A Project Report on

DUAL-PURPOSE HYPER-HEURISTIC SUPPORT VECTOR MACHINE DESIGNED FOR ADDRESSING CYBER SECURITY ISSUES IN BIG DATA ENVIRONMENTS

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

G.Poojitha

(20H51A0512)

CH.Sridham

(20H51A05K4)

Imtiyaz Ahmad Wani

(20H51A05Q1)

Under the esteemed guidance of

Ms. G. Srividya

(Assistant Professor)

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING& TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A$^+$ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2020- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major project report entitled " DUAL-PURPOSE HYPER-HEURISTIC SUPPORT VECTOR MACHINE DESIGNED FOR ADDRESSING CYBER SECURITY ISSUES IN BIG DATA ENVIRONMENTS " being submitted by Imtiyaz Ahmad Wani (20H51A05Q1) in partial fulfillment for the award of Bachelor of Technology in **Computer Science and** Engineering is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Ms. G. Srividya	Dr. Siva Skandha Sanagala	External Exmainer
Assistant Professor	Associate Professor and HOD	
Dept. of CSE	Dept. of CSE	

ACKNOWLEDGEMENT

Imtiyaz Ahmad Wani (20H51A05Q1)

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

Cyber security in the context of big data is known to be a critical problem and presents a great challenge to the research community. Machine learning algorithms have been suggested as candidates for handling big data security problems. Among these algorithms, support vector machines (SVMs) have achieved remarkable success on various classification problems. However, to establish an effective SVM, the user needs to define the proper SVM configuration in advance, which is a challenging task that requires expert knowledge and a large amount of manual effort for trial and error. In this paper, we formulate the SVM configuration process as a bi-objective optimization problem in which accuracy and model complexity are considered as two conflicting objectives. We propose a hyper-heuristic framework for bi-objective optimization that is independent of the problem domain. This is the first time that a hyper-heuristic has been developed for this problem. The proposed hyper-heuristic framework consists of a high-level strategy and low-level heuristics. The high-level strategy uses the search performance to control the selection of which low-level heuristic should be used. To address bi-objective optimization, the proposed framework adaptively integrates the strengths of decomposition- and Pareto-based approaches. The effectiveness of the proposed framework has been evaluated on two cyber security problems: Microsoft malware big data classification and anomaly intrusion detection. The obtained results demonstrate that the proposed framework is very effective, if not superior, compared with its counterparts and other algorithms.

B. Tech (CSE)

CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

The rapid growth of big data in the field of cyber-security has posed significant challenges in effectively detecting and preventing cyber threats. Support Vector Machines (SVM) have emerged as a popular technique for addressing cyber-security challenges. However, the traditional SVM approach often struggles to handle large-scale datasets efficiently, leading to reduced accuracy and increased computational time. Moreover, cyber-security professionals face the challenge of balancing two conflicting objectives: maximizing the detection rate of cyber threats and minimizing false positives. These objectives often require different parameter settings, making it challenging to find an optimal solution that satisfies both objectives simultaneously [7]. To address these issues, this project aims to develop a Dual-purposed Hyper-heuristic approach for Support Vector Machines in the context of big data cyber-security.

The project seeks to investigate and design an algorithm that can automatically select the most suitable classification for the new incoming data to optimize both the detection rate and minimize false positives. The project will involve exploring and developing hyper-heuristic algorithms that intelligently search through a large space of potential solutions [5][8]. The proposed approach will leverage the power of big data analytics to handle large-scale datasets efficiently, improving the accuracy of cyber threat detection while reducing computational time. By developing a dual-purposed hyper-heuristic approach, this project aims to provide cyber-security professionals with a tool that can effectively and efficiently address the challenges of big data cyber-security. The resulting approach will help improve the decision-making process by enabling security analysts to focus on genuine cyber threats while minimizing false positives, ultimately enhancing the overall cyber-security posture of organizations in the face of rapidly evolving cyber threats.

1.2 RESEARCH OBJECTIVE:

The primary objective of this project is to develop a novel dual-purposed Hyper-heuristic approach for Support Vector Machines (SVM) in the domain of big data cyber-security. The project aims to address the challenges associated with handling large-scale datasets efficiently while optimizing the detection rate of cyber threats and minimizing false positives simultaneously.

One objective is to investigate and design an algorithm that can automatically select the most suitable classification for new incoming test data. By leveraging hyper heuristics, the algorithm will intelligently search through a vast space of potential solutions [5][8]. This objective will contribute to enhancing the accuracy and speed of cyber threat detection by finding an optimal way to work on big data cyber-security datasets. Another objective is to develop an approach that can handle the computational demands of big data analytics efficiently. Traditional SVM approaches often struggle with scalability, leading to increased computational time and resource requirements [7]. By employing techniques specifically tailored for big data, such as distributed computing and parallel processing, the project aims to improve the efficiency of SVM-based cyber-security systems. This objective will ensure that the developed approach can handle the growing volume and complexity of cyber-security datasets without compromising performance.

In summary, the project's objectives include developing a dual-purposed hyper-heuristic approach for SVM that can automatically select optimal solution [5] for the problem in hand thus improving the scalability and efficiency of SVM-based cyber-security systems for big data, and effectively balancing the detection rate of cyber threats with the minimization of false positives. By accomplishing these objectives, the project aims to contribute to the advancement of cyber-security practices in the era of big data, ultimately enhancing the overall cyber-security posture of organizations and protecting against evolving cyber threats.

4

1.3 PROJECT SCOPE:

The scope of the project is to develop a dual-purposed Hyper-heuristic framework that can be used to detect cyber security threats in big data environments. The project involves developing a website that would take details regarding the issue that has occurred with the user's website. Based on the data provided our algorithm classifies and provides the user with the name of the virus that user's website been attacked with. Our project aims to overcome the limitations of the support vector machine and hence, creating a more robust algorithm. Our algorithm uses Hyper-heuristics to handle and classify data. Hyper-heuristics are used to increase the efficiency of detection and classification.

ADVANTAGES:

- Higher accuracy
- Faster result
- Can handle large volumes of data

CHAPTER 2
BACKGROUND WORK

CHAPTER 2
BACKGROUND WORK

2.1   SUPPORT VECTOR MACHINE:

2.1.1 INTRODUCTION:

In today's digitally interconnected world, the proliferation of cyber threats poses significant challenges to organizations and individuals alike. The exponential growth of digital data generated by various systems and networks has made cybersecurity a paramount concern. Within this landscape, the effective management and analysis of vast amounts of data play a pivotal role in identifying and mitigating potential security risks. In this context, Support Vector Machines (SVMs) have emerged as a powerful tool for classifying big data in cybersecurity, offering a robust framework for threat detection and prevention

SVMs are renowned for their ability to handle high-dimensional data and nonlinear relationships, making them well-suited for the complexities inherent in cybersecurity datasets. This capability stems from their underlying mathematical principles, which allow them to construct an optimal hyperplane that separates different classes in the feature space. By leveraging the kernel trick, SVMs can efficiently capture intricate patterns within the data, enabling accurate classification of security threats. Moreover, SVMs exhibit resilience to overfitting, making them particularly effective in scenarios where data samples may be limited or imbalanced.

One of the distinguishing features of SVMs is their resilience to overfitting, a common pitfall in machine learning models, particularly when dealing with limited or imbalanced data. SVMs achieve this by maximizing the margin between the decision boundary and the nearest data points, thereby promoting generalization to unseen data instances. This property makes SVMs particularly well-suited for cybersecurity applications, where robustness and reliability are paramount. Furthermore, SVMs offer versatility in handling various types of data, including structured and unstructured data formats commonly encountered in cybersecurity domains. Whether analyzing network traffic logs, system event records, or textual data from security incident reports, SVMs can effectively discern patterns and anomalies indicative of potential security breaches as cybersecurity threats continue to evolve in sophistication and scale, the role of SVMs in fortifying defence

mechanisms becomes increasingly indispensable. By harnessing the analytical power of SVMs, cybersecurity professionals can enhance their capabilities in threat detection, thereby safeguarding critical assets and information in the digital realm.[2]

2.1.2 MERITS, DEMERITS AND CHALLENGES:
MERITS:

- Nonlinear Classification: SVMs excel at classifying data with complex, nonlinear relationships, making them well-suited for capturing intricate patterns in cybersecurity datasets. By utilizing kernel functions, SVMs can effectively model nonlinear decision boundaries, enhancing their accuracy in threat detection.

- Robustness to Overfitting: SVMs exhibit resilience to overfitting, particularly in scenarios with limited or imbalanced data. By maximizing the margin between classes, SVMs prioritize generalization to unseen data instances, mitigating the risk of overfitting and improving performance in real-world cybersecurity applications.

- Effective with Small Sample Sizes: In cybersecurity, datasets may be constrained due to privacy concerns or data availability. SVMs perform well even with small sample sizes, making them suitable for analyzing cybersecurity data where obtaining large labeled datasets can be challenging.

- Binary Classification: SVMs are inherently binary classifiers, making them suitable for addressing many cybersecurity tasks, such as malware detection, intrusion detection, and spam filtering, where the focus is on distinguishing between benign and malicious entities or activities.

DEMERITS:

- Computational Complexity of Kernel Selection: The choice of kernel function in SVMs significantly impacts model performance. However, selecting the appropriate kernel and tuning its parameters can be computationally intensive and require domain expertise. Moreover, an improper choice of kernel may lead to suboptimal classification results, posing a challenge in cybersecurity applications.

- Sensitivity to Outliers: SVMs are sensitive to outliers in the dataset, which can skew the decision boundary and affect classification accuracy. In cybersecurity, where outliers may indicate anomalous or malicious behaviour, the sensitivity of SVMs to outliers can impact the effectiveness of threat detection and classification.

- Limited Scalability to Big Data: SVMs may face challenges in scaling to handle massive volumes of data typical in cybersecurity applications. The computational complexity of SVM algorithms,

coupled with memory constraints, can impede their scalability, hindering their deployment in large-scale cybersecurity environments.

- Need for Expertise in Model Tuning: Achieving optimal performance with SVMs requires expertise in selecting appropriate hyperparameters, including the choice of kernel function and regularization parameters. This reliance on expert knowledge may limit the accessibility of SVMs to non-experts in cybersecurity, potentially hindering their adoption in practical applications.[7]

CHALLENGES:

- Data Preprocessing Complexity: Preparing cybersecurity data for SVM classification often involves extensive preprocessing steps such as feature extraction, normalization, and dealing with missing values. The high dimensionality and complexity of cybersecurity datasets increase the computational and labour- intensive nature of these preprocessing tasks

- Adversarial Attacks: SVMs, like other machine learning models, are susceptible to adversarial attacks where malicious actors manipulate input data to deceive the classifier. Adversarial attacks can undermine the reliability and effectiveness of SVM-based cybersecurity systems, highlighting the need for robust defence mechanisms.

- Imbalanced Data Handling: Class imbalance is common in cybersecurity datasets, where the number of instances belonging to different classes varies significantly. SVMs may struggle to effectively classify imbalanced data, leading to biased models and reduced performance in detecting minority class threats.

