

1 INTRODUCTION

The key purpose of a defensive deception technique is to mislead an attacker’s view and make it choose a suboptimal or poor action for the attack failure [33]. When both the attacker and defender are constrained in their resources, strategic interactions can be the key to beat an opponent. In this sense, non-game-theoretic defense approaches have inherent limitations due to lack of efficient and effective strategic tactics. Forms of deception techniques have been discussed based on certain classifications, such as hiding the truth vs. providing false information or passive vs. active for increasing attackers’ ambiguity or confusion [3, 9].

Game theory has been substantially used for dynamic decision making under uncertainty, assuming that players have consistent views. However, this assumption fails as players may often subjectively process asymmetric information available to them [22]. Hyper game theory [5] is a variant of game theory that provides a form of analysis considering each player’s subjective belief, misbelief, and perceived uncertainty and accordingly their effect on decision making in choosing a best strategy [22]. This paper leverages hyper game theory to resolve conflicts of views of multiple players as a robust decision making mechanism under uncertainty where the players may have different beliefs towards the same game. Hyper game theory models players, such as attackers and defenders in cyber security to deal with advanced persistent threat (APT) attacks. We dub this effort Foureye after the Foureye butterfly fish, demonstrating deceptive defense in nature [40].

To be specific, we identify the following nontrivial challenges in obtaining a solution. First of all, it is not trivial to derive realistic game scenarios and develop defensive deception techniques to deal with APT attacks beyond the reconnaissance stage. This aspect has not been explored in the state-of-the-art. Second, quantifying the degree of uncertainty in the views of attackers and defenders is challenging, although they are critical because how each player frames a game significantly affects its strategies to take. Third, given a number of possible choices under dynamic situations, dealing with a large number of solution spaces is not trivial whereas the deployment and maintenance of defensive deception techniques is costly in contested environments. We partly addressed these challenges in our prior work in [12]; however, its contribution is very limited in considering a small-scale network and

a small set of strategies with a highly simplified probability model developed using Stochastic Petri Network.

To be specific, this paper has the following **new key contributions**:

- We modeled an attack-defense game under uncertainty based on hypergame theory where an attacker and a defender have different views of the situation and are uncertain about strategies taken by their opponents.
- We reduced a player’s action space by using a sub game determined based on a set of strategies available where each sub game is formulated based on each stage of the cyber kill chain (CKC) based on a player’s belief under uncertainty.
- We considered multiple defense strategies, including defensive deception techniques whose performance can be significantly affected by an attacker’s belief and perceived uncertainty, which impacts its choice of a strategy.
- We modeled an attacker’s and a defender’s uncertainty towards its opponent (i.e., the defender and the attacker, respectively) based on how long each player has monitored the opponent and its chosen strategy. To the best of our knowledge, prior research on hyper game theory uses a predefined constant probability to represent a player’s uncertainty. In this work, we estimated the player’s uncertainty based on the dynamic, strategic interactions between an attacker and a defender.
- We conducted comparative performance analysis with or without a defender using defensive deception (DD) strategies and with or without perfect knowledge available towards actions taken by the opponent. We measured the effectiveness and efficiency of DD techniques in terms of a system’s security and performance, such as perceived uncertainty, hyper game expected utility, action cost, mean time to security failure (MTTSF or system lifetime), and improved false positive rate (FPR) of an intrusion detection by the DD strategies taken by the defender.

1.1 Defensive Deception

Defensive deception, in its essence, refers to the strategic use of misleading information or actions to protect oneself or one's interests from perceived threats or risks. It's a tactic employed in various contexts, including military strategy, cybersecurity, sports, and interpersonal relationships. Defensive deception involves creating a false perception or altering reality to gain an advantage or prevent harm. In this essay, we'll delve into the concept of defensive deception, its principles, applications, and ethical considerations.

1.1.1 Principles of Defensive Deception:

1.1.1.1 Perception Management: Defensive deception relies on manipulating the perceptions of others. This can involve presenting false information, hiding true intentions, or creating ambiguity to mislead adversaries.

1.1.1.2 Adaptability: Successful defensive deception requires flexibility and adaptability. Strategies must evolve in response to changing circumstances and the tactics of opponents.

1.1.1.3 Risk Assessment: Assessing risks accurately is crucial. Deceptive actions should mitigate threats without escalating conflicts or causing unintended consequences.

1.1.1.4 Information Control: Controlling the flow of information is essential. Revealing too much can undermine deception, while withholding critical details can enhance its effectiveness.

1.1.1.5 Maintaining Credibility: Deceptive practices should not erode trust or credibility in the long term. Balancing deception with honesty is necessary to preserve integrity and credibility.

1.1.2 Applications of Defensive Deception:

1.1.2.1 Military Strategy: In warfare, defensive deception aims to mislead enemy forces about troop movements, objectives, or capabilities. Camouflage, decoys, and misinformation are commonly used tactics to confuse adversaries and protect strategic assets.

1.1.2.2 Cybersecurity: Organizations employ defensive deception to thwart cyber threats and protect sensitive information. Techniques such as honeypots, which lure attackers into fake systems, and encryption, which hides data from unauthorized access, are examples of defensive deception in cybersecurity.

1.1.2.3 Sports Strategy: Athletes and teams use defensive deception to outmaneuver opponents and gain a competitive edge. Feints, dummy passes, and deceptive body language are common tactics in sports such as football, basketball, and martial arts.

1.1.2.4 Business Competition: In the corporate world, defensive deception is used to safeguard intellectual property, strategic plans, and market advantages. Misleading competitors about product capabilities, pricing strategies, or expansion plans can protect a company's interests and maintain market dominance.

1.1.2.5 Interpersonal Relationships: Individuals may employ defensive deception to protect themselves from harm or maintain privacy. This can involve concealing personal information, feigning emotions, or bluffing to avoid confrontation or exploitation.

1.1.3 Ethical Considerations:

1.1.3.1 Transparency: Deceptive practices should be balanced with transparency and accountability. Concealing information from stakeholders or manipulating perceptions without their consent can undermine trust and ethical integrity.

1.1.3.2 Proportionality: Deceptive actions should be proportionate to the threat and avoid causing undue harm or collateral damage. Excessive deception or manipulation can violate ethical norms and harm innocent parties.

1.1.3.3 Informed Consent: In contexts such as healthcare or negotiations, individuals have the right to make informed decisions based on accurate information. Deception should not infringe upon this right or compromise autonomy.

4.Long-Term Effects: Consideration should be given to the long-term consequences of defensive deception. While it may provide short-term advantages, erosion of trust or integrity can have lasting repercussions on relationships, reputation, and societal norms.

5.Human Dignity: Respect for human dignity should guide the use of defensive deception. Manipulating or exploiting others for personal gain undermines ethical principles and diminishes the value of individuals.

In conclusion, defensive deception is a complex and multifaceted concept that encompasses various principles, applications, and ethical considerations. While it can be a valuable tool for protecting interests and mitigating risks, it must be employed judiciously and ethically to avoid unintended consequences and preserve integrity. Balancing the need for security with respect for truth, transparency, and human dignity is essential in navigating the complexities of defensive deception in diverse contexts.

1.2 Hypergame Theory

Hypergame theory is a branch of game theory that expands upon traditional game theory by introducing the concept of nested games and considering the strategic interactions between players who have different levels of information and reasoning abilities. It was first introduced by Robert Aumann in the 1970s and has since been developed by various scholars.

At its core, hypergame theory recognizes that in real-world situations, decision-makers often face uncertainty not only about the actions of other players but also about the structure of the game itself. Traditional game theory assumes that all players have complete information about the game, know the payoffs associated with each possible action, and have perfect rationality. However, in many scenarios, players have limited information and must make decisions based on their beliefs about the game and the actions of others.

Hypergame theory addresses this limitation by allowing for multiple levels of reasoning. In a hypergame, players not only consider their own actions and the actions of others but also the potential reasoning processes of those other players. This leads to a nested structure of games within games, where players engage in strategic thinking about the decision-making processes of other players.

One of the key concepts in hypergame theory is the notion of meta-games, which are games about games. In a meta-game, players make decisions based on their beliefs about how other players will reason about the game. This introduces a level of strategic complexity beyond traditional game theory, as players must consider not only their immediate actions but also how those actions will influence the beliefs and strategies of other players.

Hypergame theory has applications in various fields, including economics, political science, and evolutionary biology. In economics, it can be used to analyze strategic interactions in markets where players have imperfect information about each other's preferences and strategies. In political science, hypergame theory can help understand how actors in a political system make decisions based on their beliefs about the motivations and strategies of other actors. In evolutionary biology, it can shed light on the strategic interactions between organisms in a population, where individuals must make decisions based on their beliefs about the behavior of other individuals.

Overall, hypergame theory provides a framework for analyzing strategic interactions in situations where players have limited information and must make decisions based on their beliefs about the actions and reasoning processes of others. By incorporating multiple levels of reasoning and considering the strategic implications of those reasoning processes, hypergame theory offers a more realistic and nuanced understanding of decision-making in complex, uncertain environments

2 LITERATURE SURVEY

2.1 Existing System

Garg and Grosu [15] proposed a game-theoretic deception framework in honey nets with imperfect information to find optimal actions of an attacker and a defender and investigated the mixed strategy equilibrium. Carroll and Grosu [10] used deception in attacker-defender interactions in a signaling game based on perfect Bayesian equilibria and hybrid equilibria. They considered defensive deception techniques, such as honeypots, camouflaged systems, or normal systems. Yin et al. [41] considered a Stackelberg attack defense game where both players make decisions based on their perceived observations and identified an optimal level of deceptive protection using fake resources.

Casey et al. [11] examined how to discover Sybil attacks based on an evolutionary signaling game where a defender can use a fake identity to lure the attacker to facilitate cooperation. Schlenker et al. [32] studied a sophisticated and naive APT attacker in the reconnaissance stage to identify an optimal defensive deception strategy in a zero-sum Stackelberg game by solving a mixed integer linear program. Unlike the above works cited [10, 11, 15, 32, 41], our work used hypergame theory which offers the powerful capability to model uncertainty, different views, and bounded rationality by different players. This way reflects more realistic scenarios between the attacker and defender. Hypergame theory has emerged to better reflect real world scenarios by capturing players’ subjective and imperfect belief, aiming to mislead them to adopt uncertain or non-optimized strategies. Although other game theories deal with uncertainty by considering probabilities that a certain event may happen, they assume that all players play the same game [34]. Hyper-game theory has been used to solve decision-making problems in military and adversarial environments House and Cybenko [20], Vane [37], Vane and Lehner [39]. Several studies [16, 17] investigated how players’ beliefs evolve based on hyper-game theory by developing a misbelief function measuring the differences between a player’s belief and the ground truth payoff of other players’ strategies. Kanazawa et al. [21] studied an individual’s belief in an evolutionary hyper-game and how this belief can be modelled by interpreter functions. Sasaki [31] discussed the concept of subjective rationalizability where an agent believes that its action is a best response to the other agent’s choices based on its perceived game.

Putro et al. [30] proposed an adaptive, genetic learning algorithm to derive optimal strategies by players in a hyper-game. Ferguson-Walter et al. [13] studied the placement of decoys based on a hypergame. This work developed a game tree and investigated an optimal move for both an attacker and defender in an adaptive game. Aljefri et al. [2] studied a first level hyper-game involving misbeliefs to resolve conflicts for two and then more decision makers. Bakker et al. [4] modeled a repeated hypergame in a dynamistochastic setting against APT attacks primarily in cyberphysical systems.

Disadvantages

- The system can't track attack which can be performed to exploit unknown vulnerabilities of software, which are not patched yet.
- The system can't track Fake identity attack which can be performed when packets are transmitted without authentication or internal nodes spoofing the ID of a source node

2.2 Proposed System

- The system modelled an attack defense game under uncertainty based on hyper-game theory where an attacker and a defender have different views of the situation and are uncertain about strategies taken by their opponents.
- The system reduced a player's action space by using a subgame determined based on a set of strategies available where each sub-game is formulated based on each stage of the cyber kill chain (CKC) based on a player's belief under uncertainty.
- The system considered multiple defense strategies, including defensive deception techniques whose performance can be significantly affected by an attacker's belief and perceived uncertainty, which impacts its choice of a strategy.
- The system modeled an attacker's and a defender's uncertainty towards its opponent (i.e., the defender and the attacker, respectively) based on how long each player has monitored the opponent and its chosen strategy. To the best of our knowledge, prior research on hypergame theory uses a predefined constant probability to represent a player's uncertainty. In this work,

we estimated the player’s uncertainty based on the dynamic, strategic interactions between an attacker and a defender.

