cm1=confusion_matrix(y_test, predictLR)
print('Confusion matrix-Logistic Regression:')
print(cm)
plot_confusion_matrix(cm)
```

CHAPTER 10
RESULT AND DISCUSSION

10.1 Result

The system get input from the user and then it would be processed by already we trained data that would be checked by the input of users and the find out which prediction method is suited and what kind attack would be taken that place and who is perpetrator.



CRIME: Through Informatics Theft
MARITAL STATUS: Single
GENDER: Male
EDUCATION: High School
AGE: 27 and Under
HARM: Fraud
INCOME: Low
ATTACK: Hacking Social Media Accounts
JOB

Fig. 10.1.1 Cyber Attack Input Page

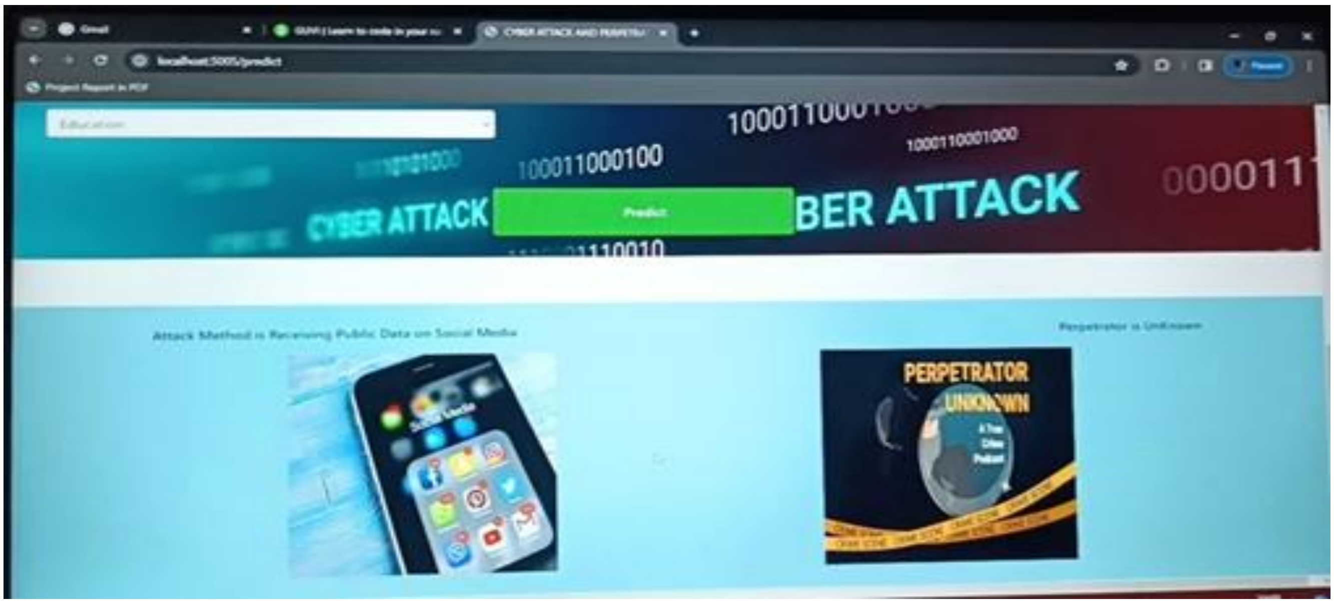


Fig. 10.1.2 Cyber Attack Output Page

10.2 ANALYSIS

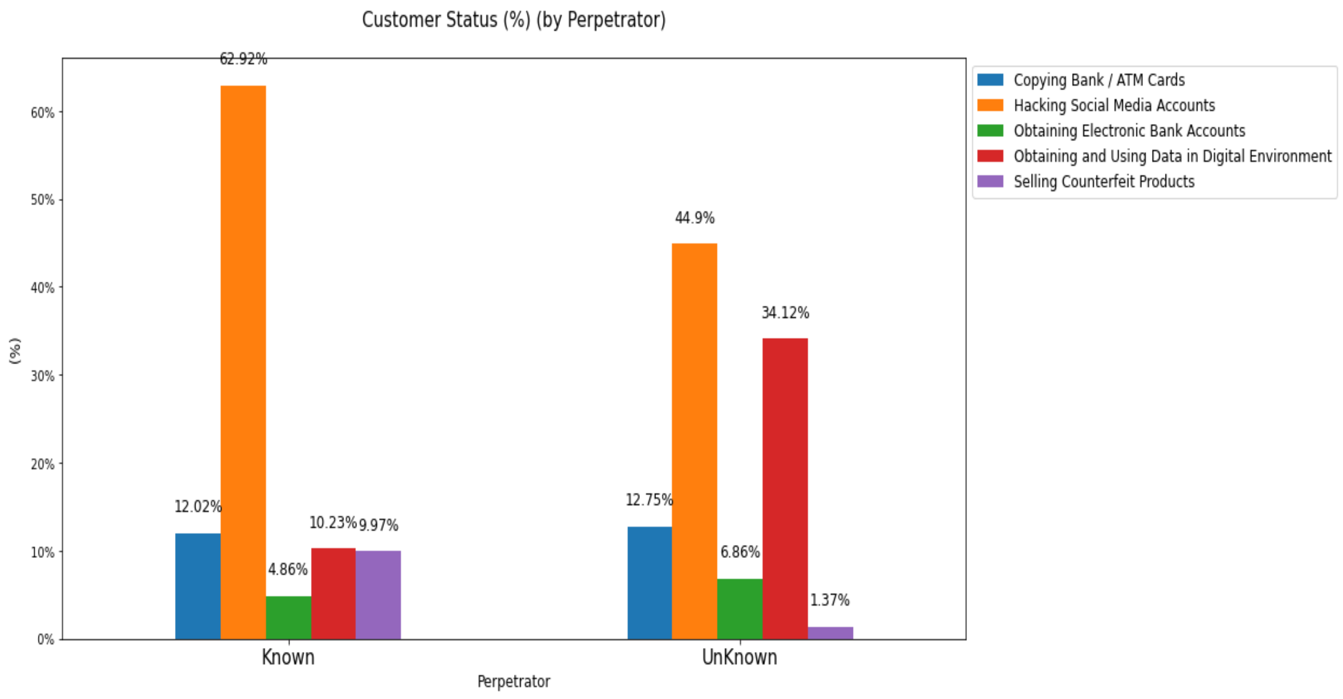


Fig. 10.2.1 Cyber Attack Bar Chart

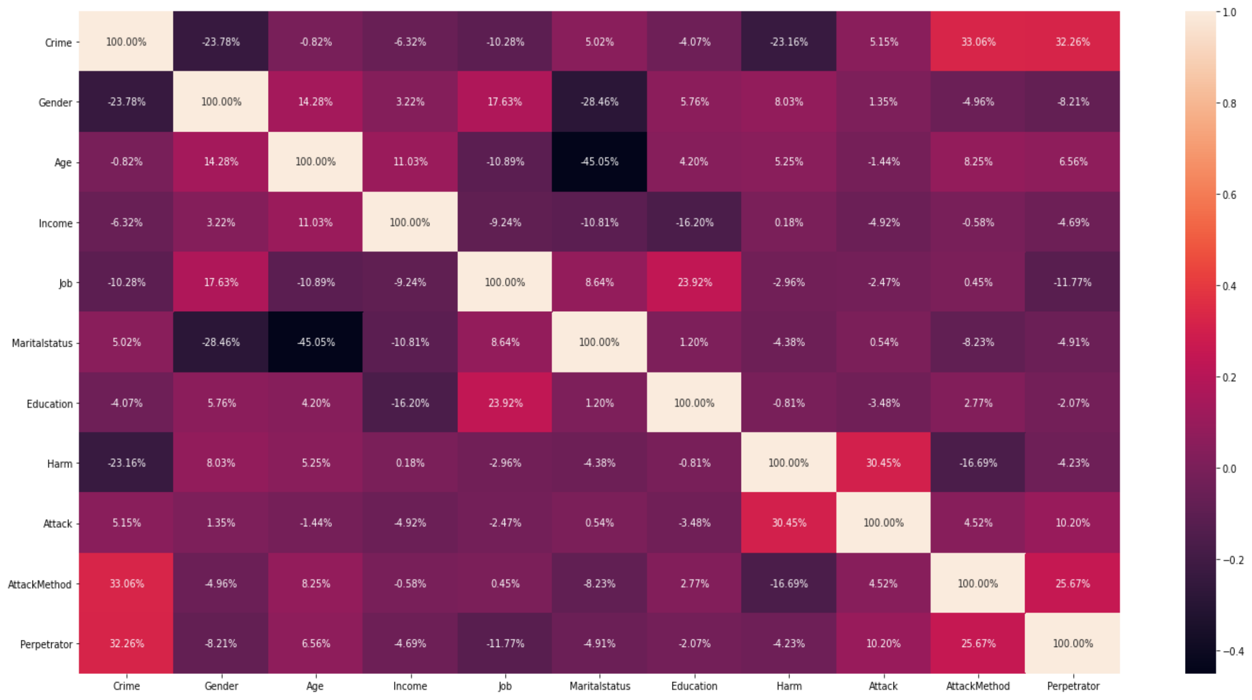


Fig. 10.2.2 Cyber Attack Warm Box Chart

10.3 CONCLUSION

This paper provides numerous surveys on cyber-attack analysis and methodologies for prediction. This problem is set to find the what kind of attack would be used and analyzed. And this method would be compared and analyzed by the prediction methods. We can separate in three findings method.

Firstly, we most check the cybersecurity prediction methods employ a framework to analyze and anticipate potential attacks or security incidents in the future.