2.1.3 IMPLEMENTATION:
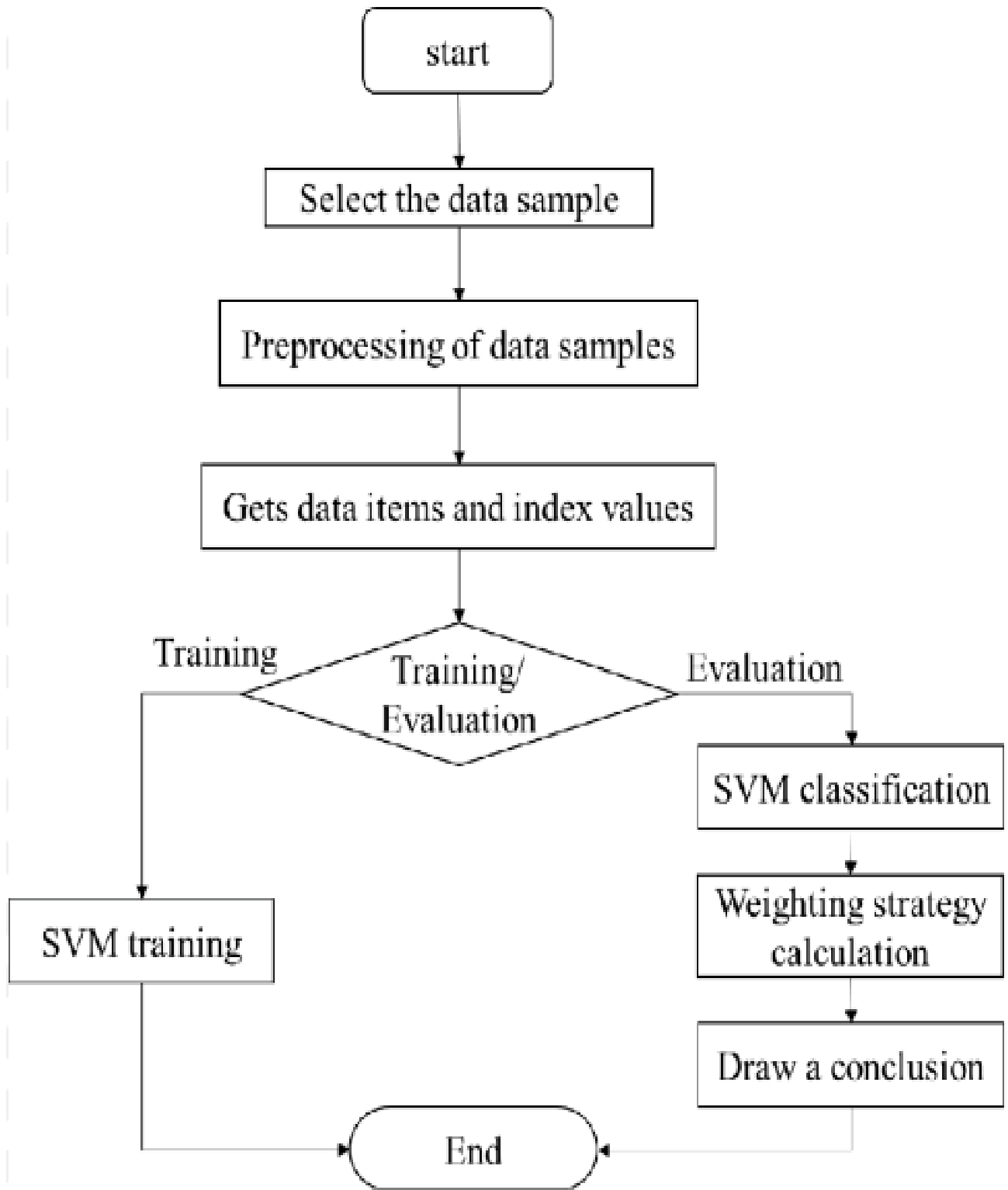
Fig 2.1: Support Vector Machine Working

9

2.2 RANDOM FOREST ALGORITHM:

2.2.1 INTRODUCTION:

In today's interconnected digital landscape, the exponential growth of digital data has ushered in unprecedented opportunities and challenges, particularly in the realm of cybersecurity. The proliferation of cyber threats poses significant risks to organizations, governments, and individuals alike, necessitating robust mechanisms for threat detection and prevention. In this dynamic and evolving landscape, the effective analysis and classification of vast amounts of data play a pivotal role in identifying, mitigating, and responding to cybersecurity threats. In response to these challenges, advanced machine learning techniques, such as Random Forest, have emerged as powerful tools for classifying big data in cybersecurity, offering a promising avenue for enhancing cybersecurity defences.

Random Forest, a versatile ensemble learning algorithm, holds immense potential for addressing the complexities inherent in cybersecurity datasets. By harnessing the collective intelligence of multiple decision trees, Random Forest offers a robust framework for classification tasks, including the detection of malware, intrusion attempts, and anomalous behaviour within network traffic and system logs. The ensemble nature of Random Forest mitigates the limitations of individual decision trees, such as overfitting and high variance, by aggregating predictions and averaging outputs, thereby enhancing classification accuracy and generalization to unseen data. Random Forest is an ensemble learning method that builds multiple decision trees during training and combines their predictions to make a final classification

The application of Random Forest in cybersecurity is further bolstered by its ability to handle high-dimensional and heterogeneous data sources commonly encountered in cybersecurity environments. Random Forest can accommodate diverse data types and extract valuable insights for threat detection and response. Moreover, Random Forest provides a measure of feature importance, enabling cybersecurity analysts to identify the most discriminative features and prioritize security measures accordingly.

The versatility and effectiveness of Random Forest in classifying big data in cybersecurity have made it a popular choice among cybersecurity professionals and researchers. From detecting

malware and intrusions to identifying suspicious network activities, Random Forest offers a robust framework for enhancing cybersecurity defences in an increasingly interconnected world.[1]

MERITS:

- Ensemble Learning: Random Forest leverages the concept of ensemble learning, where multiple decision trees are trained on different subsets of the data and combined to make predictions. This ensemble approach enhances classification accuracy and generalization by mitigating the risk of overfitting and reducing variance.

- Robustness to Overfitting: Random Forest is inherently less prone to overfitting compared to individual decision trees. By aggregating predictions from multiple trees and averaging their outputs, Random Forest reduces the likelihood of capturing noise or irrelevant features in the data, leading to more robust and reliable classification results.

- Feature Importance: Random Forest provides a measure of feature importance, allowing cybersecurity analysts to identify the most discriminative features for threat detection. This insight into feature importance can aid in understanding the underlying patterns of cyber threats and prioritizing security measures accordingly.

- Handling of High-Dimensional Data: Random Forest can effectively handle high-dimensional data typical in cybersecurity applications. It can accommodate a large number of features without the need for dimensionality reduction techniques, making it well-suited for analyzing, diverse data sources such as network logs and system events.

- Parallelization and Scalability: Random Forest algorithms can be parallelized and scaled across multiple processors or clusters, enabling efficient processing of big data in cybersecurity environments. This scalability feature facilitates real-time or near-real-time threat detection and response, enhancing the agility of cybersecurity operations.

- Resilience to Noisy Data: Random Forests are generally robust to noisy data. Since they aggregate predictions from multiple decision trees, they are less sensitive to individual noisy data points or outliers. By averaging the predictions of many trees, Random Forests can effectively filter out the effects of noise, leading to more stable and reliable classification results. This resilience to noisy data makes Random Forests well-suited for handling real-world cybersecurity datasets that may contain imperfect or incomplete information.

DEMERITS:

- Black Box Nature: Despite its high classification accuracy, Random Forest is often considered a "black box" model, providing limited interpretability of the underlying decision-making process. Understanding how individual trees contribute to the ensemble prediction can be challenging, particularly in cybersecurity applications where interpretability is crucial for trust and accountability.

- Computational Resource Requirements: Training a Random Forest model can be computationally intensive, especially when dealing with large-scale or high-dimensional data. The ensemble nature of Random Forest, which involves training multiple decision trees, increases the computational resource requirements, including memory and processing power.

- Model Complexity and Interpretability Trade-off: As the number of trees in a Random Forest increases the model's complexity also increases, making it more challenging to interpret. While Random Forests provide insights into feature importance, interpreting the collective decision-making process of multiple trees can be complex, especially for large ensembles. Balancing model complexity and interpretability is a trade-off in Random Forests, and in some cases, the increased complexity may hinder the explainability of the model, limiting its utility in scenarios where interpretability is crucial, such as cybersecurity audits or regulatory compliance.[1]

.**CHALLENGES :**

- Imbalanced Data Handling: Class imbalance is prevalent in cybersecurity datasets, where the number of instances belonging to different classes may vary significantly. Random Forest may struggle to effectively classify imbalanced data, leading to biased models and reduced performance in detecting minority class threats.

- Model Tuning and Optimization: Optimizing Random Forest hyperparameters, such as the number of trees, tree depth, and feature subsampling rate, can be challenging and require extensive experimentation. Finding the right balance between model complexity and generalization performance is crucial for achieving optimal classification results in cybersecurity applications.

- Adversarial Attacks: Random Forest, like other machine learning models, is susceptible to adversarial attacks where malicious actors manipulate input data to deceive the classifier. Adversarial attacks can undermine the reliability and effectiveness of Random Forest-based cybersecurity systems, highlighting the need for robust defence mechanisms. [1]

IMPLEMENTATION:
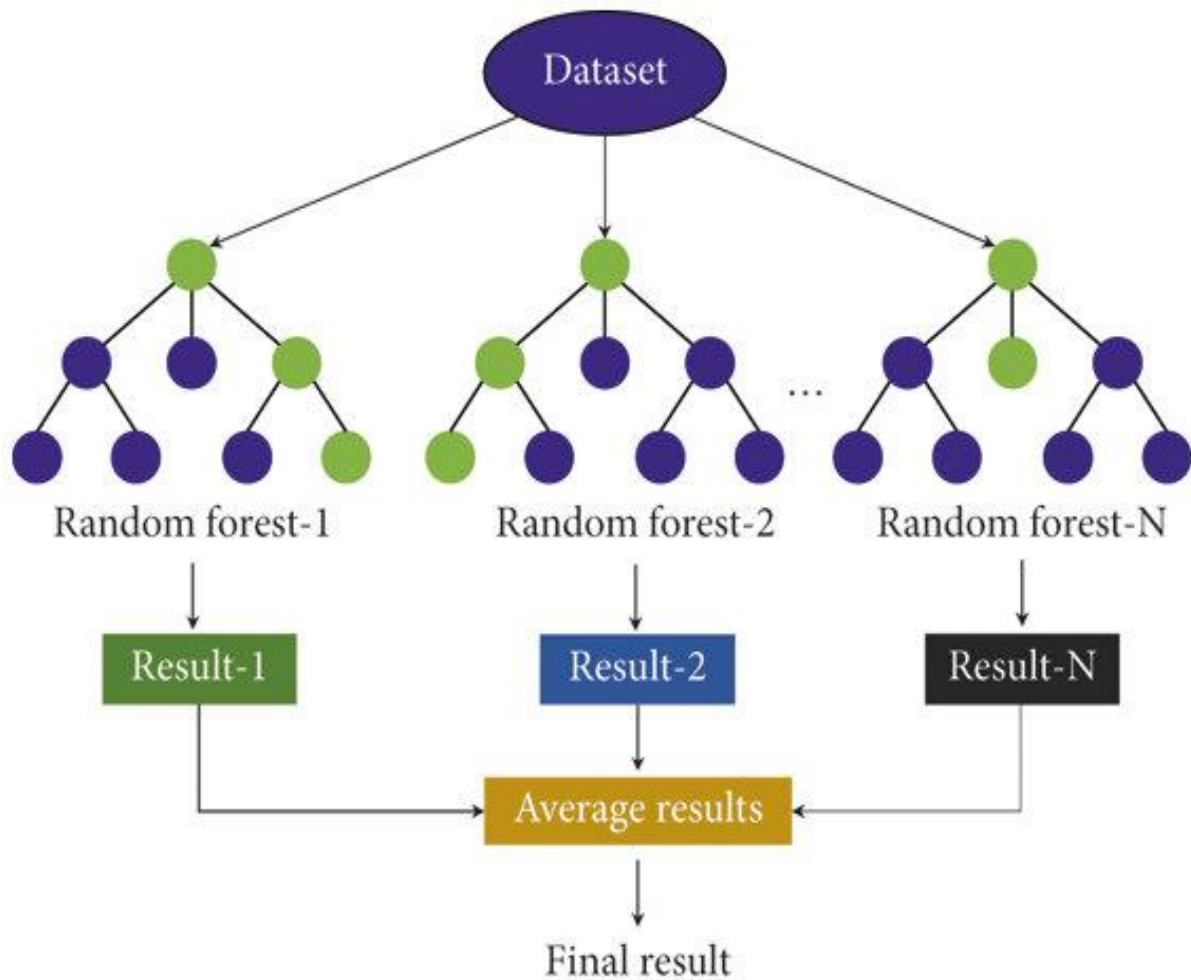
Fig 2.2: Random Forest

2.3 DECISION TREE ALGORITHM:

2.3.1 INTRODUCTION:

In the rapidly evolving landscape of cybersecurity, the exponential growth of digital data has brought about unparalleled challenges and opportunities. With each passing day, organizations and individuals are confronted with a deluge of data generated from diverse sources such as network traffic logs, system event records, and user activity logs. Amidst this torrent of information lies the key to identifying and mitigating emerging security threats, making the effective

13

analysis and classification of big data a critical imperative. In this context, machine learning algorithms have emerged as indispensable tools for cybersecurity professionals, offering powerful techniques for extracting actionable insights from vast and complex datasets.

At the forefront of these machine learning techniques is the venerable Decision Tree algorithm, which has garnered widespread acclaim for its transparency, interpretability, and versatility. Decision Trees offer an intuitive framework for classifying data by recursively partitioning the feature space into hierarchical decision nodes based on simple if-else conditions. This hierarchical structure not only provides a clear and interpretable representation of the decision-making process but also facilitates the identification of key features driving classification outcomes. By examining the structure of the tree and the importance scores assigned to each feature, cybersecurity analysts can identify the most discriminative features that contribute to the classification decisions. In the realm of cybersecurity, where transparency and explainability are paramount, Decision Trees offer a compelling solution for understanding and mitigating security threats.

One of the primary advantages of Decision Trees lies in their ability to handle nonlinear relationships and interactions within the data. Unlike traditional linear models, Decision Trees can capture complex patterns and dependencies, making them well-suited for analyzing diverse and heterogeneous cybersecurity datasets. Whether it's identifying anomalous network behaviour, detecting malicious software, or classifying security incidents, Decision Trees offer a flexible and adaptable approach for addressing a wide range of cybersecurity challenges.