- The system conducted comparative performance analysis with or without a defender using defensive deception (DD) strategies and with or without perfect knowledge available towards actions taken by the opponent. We measured the effectiveness and efficiency of DD techniques in terms of a system’s security and performance, such as perceived uncertainty, hyper-game expected utility, action cost, mean time to security failure (MTTSF or system lifetime), and improved false positive rate (FPR) of an intrusion detection by the DD strategies taken by the defender.

Advantages:

- APT Attack Procedure to Achieve Data Exfiltration in which the system define an APT attacker’s goal in that the attacker has reached and compromised a target node and successfully exfiltrated its confidential data.
- The system proposed many ML Classifiers to test and train the different types of attacks and can be predicted by using same classifiers.

3 SYSTEM STUDY

3.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

3.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.2 Software Requirements:

- ❖ **Operating system** : Windows 7 Ultimate.
- ❖ **Coding Language** : Python.
- ❖ **Front-End** : Python.
- ❖ **Back-End** : Django-ORM
- ❖ **Designing** : Html, css, javascript.
- ❖ **Data Base** : MySQL (WAMP Server).

5 DESIGN

5.1 Input Design:

In the realm of Foureye: Defensive Deception based on Hypergame Theory Against Advanced Persistent Threats, the Input Design serves as the crucial bridge between the intricate information system and its human operators. This facet involves crafting precise specifications and procedures for data preparation, ensuring that transaction data is seamlessly transformed into a usable format for processing. Such transformation may entail various methods, whether it's automated extraction from documents or direct data entry by personnel. The paramount goals of input design encompass minimizing input requirements, error control, expediency, simplicity, and maintaining privacy and security protocols.

Objectives:

- Input Design stands as the pivotal process of translating user-centric input descriptions into a meticulously structured computer-based system. This design phase is indispensable for averting data input errors and guiding management towards accurate insights derived from the computerized system.
- Achieving this objective necessitates the creation of user-friendly interfaces for data entry, capable of handling substantial data volumes. The overarching aim of input design is to streamline data entry processes while ensuring accuracy. The interface is tailored to accommodate all necessary data manipulation functions and provides intuitive record viewing features
- Validity checks are inherent to the data entry process, implemented through user-friendly interfaces. Prompt feedback messages are integrated to guide users through the data input process seamlessly.

5.2 Output Design:

Within the scope of the Foureye project, Defensive Deception based on Hypergame Theory Against Advanced Persistent Threats, Output Design assumes paramount importance. A quality output, in this context, is one that precisely meets the requirements of end-users, presenting information clearly and effectively. Outputs serve as the primary means of communicating processed information to users and other systems. The output design phase determines the format for immediate display and hard copy output, playing a pivotal role in enhancing user decision-making processes.

Objectives:

- Designing computer outputs necessitates a systematic and well-organized approach, ensuring the development of outputs that are easily navigable and comprehensible to users. During the analysis and design phases, specific output requirements are identified to meet the project's objectives.
- Various methods for presenting information are evaluated and selected based on their effectiveness in conveying the required insights.
- Documents, reports, or other formats containing information produced by the system are meticulously crafted to meet user needs and project goals.

The output forms of the Foureye project are designed to fulfill multiple objectives, including conveying information about past activities, current status, or future projections, signaling critical events or opportunities, and triggering or confirming actions as necessary for defensive deception against advanced persistent threats.

5.3 System Architecture

The architecture diagram outlines a comprehensive system catering to both service providers and remote users in the context of threat detection and data analysis. For service providers, the system offers functionalities such as login, browsing network data sets, training and testing models, and visualizing accuracy results through bar charts. Additionally, it provides tools to view both individual and overall threat detection statuses, alongside their ratios, facilitating a nuanced understanding of security measures. The ability to download predicted data sets empowers service providers with tangible resources for further analysis and decision-making. Moreover, features like browsing and training/testing data sets contribute to ongoing model refinement, while insights into popularity prediction types offer additional layers of analysis beyond threat detection.

5.3.1 Architecture Diagram

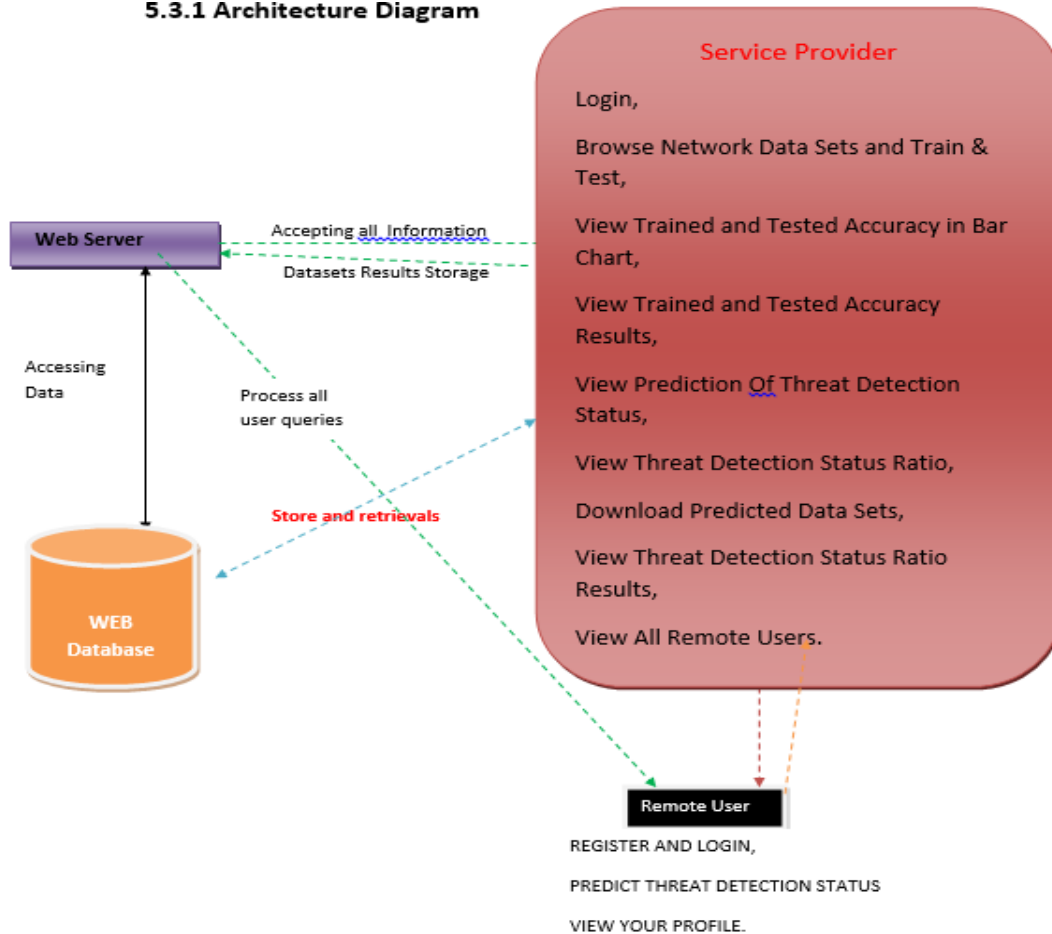


Fig : 5.3.1 Architecture Diagram

On the remote user front, the system allows for registration and login, enabling personalized experiences. Remote users can directly engage with threat detection by predicting status and accessing their profiles for a holistic view of their activities within the system. By bridging user interaction with security analytics, the architecture ensures that individuals can actively contribute to and benefit from threat detection efforts. This user-centric approach emphasizes empowerment through participation and visibility, fostering a collaborative ecosystem for addressing security challenges. Overall, the architecture balances the needs of service providers for robust analytics and tools with the user's desire for seamless engagement and insight into their own security status.

5.4 UML Diagram

5.4.1 Class Diagram

A class diagram stands as a cornerstone of static structure diagrams, offering a comprehensive portrayal of a system's underlying architecture. It meticulously outlines the system's classes, encapsulating their attributes, methods, and other structural elements, while also depicting the myriad relationships that interconnect them. These relationships encompass associations, aggregations, compositions, generalizations, and dependencies, each contributing to the holistic understanding of the system's design. By visually representing the class structure and relationships, class diagrams serve as a powerful communication tool, fostering clear and concise discourse among project stakeholders, including developers, designers, and domain experts. Moreover, they facilitate the identification of key system components, the allocation of responsibilities, and the detection of potential design flaws or inefficiencies. Thus, class diagrams play a pivotal role throughout the software development lifecycle, from initial analysis and design to implementation, maintenance, and beyond, ensuring the integrity and coherence of the system's architecture.

➤ 5.4.1 Class Diagram :

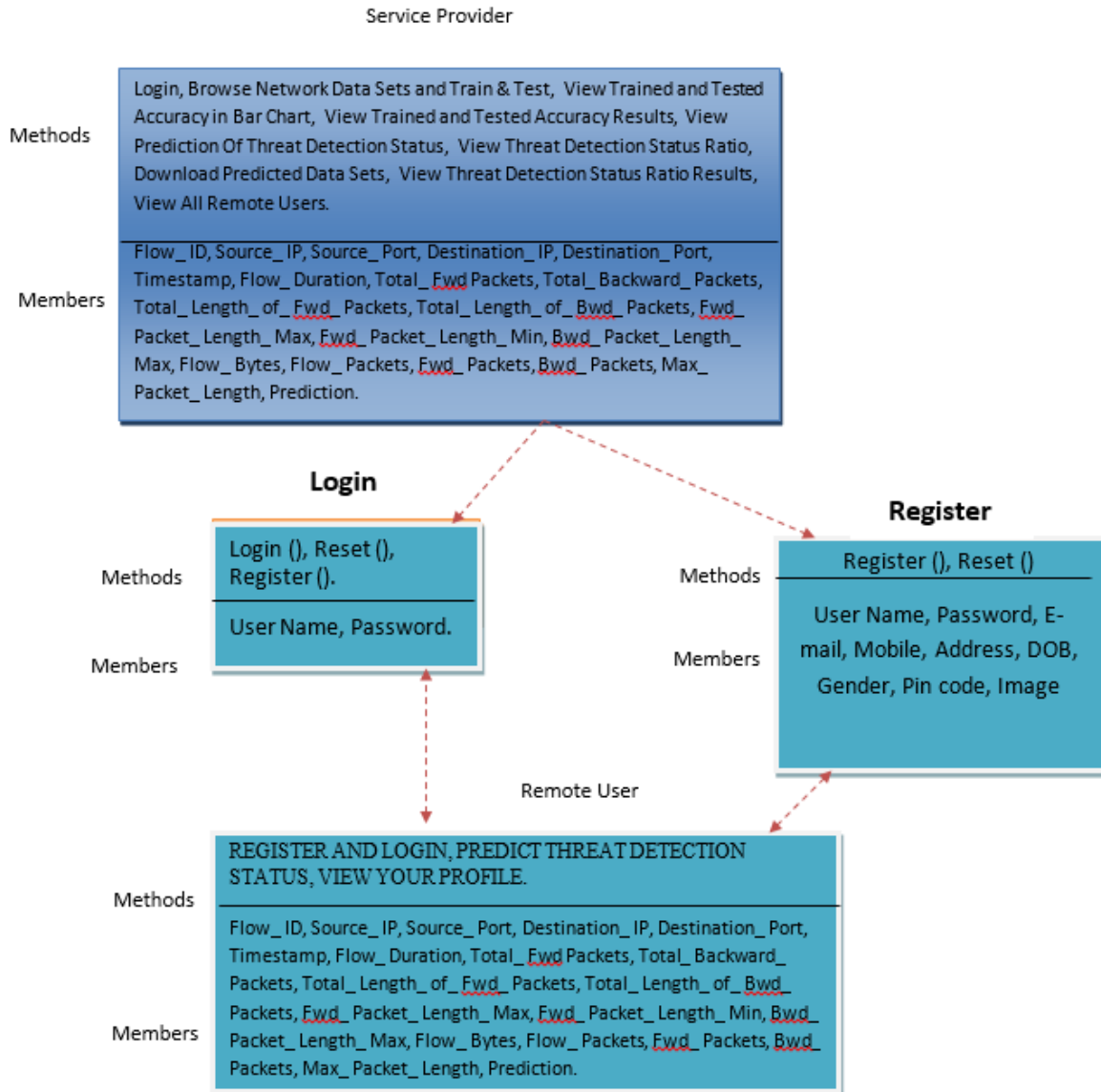


Fig : 5.4.1.Class Diagram

5.4.2 Flow Chart

A flowchart is a visual representation of a process, system, or algorithm, using standardized symbols to depict the sequence of steps or actions involved. It serves as a clear and concise tool for illustrating the logical flow of activities, decision points, and potential outcomes within a given framework or procedure.

