DETECTION AND ANALYSIS OF CYBER THREATS

A report submitted in partial fulfilment of the requirements for the Degree of

Bachelor of Technology

in

Computer Science and Engineering (Cyber Security)

BY

Ch. Sree Kiran	2011CS040004
B. Ajay Kumar Reddy	2011CS040006
K. Vinay	2011CS040043
N. Harthik Reddy	2011CS040053

Under the esteemed guidance

Mrs. A. Ramya

(Assistant Professor)

Department of Computer Science & Engineering (Cyber Security)

School of Engineering

MALLA REDDY UNIVERSITY

Maisammaguda, Dulapally, Hyderabad, Telangana 500100

2024

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

Department of Computer Science & Engineering (Cyber Security)

CERTIFICATE

This is to certify that the project report entitled **"Detection and analysis of cyber threats"**, submitted by **Ch. Sreekiran(2011CS040004), B. AjaykumarReddy(2011CS040004), K. Vinay(2011CS040004), N. Harthik Reddy(2011CS040004)** towards the partial fulfilment for the award of Bachelor's Degree in (Cyber Security) from the Department of (Computer Science and Engineering), Malla Reddy University, Hyderabad, is a record of bonafide work done by him/ her. The results embodied in the work are not submitted to any other University or Institute for award of any degree or diploma.

Internal Guide:	**Head of the Department**
Mrs. A. Ramya	**Dr. G. Anand Kumar**
(Assistant Professor)	

External Examiner

DECLARATION

We hereby declare that the project report entitled "**Detection and analysis of cyber threats**" has been carried out by us and this work has been submitted to the Department of Computer Science and Engineering (Cyber Security), Malla Reddy University, Hyderabad in partial fulfilment of the requirements for the award of degree of Bachelor of Technology. We further declare that this project work has not been submitted in full or part for the award of any other degree in any other educational institutions.

Place:

Date:

Ch. Sree Kiran	2011CS040004
B. Ajay Kumar Reddy	2011CS040006
K. Vinay	2011CS040043
N. Harthik Reddy	2011CS040053

ACKNOWLEDGEMENT

We extend our sincere gratitude to all those who have contributed to the completion of this project report. Firstly, we would like to extend our gratitude to Dr. V. S. K Reddy, Vice-Chancellor, for his visionary leadership and unwavering commitment to academic excellence.

We would also like to express my deepest appreciation to our project guide [Mrs. A. Ramya, Assistant Professor], whose invaluable guidance, insightful feedback, and unwavering support have been instrumental throughout the course of this project for successful outcomes.

We are also grateful to Dr. G. Anand Kumar, Head of the Department of Computer Science and Engineering Cyber Security, for providing us with the necessary resources and facilities to carry out this project.

We would like to thank Dr. Kasa Ravindra, Dean, School of Engineering, for his encouragement and support throughout my academic pursuit.

My heartfelt thanks also go to Dr. Harikrishna Kamatham, Associate Dean School of Engineering for his guidance and encouragement.

We are deeply indebted to all of them for their support, encouragement, and guidance, without which this project would not have been possible.

Ch. Sree Kiran(2011CS040004)

B. Ajay Kumar Reddy(2011CS040006)

K. Vinay(2011CS040043)

N. Harthik Reddy (2011CS040053)

ABSTRACT

Intrusion Detection System (IDS) leveraging advanced machine learning techniques, notably the Random Forest algorithm, to enhance network security against evolving cyber threats. Traditional IDS systems often struggle with new attack types and processing efficiency. By employing Gaussian Naive Bayes, Decision Trees, Logistic Regression, Random Forest, and Gradient Classifier, the proposed system aims to effectively recognize patterns and identify intrusions even in complex and limited data scenarios.

Specifically, the Random Forest algorithm is highlighted for its ability to handle high-dimensional and noisy network traffic data, adapt to diverse intrusion scenarios, and exhibit robust classification performance. The system architecture involves data pre-processing, feature engineering, and model development, culminating in real-time implementation. Advantages include robustness to complexity, adaptability to diverse scenarios, efficient noise handling, high performance, precision in detection, minimized false alarms, scalability, and computational efficiency.

The system architecture encompasses multi-tiered data collection, pre-processing, detection engines, decision-making modules, centralized management consoles, logging, and reporting mechanisms. Continuous adaptation and integration within the broader security ecosystem ensure swift detection, response, and mitigation of intrusions in network environments. Overall, the proposed IDS system presents a comprehensive approach to bolstering network security against modern cyber threats.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the rapidly evolving landscape of cybersecurity, the need for robust defence mechanisms against sophisticated cyber threats is more pressing than ever. Intrusion Detection Systems (IDS) stand as crucial sentinels, constantly surveillant network traffic to identify and mitigate potential intrusions. However, the complexity and diversity of modern attacks pose significant challenges to traditional rule-based IDS, often resulting in inadequate adaptability and detection accuracy.

In response to these limitations, the fusion of machine learning techniques with IDS has emerged as a promising avenue for fortifying network security. This thesis focuses on exploring the integration of machine learning algorithms within IDS frameworks to enhance their capabilities. By harnessing the power of machine learning, these systems can autonomously learn and adapt to evolving threats, enabling the identification of anomalies and malicious activities with higher accuracy and efficiency.

The primary objective of this thesis is to investigate and demonstrate the potential of machine learning-driven IDS in transforming the landscape of intrusion detection. Through the utilization of diverse machine learning models, such as neural networks, decision trees, or anomaly detection algorithms, the aim is to empower IDS to discern subtle patterns within network traffic, differentiate normal behaviour from anomalies, and effectively detect and respond to potential threats in real-time.

Expanding on this research, an additional focus will be given to evaluating the scalability of machine learning-integrated IDS solutions and their adaptability to diverse network environments. By addressing these aspects, the study aims to provide comprehensive insights into the practical applicability and potential challenges associated with deploying machine learning-driven IDS at scale.

This thesis endeavours to delve into the technical intricacies and practical implications of integrating machine learning into IDS, aiming to not only improve detection accuracy but also to minimize false positives and enhance the overall resilience of network security. By exploring this innovative amalgamation, the intent is to contribute to the advancement of intrusion detection systems, paving the way for more adaptive, efficient, and robust cybersecurity measures in today's digital landscape.

1.2 Objective

The primary objectives of this project revolve around revolutionizing intrusion detection systems (IDS) by integrating machine learning methodologies. The foremost goal is to elevate the accuracy and efficiency of threat detection within network environments. By employing machine learning algorithms, the aim is to enhance the IDS's capabilities, enabling it to swiftly identify and categorize potential threats with precision in real-time. Additionally, the project targets the development of an adaptive IDS, capable of autonomously learning and evolving alongside emerging cyber threats. This involves minimizing false positives by training the system to discern between normal network behaviour and genuine security threats accurately. Moreover, the project seeks to optimize response mechanisms, ensuring rapid and efficient actions upon threat detection. Comprehensive validation and performance testing are integral components, aiming to validate the effectiveness and reliability of the machine learning-enhanced IDS in diverse scenarios. The project also emphasizes documentation and analysis to facilitate a thorough understanding of the developed system's functionality and performance.

1.3 Scope

The scope of this project is centred on the integration of machine learning techniques into the realm of Intrusion Detection Systems (IDS) to fortify their capabilities within network security. Primarily, the project aims to focus on network traffic analysis using machine learning algorithms, such as neural networks, decision trees, or anomaly detection models. Specific emphasis will be placed on the implementation and optimization of these algorithms within the IDS framework to enhance threat detection accuracy and real-time monitoring. Additionally, the scope encompasses comprehensive validation and performance testing to assess the reliability and effectiveness of the machine learning-driven IDS across diverse simulated and real-world scenarios. While the project will delve into the integration of machine learning methodologies, it acknowledges resource limitations and operational constraints, considering available hardware, software, and expertise for development and implementation. Furthermore, the project scope explicitly defines exclusions, such as physical security aspects or certain attack types beyond the project's coverage. A vital aspect within this scope is thorough documentation, facilitating detailed analysis of the developed system's functionality, algorithms utilized, and performance metrics achieved. the project will explore avenues for scalability and interoperability to facilitate seamless integration with existing network infrastructure and security frameworks. Continuous training and skill development initiatives will also be implemented to empower team members with the latest tools and techniques relevant to the project's objectives.