Moreover, Decision Trees exhibit scalability to big data, enabling efficient processing of large-scale datasets commonly encountered in cybersecurity environments. Recent advancements in parallel and distributed computing techniques have further bolstered the scalability of Decision Tree algorithms, enabling real-time or near-real-time threat detection and response.[3]

2.3.2 MERITS, DEMERITS AND CHALLENGES:
MERITS:

- Interpretability: Decision Trees provide a transparent and interpretable representation of the decision-making process, making it easier for cybersecurity analysts to understand and explain the

classification rules. This interpretability is crucial for gaining insights into cybersecurity threats and devising effective countermeasures

- Handling of Nonlinear Relationships: Decision Trees can capture complex nonlinear relationships within the data, enabling them to effectively classify cybersecurity data with intricate patterns and interactions. This flexibility makes Decision Trees well-suited for analyzing diverse and heterogeneous cybersecurity datasets.

- Scalability to Big Data: Decision Trees can handle large-scale datasets with ease, making them suitable for analyzing the vast amounts of data generated in cybersecurity environments. Moreover, advancements in parallel and distributed computing techniques have further enhanced the scalability of Decision Tree algorithms, enabling real-time or near-real-time threat detection and response.

- Feature Importance Analysis: Decision Trees provide a measure of feature importance, allowing cybersecurity analysts to identify the most discriminative features for threat detection. This insight into feature importance aids in prioritizing security measures and focusing resources on mitigating the most significant cybersecurity risks.

DEMERITS:

- Overfitting: Decision Trees are prone to overfitting, particularly when the tree depth is not appropriately constrained or when the dataset contains noise or irrelevant features. Overfitting can lead to overly complex trees that perform well on the training data but generalize poorly to unseen data, reducing the effectiveness of the classifier in real-world cybersecurity scenarios.

- High Variance: Decision Trees can be sensitive to small changes in the training data, leading to high variance in the model's predictions. This sensitivity can result in instability and inconsistency in the classification results, particularly when dealing with noisy or imbalanced cybersecurity datasets.

- Limited Handling of Nonlinear Relationships: While Decision Trees can capture complex nonlinear relationships within the data, they may struggle to model highly nonlinear relationships effectively. In cybersecurity, where data may exhibit intricate and nonlinear patterns, Decision Trees alone may not be sufficient to capture all nuances of the data.

- Model Complexity Management: Managing the complexity of Decision Trees, particularly in deep or wide trees, can be challenging. Complex trees are prone to overfitting and can become difficult to interpret, limiting their usability and effectiveness in cybersecurity applications.

CHALLENGES:

- Imbalanced Data Handling: Class imbalance is common in cybersecurity datasets, where the number of instances belonging to different classes may vary significantly. Decision Trees may struggle to effectively classify imbalanced data, leading to biased models and reduced performance in detecting minority class threats.

- Model Complexity Management: Managing the complexity of Decision Trees, particularly in deep or wide trees, can be challenging. Complex trees are prone to overfitting and can become difficult to interpret, limiting their usability and effectiveness in cybersecurity applications.

- Ensemble Learning Integration: While ensemble methods like Random Forests and Gradient Boosting Machines (GBMs) can enhance the performance of Decision Trees, integrating these techniques into cybersecurity workflows requires careful consideration of computational resources, model interpretability, and scalability.[3]

2.3.3 IMPLEMENTATION:
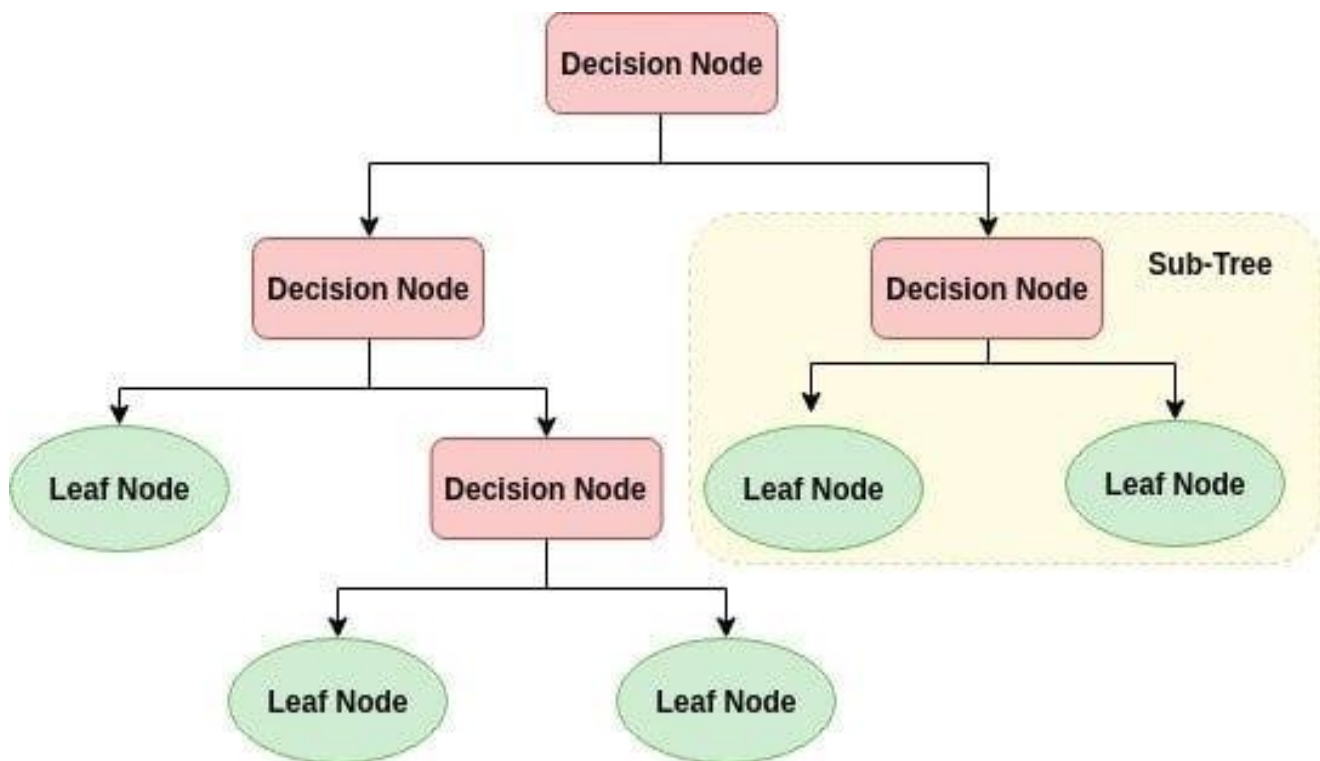
Fig 2.3: Decision tree

16

CHAPTER 3
PROPOSED SYSTEM

CHAPTER 3

PROPOSED SYSTEM

3.1 OBJECTIVE OF PROPOSED MODEL

The main objective of this project is to develop a framework that can be used as a malware detection tool in cybersecurity realm. With the increase in the amount of data in today's world analyzing it and understanding it becomes crucial identify key insights. These insights can be used in various ways in realm of cyber security, understanding loop holes etc. To analyze and classify this big data there are many machine algorithms but each has its own merits and demerits. SVM was one of the most frequently used among all the algorithms [2]. We propose a new framework that tries to do same work as svm but also covers all of its drawbacks.

Our proposed framework Can analyze big data related to cybersecurity to analyze key insights from it. our proposed framework can handle huge amounts of data.it can also give accurate and speedy result.

Our project mainly focusses on categorizing the problem that a client's website is going through under a relevant virus name. if client knows the name of the virus his/her website has attacked with it can be better delt with our algorithm compares input given with training data to tell the client the name of virus associated with that problem.

3.2 ALGORITHMS

3.2.1 Support Vector Machines (Svm)

Support Vector Machines (SVM) is a widely used machine learning algorithm that forms the core of the proposed project. SVM is a binary classification algorithm that aims to find an optimal decision boundary that separates data points belonging to different classes [2]. Here is a detailed explanation of the SVM algorithm:

1. **Data Pre-processing:**

The SVM algorithm begins with pre-processing the input data. This may involve tasks such as data cleaning, normalization, feature scaling, and handling missing values. The data is then split into training and testing datasets.

2. **Feature Selection:**

In the context of big data cyber-security, selecting relevant features is crucial. Feature selection techniques, such as information gain, chi-square, or recursive feature elimination, can be applied to

identify the most informative features that contribute to the classification task.

3. **Kernel Selection:**

SVM allows for the use of different kernel functions to transform the input data into a higher-dimensional feature space. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.[2]

The choice of kernel function depends on the characteristics of the data and the desired decision boundary.

4. **Training:**

SVM aims to find an optimal hyperplane that maximizes the margin between the support vectors, which are the data points closest to the decision boundary. The training phase involves solving a quadratic optimization problem to find the optimal hyperplane parameters (weights and bias) that minimize the classification error.

5. **Hyper parameter Tuning:**

SVM has several hyper parameters that need to be tuned to achieve optimal performance. These include the regularization parameter (C), kernel-specific parameters [2](such as the degree for polynomial kernels or the gamma parameter for RBF kernels), and the choice of kernel function itself. Techniques such as cross-validation or grid search can be used to find the best hyper parameter values.

6. **Classification:**

Once the SVM model is trained, it can be used for classification of new, unseen data points. The algorithm computes the decision function using the learned hyperplane parameters and assigns class labels based on the sign of the decision function output. The classification output can be binary (e.g., malicious or benign) or multi-class, depending on the specific cyber-security scenario.

7. **Evaluation:**

The performance of the SVM model is assessed using various evaluation metrics, such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic (ROC) curve. These metrics provide insights into the model's ability to correctly classify cyber security instances and its overall effectiveness in threat detection.

3.2.2 Bi-Objective Optimization

To handle the conflicting objectives of maximizing the detection rate of cyber threats while minimizing false positives, a bi-objective optimization algorithm is used. Bi-objective optimization aims to find a set of solutions that represent the trade-offs between the two objectives. Here is an overview of the bi-objective optimization algorithm:

1. **Objective Definition:**

   The two objectives in the project are maximizing the detection rate of cyber threats and minimizing false positives. Each objective is quantified using appropriate performance metrics, such as true positive rate (detection rate) and false positive rate.

2. **Pareto Dominance:**

   The bi-objective optimization algorithm utilizes the concept of Pareto dominance to identify the non-dominated solutions. A solution is considered dominant if it is better than or equal to another solution in both objectives and strictly better in at least one objective. Dominated solutions are eliminated from consideration.

3. **Population Initialization:**

   The algorithm initializes a population of candidate solutions, typically using random or heuristic-based methods[5]. Each solution represents a specific combination of hyper parameters, feature selection techniques, or other system configuration settings.

4. **Fitness Evaluation:**

   The fitness of each solution is evaluated by measuring its performance in terms of the two objectives. The solutions are ranked based on their Pareto dominance status, with non-dominated solutions forming the Pareto front.

5. **Selection and Reproduction:**

   The algorithm selects a subset of solutions from the Pareto front, typically using selection methods such as tournament selection or fitness-based selection. The selected solutions are then used to create new candidate solutions through reproduction techniques such as crossover and mutation.

6. **Iterative Improvement:**

   The bi-objective optimization algorithm iteratively improves the population by repeating the selection, reproduction, and evaluation steps. This iterative process explores the solution space, gradually converging towards a set of non-dominated solutions that represent the optimal trade offs between the detection rate and false positives.

7. **Decision Support:**

   The final set of non-dominated solutions provides decision support to cyber-security professionals. They can choose a solution that aligns with their specific requirements, striking the desired balance between detection rate and false positives.

3.2.3 Hyper-Heuristics

Hyper-heuristics are employed to automate the selection of optimal hyper parameters and feature

selection techniques for SVM. Hyper-heuristics are search algorithms that intelligently explore the solution space, evaluating different combinations and determining the best configuration for optimal performance [5][8]. Here is an overview of the hyper-heuristic's algorithm:

1. **Initialization:**

The hyper-heuristics algorithm initializes a population of candidate configurations by randomly selecting hyper parameters and feature selection techniques. Each configuration represents a potential combination to be evaluated.

2. **Evaluation:**

Each candidate configuration is evaluated by training and testing the SVM model using the selected hyper parameters and feature selection techniques. The performance of the configuration is measured using appropriate metrics, such as accuracy, precision, or F1-score.

3. **Selection and Variation:**

Based on the performance evaluation, the algorithm selects promising candidate configurations for further exploration. Variation operators, such as mutation or recombination, are applied to these selected configurations to create new candidate configurations[8]. This introduces diversity and allows the algorithm to explore different regions of the solution space.