5.4.2.1 Flow Chart : Remote User

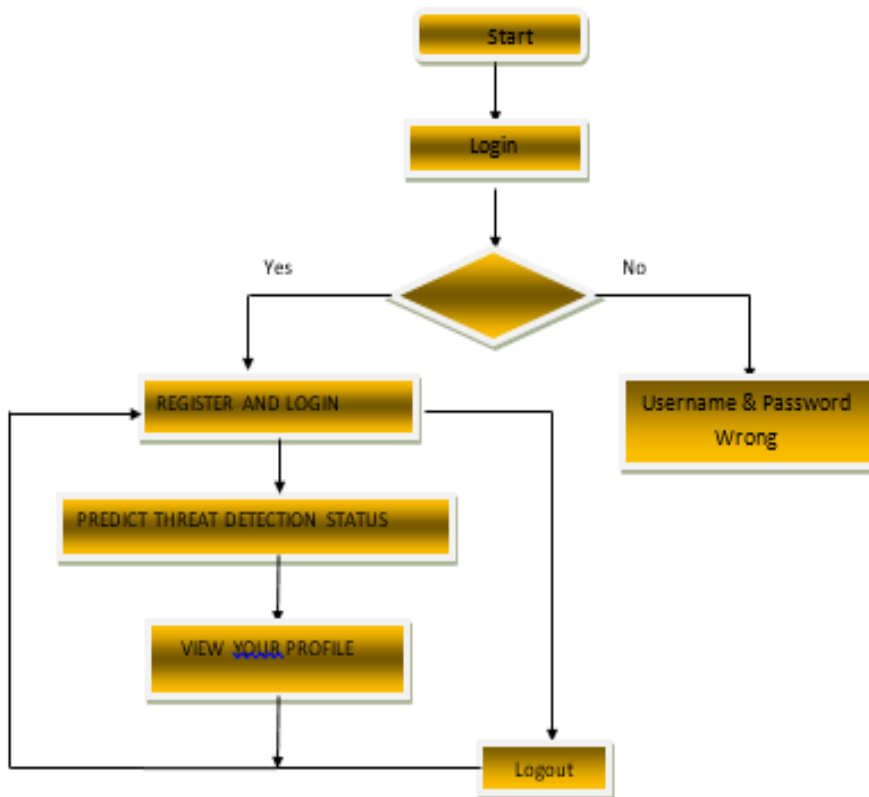


Fig : 5.4.2.1. Flow Chart : Remote User

5.4.2.2 Flow Chart : Service Provider

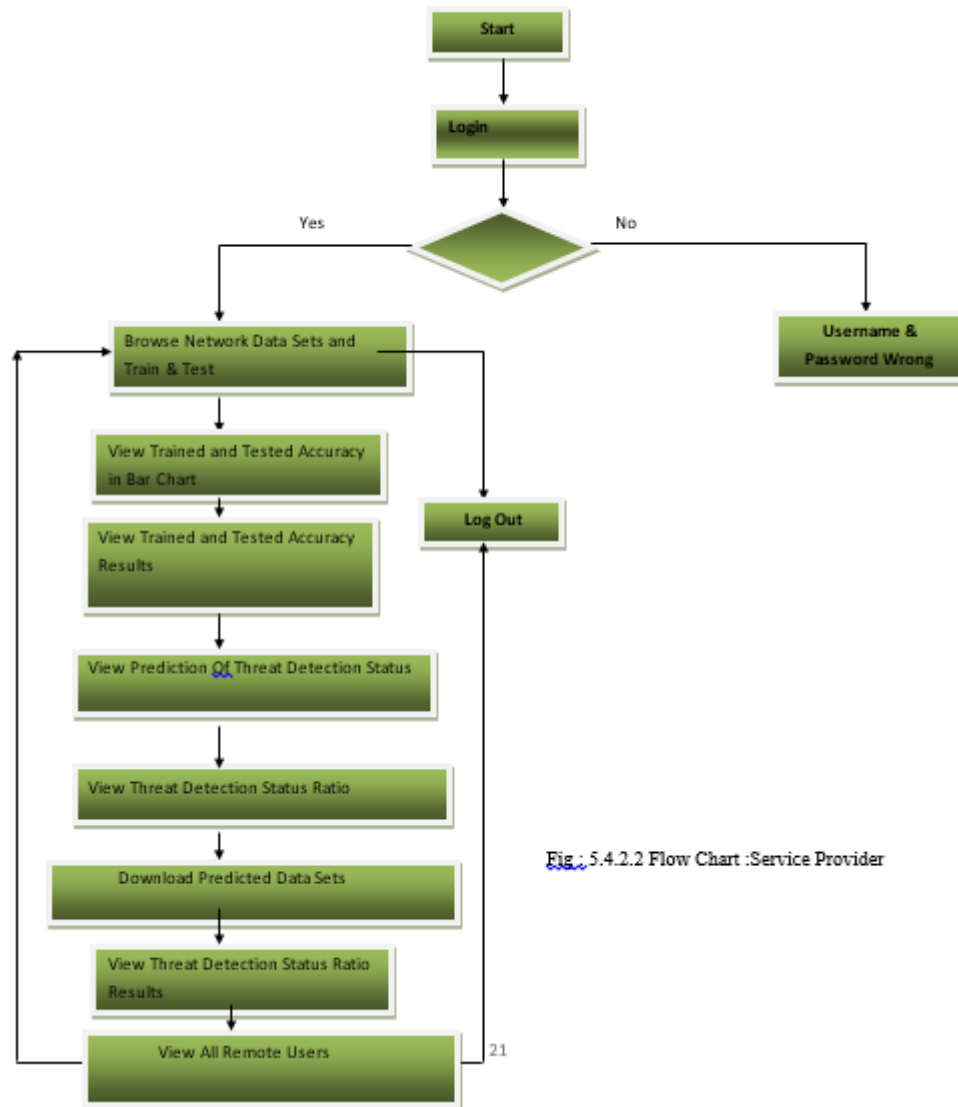


Fig : 5.4.2.2 Flow Chart :Service Provider

5.4.3 Data Flow Diagrams

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its process aspects. ADFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing.

5.4.3 Data Flow Diagram :

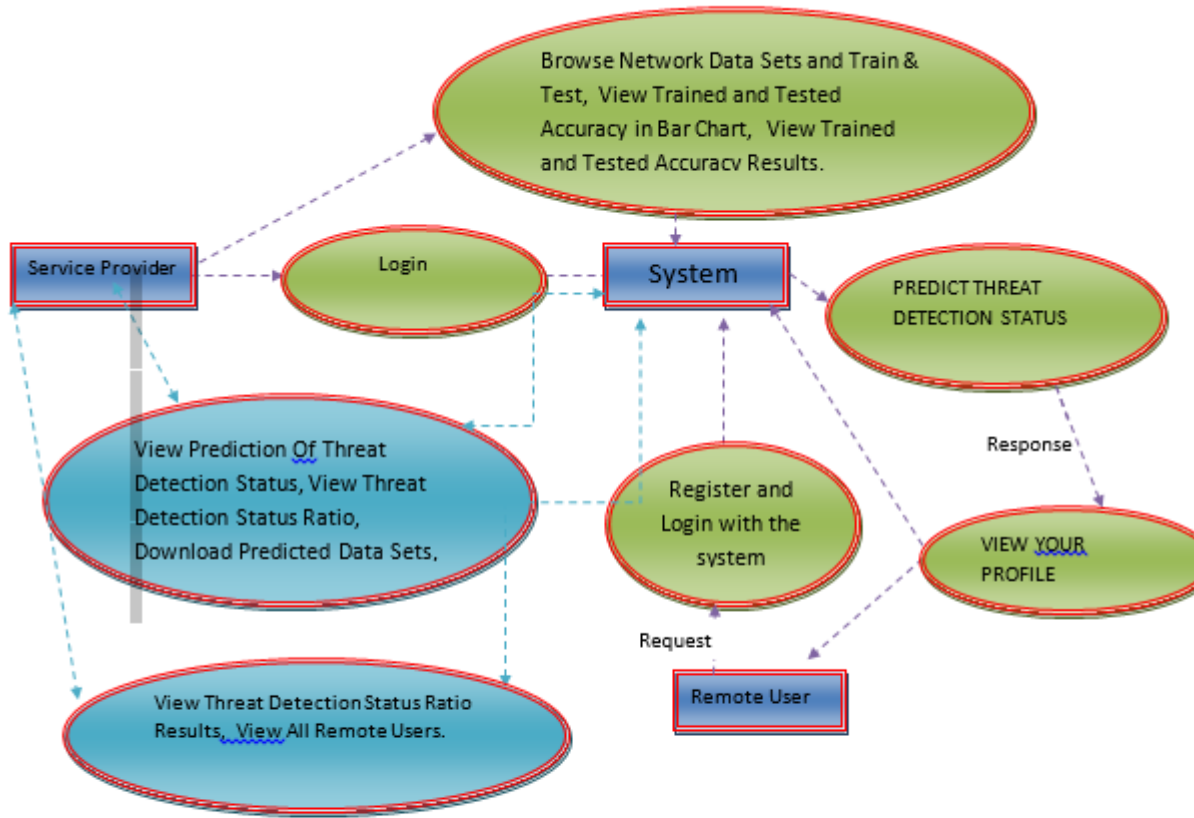


Fig.:5.4.3 Data Flow Diagram

5.4.4 Use Case Diagrams

A use case diagram in the Unified Modeling Language (UML) is a behavioral diagram that illustrates the interactions between actors (external entities such as users or systems) and a system to accomplish specific goals. It provides a high-level overview of the functionality of a system from the perspective of its users. Use case diagrams consist of actors, use cases, and relationships between them. Actors

represent roles played by users or other systems interacting with the system being modeled. Use cases represent specific functionalities or actions that the system can perform to fulfill the needs of its actors. Relationships between actors and use cases, such as associations and includes/extends relationships, depict how actors interact with the system to achieve their goals and how different use cases may relate to each other. Use case diagrams are valuable for capturing system requirements, understanding user needs, and guiding system design and development processes.