1.4 Technical Approach

A structured process beginning with comprehensive data collection from network sources, followed by meticulous pre-processing to ensure data quality and relevance. Feature selection and engineering techniques are employed to identify and extract key attributes from the data, reducing complexity and enhancing model performance. Subsequently, a strategic selection of machine learning algorithms, such as neural networks or decision trees, is made based on the nature of the problem and data characteristics. These algorithms are then trained and optimized using labelled data to recognize patterns and anomalies in network traffic. Upon successful training, the models are integrated into the IDS framework for real-time monitoring, with mechanisms developed to process incoming data and trigger alerts upon detecting potential intrusions or anomalies. Rigorous validation and performance assessment, incorporating metrics like accuracy and false-positive rates, ensure the effectiveness of the system across diverse scenarios. The process involves iterative improvements, where feedback from validation results guides refinements in model tuning and system adaptation to evolving threats. Comprehensive documentation of methodologies, algorithms utilized, and performance results facilitates an in-depth understanding and further enhancements of the developed IDS.

1.5 Organization of Thesis

The thesis is structured into eight distinct chapters, each serving a crucial role in the comprehensive exploration of the research topic. Chapter 1 serves as the gateway to the study, setting the stage by introducing the research topic, outlining the objectives, defining the scope, detailing the technical approach, and providing an overview of the organization of the thesis. Following this introduction, Chapter 2 delves into the fundamental concepts of machine learning, offering a thorough discussion of both supervised and unsupervised learning methodologies, supplemented by a diverse array of real-world applications. In Chapter 3, a comprehensive literature survey is conducted, synthesizing existing research findings and insights relevant to the research topic, thereby providing a solid foundation for the subsequent chapters. Chapter 4 conducts a critical analysis of the existing system, pinpointing its limitations and proposing a novel system design, complete with an architectural framework to address the identified shortcomings. The exploration of machine learning algorithms takes centre stage in Chapter 5, where a detailed examination of algorithms such as Gaussian Naive Bayes, Decision Tree, Random Forest, Logistic Regression, and Gradient Classifier is conducted, alongside an elucidation of evaluation parameters crucial for assessing their performance. Chapter 6 meticulously details the dataset employed in the study, elucidating the data processing techniques, feature extraction methods, and classification methodologies utilized. Chapter 7 presents the culmination of the study's empirical endeavours, showcasing the experimental results obtained and providing insights into the efficacy of the proposed system. Finally, Chapter 8 serves as the conclusion and reflection of the thesis, offering a succinct summary of key findings, discussing their implications, proposing avenues for future research, and providing a comprehensive list of references cited throughout the document. Together, these chapters form a cohesive narrative that encapsulates the breadth and depth of the research endeavour, offering valuable contributions to the field.

CHAPTER 2

BACKGROUND CONCEPTS

2.1 Introduction

In the ever-evolving landscape of cybersecurity, the safeguarding of network systems against malicious intrusions stands as an ongoing challenge. Intrusion Detection Systems (IDS) play a pivotal role in fortifying network security by continuously monitoring and analysing network traffic for potential threats. Traditionally, IDS have relied on predefined rules and signatures, often struggling to adapt swiftly to emerging and sophisticated cyber threats. In response to this limitation, the integration of machine learning methodologies has emerged as a transformative approach in enhancing the capabilities of IDS. Machine learning, a subset of artificial intelligence, offers the promise of imbuing IDS with adaptive learning capabilities, allowing these systems to autonomously learn from data patterns, discern anomalies, and proactively identify potential security breaches in real-time. By leveraging the power of machine learning algorithms, such as neural networks, decision trees, or clustering techniques, IDS can evolve beyond static rule-based approaches, potentially revolutionizing the field of intrusion detection by enabling more accurate, dynamic, and efficient threat identification.

2.2 Background Concepts of Machine Learning

Machine learning, a subset of artificial intelligence (AI), encompasses a set of algorithms and techniques that enable computer systems to learn and improve from experience without explicit programming. At its core, machine learning revolves around the ability of algorithms to identify patterns, learn from data, and make data-driven predictions or decisions. Three primary categories define machine learning approaches: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training models on labelled data, where algorithms learn patterns and relationships between input and output variables. Unsupervised learning deals with unlabelled data, aiming to uncover hidden patterns or structures within datasets. Reinforcement learning focuses on an agent learning from interactions with an environment by receiving feedback in the form of rewards or penalties based on its actions. Key techniques within machine learning include regression, classification, clustering, and neural networks, each serving specific purposes in analysing and understanding data. The versatility and adaptability of machine learning techniques have found applications in diverse fields, including cybersecurity, where these methods play a crucial role in augmenting security measures, such as the development of advanced IDS capable of autonomously identifying and mitigating potential threats.

2.2.1 Supervised Machine Learning

In supervised learning, both input and output variables are provided. The algorithms are trained on labelled data, using the training dataset to understand the relationships between input and output variables. Consequently, the outputs derived from this method tend to be more accurate. This training data serves as the guiding force, essentially acting as the supervisor for the machine, aiding it in making precise predictions. The process of supervised learning is illustrated in Figure 2.1, depicting the operational flow of supervised machine learning.

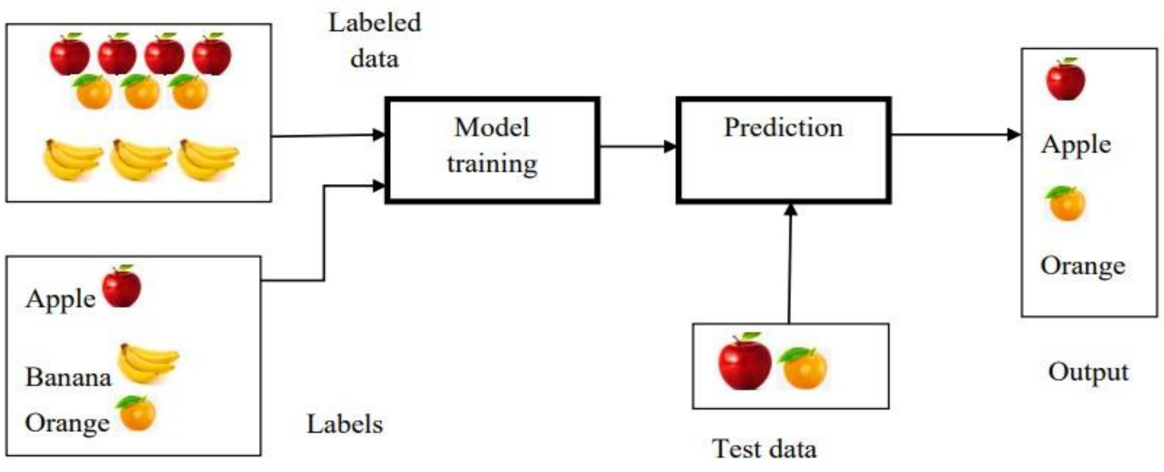


Figure 2.1 Working of supervised machine learning technique

Steps for Supervised Learning

Supervised learning involves a series of steps to train a model on labelled data, enabling it to make accurate predictions on new, unseen data. The process begins with data collection, where a dataset containing both input features and corresponding output labels is assembled. Subsequently, the data undergoes pre-processing to enhance its quality and relevance. The dataset is then split into training and testing subsets, with the training set used to educate the model. During the training phase, various machine learning algorithms, such as decision trees or neural networks, learn the underlying patterns and relationships between input features and output labels. Once the model is trained, it enters the prediction phase, where it utilizes these learned patterns to make predictions on new, unlabelled data. The accuracy and efficacy of the model are evaluated using the testing subset, and based on the evaluation results, the model may undergo refinement for improved performance. This iterative process ensures that the supervised learning model can generalize well to new, unseen data, making it a valuable tool for various predictive tasks. Continuous feedback loops will facilitate model adaptation and refinement over time, ensuring its relevance and effectiveness in dynamic environments. Additionally, efforts will be made to streamline the model's implementation and deployment process to ensure ease of use for end-users. Collaboration with stakeholders will provide valuable insights for refining the model and addressing any usability concerns. Overall, these iterative improvements will contribute to the model's effectiveness and usability in real-world applications.