Secondly, we have introduced the new ways for the core basic of machine learning and data mining that would have the likelihood to find which kind of attack would be used and that would help to study about the crimes happen on over the network.

It helps to taking the quick response for the attack on the network and how to take action for that. In conclusion, predicting cyber-attacks remains a captivating subject of research that has garnered significant attention from numerous scholars on multiple occasions.

10. 4 Future Work

For future endeavors’, the prediction of wrongdoing, criminal behaviour, victim profiling, and cyber-attacks can be facilitated through the application of deep learning algorithms, with the outcomes subject to comparison. Collaborations with authorized units possessing crime databases could further enrich the analysis by incorporating cybercrime data from diverse regions. This comparative analysis across different areas can provide valuable insights.

11. REFERENCES

[1] Zahid Anwar a, b, Asad Waqar Malik a, Sharifullah Khan a, umara Noor a c, Shahzad Saleem a, “A Machine Learning Framework for Investigating Data Breaches Based on Semantic Analysis of Adversary’s Attack Patterns in Threat Intelligence Repositories”,2019.

[2] Brandon Amos, Hamilton Turner, Jules White Dept. of Electrical and Computer Engineering, Virginia Tech Blacksburg, Virginia, USA, "Applying machine learning classi? ers to dynamic Android malware detection at scale",2013.