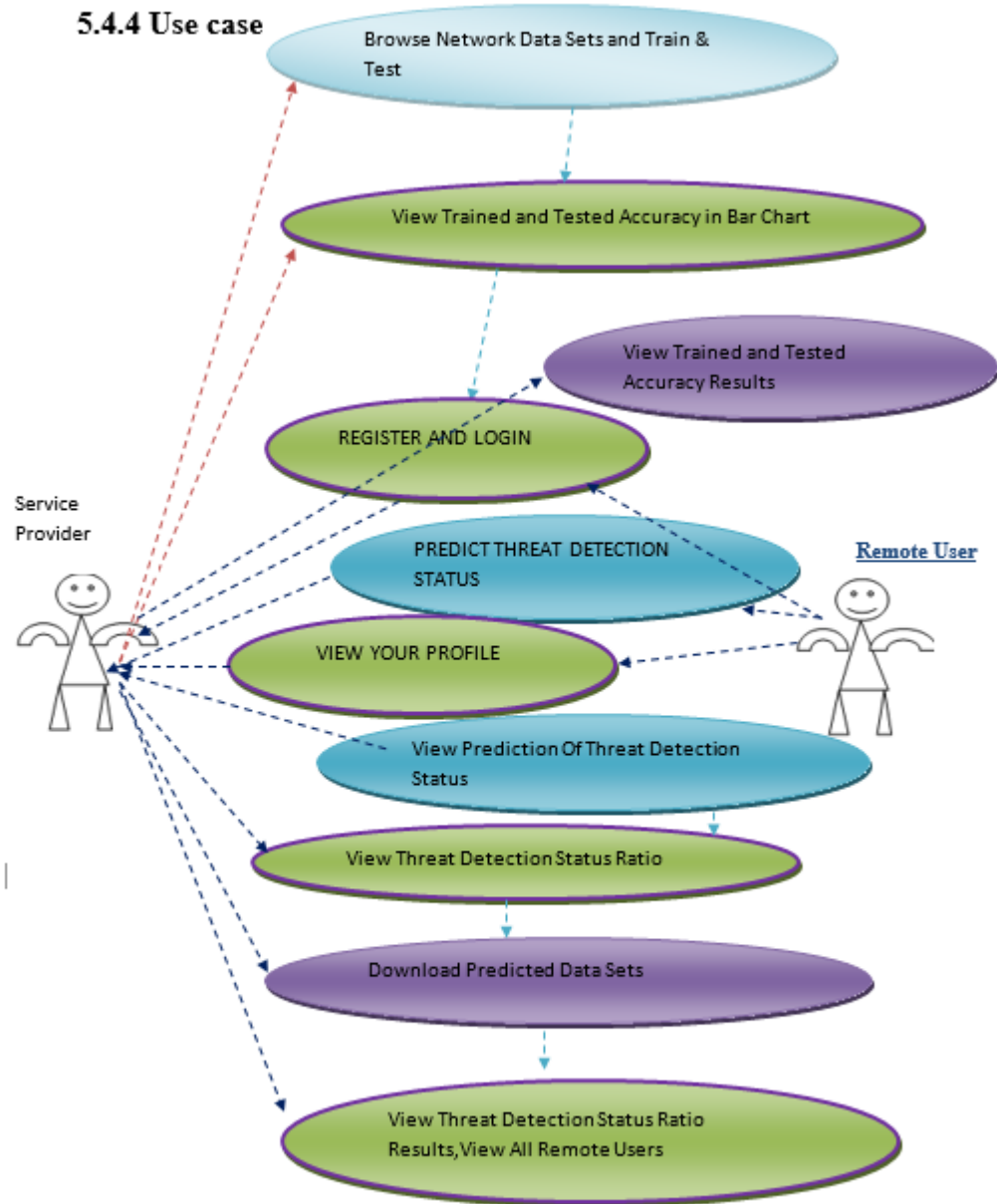
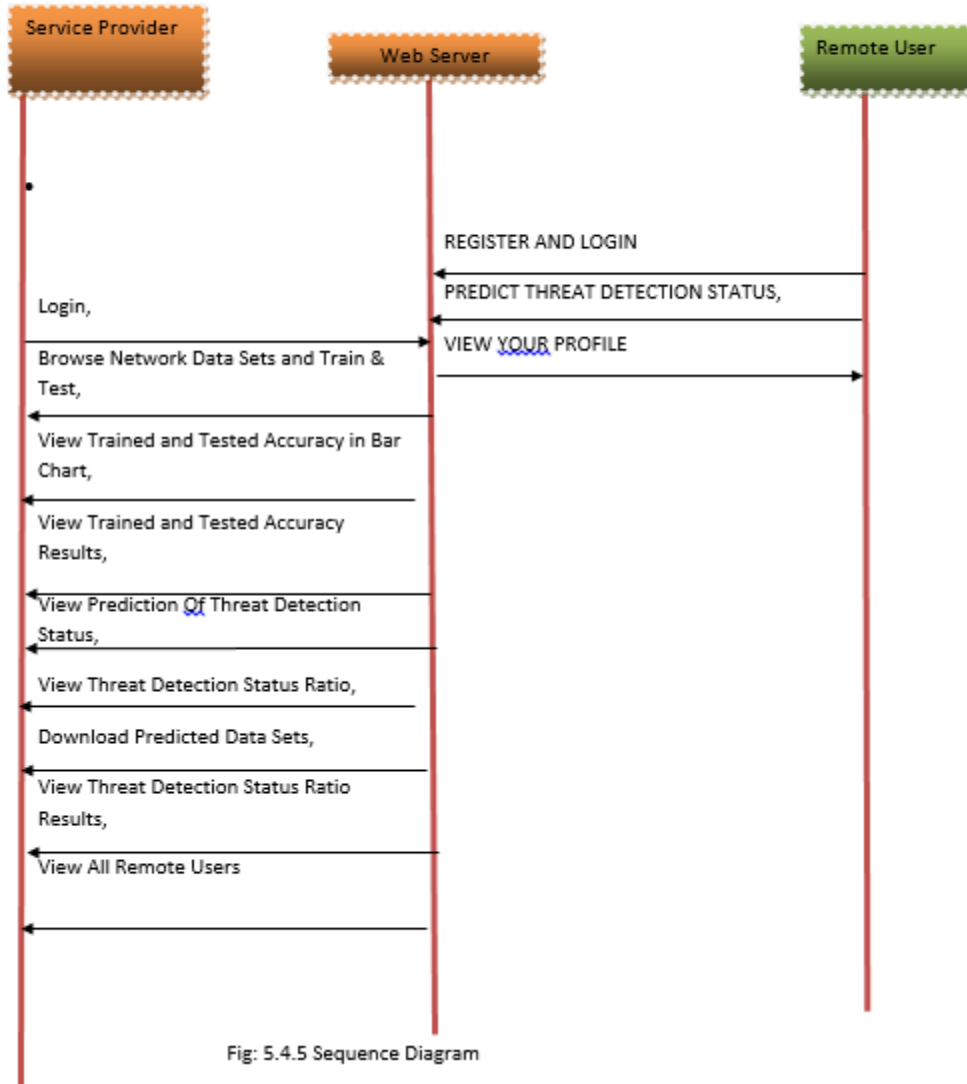


Fig :5.4.4 UseCase Diagram

5.4.5 SEQUENCE DIAGRAM

A sequence diagram in the Unified Modeling Language (UML) is a dynamic diagram that illustrates how objects interact with each other in a particular scenario or sequence of events. It portrays the flow of messages exchanged between objects over time, depicting the chronological order of these interactions. Sequence diagrams consist of lifelines, messages, activations, and optionally, fragments. Lifelines represent individual objects participating in the interaction, typically depicted as vertical lines. Messages are depicted as horizontal arrows between lifelines, indicating communication between objects. These messages can be synchronous (request-response), asynchronous (fire-and-forget), or self-referential (a message sent by an object to itself). Activations represent the period during which an object is actively processing a message. Additionally, sequence diagrams may include fragments, such as alternative, optional, or loop fragments, to represent conditional or repetitive behavior within the sequence of interactions. Sequence diagrams are valuable for visualizing the dynamic behavior of a system, understanding the sequence of operations during execution, identifying potential bottlenecks or concurrency issues, and verifying the correctness of system interactions. They are widely used in software design, development, and documentation processes.

5.4.5 Sequence Diagram



6 IMPLEMENTATION

6.1 MODULES

6.1.1 Service Provider

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as Login, Browse Network Data Sets and Train & Test, View Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results, View Prediction Of Threat Detection Status, View Threat Detection Status Ratio, Download Predicted Data Sets, View Threat Detection Status Ratio Results, View All Remote Users.

6.1.2 View and Authorize Users

In this module, the admin can view the list of users who all registered. In this, the admin can view the user’s details such as, user name, email, address and admin authorizes the users.

6.1.3 Remote User

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like REGISTER AND LOGIN,PREDICT THREAT DETECTION STATUS,VIEW YOUR PROFILE.