Types of Supervised Learning

Supervised learning encompasses several types, each tailored to different prediction or classification tasks. One fundamental type is Classification**,** where the model learns to assign inputs to discrete categories or classes. For instance, email spam detection classifies emails as either spam or not spam based on features like content and sender. Another type is Regression, used for predicting continuous numerical outputs. This type is applied in scenarios such as predicting house prices based on features like area, location, and amenities. Additionally, there's Sequence Generation where the model generates sequences, like predicting a sentence completion in natural language processing. Moreover, Object Detection identifies and localizes objects within an image, crucial in computer vision applications like identifying cars in images. Lastly, Semantic Segmentation assigns class labels to each pixel in an image, facilitating tasks such as identifying different objects within an image.

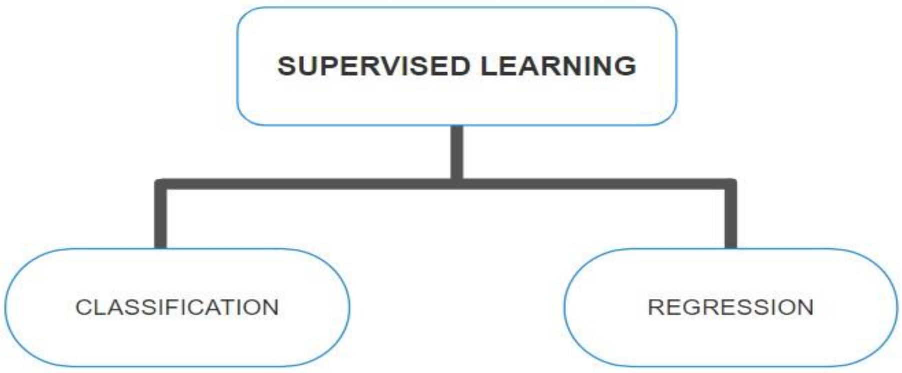


Figure 2.2 Type of Supervised Learning

These diverse types of supervised learning cater to a wide array of applications, from straightforward classifications to complex sequential predictions and spatial recognition tasks, illustrating the versatility and applicability of this learning paradigm.

Advantages of Supervised Learning

- Adaptability to Evolving Threats: Using historical data with clear labels, supervised learning models can continuously adapt to evolving attack patterns. This adaptability ensures that IDS remain effective in detecting new and emerging threats, improving overall network security.
- Reduced False Positives: By learning from labelled data, supervised learning models in IDS can minimize false alarms or false positives. This reduction in false alerts enables security teams to focus on genuine threats, optimizing response efforts, and resource allocation.

- Efficient Resource Utilization: The utilization of labelled data streamlines the learning process, enabling efficient utilization of resources. Supervised learning allows for the optimal use of available data, computing power, and expertise, enhancing the effectiveness of IDS.
- Interpretability and Understanding: Certain supervised learning models provide interpretability, enabling security experts to understand how the system makes decisions. This transparency aids in comprehending the reasoning behind threat identifications, facilitating informed actions and response strategies.

Disadvantages of Supervised Learning

- Dependency on Labelled Data: Supervised learning heavily relies on labelled data for training, and acquiring large volumes of accurately labelled data in cybersecurity, especially for rare or novel attacks, can be challenging. Limited or biased labelled datasets may result in incomplete learning and detection of new threats.
- Inability to Detect Novel Attacks: Supervised learning models primarily recognize patterns from known labelled data. As a result, they might struggle to identify entirely new or previously unseen attacks, as they lack prior knowledge or labelled examples of such threats.
- Overfitting and Generalization Issues: There's a risk of overfitting, where the model memorizes noise or specific characteristics of the training data, reducing its ability to generalize well to new, unseen data. This can result in false positives or poor performance on real-world data.
- Difficulty in Interpretation: Some advanced supervised learning models, like deep neural networks, lack interpretability, making it challenging for security analysts to comprehend the rationale behind the model's decisions, hindering their trust and understanding of the system.

2.2.2 Unsupervised Machine Learning

The absence of labelled data challenges models to uncover hidden patterns autonomously. Here, models process unlabelled data without explicit guidance, aiming to extract insights independently. This approach aligns closely with human-like learning, allowing models to draw conclusions based on experience rather than predefined labels. It operates on uncategorized, unlabelled data, making it immensely valuable. Figure 2.3 illustrates the workings of unsupervised machine learning. The input, unlabelled data lacks specific categorization for its respective outputs. This unannotated data undergoes training within an appropriate machine learning model.

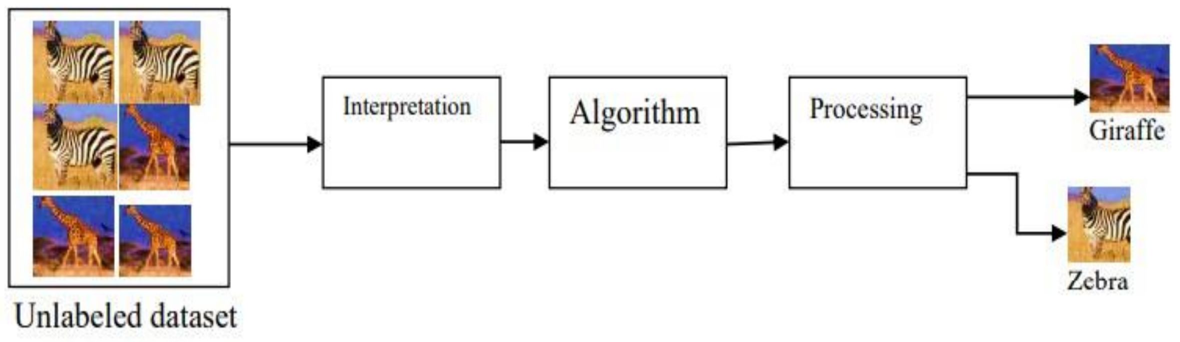


Figure 2.3 Working of unsupervised machine learning technique

Initially, the model discerns concealed patterns within the raw data, followed by the application of suitable algorithms. Subsequently, the model groups data based on their similarities and dissimilarities, establishing a method to interpret and structure unlabelled information.

Types of Unsupervised Learning Algorithms

Unsupervised learning encompasses diverse algorithms designed to explore and extract patterns or structures from unlabelled data. Clustering algorithms stand as a fundamental type, segregating data points into groups based on similarities, allowing for the discovery of natural groupings within datasets. Dimensionality reduction techniques like Principal Component Analysis (PCA) or distributed Stochastic Neighbour Embedding (t-SNE) aid in simplifying complex data by condensing it into a lower-dimensional space while retaining essential information. Anomaly detection algorithms focus on identifying outliers or irregularities within datasets, highlighting instances significantly deviating from the norm. Additionally, association rule learning algorithms uncover relationships or associations between variables, revealing co-occurrences or correlations within the data. Each type serves distinct purposes; clustering aids in uncovering natural groupings, dimensionality reduction simplifies data representation, anomaly detection flags unusual instances, and association rule learning unveils interesting connections between variables.

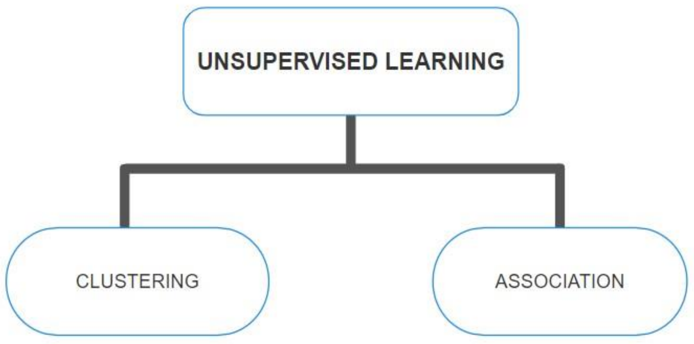


Figure 2.4 Types of Unsupervised Learning

These algorithms collectively enable unsupervised learning to derive valuable insights, identify patterns, and uncover hidden structures within unlabelled data across various domains and applications.