[3] Q. K. A. Mirza, I. Awan, M. Younas, Cloudintell: An intelligent malware detection system, Future Generation Computer Systems 86 (2018) 1042–1053.

[4] Martin Hus´ak, Jana Kom´arkov´a, Elias Bou-Harb, and Pavel?Celeda, “Survey of Attack Projection, Prediction, and Forecasting in Cyber Security”,2018.

[5] UmaraNoora,c, ZahidAnwara,b, TehminaAmjadc,Kim-KwangRaymond Chood, "A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise",2019

[6] S. More, M. Matthews, A. Joshi, T. Finin, A knowledge-based approach to intrusion detection modelling, in: IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, IEEE, 2012, pp. 75–81.

[7] V. Mulwad, W. Li, A. Joshi, T. Finin, K. Viswanathan, Extracting information about security vulnerabilities from web text, in: Proceedings of the 2011 IEEE ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology Workshops, WI-IAT 2011, Lyon, France, IEEE, 2011, pp. 257–260.

[8] C. Sabottke, O. Suciu, T. Dumitras, Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits, in: 24th

USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, USENIX, 2015, pp. 1041–1056.

[9] I. Gha? r, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, F. J. Aparicio-Navarro, Detection of advanced persistent threat using machine-learning correlation analysis, Future Generation Computer Systems 89 (2018) 349–359.