6.2 SAMPLE CODE

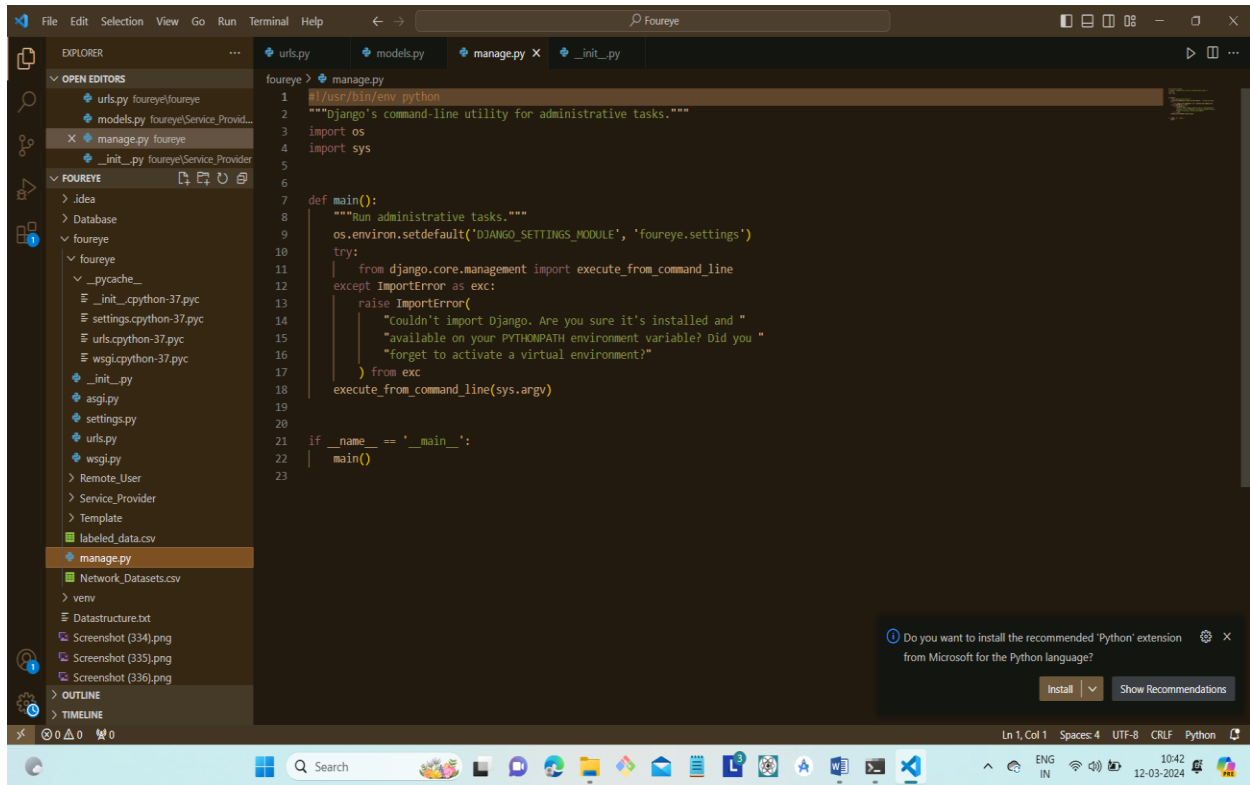


Fig 6.2.1: Sample code-1

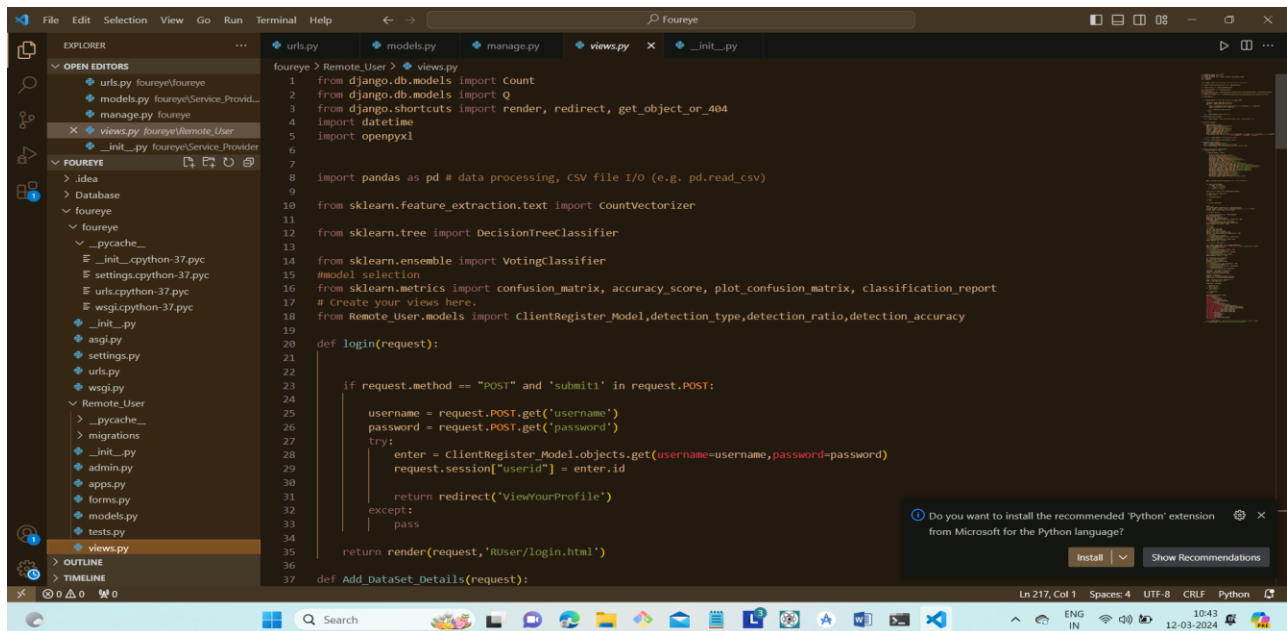


Fig 6.2.2: Sample code-2

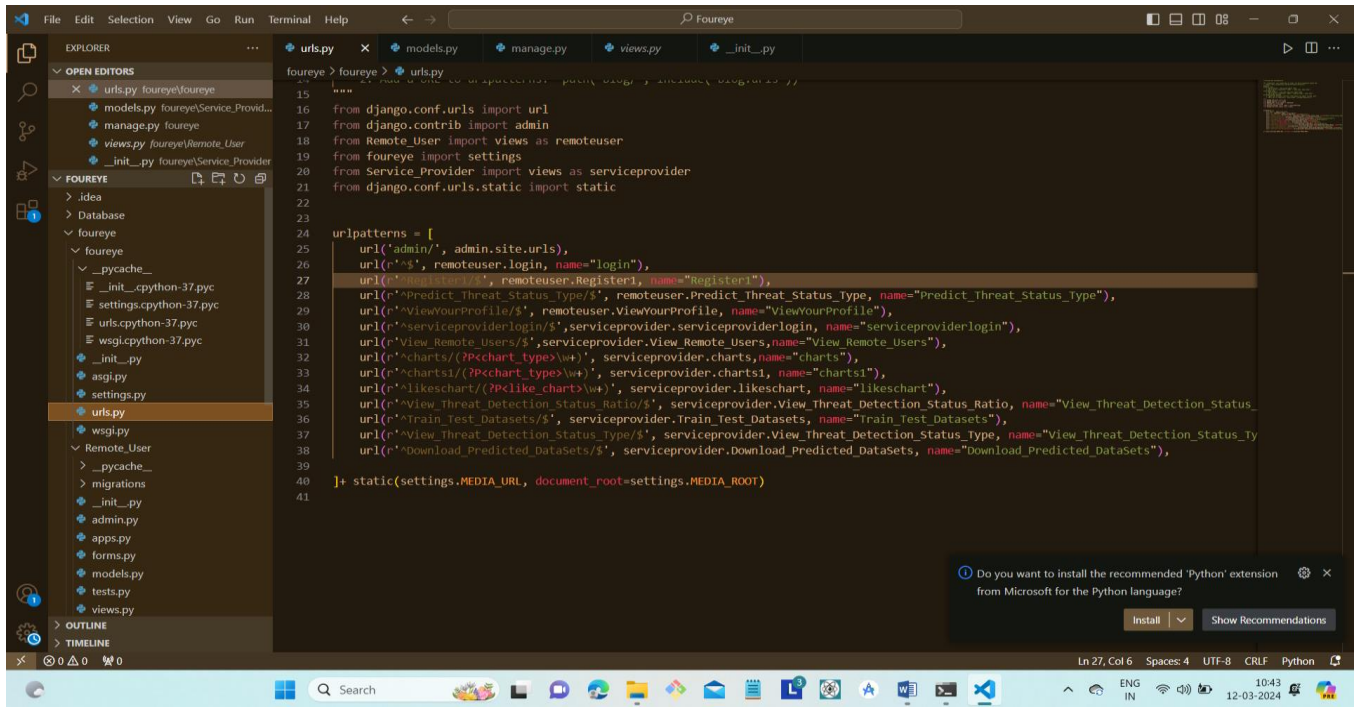


Fig 6.2.3: Sample code-3

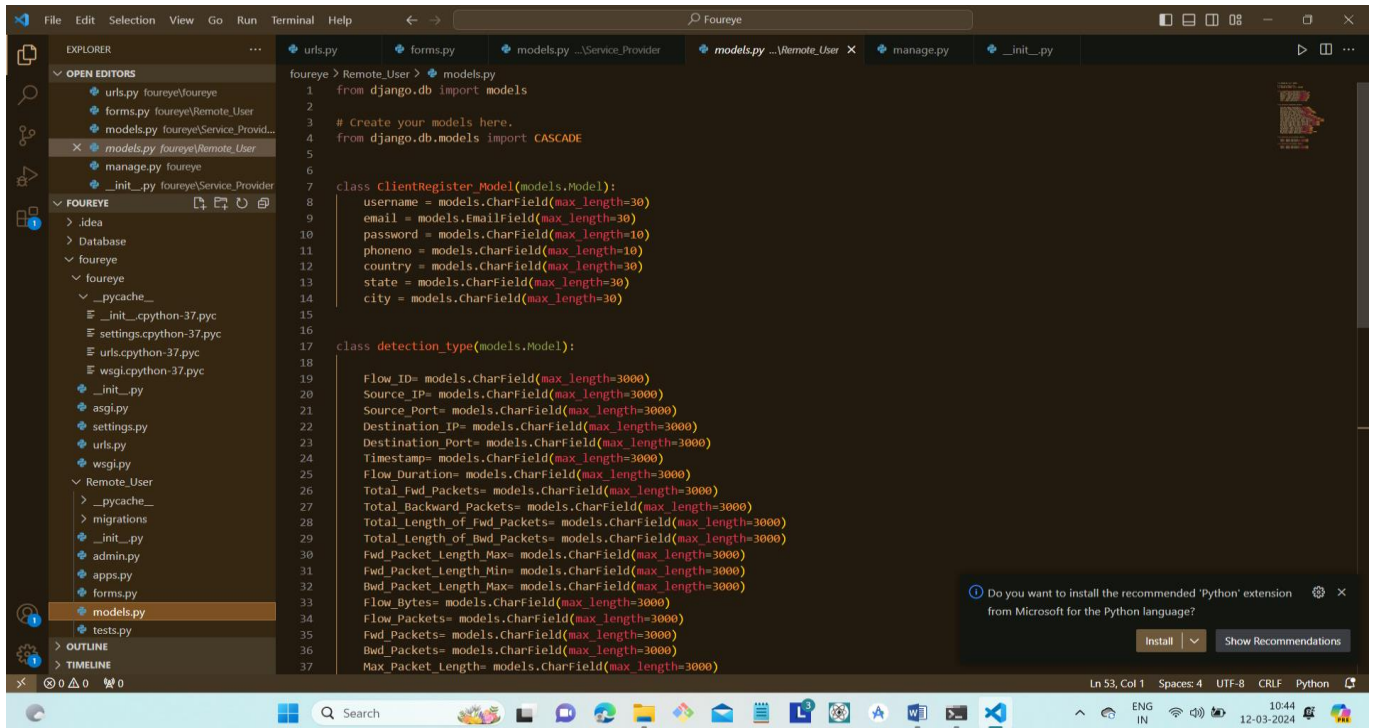


Fig 6.2.4: Sample code-4

6.3 DATASET

6.3.1 Labelled dataset

labeled_data - Microsoft Excel

FILE	HOME	INSERT	PAGE LAYOUT	FORMULAS	DATA	REVIEW	VIEW										
<div><div><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></</div></div></div></div></div></div>																	

Fig 6.3.1 :Labelled dataset

6.3.2 Network Datasets

FILE

HOME

INSERT

PAGE LAYOUT

FORMULAS

DATA

REVIEW

VIEW

Cut

Copy

Format Painter

Clipboard

Calibri

11

7 ALGORITHMS

7.1 Decision Tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C_1, C_2, \dots, C_k is as follows:

Step 1. If all the objects in S belong to the same class, for example C_i , the decision tree for S consists of a leaf labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O_1, O_2, \dots, O_n . Each object in S has one outcome for T so the test partitions S into subsets S_1, S_2, \dots, S_n where each object in S_i has outcome O_i for T. T becomes the root of the decision tree and for each outcome O_i we build a subsidiary decision tree by invoking the same procedure recursively on the set S_i .

7.2 Gradient Boosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.^{[1][2]} When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradientboosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable

7.3 K-Nearest Neighbors (KNN)

- Simple, but a very powerful classification algorithm
- Classifies based on a similarity measure
- Non-parametric
- Lazy learning
- Does not “learn” until the test example is given

- Whenever we have a new data to classify, we find its K-nearest neighbors from the training data

Example

- Training dataset consists of k-closest examples in feature space
- Feature space means, space with categorization variables (non-metric variables)
- Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset

7.4 Logistic Regression Classifiers

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name *logistic regression* is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name *multinomial logistic regression* is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar.

Logistic regression competes with discriminant analysis as a method for analyzing categorical response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does.

This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits, likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values

and provides ROC curves to help determine the best cutoff point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

7.5 Naïve Bayes

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature .

Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0b and RapidMiner 4.6.0). We try above all to understand the obtained results.

7.6 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configurat

7.7 SVM

In classification tasks a discriminant machine learning technique aims at finding, based on an *independent and identically distributed (iid)* training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point x and assigns it to one of the different classes that are a part of the classification task.

Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space.

SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to *genetic algorithms* (GAs) or *perceptrons*, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and Perceptrons is only to minimize error during training, which will translate into several hyper planes’ meeting this requirement.

8 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.1 TYPES OF TESTS

8.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.1.7 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

8.1.8 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

8.1.9 Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

8.1.10 Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.1.11 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.1.12 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.2 Testing Methodologies

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- User Acceptance Testing.
- Output Testing.
- Validation Testing.

8.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

8.2.2 Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

8.2.2.1 Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

8.2.2.2 Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

8.2.3 OTHER TESTING METHODOLOGIES

8.2.3.1 User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

8.2.3.2 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

8.2.3.3 Validation Checking

Validation checks are performed on the following fields.

- **Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

- **Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each

module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

8.3 Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

8.3.1 Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

8.3.2 Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

8.4 USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

8.5 _MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

8.6 TESTING STRATEGY :

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-

operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

8.6.1 SYSTEM TESTING:

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

8.6.2 UNIT TESTING:

In unit testing different modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

9 SOURCE CODE

```
from django.db.models import Count

from django.db.models import Q

from django.shortcuts import render, redirect, get_object_or_404

import datetime

import openpyxl


import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)


from sklearn.feature_extraction.text import CountVectorizer


from sklearn.tree import DecisionTreeClassifier


from sklearn.ensemble import VotingClassifier


#model selection


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


# Create your views here.


from Remote_User.models import
ClientRegister_Model,detection_type,detection_ratio,detection_accuracy


def login(request):
```

```
if request.method == "POST" and 'submit1' in request.POST:

    username = request.POST.get('username')
    password = request.POST.get('password')

    try:
        enter = ClientRegister_Model.objects.get(username=username,password=password)
        request.session["userid"] = enter.id

        return redirect('ViewYourProfile')
    except:
        pass

return render(request,'RUser/login.html')

def Add_DataSet_Details(request):

    return render(request, 'RUser/Add_DataSet_Details.html', {"excel_data": ""})

def Register1(request):

    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
```

```
password = request.POST.get('password')
phoneno = request.POST.get('phoneno')
country = request.POST.get('country')
state = request.POST.get('state')
city = request.POST.get('city')

ClientRegister_Model.objects.create(username=username, email=email, password=password,
phoneno=phoneno,

                                country=country, state=state, city=city)

return render(request, 'RUser/Register1.html')
else:
    return render(request, 'RUser/Register1.html')

def ViewYourProfile(request):
    userid = request.session['userid']
    obj = ClientRegister_Model.objects.get(id= userid)
    return render(request, 'RUser/ViewYourProfile.html', {'object':obj})

def Predict_Threat_Status_Type(request):
    if request.method == "POST":

        if request.method == "POST":

            Flow_ID= request.POST.get('Flow_ID')
```

```
Source_IP= request.POST.get('Source_IP')
Source_Port= request.POST.get('Source_Port')
Destination_IP= request.POST.get('Destination_IP')
Destination_Port= request.POST.get('Destination_Port')
Timestamp= request.POST.get('Timestamp')
Flow_Duration= request.POST.get('Flow_Duration')
Total_Fwd_Packets= request.POST.get('Total_Fwd_Packets')
Total_Backward_Packets= request.POST.get('Total_Backward_Packets')
Total_Length_of_Fwd_Packets= request.POST.get('Total_Length_of_Fwd_Packets')
Total_Length_of_Bwd_Packets= request.POST.get('Total_Length_of_Bwd_Packets')
Fwd_Packet_Length_Max= request.POST.get('Fwd_Packet_Length_Max')
Fwd_Packet_Length_Min= request.POST.get('Fwd_Packet_Length_Min')
Bwd_Packet_Length_Max= request.POST.get('Bwd_Packet_Length_Max')
Flow_Bytes= request.POST.get('Flow_Bytes')
Flow_Packets= request.POST.get('Flow_Packets')
Fwd_Packets= request.POST.get('Fwd_Packets')
Bwd_Packets= request.POST.get('Bwd_Packets')
Max_Packet_Length= request.POST.get('Max_Packet_Length')

data = pd.read_csv("Network_Datasets.csv", encoding='latin-1')

def apply_results(label):
```

```
    if (label == "Benign"):
        return 0 # No Threat

    elif (label == "Threat"):
        return 1 # Threat


data['Label'] = data['Class'].apply(apply_results)


x = data['Flow_ID'].apply(str)
y = data['Label']


cv = CountVectorizer()


print(x)
print(y)


x = cv.fit_transform(x)


models = []

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
X_train.shape, X_test.shape, y_train.shape


print("Naive Bayes")
```

```
from sklearn.naive_bayes import MultinomialNB

NB = MultinomialNB()

NB.fit(X_train, y_train)

predict_nb = NB.predict(X_test)

naivebayes = accuracy_score(y_test, predict_nb) * 100

print(naivebayes)

print(confusion_matrix(y_test, predict_nb))

print(classification_report(y_test, predict_nb))

models.append(('naive_bayes', NB))


# SVM Model

print("SVM")

from sklearn import svm

lin_clf = svm.LinearSVC()

lin_clf.fit(X_train, y_train)

predict_svm = lin_clf.predict(X_test)

svm_acc = accuracy_score(y_test, predict_svm) * 100

print(svm_acc)

print("CLASSIFICATION REPORT")

print(classification_report(y_test, predict_svm))

print("CONFUSION MATRIX")

print(confusion_matrix(y_test, predict_svm))

models.append(('svm', lin_clf))


print("Logistic Regression")
```



```
from sklearn.linear_model import LogisticRegression

reg = LogisticRegression(random_state=0, solver='lbfgs').fit(X_train, y_train)

y_pred = reg.predict(X_test)

print("ACCURACY")

print(accuracy_score(y_test, y_pred) * 100)

print("CLASSIFICATION REPORT")

print(classification_report(y_test, y_pred))

print("CONFUSION MATRIX")

print(confusion_matrix(y_test, y_pred))

models.append(('logistic', reg))


print("Decision Tree Classifier")

dtc = DecisionTreeClassifier()

dtc.fit(X_train, y_train)

dtcpredict = dtc.predict(X_test)

print("ACCURACY")

print(accuracy_score(y_test, dtcpredict) * 100)

print("CLASSIFICATION REPORT")

print(classification_report(y_test, dtcpredict))

print("CONFUSION MATRIX")

print(confusion_matrix(y_test, dtcpredict))


classifier = VotingClassifier(models)

classifier.fit(X_train, y_train)
```

```
y_pred = classifier.predict(X_test)

Flow_ID1 = [Flow_ID]
vector1 = cv.transform(Flow_ID1).toarray()
predict_text = classifier.predict(vector1)

pred = str(predict_text).replace("[", "")
pred1 = pred.replace("]", "")

prediction = int(pred1)

if prediction == 0:
    val = 'Benign'
elif prediction == 1:
    val = 'Threat'

print(prediction)
print(val)

detection_type.objects.create(
    Flow_ID=Flow_ID,
    Source_IP=Source_IP,
    Source_Port=Source_Port,
    Destination_IP=Destination_IP,
    Destination_Port=Destination_Port,
```