Advantages of Unsupervised Learning

- Anomaly Detection: Unsupervised learning excels in anomaly detection by identifying irregular patterns or outliers within network traffic without relying on labelled data. This capability allows the system to detect novel threats or unusual behaviours that might go unnoticed in labelled datasets.
- Adaptability to New Threats: Since unsupervised learning operates without labelled examples, it can adapt to new and emerging threats that lack predefined signatures. This adaptability is crucial in detecting previously unseen attack patterns or zero-day attacks.
- Discovering Unknown Patterns: Unsupervised learning algorithms can unveil hidden patterns or trends within network traffic that might not be explicitly defined in labelled datasets. This capability aids in understanding evolving attack methodologies that might not conform to known patterns.
- Reducing False Positives: By comprehensively analysing the entire dataset and identifying deviations from normal behaviour, unsupervised learning helps in reducing false alarms or false positives, focusing attention on genuine anomalies.
- Scalability and Flexibility: Unsupervised learning techniques can scale effectively to large and diverse datasets, offering flexibility in handling varied and complex network environments without the need for labelled data, which might be scarce or limited.

Disadvantages of Unsupervised Learning

- Difficulty in Interpreting Results: Unsupervised learning often lacks explicit labels or guidance, making it challenging to interpret and validate the findings. Understanding the rationale behind anomalies or patterns identified by unsupervised algorithms can be complex, requiring expert analysis.
- Inability to Detect Specific Known Threats: Unlike supervised learning, unsupervised methods might struggle to detect specific known threats that are well-defined with distinct patterns. These algorithms may not effectively identify known attack signatures without prior labelled examples.
- Complexity and Computational Resources: Certain unsupervised learning algorithms, especially those dealing with large-scale data, can be computationally intensive and complex, demanding substantial resources in terms of computational power and time.
- Limited Guided Learning: Unsupervised learning lacks the explicit guidance provided by labelled data, which could limit its ability to focus on specific types of threats or attack patterns, especially those with distinct signatures.

2.3 Applications

Machine learning's applications within Intrusion Detection Systems (IDS) are instrumental in fortifying cybersecurity measures. These applications encompass a spectrum of techniques designed to detect, prevent, and respond to potential cyber threats effectively. Anomaly detection stands as a cornerstone, leveraging machine learning models to distinguish between normal network behaviour and suspicious activities, swiftly flagging anomalies that deviate from established patterns. Signature-based detection methods utilize predefined signatures of known attacks, enabling the identification and mitigation of recognized threats, providing a robust defence mechanism against well-established attack patterns. Behavioural analysis facilitated by machine learning enables the detection of subtle deviations in network behaviour, allowing systems to identify potential security breaches. Real-time threat response, empowered by machine learning algorithms, enables IDS to respond promptly upon detecting threats, minimizing potential damages caused by cyber intrusions.

CHAPTER 3

LITERATURE SURVEY

[1] A Novel Network Intrusion Detection System Based on CNN.

Network Intrusion Detection Systems (NIDS) have traditionally relied on machine learning techniques like Support Vector Machines, Bayesian Classification, and k-Means clustering for enhancing network security. However, these methods often require manual feature selection and are constrained by inherent limitations. To overcome these challenges, this study introduces a novel NIDS framework leveraging Convolutional Neural Networks (CNNs). Unlike conventional methods, CNNs autonomously extract discriminative features directly from raw network traffic data, enabling the development of sophisticated detection models capable of discerning subtle intrusion patterns. Comprehensive experiments conducted on established benchmark datasets validate the superiority of the CNN-based NIDS framework. Notably, models trained on raw traffic data consistently demonstrate higher accuracy compared to those trained solely on extracted features, underscoring the intrinsic advantage of leveraging raw data in deep-learning-based NIDS models. This research represents a significant advancement in network security, offering enhanced detection capabilities while mitigating the limitations associated with traditional machine learning techniques.

[2] Dynamic detection of malicious intrusion in wireless network based on improved random forest To enhance the detection capability of malicious nonlinear scrambling intrusions in wireless networks and ensure network security, this paper proposes a method based on an improved random forest algorithm. The method begins by constructing a model for the malicious nonlinear scrambling intrusion signal in wireless networks. Subsequently, it employs a terminal equipment's physical layer characteristic detection method to decompose the signal's characteristics. Through the analysis of test statistics and detection thresholds, spectrum characteristic quantities of the intrusion signal are extracted. Next, an improved random forest algorithm is utilized for signal characteristic reorganization and fuzzy clustering analysis. By leveraging the clustering distribution of spectral features, the method optimizes the detection of malicious nonlinear scrambling intrusions under random forest learning. This is achieved through a combination of reinforcement learning decision techniques and static feature fusion methods. Simulation results demonstrate the effectiveness of the proposed approach, showing strong feature clustering in detecting malicious nonlinear scrambling intrusions, leading to high accuracy in identifying intrusion information.

[3] Naive Bayes modification for intrusion detection system classification with zero probability.

One approach commonly employed in intrusion detection systems is the implementation of the Naïve Bayes algorithm. However, Naïve Bayes encounters a significant challenge when encountering probabilities of zero, leading to inaccurate predictions or even failure to produce any predictions at all. To address this limitation, this paper proposes two modifications to the Naïve Bayes algorithm. The first modification involves eliminating variables associated with zero probabilities, while the second modification replaces multiplication operations with addition operations. These modifications are selectively applied only when the Naïve Bayes algorithm fails to produce predictions due to zero probabilities. The results of this research demonstrate that the proposed modifications yield improvements in precision, recall, and accuracy compared to the original Naïve Bayes algorithm. Specifically, the modification involving addition operations achieves the highest precision, recall, and accuracy. Precision increases by up to 4%, recall by up to 2%, and accuracy by up to 2%, highlighting the efficacy of the proposed modifications in enhancing the performance of the Naïve Bayes algorithm for intrusion detection.

[4] An effective intrusion detection approach using SVM with naïve Bayes feature embedding.

Over the past decades, the increasing significance of network security has underscored the pivotal role of intrusion detection systems (IDS) in safeguarding networks. While machine learning techniques have been extensively applied to intrusion detection, particularly Support Vector Machine (SVM) has garnered recognition for its effectiveness. Nonetheless, a notable gap exists in current studies, as they often overlook the crucial aspect of data quality, which is fundamental for constructing a robust intrusion detection system beyond machine learning algorithms. To address this gap, this paper introduces an innovative intrusion detection framework that combines SVM with naïve Bayes feature embedding. Specifically, the framework utilizes the naïve Bayes feature transformation technique to enhance the quality of the original features, thereby generating new data of higher quality. Subsequently, an SVM classifier is trained on the transformed data to construct the intrusion detection model. Experimental evaluations conducted on multiple datasets within the intrusion detection domain validate the efficacy and robustness of the proposed detection method. Notably, the results demonstrate promising performance metrics, achieving an accuracy of 93.75% on the UNSW-NB15 dataset, 98.92% on the CICIDS2017 dataset, 99.35% on the NSL-KDD dataset, and 98.58% on the Kyoto 2006+ dataset. Furthermore, comparative analysis reveals substantial advantages of our method in terms of accuracy, detection rate, and false alarm rate when juxtaposed with other state-of-the-art methods. Overall, our proposed framework represents a significant advancement in intrusion detection, offering improved performance and reliability by incorporating data quality considerations alongside machine learning techniques.

[5] A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM.

Pattern matching methods, particularly signature-based techniques, have long been utilized in basic network intrusion detection systems (IDS). However, these methods may struggle to effectively detect uncommon attacks, such as Remote2Local (R2L) and User2Root (U2R), due to significant variations in attack patterns.