4. **Fitness-based Decision Making:**

The algorithm employs fitness-based decision-making strategies to determine which candidate configurations to retain and which to discard. This decision-making process is guided by the performance of the configurations and may involve ranking, tournament selection, or probabilistic selection methods.

5. **Iterative Improvement:**

The hyper-heuristics algorithm iteratively improves the population of candidate configurations by repeating the evaluation, selection, variation, and decision-making steps. Through successive iterations, the algorithm converges towards a set of configurations that exhibit superior performance in terms of both detection rate and false positives.

6. **Termination Criterion:**

The algorithm terminates when a predefined stopping criterion is met, such as reaching a maximum number of iterations or achieving a satisfactory level of performance. The best performing configuration(s) are then selected as the optimal solution(s) for the SVM model. The combination of Support Vector Machines (SVM), bi-objective optimization, and hyper-heuristics [2][5][8][6] forms a comprehensive approach for the project on A Dual-purpose Hyper Heuristic Support Vector Machines for Addressing Cyber-Security issues in Big Data Environments. These algorithms work

together to enhance the accuracy, efficiency, and decision-making capabilities of the cyber-security system, ultimately improving threat detection and mitigation in the realm of big data.

3.3 DESIGNING

3.3.1   UML DIAGRAM

A. Class diagram: -

A class diagram is a type of static structure diagram in the Unified Modeling Language (UML) used in software engineering to visually represent the structure and relationships of classes in a system. a class diagram provides a high-level overview of the various classes that make up the system, along with their attributes, methods, and the associations between them. In a class diagram, each class is represented as a box divided into three compartments. The top compartment typically contains the name of the class, the middle compartment lists the attributes or properties of the class, and the bottom compartment lists the methods or behaviors of the class. The relationships between classes are depicted using various types of lines and symbols, such as associations, generalization (inheritance), aggregation, and composition. These relationships help to illustrate how classes interact with each other within the system, including how they inherit behavior from parent classes, collaborate with other classes, and form larger structures. Overall, a class diagram serves as a blueprint for developers, providing a visual roadmap of the system's structure and guiding the implementation process by defining the key building blocks and their relationships. It helps stakeholders to understand the architecture of the system and facilitates communication among developers, designers, and other project stakeholders throughout the software development lifecycl
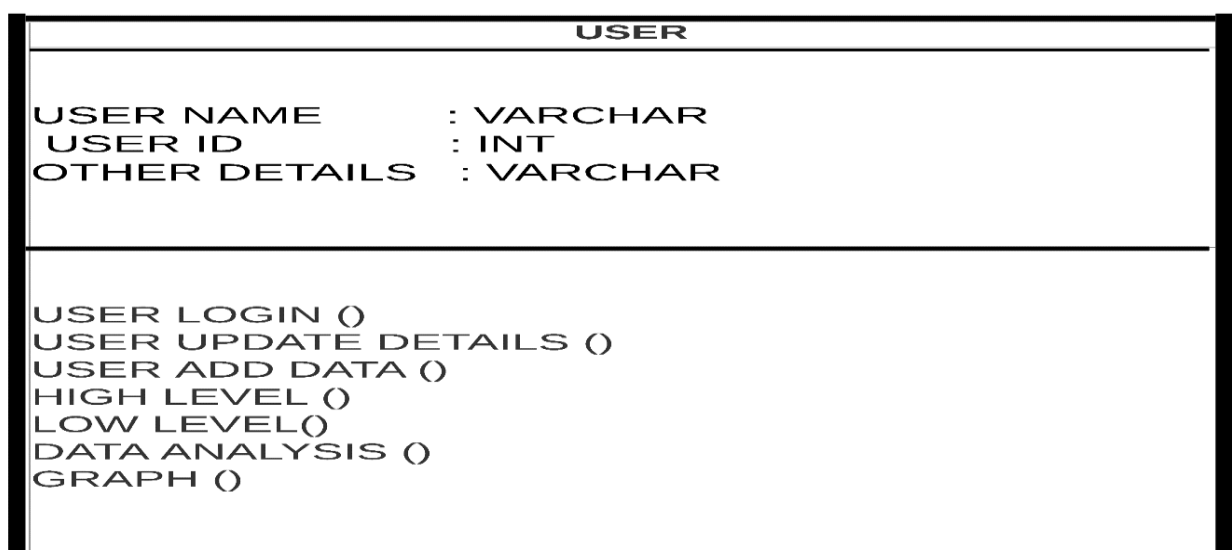
Fig 3.1: Class Diagram

4   Use case diagram:

A use case diagram is a type of behavioral diagram in the Unified Modeling Language (UML) used in software engineering to visually represent the functional requirements of a system from the perspective of its users. In a use case diagram:

Actors: Actors represent the users or external systems that interact with the system being modeled. They are typically depicted as stick figures or other simple shapes. Actors can be individuals, roles, or other systems that interact with the system being modeled.

Use Case: Use cases represent the specific functionalities or tasks that users can perform with the system. Each use case describes a discrete unit of functionality that the system provides to its users. Use cases are represented as ovals or ellipses and are connected to actors to indicate which actors are involved in each use case.

Relationships: Relationships between actors and use cases are depicted with solid lines connecting actors to the use cases they participate in. These relationships represent the interactions between users and the system and show which users are involved in each use case.

Use case diagrams are valuable tools for capturing and communicating the functional requirements of a system in a clear and concise manner. They help stakeholders, including developers, designers, and clients, to understand the system's behavior, identify user requirements, and ensure that the system meets the needs of its users. Use case diagrams serve as a foundation for further analysis and design activities in the software development process.

Use case diagram:
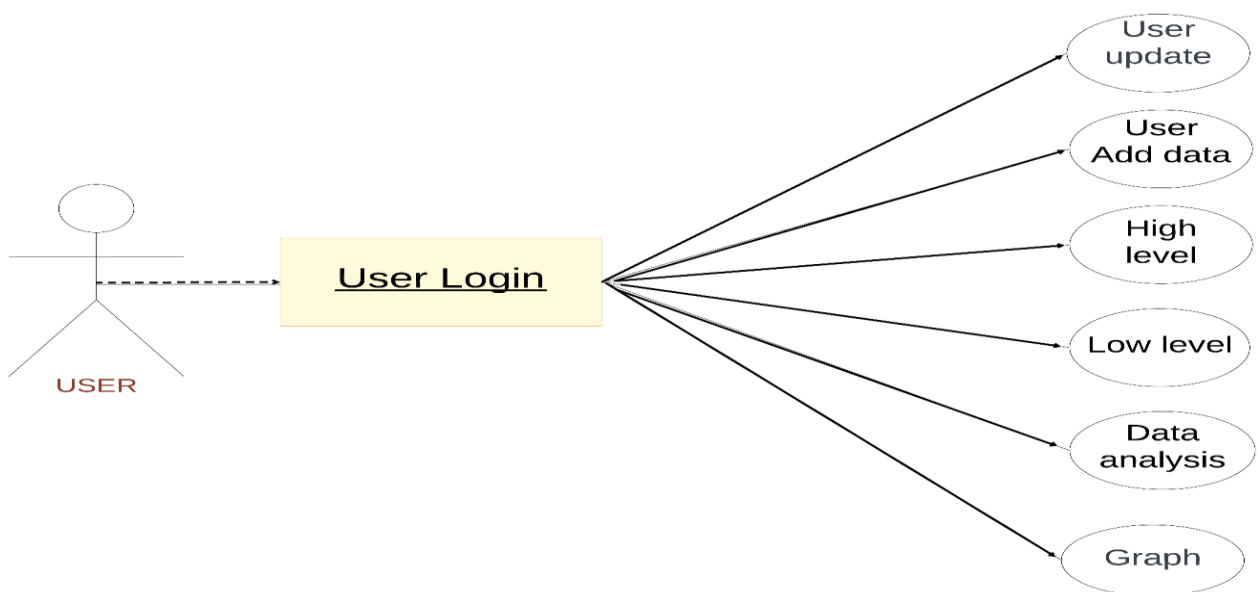
Fig 3.2: Use case Diagram

5   Sequence Diagram:

A sequence diagram is a type of interaction diagram in the Unified Modeling Language (UML) that illustrates how objects in a system interact over time to achieve a specific functionality. It depicts the sequence of messages exchanged between objects or components within the system to accomplish a particular task or scenario. Sequence diagrams are valuable tools for visualizing the dynamic behavior of a system, capturing the flow of control and data between objects, and understanding the timing and ordering of interactions.

Key Elements of a Sequence Diagram:

Objects: Objects represent the entities or components within the system that participate in the interaction. Each object is depicted as a box with the object's name at the top.

Lifelines: Lifelines represent the lifespan of an object during the sequence of interactions. They are depicted as vertical dashed lines extending downwards from the object's box. Lifelines show the existence of an object over time and provide a visual context for the sequence of messages

Messages: Messages represent the communication between objects in the system. They indicate the flow of control and data between objects and are depicted as arrows or lines connecting lifelines. Messages can be synchronous (e.g., method calls), asynchronous (e.g., signal or event notifications), or self-referential (e.g., a message sent by an object to itself).

Activation Bars: Activation bars represent the period during which an object is actively processing a message. They are depicted as horizontal bars extending from the lifeline at the point where a message is sent or received. Activation bars show when an object is busy executing a message and help visualize the concurrency of interactions between objects.

Return Messages: Return messages represent the response or result returned by an object after processing a message. They indicate the flow of control back to the sender and are depicted as arrows or lines with a dashed line indicating the return path. Return messages show the completion of a method call or the result of an operation.