[10] R. A. Ahmadian and A. R. Ebrahimi, “A survey of IT early warning systems: architectures, challenges, and solutions,” Security and Communication Networks, vol. 9, no. 17, pp. 4751–4776, 2016

[11] H. Debar and A. Wespi, "Aggregation and correlation of intrusion detection alerts," in International Workshop on Recent Advances in Intrusion Detection. Springer, 2001, pp. 85–103.

[12] S. Shin, S. Lee, H. Kim, and S. Kim, “Advanced probabilistic approach for network intrusion forecasting and detection,” Expert Systems with Applications, vol. 40, no. 1, pp. 315 – 322, 2013.

[13] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin,S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, ``Tensor_ow: A system for large-scale machine learning," in Proc. 12th USENIX Symp. Operating Syst. Design Implement., 2016, pp. 265_283.

[14] D. Baylor, ``TFX: A tensor_ow-based production-scale machine learning platform," inProc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2017, pp. 1387_1395.

[15] Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, ``Learning activation functions to improve deep neural networks," 2014, arXiv:1412.6830. [Online]. Available: https://arxiv.org/abs/1412.6830.

[16] Prabha, R., Senthil, G.A., Razmah, M., Akshaya, S.R., Sivashree, J., Cyrilla Swathi, J. (2023). A Comparative Study of SVM, CNN, and DCNN Algorithms for Emotion Recognition and Detection. In: Jacob, I.J., Kolandapalayam Shanmugam, S., Izonin, I. (eds) Data Intelligence and

Cognitive Informatics. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-19-6004-8_64.

[17] M. Razmah, S. G. A, R. Prabha, D. B, S. S and A. Naveen, "LSTM Method for Human Activity Recognition of Video Using PSO Algorithm," 2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 2022, pp. 1-6, doi: 10.1109/ICPECTS56089.2022.10046783.

[18] S. G. A, R. Prabha, M. Razmah, T. Veeramakali, S. S and Y. R, "Machine Learning Heart Disease Prediction Using KNN and RTC Algorithm," *2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, Chennai, India, 2022, pp. 1-5, doi: 10.1109/ICPECTS56089.2022.10047501.

[19] R.M. Asha, P. Pondeepak, R. Prabha, G.A. Senthil, A. Padma Bharrathi, A novel approach effect of ocean acidification on oysters, Materials Today: Proceedings,2023,ISSN 2214-7853,https://doi.org/10.1016/j.matpr.2023.01.194.

[20] G. A. Senthil, P. Suganthi, R. Prabha, M. Madhumathi, S. Prabhu and S. Sridevi, "An Enhanced Smart Intelligent Detecting and Alerting System for Industrial Gas Leakage using IoT in Sensor Network," *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, 2023, pp. 397-401, doi: 10.1109/ICSSIT55814.2023.10060907.

[21] B. Zhao, D. Wang, G. Shi, D. Liu, and Y. Li, "Decentralized control for large-scale nonlinear systems with unknown mismatched interconnections via policy iteration," IEEE Trans. Syst., Man, Cybern., Syst., vol. 48, no. 10, pp. 1725–1735, Oct. 2018.

[22] V. Narayanan, A. Sahoo, S. Jagannathan, and K. George, "Approximate optimal distributed control of nonlinear interconnected systems using event-triggered nonzero-sum games," IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 5, pp. 1512–1522, May 2019.

[23] X. Yang and H. He, “Adaptive critic learning and experience replay for decentralized event-triggered control of nonlinear interconnected systems,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 50, no.11, pp. 4043–4055, Nov. 2020.

[24] C. Liu, H. Zhang, G. Xiao, and S. Sun, “Integral reinforcement learnin based decentralized optimal tracking control of unknown nonlinear large-scale interconnected systems with constrained-input,” Neurocomputing, vol. 323, pp. 1–11, Jan. 2019.

[25] X. Yang, H. He, and X. Zhong, “Approximate dynamic programming for nonlinear-constrained optimizations,” IEEE Trans. Cybern., early access, Jul. 19, 2019, doi: 10.1109/TCYB.2019.2926248.