To address this challenge, a more robust approach involves employing machine learning classifiers to detect anomalies and unseen attacks. Nevertheless, relying solely on a single classifier may not accurately identify all types of attacks. In response, hybrid approaches, which combine multiple classifiers, offer a promising solution. In this paper, we introduce a novel Double-Layered Hybrid Approach (DLHA) explicitly designed to tackle this issue. Through our research, we identified common characteristics among different attack categories using Principal Component Analysis (PCA) variables, which maximize variance from each attack type. Interestingly, we discovered that R2L and U2R attacks exhibit behaviour similar to that of normal users. DLHA employs a Naive Bayes classifier as Layer 1 to detect Denial of Service (DoS) and Probe attacks, while utilizing Support Vector Machine (SVM) as Layer 2 to differentiate R2L and U2R attacks from normal instances. We conducted experiments using the NSL-KDD dataset and compared our approach with other state-of-the-art IDS techniques. The results demonstrate that DLHA surpasses existing methods by a significant margin and outperforms any single machine learning classifier. Moreover, DLHA exhibits outstanding performance in detecting rare attacks, achieving detection rates of 96.67% and 100% for R2L and U2R attacks, respectively. These findings underscore the effectiveness and superiority of DLHA in enhancing network intrusion detection capabilities.

[6] A Survey on Machine Learning Techniques for Cyber Security in the Last Decade.

The pervasive expansion of the Internet and mobile applications has significantly widened cyberspace, rendering it more susceptible to automated and prolonged cyberattacks. In response, cyber security techniques have evolved to bolster security measures, aiming to detect and respond to these threats effectively. However, traditional security systems have proven inadequate, as cybercriminals continually outsmart conventional defences. They employ tactics that evade detection, posing challenges in combating previously unseen and polymorphic security attacks. Machine learning (ML) techniques have emerged as indispensable tools in addressing cyber security challenges, finding applications across various domains. Despite their successes, ensuring the reliability of ML systems remains a significant hurdle. Malicious actors in cyberspace are motivated to exploit vulnerabilities in ML systems for their gain, thereby threatening the integrity of these defences. This paper seeks to offer a comprehensive examination of the challenges faced by ML techniques in safeguarding cyberspace against attacks. It accomplishes this by reviewing literature on ML applications in cyber security, encompassing intrusion detection, spam detection, and malware detection across computer and mobile networks over the past decade. Additionally, it provides succinct descriptions of each ML method, commonly used security datasets, essential ML tools, and evaluation metrics for assessing classification models. Moreover, the paper deliberates on the inherent challenges associated with integrating ML techniques into cyber security practices. By compiling the latest extensive bibliography and highlighting current trends in ML for cyber security, this paper serves as a valuable resource for researchers and practitioners navigating the complex landscape of cyber defence.

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 Existing System

Ensuring network and system security remains critical in today's data communication landscape. Hackers and intruders continuously devise numerous successful strategies to disrupt networks and web services through unauthorized access. As new threats emerge alongside advancements in secure systems, the need for associated preventive measures becomes increasingly apparent. Intrusion Detection Systems (IDS) serve as a crucial solution within this context. Their primary role is to safeguard resources from potential threats by analysing and forecasting user behaviours, distinguishing between what constitutes an attack and normal behaviour.

In addressing this imperative, researchers proposed an innovative intrusion detection method that leverages a Support Vector Machine (SVM) within a system based on Rough Set Theory (RST). The objective of this approach is to streamline the initial 41 features down to 29, utilizing RST as a pre-processing step to discern the most relevant features for intrusion detection. The SVM model is then employed to learn from these reduced features, aiming to enhance the system's efficiency in accurately detecting network intrusions while minimizing the feature space's dimensionality.

Rough Set Theory serves as a crucial pre-processing step, effectively reducing data dimensions by discerning crucial features from network packets. These selected features are then forwarded to the SVM model. SVM, known for its robust machine learning capabilities, learns from these features to differentiate between normal and anomalous network behaviour.

This combined approach not only effectively decreases data density but also enables the SVM model to learn and discern patterns indicative of potential intrusions. By leveraging RST's feature reduction and SVM's classification abilities, this IDS aims to accurately detect and mitigate network-based threats, significantly enhancing the system's efficiency in identifying anomalies and potential attacks. The integration of these advanced techniques showcases a proactive stance in enhancing network security, contributing to the ongoing efforts to stay ahead of evolving cyber threats. this integrated approach seamlessly combines the feature discernment power of Rough Set Theory with the robust classification capabilities of Support Vector Machines, providing a sophisticated and efficient means to bolster network security against evolving cyber threats.

4.1.1. Disadvantages of Existing System

- Integrating RST and SVM into an IDS can be intricate and require expertise. Configuring the system, selecting appropriate parameters, and optimizing the algorithms might pose challenges.
- RST-based reduction might struggle to adapt quickly to evolving attack strategies or new threat patterns not present in the training data, potentially affecting the system's ability to detect novel intrusions.
- SVM models, particularly with large datasets, might demand significant computational resources and time for training, affecting real-time intrusion detection capabilities.

4.2 Problem Statement

Intrusion detection steps in when firewalls can no longer prevent unauthorized access entirely. While preventing access is ideal, it's not always achievable. Hence, the system's reliability, accuracy, and security become paramount. Intrusion detection involves continuously monitoring and scrutinizing network activities and data in real-time, aiming to uncover vulnerabilities and ongoing attacks.

4.3 Proposed System

Our idea for a smart Intrusion Detection System (IDS) aims to protect networks from outside threats. The current IDS systems sometimes struggle to spot new kinds of attacks and can be slow when dealing with certain data. So, we're using smart computer methods called Gaussian Naive Bayes, Decision Trees, Logistic Regression, Random Forest, and Gradient Classifier. These methods are good at recognizing patterns and spotting intrusions even when we don't have a lot of examples or when the data is really complicated. They're like powerful tools helping us find and stop attacks on computer networks.

The proposed system for an Intrusion Detection System (IDS) leverages the efficiency and adaptability of the Random Forest algorithm within the domain of machine learning. The primary objective is to develop a robust IDS that effectively identifies and categorizes network intrusions by harnessing the ensemble learning capabilities of Random Forest. This system aims to address the complexities of modern cyber threats by analysing diverse network traffic data, distinguishing normal behaviour from potential intrusions with higher accuracy. The Random Forest algorithm stands out as a prime candidate due to its ability to handle high-dimensional datasets, manage noisy data, and exhibit robust performance in classification tasks. The envisioned IDS involves a comprehensive approach, encompassing data pre-processing, feature engineering, model development, and real-time implementation. Through meticulous model training, parameter tuning, and continuous monitoring, the system aims to create a dependable and adaptable defence mechanism capable of detecting and mitigating various intrusion attempts in dynamic network environments. Ultimately, the integration of Random Forest into this IDS aims to enhance detection precision, minimize false alarms, and fortify network security against evolving cyber threats. The research endeavours to explore the efficacy of the Random Forest algorithm in handling high-dimensional network data and adaptively learning from diverse intrusion scenarios. Additionally, the study aims to optimize the IDS for scalability, computational efficiency, and Realtime detection in dynamic network environments.

4.3.1 Advantages of Proposed System

- Robustness to Complexity: Random Forest handles intricate and complicated data effectively. It copes well with diverse and high-dimensional network traffic data, essential in modern cyber threat analysis.
- Adaptability: The algorithm's ensemble learning capabilities enable it to adapt to various intrusion scenarios. It can distinguish between normal behaviour and potential intrusions with higher accuracy, even with limited examples.
- Noise Handling: Random Forest is adept at managing noisy data, crucial in real-world network environments where data can be unreliable or incomplete.
- High Performance: It exhibits robust performance in classification tasks, which is vital for an IDS to accurately identify and categorize network intrusions.
- Precision and Minimized False Alarms: By leveraging Random Forest, the IDS aims to enhance detection precision while reducing false alarms, improving overall system reliability.
- Scalability and Efficiency: The envisioned IDS intends to optimize the Random Forest algorithm for scalability, computational efficiency, and real-time detection.