24
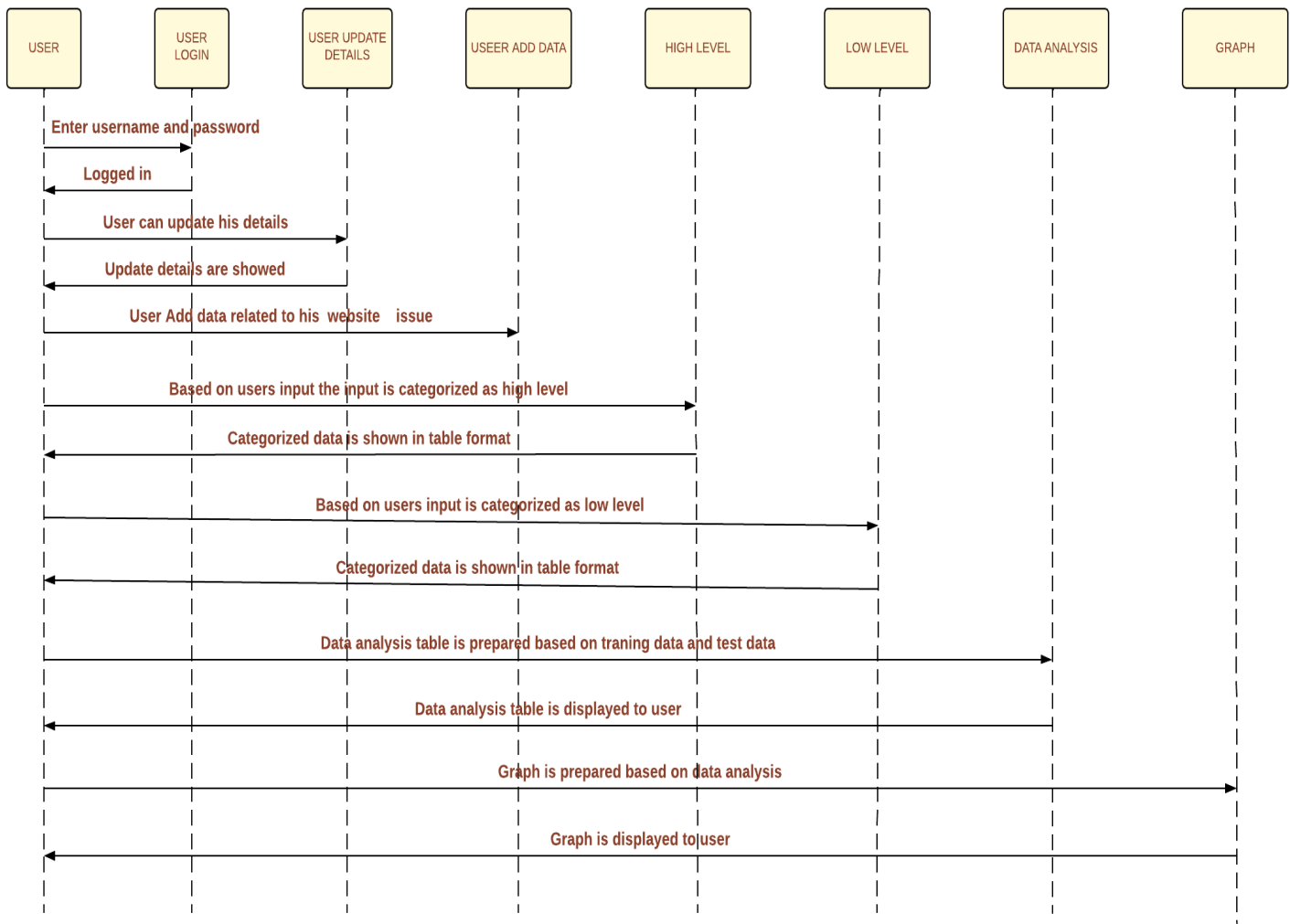
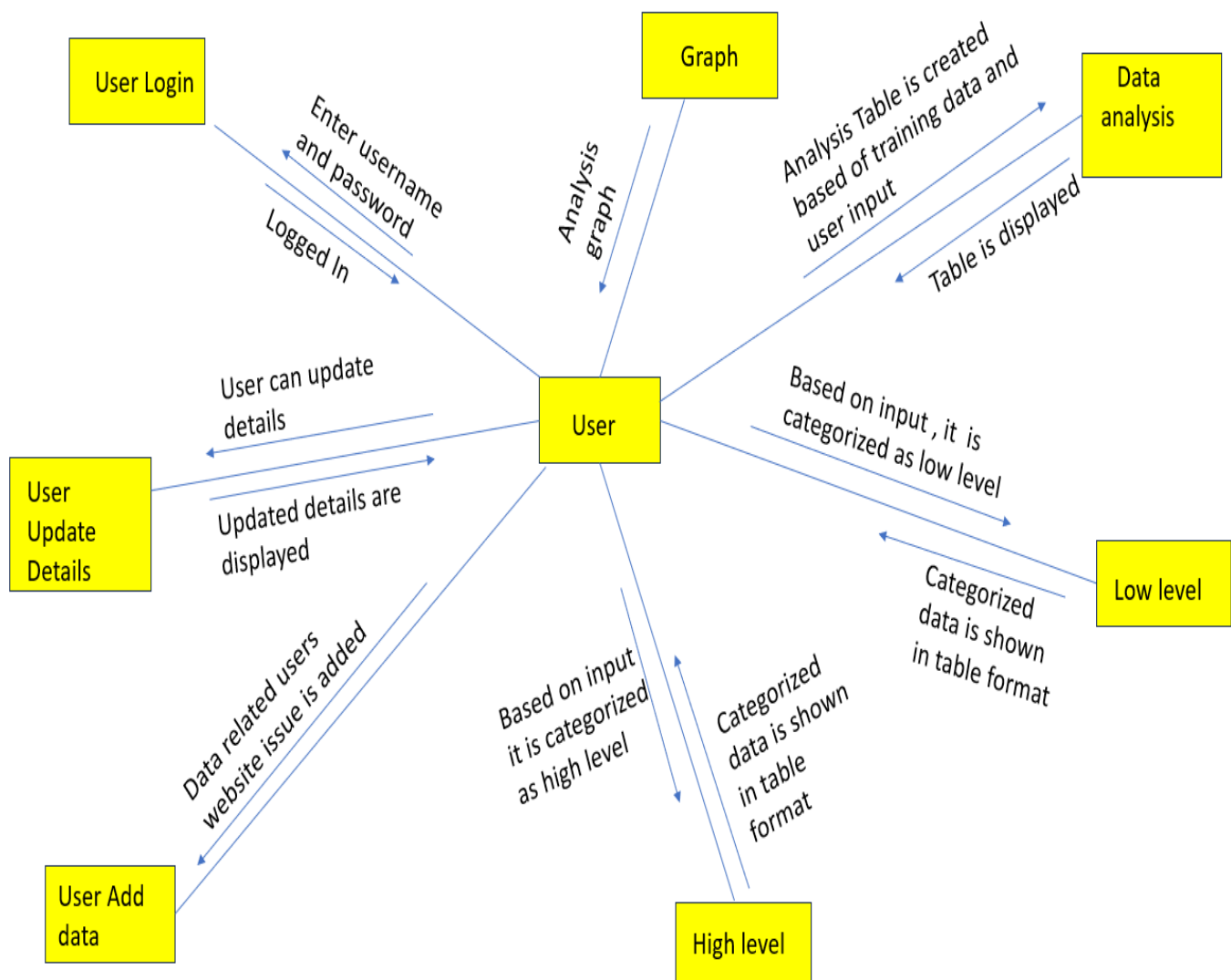**Sequence diagram:**

Fig 3.3: Sequence Diagram

6   Collaboration Diagram:

A collaboration diagram, also known as a communication diagram, is a type of interaction diagram in the Unified Modeling Language (UML) used in software engineering to visualize the interactions between objects or components within a system. Unlike sequence diagrams which focus on the sequence of messages exchanged between objects over time, collaboration diagrams emphasize the structural organization of objects and their relationships in a scenario.

25

In a collaboration diagram, objects are depicted as rectangles or boxes, each representing an entity or component in the system. Links, represented as lines connecting the objects, illustrate the relationships between them.

These relationships may indicate associations, dependencies, aggregations, or other connections. Messages, depicted as arrows or lines between objects, showcase the communication or interaction between them. They represent method calls, signals, or events exchanged during the collaboration. Roles, often labeled next to objects, clarify the responsibilities or behaviors associated with each object in the collaboration

**Collaboration diagram:**

Fig 3.4: Collaboration Diagram

7   Activity diagram:

Activity diagrams are a type of behavioural diagram in UML that illustrate the flow of control in a system, focusing on the sequence and conditions of activities. They are used to depict both sequential and concurrent processing of activities, making them essential for modeling the dynamic aspects of a system, such as the steps involved in the execution of a use case. Activity diagrams are particularly useful in software engineering for modeling and designing software systems at a high level, representing algorithms, decision structures, and program flows. They help in visualizing the workflow of activities within a use case or business process, showing the individual steps and the order in which they are presented, as well as the flow of data between activities. This makes them invaluable tools in system design and analysis, enhancing communication among project stakeholders and contributing to effective documentation.

Activity diagram
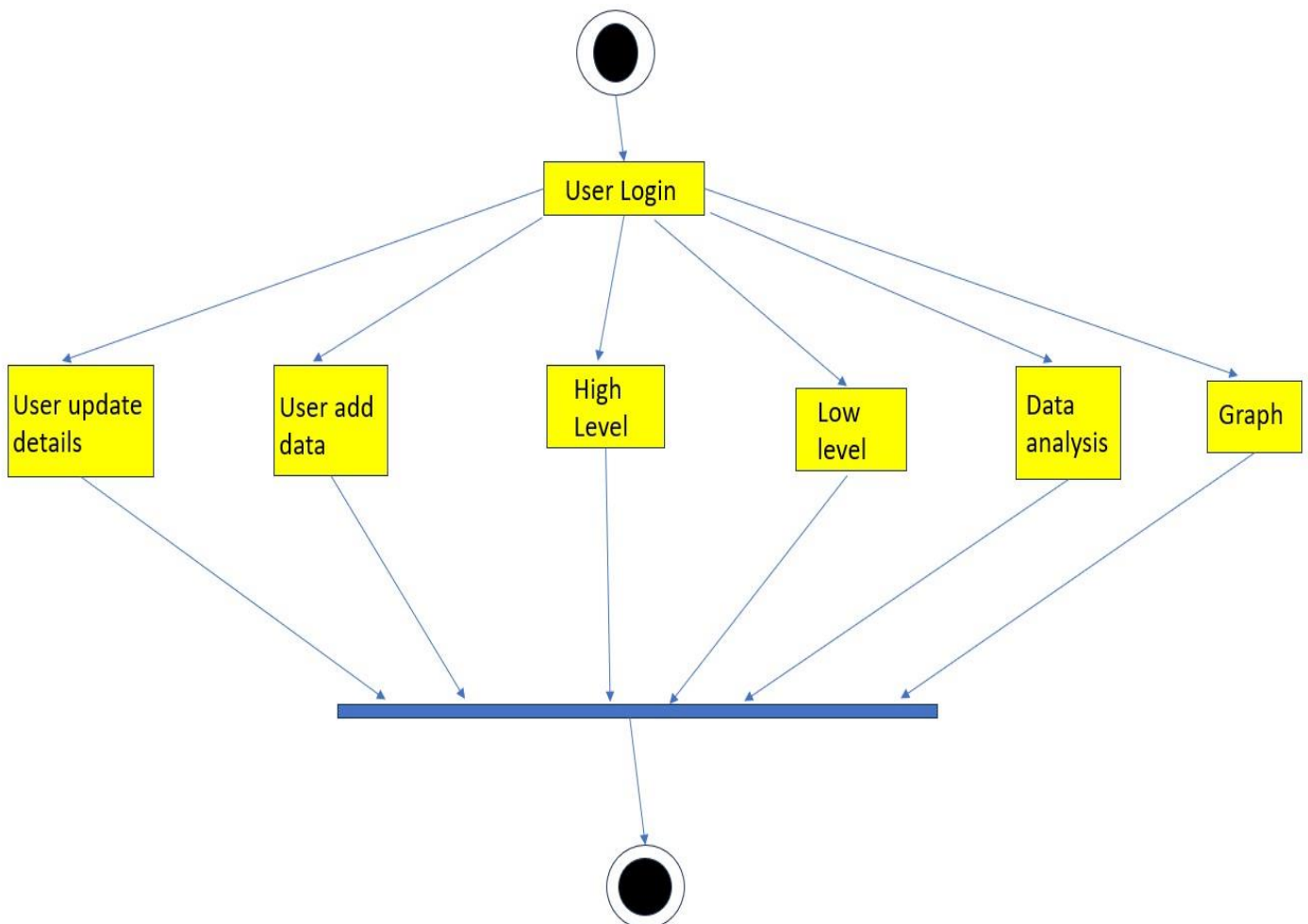
Fig 3.5: Activity Diagram

F. Data Flow Diagram:

Data Flow Diagrams (DFDs) are a graphical tool used in software engineering to represent the flow of data within a system. They are particularly useful for understanding the overall structure and functionality of a system, providing a high-level overview of the system's data flow. DFDs use a simple set of symbols, such as circles, arrows, and rectangles, to represent processes, data stores, and data flows. They are commonly used during the requirements analysis and system design stages of software development.

DFDs are distinct from Unified Modeling Language (UML) diagrams, which are a more comprehensive modelling language used in object-oriented software development. UML diagrams cover a broader scope, providing a detailed representation of the system's structure, behaviour and interactions. UML includes various types of diagrams, such as class diagrams, use case diagrams, sequence diagrams, and activity diagrams, and uses standardized symbols and notations for different types of diagrams. UML diagrams are used throughout the software development lifecycle and are typically created using specialized modelling tools that support the UML notation. In contrast to DFDs, which focus on the flow of data and provide a high-level overview of the system's processes and data dependencies, UML diagrams allow for detailed modelling of system components and interactions. UML diagrams are particularly useful for designing and documenting complex systems, offering a more detailed and holistic view of a system .DFDs are generally considered to be more abstract than UML diagrams, making them particularly useful during the early stages of system analysis and design. On the other hand, UML diagrams provide a more detailed and comprehensive view of the system, covering both the structural and behavioral aspects of the system. This allows for a deeper understanding of the system's components, relationships, and interactions

In summary, DFDs and UML diagrams serve different purposes in software development. DFDs are useful for understanding the flow of data within a system and are particularly suited for the early stages of system analysis and design. UML diagrams, on the other hand, provide a more detailed and comprehensive view of the system, covering both the structural and behavioral aspects, and are used throughout the software development lifecycle.