Timestamp=Timestamp,

Flow_Duration=Flow_Duration,

Total_Fwd_Packets=Total_Fwd_Packets,

Total_Backward_Packets=Total_Backward_Packets,

Total_Length_of_Fwd_Packets=Total_Length_of_Fwd_Packets,

Total_Length_of_Bwd_Packets=Total_Length_of_Bwd_Packets,

Fwd_Packet_Length_Max=Fwd_Packet_Length_Max,

Fwd_Packet_Length_Min=Fwd_Packet_Length_Min,

Bwd_Packet_Length_Max=Bwd_Packet_Length_Max,

Flow_Bytes=Flow_Bytes,

Flow_Packets=Flow_Packets,

Fwd_Packets=Fwd_Packets,

Bwd_Packets=Bwd_Packets,

Max_Packet_Length=Max_Packet_Length,

Prediction=val)

return render(request, 'RUser/Predict_Threat_Status_Type.html',{'objs': val}))

return render(request, 'RUser/Predict_Threat_Status_Type.html')

10 RESULTS

10.1 REMOTE USER PAGES

10.1.1 Registration Page

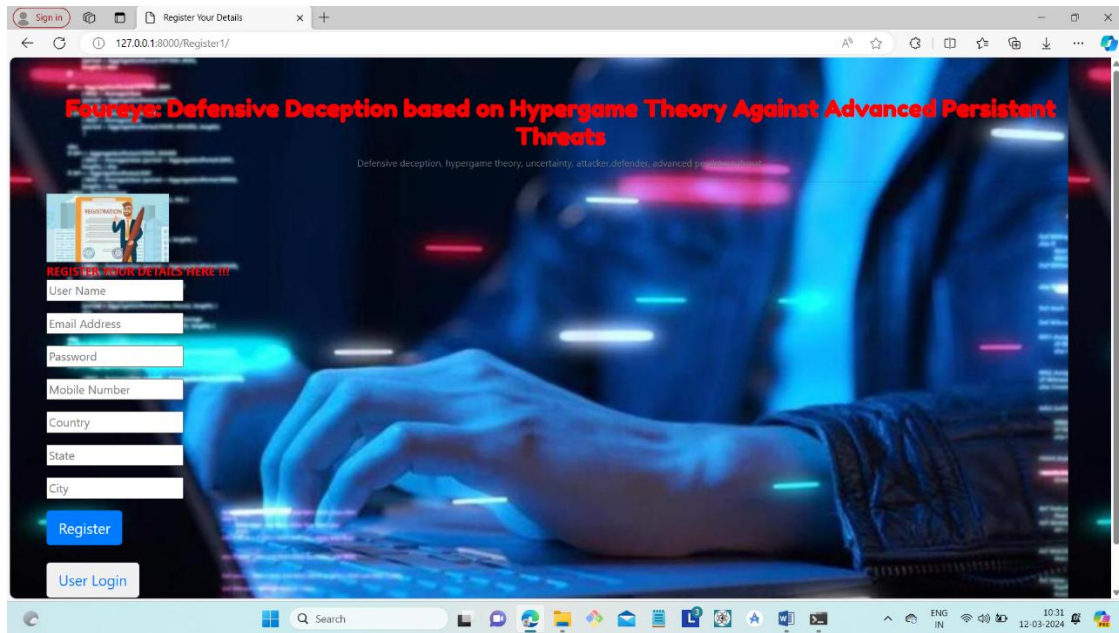


Fig 10.1.1 Registration Page

10.1.2 User Login Page

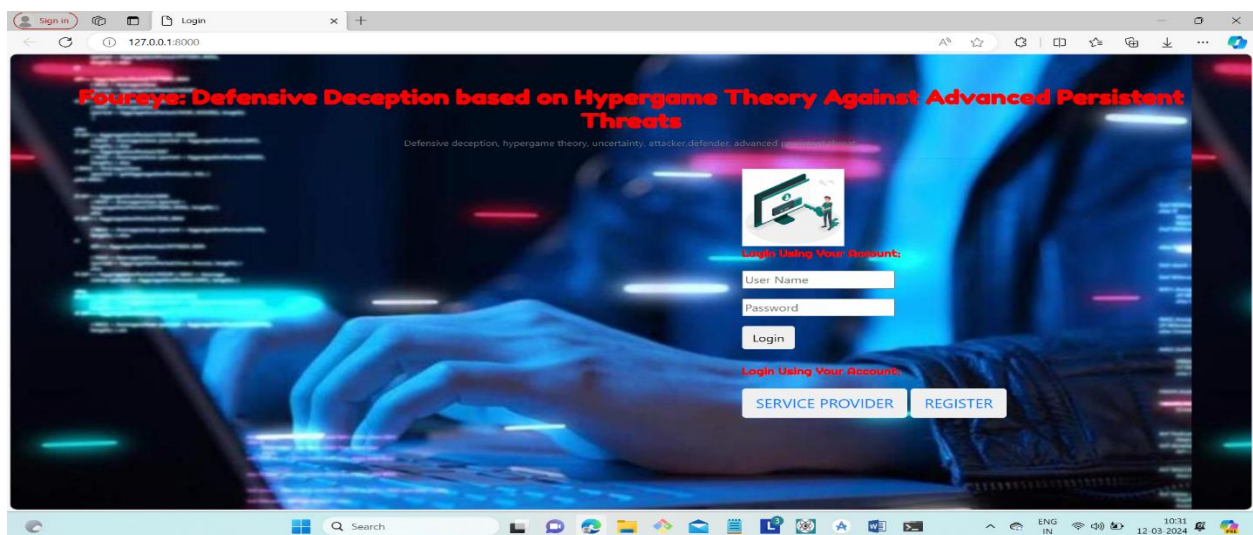


Fig : 10.1.2 User Login Page

10.1.3 View Your Profile Page

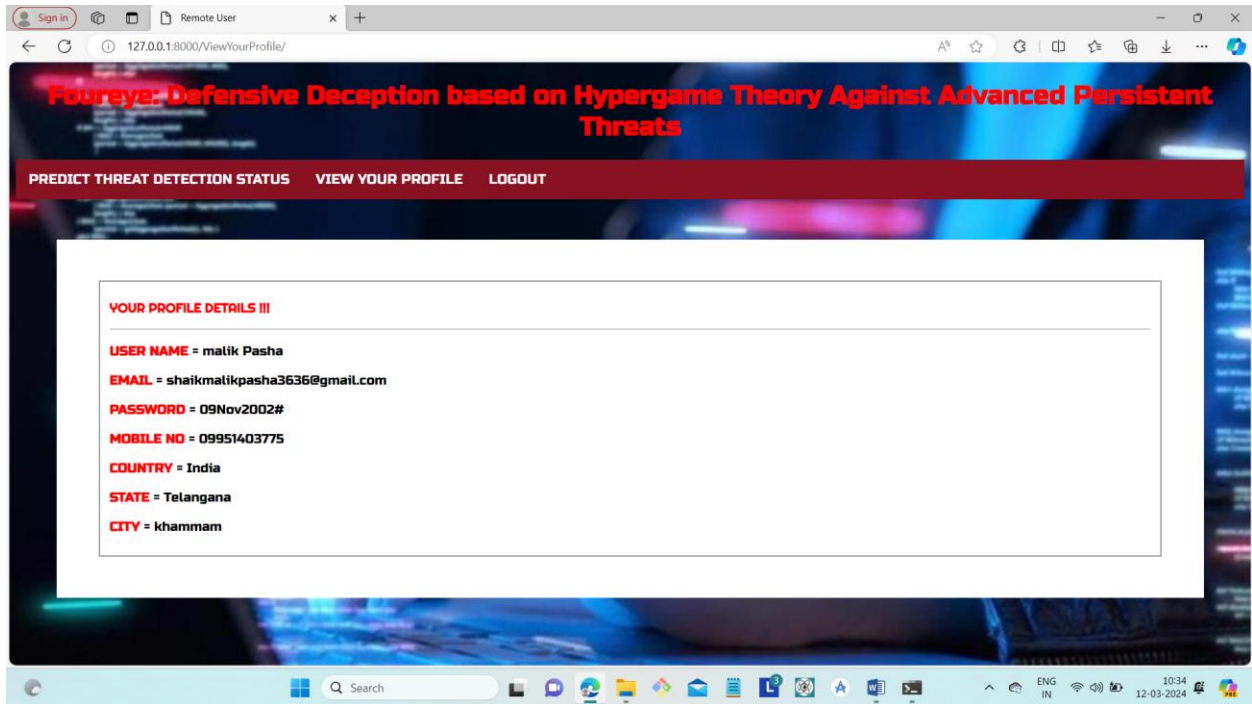


Fig : 10.1.3 View Your Profile Page

10.1.4 Predict Threat Detection Status Page

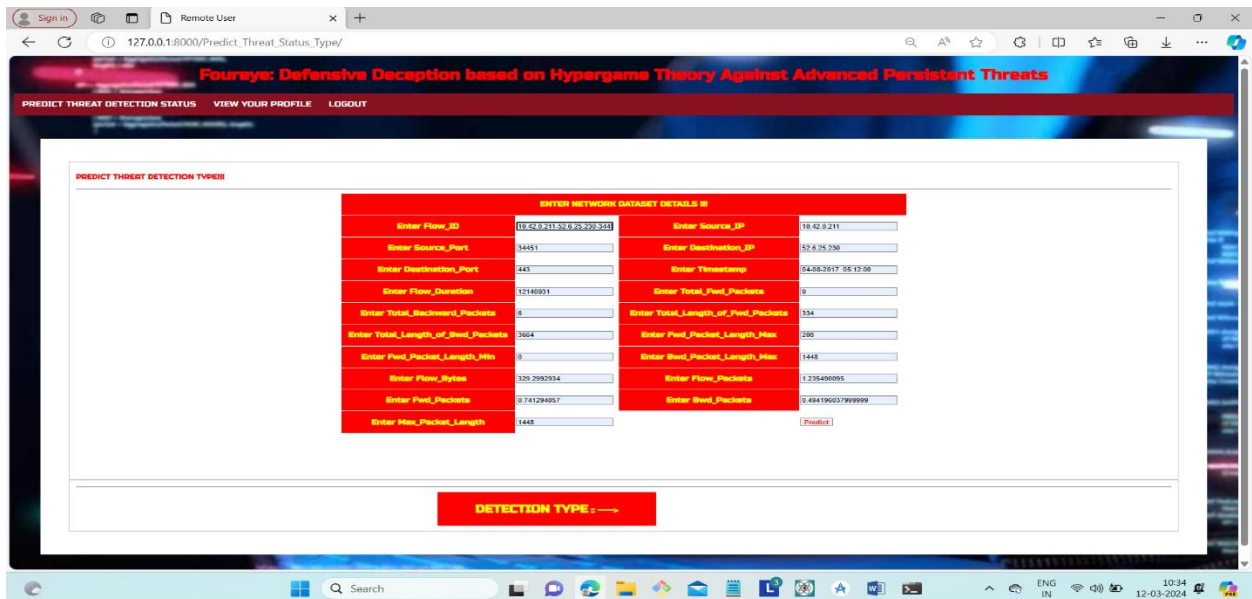


Fig : 10.1.4 Predict Threat Detection Status Page

Foureye: Defensive Deception based on Hypergame Theory Against Advanced Persistent Threats

PREDICT THREAT DETECTION STATUS VIEW YOUR PROFILE LOGOUT

PREDICT THREAT DETECTION TYPE!!!