4.4 System Architecture

Intrusion Detection Systems (IDS) utilize a multi-tiered approach, beginning with the collection of data from diverse network devices and hosts. This data is then subjected to pre-processing, where it undergoes filtering and normalization to streamline subsequent analysis. Following preprocessing, feature extraction and selection techniques are employed to identify the most relevant attributes for intrusion detection. These attributes serve as the basis for the detection engines, which utilize both signature-based and anomaly-based methods to identify potential threats. Signature-based detection involves comparing extracted features against known patterns of malicious activity, while anomaly-based detection identifies deviations from normal behaviour within the network. By incorporating these methods, IDS can effectively detect and respond to a wide range of security threats, safeguarding network integrity and protecting against potential breaches.

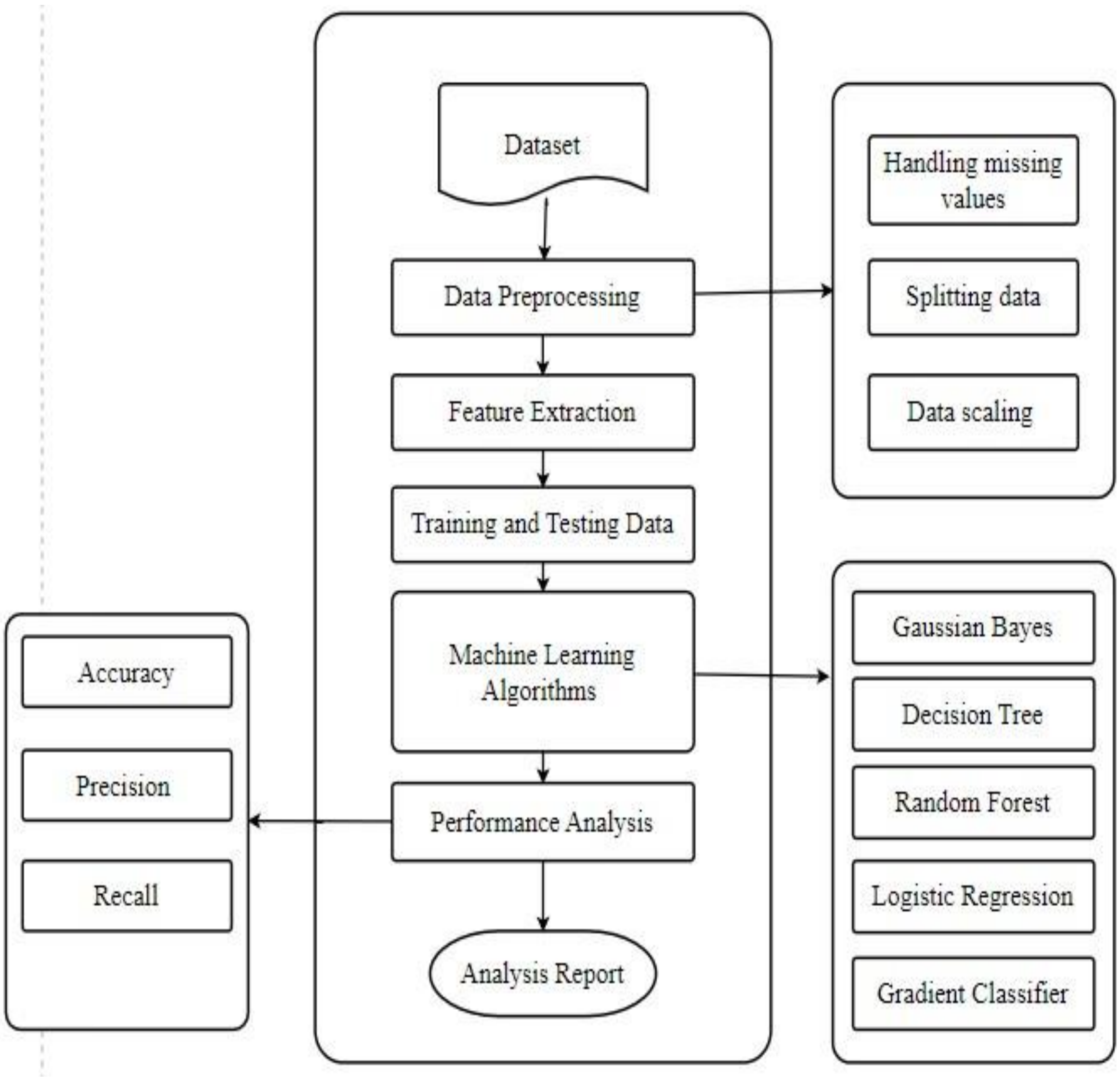


Figure 4.1 System Architecture

These engines analyse patterns and deviations, triggering alerts upon detecting potential intrusions. Logging and reporting mechanisms document activities, aiding in audits and incident investigations. Continuous adaptation is ensured through feedback loops and periodic updates, integrating the IDS within the broader security ecosystem for comprehensive threat management and coordination with other security measures like firewalls and SIEM systems. This interconnected architecture aims to swiftly detect, respond to, and mitigate various intrusions within network environments.