Dataflow Diagram:
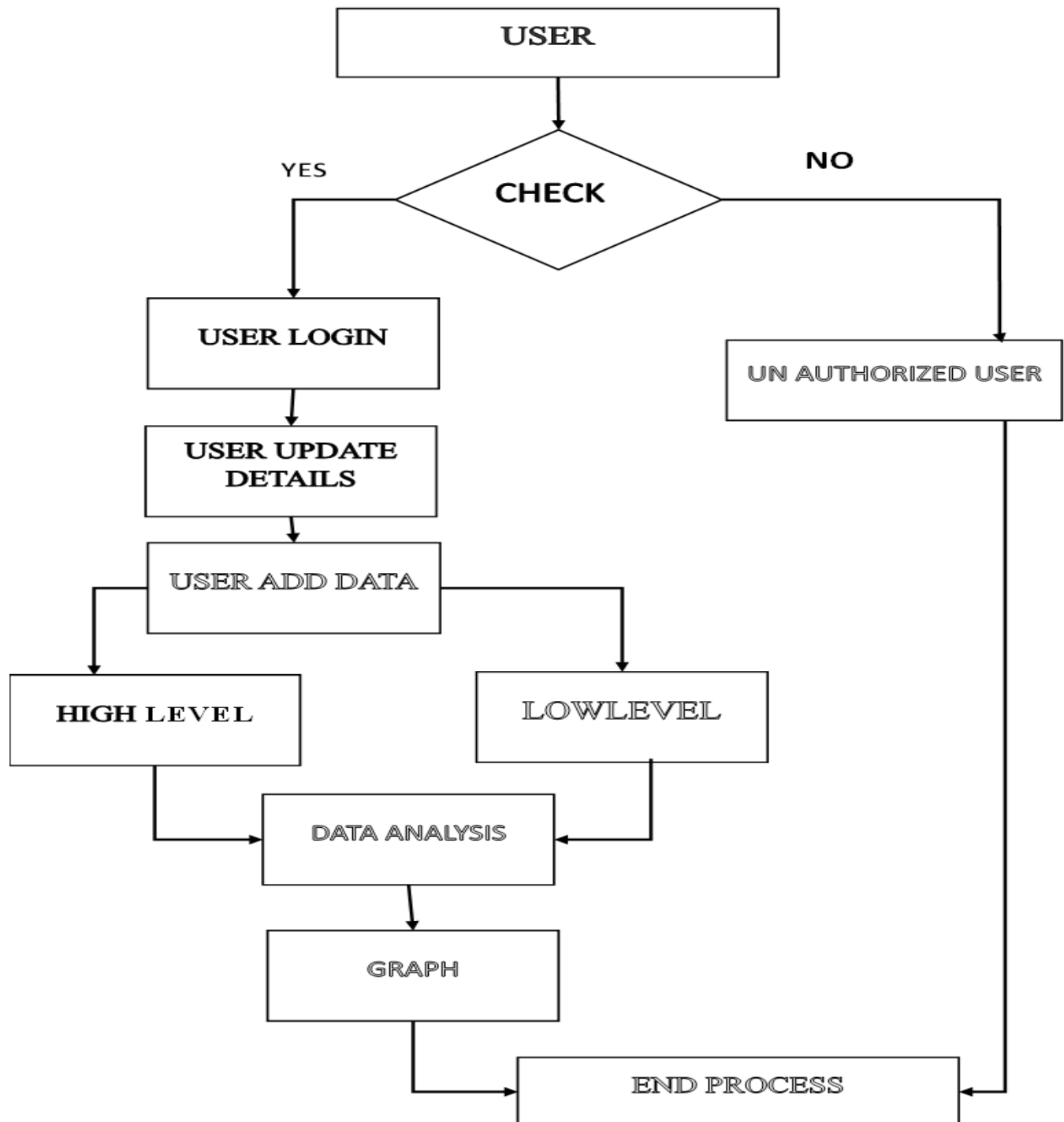
Fig 3.6: Data Flow Diagram

3.3 STEPWISE IMPLEMENTATION AND TESTING AND CODE

Implementation

The proposed framework has levels in it. One is called the high position strategy. Other one is called as low-position heuristics. In each replication, the high-position strategy selects one from the being pool of low-position heuristic, applies it to the Present result, to come up with a updated result and also makes a decision on whether to accept this new result [5][8]. The low positions heuristics are a collection of problem-specific heuristic that try to provide solutions to specific set of problems. We try to apply these two-position framework differently as explained below:

our website would allow users to enter details such as "what kind of problem the user's website is undergoing (such as got hacked, lost or stolen data from the website, etc). Based on user input our algorithm will classify the issues into various groups and inform what kind of virus users' websites might be subjected to (such as polymorphic virus or macro virus). here we are doing the same work as SVM which would be the classification of data here our algorithms group training data into various groups each titled with a unique virus name. whenever a user inputs his website issues, based on input the algorithm will decide which group the input issues (given by the user) belong to, hence providing the user the name of virus the user's website must have undergone. here our algorithm does similar work as work svm which is classification and detection [2] but our algorithm gives speedy and accurate results without the need to set configurations frequently (which might be necessary if we are using svm[7]).

user accesses the website and enter the information. The data entered by user is the sole input for the developed algorithm. There are various fields of information that must be entered by user/client

- Corporate name
- Department
- Description
- Website
- Methods (this field has description about the issue users website is going through)
- Records

The proposed algorithm accesses the client's input data and then choses to which data cluster (made of training data) that new client input might belong to. after clustering stores data in database in organized way .as we working on cyber security training data would be viruses name and their impact based on clients input (about their website) algorithm decides which virus must be associated to user input. then it stores all of this in database. Here our algorithm focusses on categorizing the issue given by user as high level or low level. High level includes (macro virus, polymorphic virus, worms, droppers, file infectors, system or boot record infectors, Ransome ware, spyware, logic bombs).

Testing

TEST

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.

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.

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.

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.

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.

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.

Acceptance testing Acceptance testing is the final phase of the software testing process, where the system is evaluated to determine whether it meets the specified requirements and is ready for deployment. Requirements Validation:

The system's functionality is tested against the specified requirements to ensure that it meets the needs of the end users and stakeholders.

User Experience:

The user interface, usability, and overall user experience are evaluated to ensure that they are intuitive and meet user expectations.

Real-World Scenarios:

The system is tested using real-world scenarios to verify that it behaves correctly in different usage situations.Any feedback or issues identified during acceptance testing are addressed, and necessary modifications are made to the system to ensure its readiness for deployment.

Test Case ID	Description	Expected Result	Actual Result
TC001	User login with valid credentials	Successful login	Successful login
TC002	User login with invalid credentials	Not able to login	Not able to login
TC003	User updates his/her details	Updated details are stored and displayed	Updated details are stored and displayed
TC004	User adds the data regarding the issue user's website is going through	Data is added to user's page	Data is added to user's page
TC005	User Adding data with keywords associated with attack 1(which is associated with macro virus in code)	Detection of "Macro viruses"	Detection of "Macro viruses"
TC005	User Adding data with keywords associated with attack 2(which is associated with file infectors in code)	Detection of "File Infectors"	Detection of "File Infectors"
TC006	User Adding data with keywords associated with attack 6(which is associated with logic bombs in code)	Detection of "Logic bombs"	Detection of "Logic bombs"

Table 3.7: Testing

CODE:

Manage.py:

```python
#!/usr/bin/env python
import os
import sys
if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE",
"Bi_objective_Hyper_Heuristic.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

Users views.py:

```python
import re
from django.contrib import messages
from django.db.models import Q, Count
from django.shortcuts import render, redirect, get_object_or_404


# Create your views here.
from Users.forms import Userregister_Form
from Users.models import Userregister_Model, UserAdd_Model
def user_login(request):
    if request.method == "POST":
        name = request.POST.get('name')
```

```python
        password = request.POST.get('password')
        try:
            enter = Userregister_Model.objects.get(name=name,password=password)
            request.session['name']=enter.id
            return redirect('user_mydetails')
        except:
            pass
    return render(request, 'users/user_login.html')
def user_register(request):
    if request.method == "POST":
        forms = Userregister_Form(request.POST)
        if forms.is_valid():
            forms.save()
            messages.success(request, 'You have been successfully registered')
            return redirect('user_login')
    else:
        forms = Userregister_Form()
    return render(request, 'users/user_register.html',{'form':forms})
def user_mydetails(request):
    name = request.session['name']
    ted = Userregister_Model.objects.get(id=name)
    return render(request, 'users/user_mydetails.html',{'object':ted})
def user_updatedetails(request):
    name = request.session['name']
    obj = Userregister_Model.objects.get(id=name)
    if request.method == "POST":
        UserName = request.POST.get('name', '')
        Email = request.POST.get('email', '')
        Password = request.POST.get('password', '')
        Phone_Number = request.POST.get('phoneno', '')
        Address = request.POST.get('address', '')
        Location = request.POST.get('location', '')
        obj = get_object_or_404(Userregister_Model, id=name)
```

35

```python
        obj.name = UserName
        obj.email = Email
        obj.password = Password
        obj.phoneno = Phone_Number
        obj.address = Address
        obj.location = Location
        obj.save(update_fields=["name", "email", "password", "phoneno", "address","location"])
        return redirect('user_mydetails')
    return render(request, 'users/user_updatedetails.html',{'form':obj})
def user_search(request):
    objw = ''
    if request.method == "POST":
        usid = request.POST.get('search')
        objw = UserAdd_Model.objects.filter(Q(cname=usid) | Q(dept=usid))
    return render(request,'users/user_search.html', {'objes': objw})
def user_adddata(request):
    name = request.session["name"]
    obj = Userregister_Model.objects.get(id=name)
    attack1 = []
    attack2, attack3, attack4, attack5, attack6, attack7, attack8, attack9 = [], [], [], [], [], [], [], []
    splt = ''
    Cname = ''
    Dept = ''
    Description = ''
    Record = ''
    Method = ''
    txt =''
    ans = ''
    if request.method == "POST":
        Cname = request.POST.get("cname")
        Dept = request.POST.get("dept")
        Description = request.POST.get("description")
        txt = request.POST.get("name")
```

```python
        Method = request.POST.get("method")

        Record = request.POST.get("record")

        splt = (re.findall(r"[\w']+", str(txt)))

    for f in splt:

        if f in ('IPid', 'FDDI', 'x25', 'rangingdistance'):

            attack1.append(f)

        elif f in ('tcpchecksum', 'mtcp', 'controlflags', 'tcpoffset', 'tcpport'):

            attack2.append(f)

        elif f in ('ICMPID', 'udptraffic', 'udpunicorn', 'datagramid', 'NTP', 'RIP', 'TFTP'):

            attack3.append(f)

        elif f in ('GETID', 'POSTID', 'openBSD', 'appid', 'sessionid', 'transid', 'physicalid'):

            attack4.append(f)

        elif f in ('SYN', 'ACK', 'synpacket', 'sycookies'):

            attack5.append(f)

        elif f in ('serverattack', 'serverid', 'blockbankwidth'):

            attack6.append(f)

        elif f in ('monlist', 'getmonlist', 'NTPserver'):

            attack7.append(f)

        elif f in ('portid', 'FTPID', 'tryion', 'fragflag'):

            attack8.append(f)

        elif f in ('malwareid', 'gethttpid', 'httpid'):

            attack9.append(f)

    if len(attack1) > len(attack2) and len(attack1) > len(attack3) and len(attack1) > len(attack4) and
len(attack1) > len(attack5) and len(attack1) > len(attack6) and len(attack1) > len(attack7) and len(
attack1) > len(attack8) and len(attack1) > len(attack9):

        ans = "Macro viruses"

    elif len(attack2) > len(attack1) and len(attack2) > len(attack3) and len(attack2) > len(attack4)
and len(attack2) > len(attack5) and len(attack2) > len(attack6) and len(attack2) > len(attack7) and
len( attack2) > len(attack8) and len(attack2) > len(attack9):

        ans = "File infectors"

    elif len(attack3) > len(attack2) and len(attack3) > len(attack1) and len(attack3) > len(attack4)
and len( attack1) > len(attack5) and len(attack1) > len(attack6) and len(attack1) > len(attack7) and
len(attack1) > len(attack8) and len(attack1) > len(attack9):
```

```
    ans = "System or boot-record infectors"
  elif len(attack4) > len(attack2) and len(attack4) > len(attack3) and len(attack4) > len(attack1)
and len( attack4) > len(attack5) and len(attack4) > len(attack6) and len(attack4) > len(attack7) and
len( attack4) > len(attack8) and len(attack4) > len(attack9):
    ans = "Polymorphic viruses"
  elif len(attack5) > len(attack2) and len(attack5) > len(attack3) and len(attack5) > len(attack4)
and len(attack5) > len(attack1) and len(attack5) > len(attack6) and len(attack5) > len(attack7) and
len(attack5) > len(attack8) and len(attack5) > len(attack9):
    ans = "Ransomware"
  elif len(attack6) > len(attack2) and len(attack6) > len(attack3) and len(attack6) > len(attack4)
and len(attack6) > len(attack5) and len(attack6) > len(attack1) and len(attack6) > len(attack7) and
len(attack6) > len(attack8) and len(attack6) > len(attack9):
    ans = "Logic bombs"
  elif len(attack7) > len(attack2) and len(attack7) > len(attack3) and len(attack7) > len(attack4)
and len( attack7) > len(attack5) and len(attack7) > len(attack6) and len(attack7) > len(attack1) and
len(attack7) > len(attack8) and len(attack7) > len(attack9):
    ans = "Worms"
  elif len(attack8) > len(attack2) and len(attack8) > len(attack3) and len(attack8) > len(attack4)
and len( attack8) > len(attack5) and len(attack8) > len(attack6) and len(attack8) > len(attack7) and
len(attack8) > len(attack1) and len(attack8) > len(attack9):
  ans = "Droppers"
  elif len(attack9) > len(attack2) and len(attack9) > len(attack3) and len(attack9) > len(attack4)
and len( attack9) > len(attack5) and len(attack9) > len(attack6) and len(attack9) > len(attack7) and
len(attack9) > len(attack8) and len(attack9) > len(attack1):
    ans = "Spyware "
  else:
    ans = "lowlevel"