ENTER NETWORK DATASET DETAILS IN

Enter Flow_ID	Enter Source_IP
Enter Source_Port	Enter Destination_IP
Enter Destination_Port	Enter Timestamp
Enter Flow_Duration	Enter Total_Fwd_Packets
Enter Total_Backward_Packets	Enter Total_Length_of_Fwd_P
Enter Total_Length_of_Fwd_Packets	Enter Fwd_Packet_Length_Max
Enter Fwd_Packet_Length_Min	Enter Fwd_Packet_Length_M
Enter Flow_Bytes	Enter Fwd_Packet_Length_Max
Enter Fwd_Packets	Enter Flow_Packets
Enter Max_Packet_Length	Enter Fwd_Packets

Predict

DETECTION TYPE Design

Fig: 10.1.4 Predict Threat Detection Status Page

10.2 SERVICE PROVIDER

10.2.1 Service Provider Login Page

Foureye: Defensive Deception based on Hypergame Theory Against Advanced Persistent Threats

Defensive deception, hypergame theory, uncertainty, attacker/defender, advanced persistent threats

Login Service Provider:

User Name

Password

Login

User Login

Fig : 10.2.1Service Provider Login Page

10.2.2 Browse Network Dataset and Train &Test Page

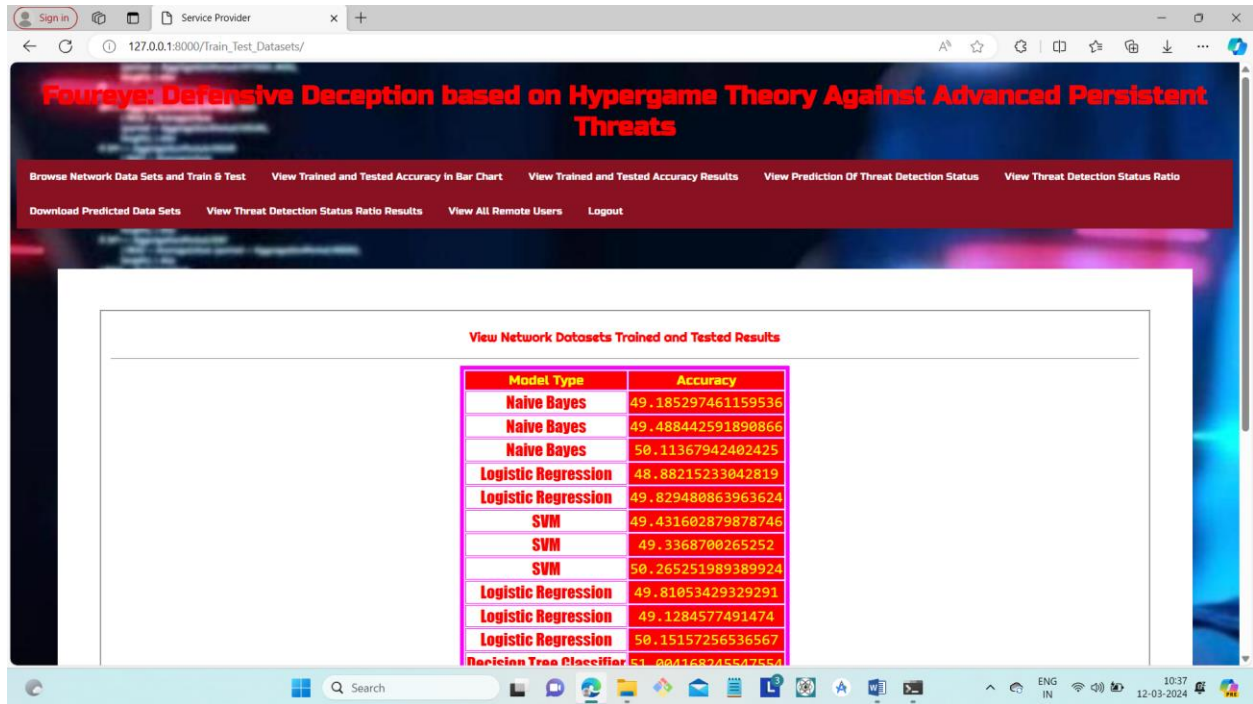


Fig : 10.2.2 Browse Network Dataset and Train &Test Page



Fig : 10.2.2 Browse Network Dataset and Train &Test Page

10.2.3 View Trained and Tested Accuracy in Bar Chart Page

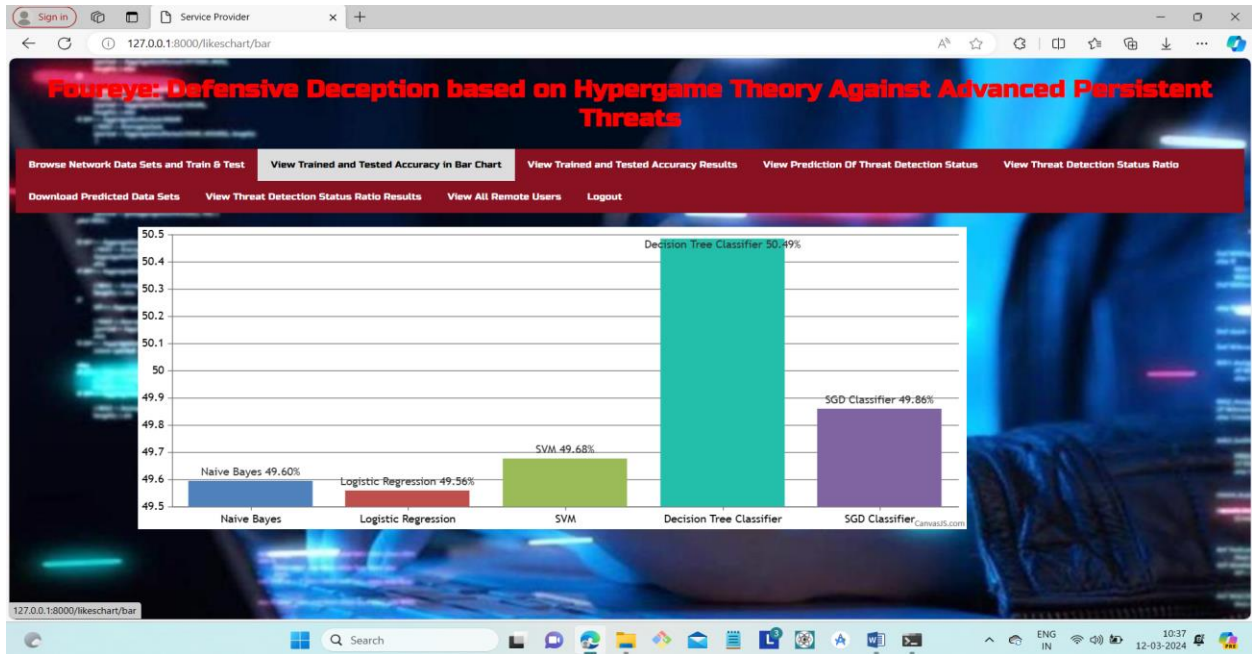


Fig : 10.2.3 View Trained and Tested Accuracy in Bar Chart Page

10.2.4 View trained and Tested Accuracy Results

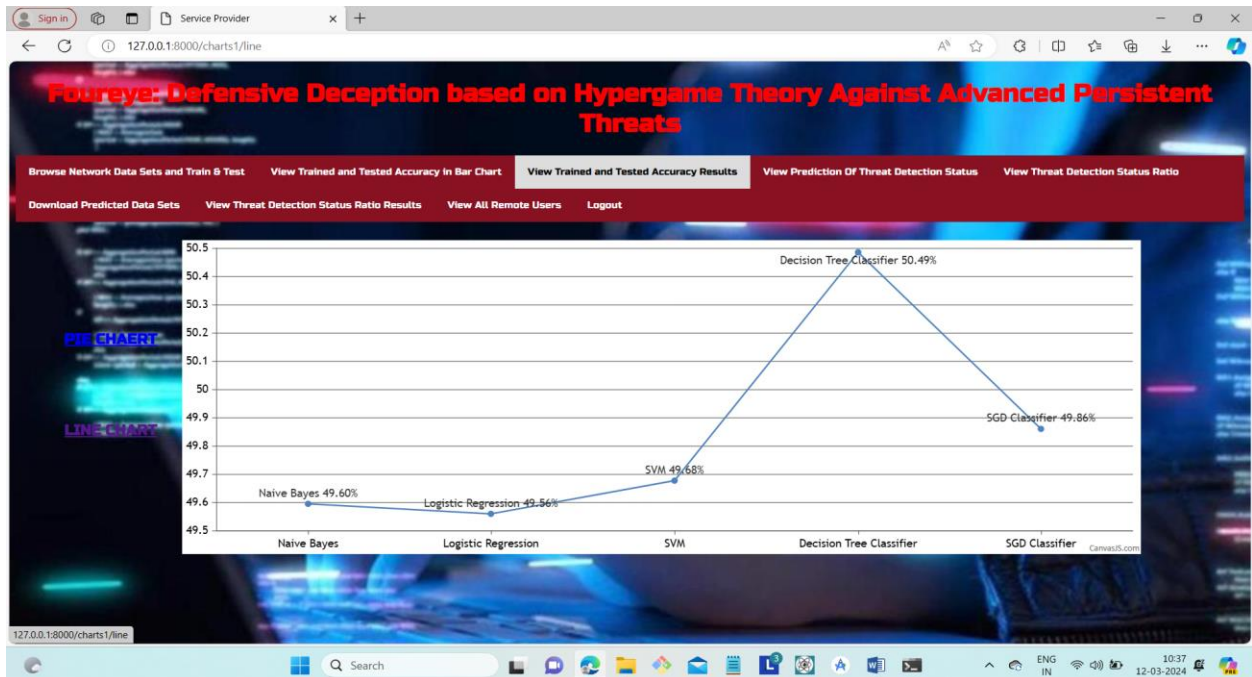


Fig : 10.2.4 View trained and Tested Accuracy Results

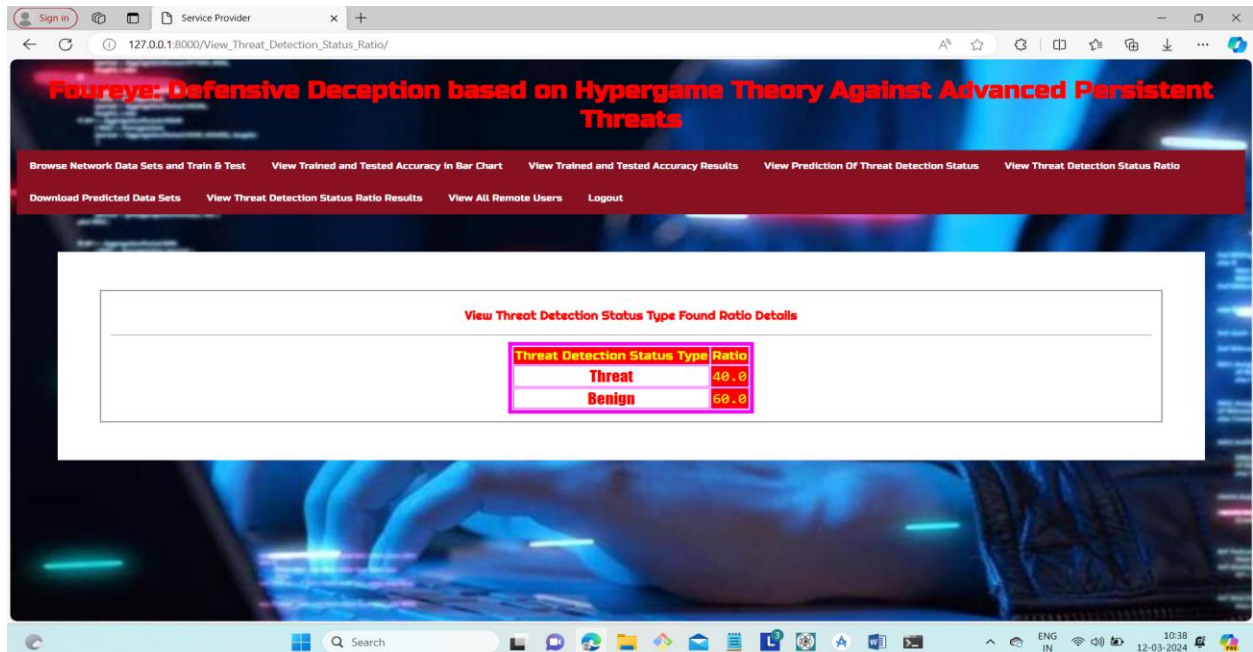
10.2.5 View Prediction of Threat Detection Status



Flow_ID	Source_IP	Source_Port	Destination_IP	Destination_Port	Timestamp	Flow_Duration	Total_Fwd_Packets	Total_Back_Packets
10.42.0.42-52.179.189.71-54061-443-6	10.42.0.42	54061	52.179.189.71	443	04-08-17 8:15	49629	1	
172.217.12.162-10.42.0.151-443-41454-6	172.217.12.162	443	10.42.0.151	41454	05-08-17 6:28	2080	2	
172.217.3.106-10.42.0.211-443-41106-6	10.42.0.211	41106	172.217.3.106	443	11-07-17 10:57	51597803	6	
173.194.175.188-10.42.0.151-5228-34789-6	10.42.0.151	34789	173.194.175.188	5228	04-08-17 10:44	128500	2	
172.217.12.161-10.42.0.151-443-54018-6	172.217.12.161	443	10.42.0.151	54018	04-08-17 10:43	36	2	

Fig : 10.2.5 View Prediction of Threat Detection Status

10.2.6 View Threat Detection Status Ratio Results



Threat Detection Status Type	Ratio
Threat	48.6
Benign	50.6

Fig : 10.2.6 View Threat Detection Status Ratio Results

10.2.7 View All Remote Users

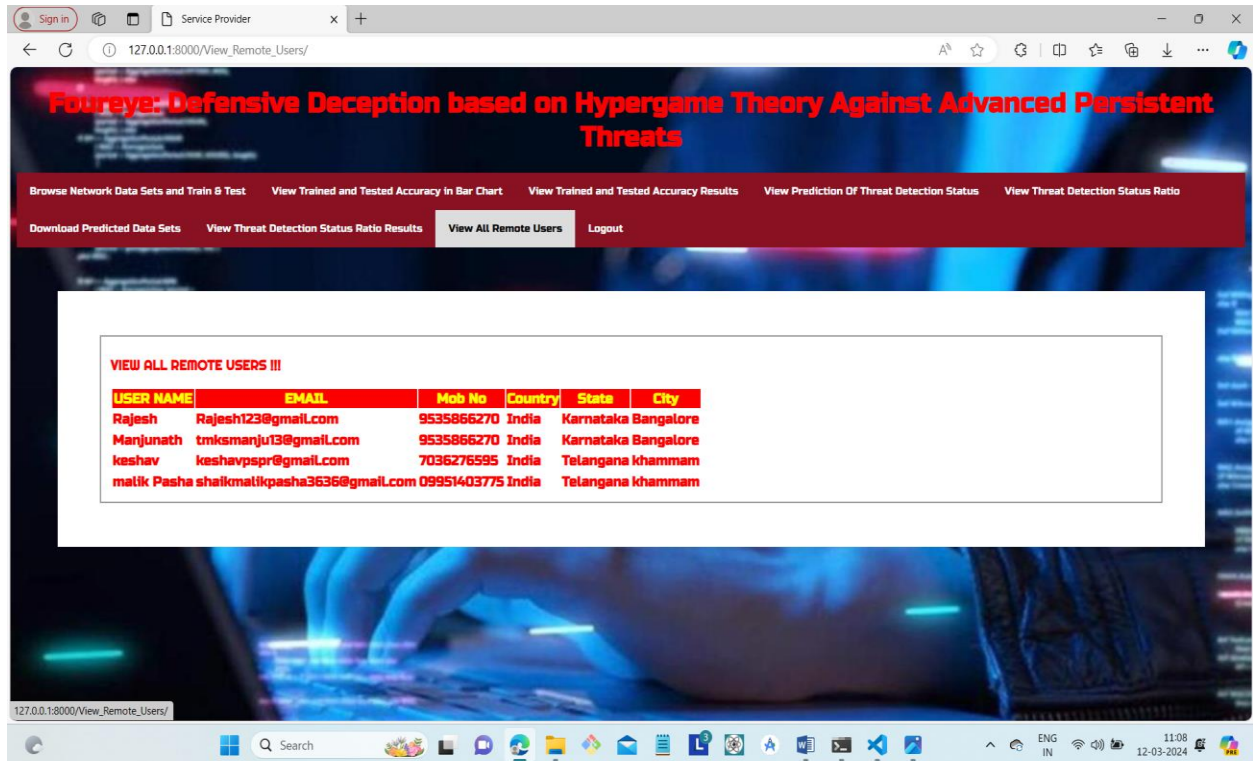


Fig : 10.2.7 View All Remote Users

11 CONCLUSION

An attacker’s and defender’s perceived uncertainty can be reduced when defensive deception (DD) is used. This is because the attacker perceives more knowledge about the system as it performs attacks as an inside attacker. On the other hand, the defender’s uncertainty can be reduced by collecting more attack intelligence by using DD while allowing the attacker to be in the system. Attack cost and defense cost are two critical factors in determining HEUs (hyper game expected utilities). Therefore, high DHEU (defender’s HEU) is not necessarily related to high system performance in MTTSF (mean time to security failure) or TPR (true positive rate) which can also be a key indicator of system security. Therefore, using DD under imperfect information (IPI) yields the best performance in MTTSF (i.e., the longest system lifetime) while it gives the minimum DHEU among all schemes. DD can effectively increase TPR of the NIDS in the system based on the attack intelligence collected through the DD strategies. This work brings up some important directions for future research by: (1) considering multiple attackers arriving in a system simultaneously in order to consider more realistic scenarios; (2) estimating each player’s belief based on machine learning in order to more correctly predict a next move of its opponent; (3) dynamically adjusting a risk threshold, i.e., Eq. (6), depending on a system’s security state; (4) introducing a recovery mechanism to restore a compromised node to a healthy node allowing the recovery delay; (5) developing an intrusion response system that can reassess a detected intrusion in order to minimize false positives while identifying an optimal response strategy to deal with intrusions with high urgency; and (6) considering another intrusion prevention mechanism, such as moving target defense, as one of the defense strategies.