CHAPTER 5

ALGORITHMS AND TECHNIQUES

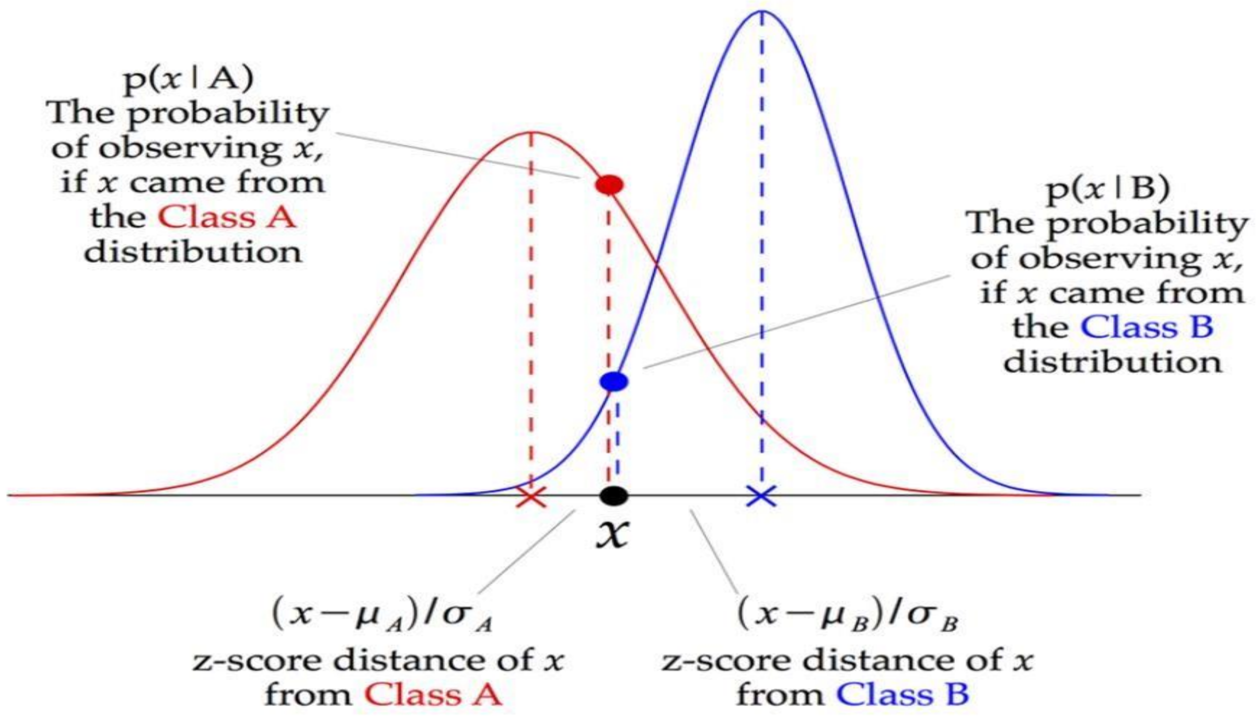
5.1 Machine Learning Algorithms

Machine learning's significance has surged due to its capacity to enhance operational efficiency, driven by rapid advancements in computer technology that have facilitated the proliferation of distributed data managed and processed globally by users. The true value of this data emerges when transformed and analysed, generating insights and predictive models crucial for intelligent decision-making. Machine learning algorithms are categorized into two principal types: supervised and unsupervised. Supervised learning focuses on understanding the influence of one variable set on another, designated as input and output variables, respectively. In this learning paradigm, the primary goal is to decipher the unknown function f(p) = q, where 'p' signifies input and 'q' represents the desired output. Within supervised learning, both 'p' and 'q' are provided, aiming to unveil the function responsible for translating 'p' into 'q' and the relationship between these variable sets. Consider a disease diagnosis scenario where the training set comprises positive and negative patient reports. Here, the task is to devise an algorithm that accurately discerns whether a patient tests positive or negative. The supervised learning technique necessitates labelled instances within the training set, represented as {(p1, q1), (p2, q2), (p3, q3) ... (pn, qn)}, each 'pi' denoting either a positive or negative result. It entails identifying instances related to a concept from given examples. For instance, distinguishing whether a specified example is an instance or not using supervised machine learning. In an instance, if confirmed positive, it's labelled as 1; otherwise, it's negative. An example includes learning a disease class based on patients' medical reports, categorizing them as positive or negative. Each patient is characterized by an ordered pair (p, L), where 'L' represents the label (1 for positive, else negative), and 'p' comprises symptoms' details (cough, headache, fever, breathlessness). The training set consists of 'N' patient reports, forming ordered pairs represented as P= {(pk, Lk), 1≤ k≤ N}. 'pk' signifies symptoms of the 'k-th' patient, visualized as data points in three-dimensional space alongside their respective labels. Within this realm of supervised learning, numerous techniques exist, with the thesis implementing four: support vector regression, linear regression, least absolute shrinkage selection operator, and exponential smoothing. The advantages of machine learning algorithms encompass adaptability, robustness, scalability, interpretability, continuous learning, and automation, making them indispensable tools across a wide array of applications in today's data-driven world. This approach facilitates the development of accurate predictive models, essential for intelligent decision-making in various domains.

5.1.1 Gaussian Naive Bayes Algorithm

The Gaussian Naive Bayes algorithm represents a cornerstone in the field of classification, embodying both simplicity and robustness in its approach. As an extension of the Naive Bayes classifier, it operates on the principles of Bayes' theorem and statistical probability, showcasing exceptional proficiency in managing continuous data characterized by features assumed to conform to a Gaussian distribution within each class. Central to its methodology is the assumption of feature independence, where each feature is considered to contribute independently to the probability of an outcome. While this assumption may not always hold true in intricate real-world scenarios, the algorithm's elegance lies in its ability to navigate this complexity with grace.

Gaussian Naive Bayes harnesses Gaussian probability density functions to ascertain the likelihood of observing a specific feature value given a class. This involves estimating the mean and variance for each feature within every class present in the training dataset. Subsequently, predictions are formulated by amalgamating these likelihood probabilities with prior probabilities through the application of Bayes' theorem. Despite its seemingly simplistic nature and dependence on assumptions like feature independence and adherence to Gaussian distribution, the algorithm demonstrates remarkable efficacy.



$p(x \mid A)$
The probability of observing x, if x came from the Class A distribution
$p(x \mid B)$
The probability of observing x, if x came from the Class B distribution
x
$(x-\mu_A)/\sigma_A$
z-score distance of x from Class A
$(x-\mu_B)/\sigma_B$
z-score distance of x from Class B

Figure 5.1 Working of GNB Algorithm

Particularly noteworthy is its performance with moderate-sized datasets and continuous features that exhibit Gaussian-like properties. This versatility and efficiency make it a stalwart choice across a spectrum of classification tasks, offering a harmonious equilibrium between simplicity and predictive prowess. In practical applications, Gaussian Naive Bayes emerges as a dependable ally, capable of delivering reliable results while navigating the complexities inherent in real-world datasets.

Working steps:

Probability Calculation:

Gaussian Distribution: Assumes that continuous features follow a Gaussian (normal) distribution within each class.

Mean and Variance: Computes the mean and variance of each feature for each class in the training dataset.

Probability Estimation:

Likelihood Estimation: Calculates the likelihood probability of observing a specific feature value given the class.

$$P\times e \qquad (x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \qquad - \frac{(xi-\mu y)2}{2\sigma y2}$$

Where x_i is the feature value, μ_y is the mean, σ_y^2 is the variance of the ith feature in class y.

Class Prediction:

Bayes' Theorem: Combines likelihood probabilities with prior probabilities (the probability of each class occurring).

$$P(y|x_1,x_2,\ldots,x_n) = \frac{P(y) \times \prod_{i=1}^{n} P(x_i|y)}{P(x_1,x_2,\ldots,x_n)}$$

Class Prediction: Assigns the class label with the highest posterior probability given the input features.

Advantages:

- Simplicity and Speed: The algorithm is straightforward and computationally efficient, making it suitable for large datasets.
- Effective with Continuous Data: Particularly effective in handling continuous features with Gaussian (normal) distribution assumptions within classes.
- Minimal Tuning Required: Requires minimal hyper parameter tuning compared to more complex algorithms.

Limitations:

- Assumption of Feature Independence: Its assumption that features are independent might not hold in real-world scenarios, impacting predictive accuracy.
- Cannot Capture Feature Interactions: Lacks the capacity to capture interactions between features since it assumes independence.
- Requires Gaussian Distribution: Performs best when the continuous features indeed follow a Gaussian distribution within each class, which might not always be the case in practice.

5.1.2 Decision Tree Algorithm

The decision tree serves as a prominent supervised learning method, particularly in classification scenarios where both categorical and continuous input and output variables are considered. This technique involves the segmentation of samples into homogeneous subsets, relying on the most significant distinguishing feature among input variables. Within the decision tree structure, internal nodes represent attribute tests, branches indicate outcomes, and leaves signify decisions reached after attribute computations.

The primary goal of employing a decision tree is to build a training model capable of predicting the class or value of target variables by assimilating decision rules derived from historical data, also known as training data. The decision tree algorithm is notably accessible compared to other classification algorithms, operating through a tree representation where each internal node corresponds to an attribute, and each leaf node represents a distinct class label. Operating recursively, the Decision Tree algorithm partitions the dataset into smaller and more homogeneous subsets based on the values of input features. This results in a tree-like structure where each internal node represents a feature, each branch signifies a decision rule, and each leaf node denotes the outcome or class label. This intuitive approach makes decision trees a valuable tool for understanding and interpreting classification processes.

This recursive partitioning of the dataset continues until a predefined stopping criterion is met, such as a certain depth of the tree or a minimum number of samples in each leaf. Decision trees have the advantage of providing clear and interpretable decision paths, allowing users to comprehend the logic behind the model's predictions. Moreover, decision trees are adept at handling both numerical and categorical data, making them versatile for various applications.