UserAdd_Model.objects.create(uregid=obj,cname=Cname,dept=Dept,description=Description,web
site=txt,method=Method,record=Record,attackresult=ans)
  return render(request,'users/user_adddata.html')
def higlevel(request):
  obj = UserAdd_Model.objects.filter(Q(attackresult='Macro viruses') | Q(attackresult='File
```

38

infectors') | Q(

    attackresult='System or boot-record infectors') | Q(attackresult='Polymorphic viruses') | Q(

    attackresult='Ransomware') | Q(attackresult='Logic bombs') | Q(attackresult='Worms') | Q(

    attackresult='Droppers') | Q(attackresult='Spyware'))

    return render(request,'users/higlevel.html',{'object':obj})

def lowlevel(request):

   obj = UserAdd_Model.objects.filter(attackresult='lowlevel')

   return render(request,'users/lowlevel.html',{'object':obj})

def data_analysis(request):

   chart =

UserAdd_Model.objects.values('attackresult','method').annotate(dcount=Count('attackresult'))

   return render(request,'users/data_analysis.html',{'objects':chart})

def chart_page(request,chart_type):

   chart = UserAdd_Model.objects.values('attackresult').annotate(dcount=Count('dept'))

   return render(request,'users/chart_page.html',{'chart_type':chart_type,'objects':chart})

def user_page(request):

   obj = UserAdd_Model.objects.all()

   return render(request,'users/user_page.html',{'object':obj})

models.py:

from tkinter import CASCADE

from django.db import models

Create your models here.

class Userregister_Model(models.Model):

   name= models.CharField(max_length=50)

   email=models.EmailField(max_length=30)

   password=models.CharField(max_length=10)

   phoneno=models.CharField(max_length=15)

   address=models.CharField(max_length=500)

   location=models.CharField(max_length=20)

class UserAdd_Model(models.Model):

   uregid = models.ForeignKey(Userregister_Model, on_delete=CASCADE)

```python
cname = models.CharField(max_length=100)

dept = models.CharField(max_length=10000)

description = models.CharField(max_length=1000)

website=models.CharField(max_length=1000)

method = models.CharField(max_length=100)

record = models.CharField(max_length=500)

attackresult = models.CharField(max_length=500)
```

forms.py

```python
from django import forms
from Users.models import Userregister_Model
class Userregister_Form(forms.ModelForm):
    class Meta:
        model = Userregister_Model
        fields = ('name','email','password', 'phoneno', 'address', 'location')
```

wsgi.py

```python
import os
from django.core.wsgi import get_wsgi_application
os.environ.setdefault("DJANGO_SETTINGS_MODULE",
"Bi_objective_Hyper_Heuristic.settings")
application = get_wsgi_application()
```

Urls.py

```python
from django.conf.urls import url
from django.contrib import admin
from Users import views as user_view
from Admin import views as admin_view
urlpatterns = [
    url('admin/', admin.site.urls),
    url('^$', user_view.user_login, name="user_login"),
    url(r'^user_register/$', user_view.user_register, name="user_register"),
```

```
url(r'^user_search/$', user_view.user_search, name="user_search"),
url(r'^user_mydetails', user_view.user_mydetails, name="user_mydetails"),
url(r'^user_updatedetails', user_view.user_updatedetails, name="user_updatedetails"),
url(r'^user_adddata/$', user_view.user_adddata, name="user_adddata"),
url(r'^higlevel/$',user_view.higlevel,name='higlevel'),
url(r'^lowlevel/$',user_view.lowlevel,name='lowlevel'),
url(r'^data_analysis/$',user_view.data_analysis,name='data_analysis'),
url(r'^user_page/$',user_view.user_page,name='user_page'),
url(r'chart_page/(?P<chart_type>\w+)', user_view.chart_page,name="chart_page"),
url(r'^admin_login/$',admin_view.admin_login, name='admin_login'),
url(r'^user_details/$',admin_view.user_details, name='user_details'),
url(r'^admin_analysis/$',admin_view.admin_analysis, name='admin_analysis'),
url(r'^graph_analysis/(?P<aachart_page>\w+)', admin_view.graph_analysis,
name="graph_analysis"),
```

apps.py