12 REFERENCES

- “Common vulnerability scoring system (CVSS).”[Online].Available: <https://www.first.org/cvss/>
- Y. M. Aljefri, M. A. Bashar, L. Fang, and k. W. Hipel, “First-level hypergame for investigating misperception in conflicts,” *IEEE Trans. Systems, Man, and Cybernetics:Systems*, vol. 48, no. 12, pp. 2158–2175, 2017.
- H. Almeshekah and H. Spafford, “Cyber security deception,” in *Cyber Deception*. Springer, 2016, pp. 25–52.
- C. Bakker, A. Bhattacharya, S. Chatterjee, and D. L.Vrabie, “Learning and information manipulation: Repeated hypergames for cyber-physical security,” *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 295–300, 2019.
- P. G. Bennett, “Toward a theory of hypergames,” *Omega*, vol. 5, no. 6, pp. 749–751, 1977.
- E. Bertino and N. Islam, “Botnets and Internet of Things security,” *Computer*, vol. 50, no. 2, pp. 76–79, Feb. 2017.
- M. Boussard, D. T. Bui, L. Ciavaglia, R. Douville, M. L.Pallec, N. L. Sauze, L. Noirie, S. Papillon, P. Peloso, and F. Santoro, “Software-defined LANs for interconnected smart environment,” in *2015 27th Int’l Teletraffic Congress*, Sep. 2015, pp. 219–227.
- U.Brandes, “A faster algorithm for betweenness centrality,” *Jour. mathematical sociology*, vol. 25, no. 2, pp.163–177, 2001.
- J.W.Caddell, “Deception 101-primer on deception,” *DTIC Document*, Tech. Rep., 2004.
- T.E.Carroll and D.Grosu, “A game theoretic investigation of deception in network security,” *Security and Communication Networks*, vol. 4, no. 10, pp. 1162–1172, 2011.
- W. Casey, A. Kellner, P. Memarmoshrefi, J.A.Morales, and B. Mishra, “Deception, identity, and security: Thegame theory of Sybil attacks,” *Comms. of the ACM*, vol. 62, no. 1, pp. 85–93, 2018.
- J.-H. Cho, M. Zhu, and M. P. Singh, *Modeling and Analysis of Deception Games based on Hypergame Theory*. Cham, Switzerland: Springer Nature, 2019, ch. 4, pp. 49–74.
- [13] K. Ferguson-Walter, S. Fugate, J. Mauger, and M. Major, “Game theory for adaptive defensive cyber deception,” in *Proc. 6th Annual Symp. on Hot Topics in the Science of Security*. ACM, 2019, p. 4.

- N. M. Fraser and K. W. Hipel, Conflict Analysis: Models and Resolutions. North-Holland, 1984.
- N. Garg and D. Grosu, “Deception in honeynets: A game-theoretic analysis,” in Proc. IEEE Information Assurance and Security Workshop (IAW). IEEE, 2007, pp. 107–113.
- B. Gharesifard and J. Cortés, “Evolution of the perception about the opponent in hypergames,” in Proc. 49th IEEE Conf. Decision and Control (CDC), Dec. 2010, pp. 1076–1081.
- “Evolution of players’ misperceptions in hypergames under perfect observations,” IEEE Trans. Automatic Control, vol. 57, no. 7, pp. 1627–1640, Jul. 2012.
- GmbH. MindNode. [Online]. Available: <https://mindnode.com/>
- J. Han, J. Pei, and M. Kamber, Data Mining: Concepts and Techniques. Elsevier, 2011.
- J. T. House and G. Cybenko, “Hypergame theory applied to cyber attack and defense,” in Proc. SPIE Conf. Sensors, and Command, Control, Comms., and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IX, vol. 766604, May. 2010.
- T. Kanazawa, T. Ushio, and T. Yamasaki, “Replicator dynamics of evolutionary hypergames,” IEEE Trans. Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 37, no. 1, pp. 132–138, Jan. 2007.
- N. S. Kovach, A. S. Gibson, and G. B. Lamont, “Hypergame theory: A model for conflict, misperception, and deception,” Game Theory, 2015, article ID 570639, 20 pages.
- K. Krombholz, H. Hobel, M. Huber, and E. Weippl, “Advanced social engineering attacks,” Jour. Information Security and Applications, vol. 22, pp. 113–122, 2015.
- S. Kyung, W. Han, N. Tiwari, V. H. Dixit, L. Srinivas, Z. Zhao, A. Doupe, and G. Ahn, “Honeyproxy: Design and implementation of next-generation honeynet via SDN,” in 2017 IEEE Conf. Comms. and Network Security (CNS), Oct. 2017, pp. 1–9.
- O. Leiba, Y. Yitzchak, R. Bitton, A. Nadler, and A. Shabtai, “Incentivized delivery network of IoT software up-dates based on trustless proof-of distribution,” in 2018 IEEE European Symp. on Security and Privacy Workshops (EuroS PW), Apr. 2018, pp. 29–39.