```
from django.apps import AppConfig
class UsersConfig(AppConfig):
    name = 'Users'
```

3.4 MODEL ARCHITECTURE

Fig 3.8: Model Architecture

- **3.6   SYSTEM REQUIRMENTS**

- Processor   -   Intel i3(min)

- Speed   -   1.1 Ghz

- RAM   -   4GB (min)

- Hard Disk   -   500 GB

   SOFTWARE REQUIREMENTS:

- Operating System   -   Windows7(min)
- Programming Language   -   Python 3.6.2
- Framework   -   Django
- Database   -   mysql
- Wamp server
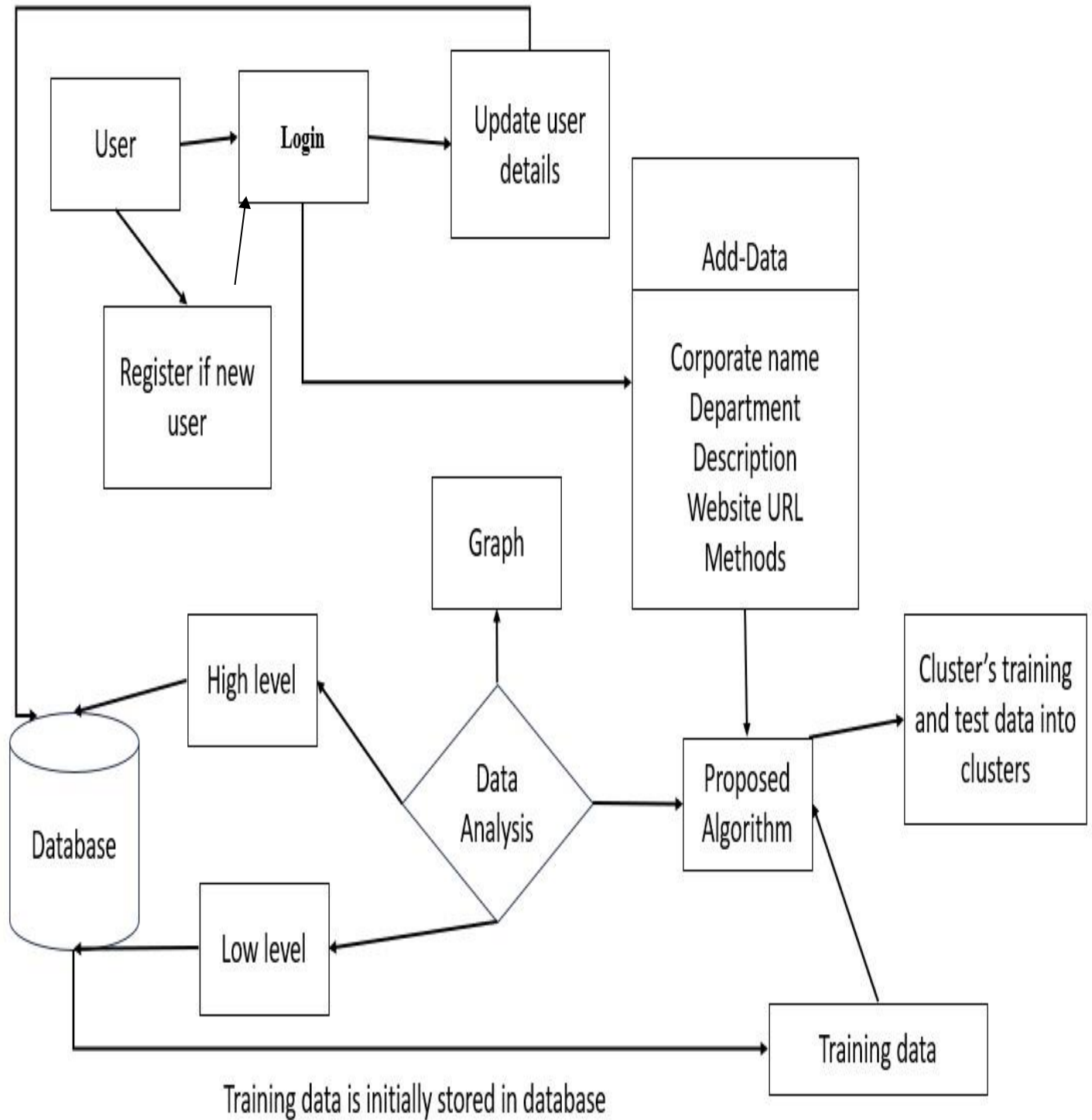
CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

4.1   OUTPUT SCREENS:

First the user enters the website by entering username and password

Fig 4.1: Login Page

If the user is new to website he/she can register
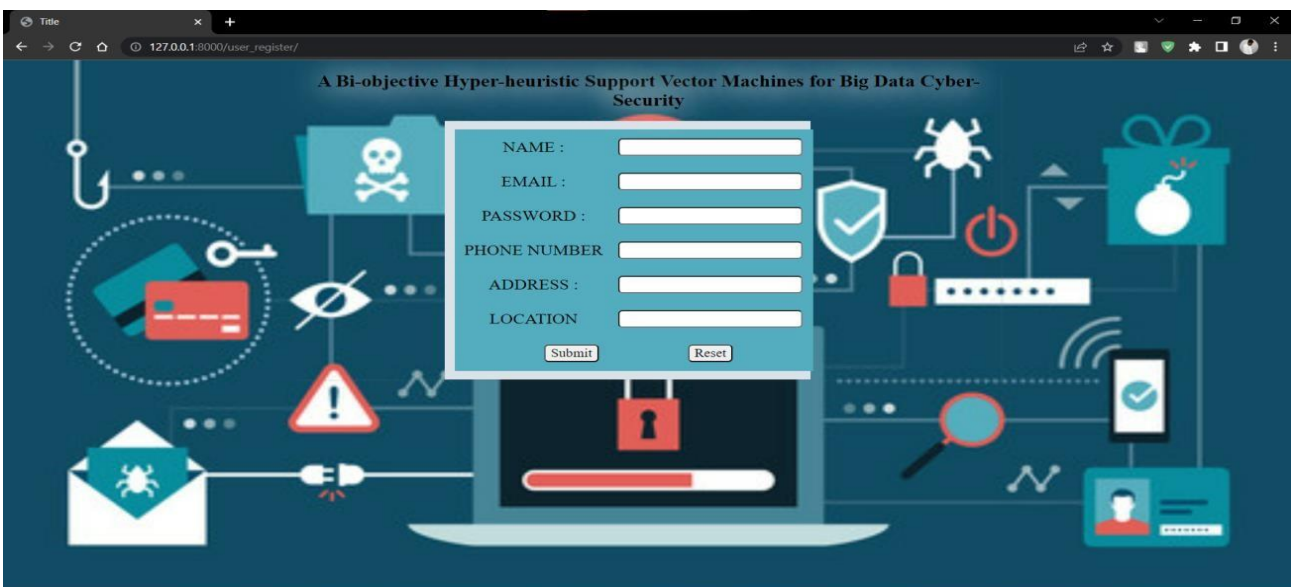
Fig 4.2: Registration Page

Details entered by user can be viewed in this section
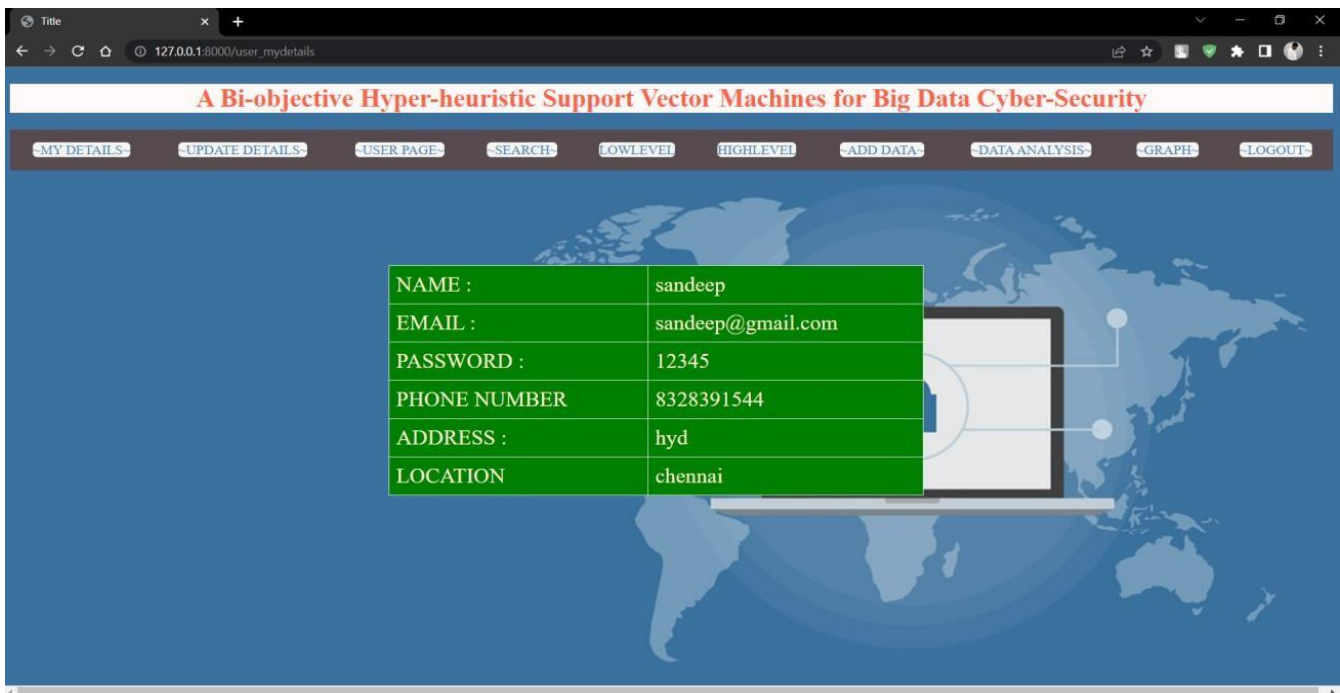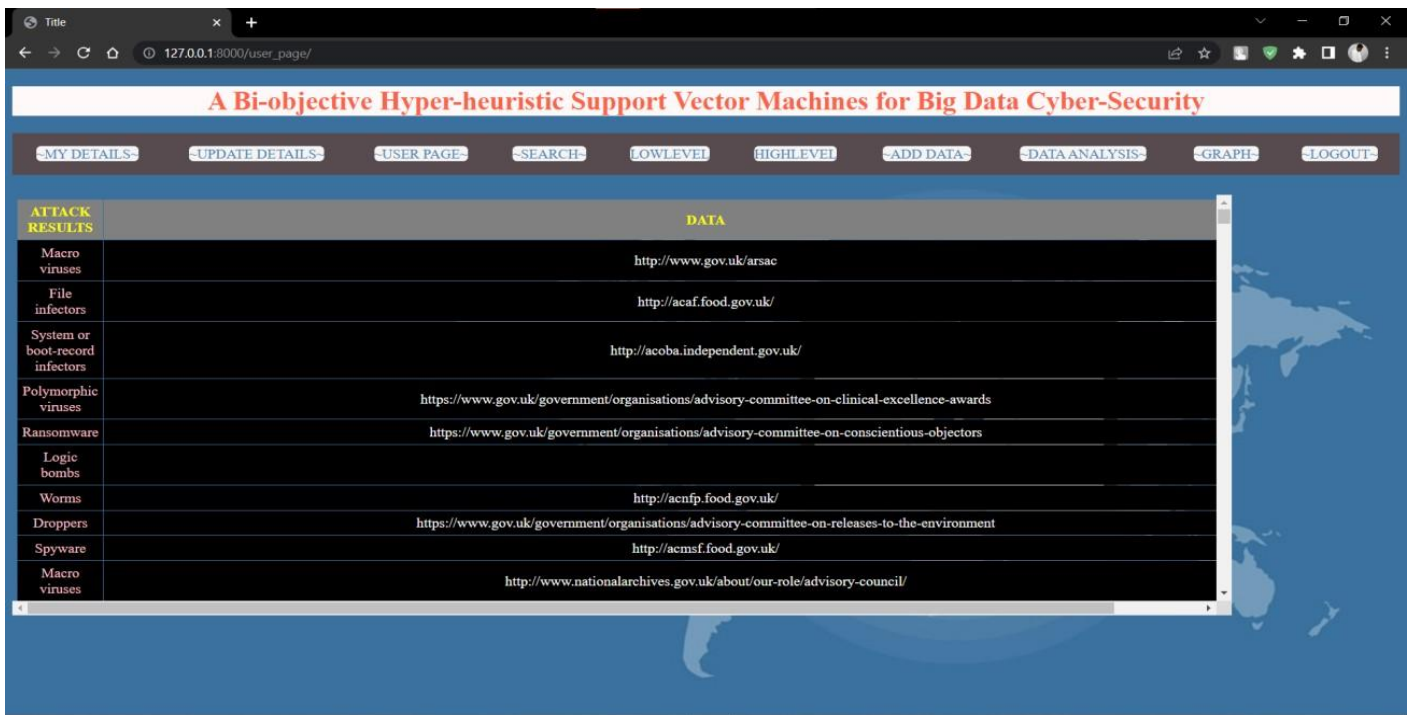
Fig 4.3: User Details Page

User can update his details

Fig 4.4: Update Details Page

46

User page has the website URL's and the name of the viruses it have been through

Fig 4.5: User Page

User can search for any data related to various pre-existing website.

Fig 4.6: Search

All the websites that are effected by low level viruses are listed in this section

Fig 4.7: Low level

All the websites that are effected by high level viruses are listed in this section. We have categorized main prominent viruses under high level and rest as low level.

Fig 4.8: High level

User can add data related to issue his/her website is going through, in the given fields

Fig 4.9: Add Data

Data analysis table is generated based of user's input and training data

Fig 4.10: Data Analysis

Data Analysis graph is generated.

Fig 4.11: Graph

4.2   PERFORMANCE METRICS:
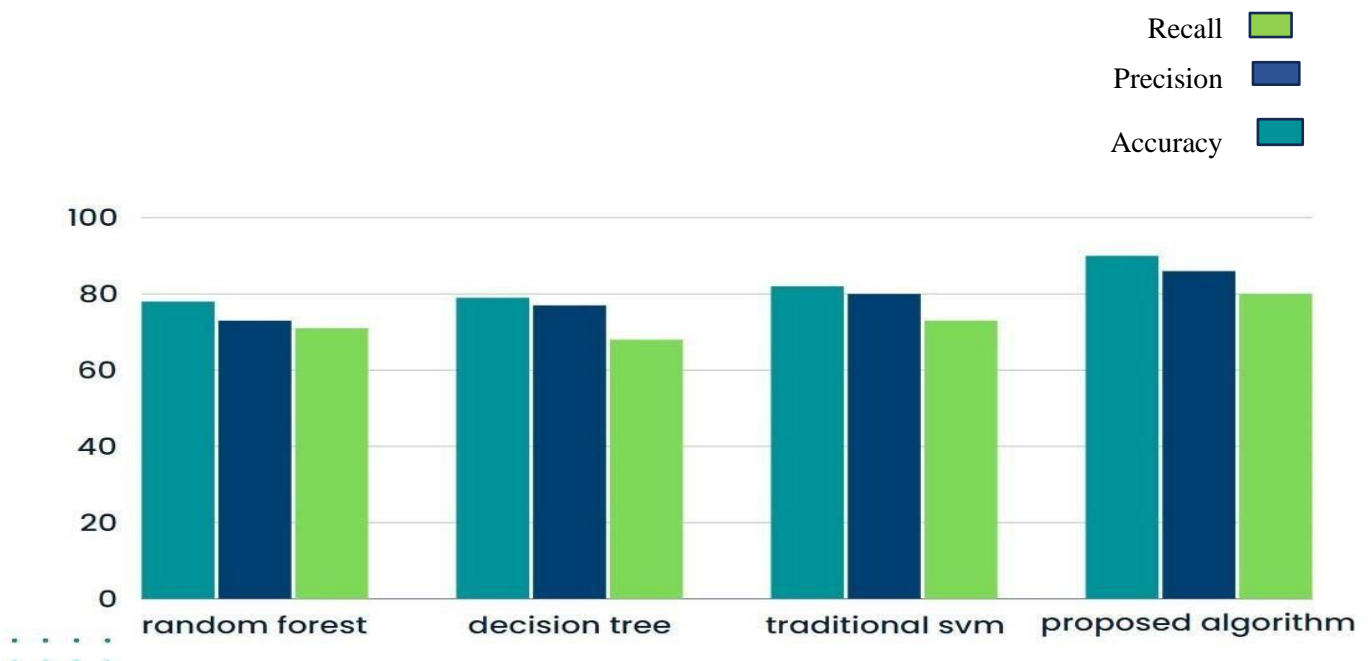
4.11 Accuracy, precision, recall comparison graph

ALGORITHMS	Accuracy	Precision	Recall
RANDOM FOREST	78.65	73.13	71.014
DECISION TREE CLASSIFIER	79.77	77.04	68.11
TRADITIONAL SVM	82.02	80.3	73.14
Modified HYPER HEURISTIC SVM (PROPOSED ALGORITHM)	90.03	86.1	80.46

Fig 4.12   comparison table

CHAPTER 5
CONCLUSION

B.Tech (CSE)

CHAPTER 5

CONCLUSION

5.1 CONCLUSION AND FUTURE ENHANCEMENT:

This study, a hyper-heuristic SVM optimization model is suggested to face with cybersecurity issues. We remodeled the SVM setup process as a dual- goal optimization issue, where precision and model intricacy are seen as opposing priorities. This dual-goal optimization issue can be dealt through the hyper-heuristic system. It merges the capabilities of decomposition- and Pareto-based methods.

Our project caters the drawbacks of svm but there is always the scope of increasing the accuracy of classification (done by our algorithm) by training the model with more as well as harder datasets. hyper heuristics are used in this project hence our work gives a speedy and accurate result. Our algorithm can also be further tuned to handle much larger dataset.

DUAL-PURPOSE HYPER-HEURISTIC SUPPORT VECTOR MACHINE DESIGNED FOR ADDRESSING CYBER
SECURITY ISSUES IN BIG DATA ENVIRONMENTS

54

REFERENCES

REFERENCE

[1] Yingchun Liu, proposed an optimized upgradation of Random forest algorithm in big data environment, COMPUTER MODELLING & NEW TECHNOLOGIES 2014.

[2] DURGESH K, SRIVASTAVA, LEKHA BHAMBHU, discussed on "DATA CLASSIFICATION USING SVM", Journal of Theoretical and Applied Information Technology,2010.

[3] Yan Hou, proposed optimized approach of Decision Tree Algorithm for Big Data Analysis, Advances in Intelligent Systems R

[4] Malak El Bakry, Soha Safwat, Osman Hegazy, researched and published paper on "Big Data Classification using Fuzzy KNN", 2015.

[5] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan & Rong Qu provided an analysis on topic "Hyper-heuristics: a survey of the state of the art", Journal of the Operational Research Society ,(2013)

[6] Cobos C, Mendoza M and Leon E (2011) put forward A hyper- heuristic approach to design and tuning heuristic methods for web document clustering. In: Evolutionary Computation (CEC). IEEE: New Orleans, LA, pp 1350–1358.

[7] Ximing Wang & Panos M. Pardalos, performed a Survey of Support Vector Machines with Uncertainties, Annals of Data Science,2015

[8] Ender Özcan,Burak Bilgin,Emin Erkan Korkmaz, did a comprehensive analysis of hyper-heuristic,Annual International Conference on Information and Sciences (AiCIS),Article in Intelligent Data Analysis

GITHUB LINK:

https://github.com/SRIDHAM25/major-project-phase-2-batch-30

ACCEPTANCE LETTER:

Dear Ch.Sridham

CMR College of Engineering &Technology,

Warm Greetings...!!!

We are greatly delighted with your interest of participation towards our conference titled "5th International Conference on Artificial Intelligence and Its Emerging Areas [NEC-ICAIEA-2K24]" organized by "Department of Computer Science and Engineering [CSE], CSE[AI&ML], CSE[DS], CSE[CS], CSE[AI], IT &MCA" which is proposed to be held on 12th & 13th April 2024 through online mode.

We are happy to inform you that your Paper Title " DUAL-PURPOSE HYPER-HEURISTIC SUPPORT VECTOR MACHINE DESIGNED FOR ADDRESSING CYBER SECURITY ISSUES IN BIG DATA ENVIRONMENTS" has been accepted for presenting in the conference [NEC-ICAIEA-2K24].

PAPER CODE :NECICAIEA-2K24-022

 PAPER TITLE: DUAL-PURPOSE HYPER-HEURISTIC SUPPORT VECTOR MACHINE DESIGNED FOR ADDRESSING CYBER SECURITY ISSUES IN BIG DATA ENVIRONMENTS

Fill google form with all author details and upload camera ready paper with IEEE format :https://forms.gle/qF79rHEbZk7XJnNo6

Shortly you will receive an mail regarding the "Joining link" and "Slot Timings" for the paper presentation.

 WhatsApp Link : https://chat.whatsapp.com/FJ4EG893YZV48TfLXUUf0V

Thank you,

With Regards,

Dr S V N SREENIVASU
CONVENER
NEC-ICAIEA-2K24,
Dean R&D, Professor in CSE
Narasaraopeta Engineering College,
Kotappakonda Road, Narasaraopet-522601, A.P.
9849866902