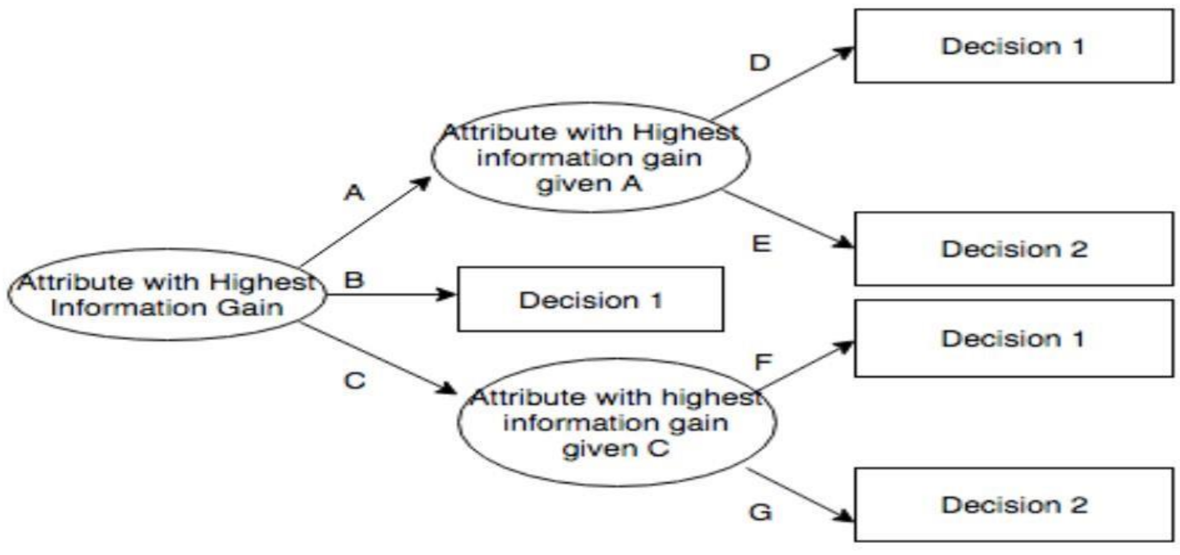

Attribute with Highest Information Gain
A
B
C
Attribute with Highest information gain given A
Decision 1
Attribute with highest information gain given C
D
E
F
G
Decision 1
Decision 2
Decision 1
Decision 2

Figure 5.2 Working of Decision Tree Algorithm

Working Steps:

- Selecting the Best Split: The algorithm identifies the most discriminative feature that best separates the data into distinct classes or categories. It evaluates various splitting criteria (e.g., Gini impurity, entropy) to determine the best feature to split upon.
- Creating Nodes: The chosen feature becomes the root node. The dataset is split into subsets based on the values of this feature. For categorical features, the tree branches to separate categories, while for continuous features, it chooses a threshold to create binary splits.
- Recursively Growing the Tree: The process continues recursively for each subset, aiming to create more homogeneous subsets. At each node, the algorithm selects the best feature to split the data, and this process continues until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).
- Stopping Criteria: The recursive splitting stops when a specified condition is reached, preventing overfitting. This could be a minimum number of samples per leaf, a maximum depth of the tree, or other pre-defined criteria.
- Leaf Node Assignment: As the tree grows, each leaf node eventually represents a class label or an outcome based on majority voting (for classification) or averaging (for regression) of the instances in that subset.
- Predictions: During prediction, new instances traverse the tree from the root node following the decision rules (feature tests) until reaching a leaf node, where a class label or outcome is assigned.

Advantages:

- Easily interpretable and visualizable.
- Handles both numerical and categorical data.
- Requires minimal data pre-processing.

Limitations:

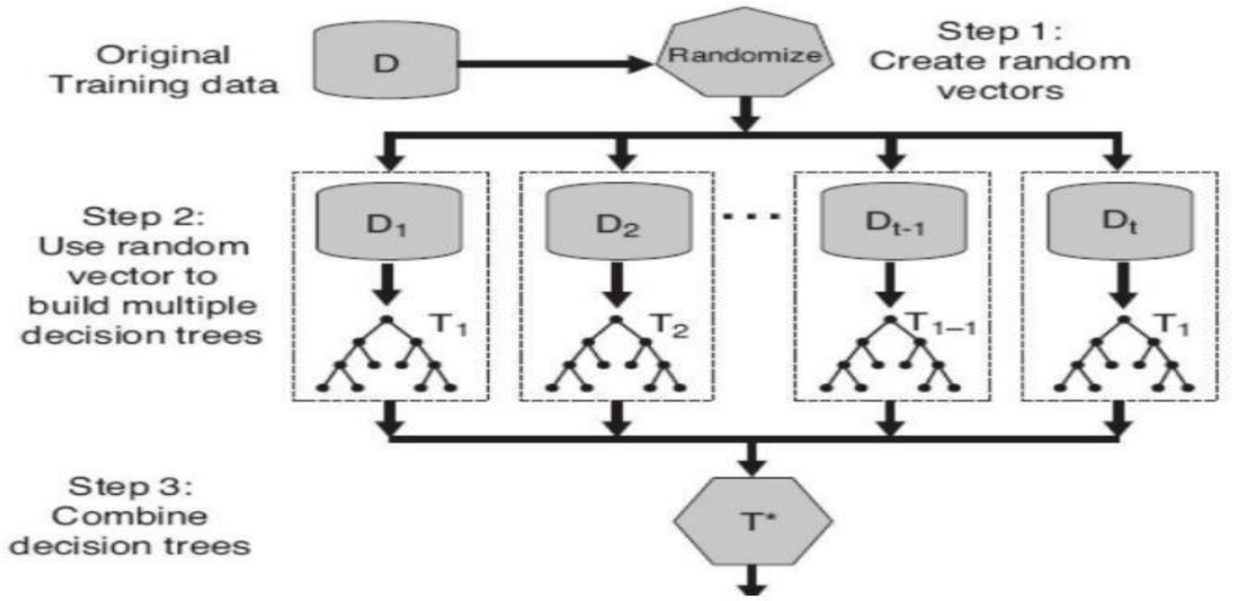
- Prone to overfitting, especially when the tree grows deep.
- Sensitive to noisy data and small variations.
- Single decision trees might not capture complex relationships present in the data.

5.1.3 Random Forest Algorithm

Random Forest stands as a versatile and influential ensemble learning method that finds widespread application in both classification and regression tasks within the expansive domain of machine learning. At its core, Random Forest operates by assembling multiple decision trees during the training phase and subsequently aggregating their outputs to reach a final prediction. This method, renowned for its robustness and effectiveness, orchestrates a symphony of random sampling and feature selection techniques, crafting an ensemble of diverse trees that collectively yield more accurate and stable predictions than any individual tree alone could achieve. The beauty of Random Forest lies in its ability to harness the wisdom of crowds, leveraging the collective insights gleaned from a multitude of decision trees to produce a final prediction that transcends the limitations of any single model. Through the process of bootstrapping and feature randomization, each tree within the forest is trained on a unique subset of the training data and a random subset of features, ensuring diversity in the trees' perspectives and enhancing the model's resilience to noise and outliers. This ensemble approach not only enhances predictive accuracy but also serves as a powerful defence mechanism against overfitting, a common pitfall in complex modelling tasks.

Random Forest's adaptability and versatility make it a stalwart choice across a myriad of domains and applications. From finance to healthcare, from marketing to ecology, Random Forest has demonstrated its efficacy in tackling a wide array of challenges posed by complex datasets and diverse problem domains. Its innate ability to handle high-dimensional data, nonlinear relationships, and missing values further solidifies its status as a go-to solution for data scientists and practitioners seeking reliable and scalable machine learning solutions.

Random Forest stands as a testament to the ingenuity of ensemble learning, embodying the collective wisdom of decision trees while simultaneously transcending their individual limitations. Its unparalleled robustness, effectiveness, and versatility have cemented its position as a cornerstone of modern machine learning, empowering researchers and practitioners alike to unlock new insights and drive innovation across a multitude of disciplines.



Original Training data
D
Randomize
Step 1: Create random vectors
Step 2: Use random vector to build multiple decision trees
D_1
D_2
D_{t-1}
D_t
T_1
T_2
T_{1-1}
T_1
Step 3: Combine decision trees
T^*

Figure 5.3 Working of Random Forest Algorithm

Working steps:

- Random Sampling: Random Forest selects random samples from the dataset with replacement (bootstrap samples), forming multiple subsets for each tree.
- Feature Randomness: For each tree in the forest, at each node, a subset of features is considered for splitting, introducing randomness and diversity among trees.
- Decision Tree Building: Multiple decision trees are grown independently based on these random subsets of data. Each tree is constructed using a subset of samples and features.
- Voting (Classification) or Averaging (Regression): During prediction: For classification tasks, the mode (most frequent class) among all individual tree predictions is taken as the final prediction.
- For regression tasks, the average of predictions from all trees is computed as the final prediction.

Advantages:

- Robustness: Resistant to overfitting due to ensemble averaging.
- Feature Importance: Provides a measure of feature importance, indicating the contribution of each feature in the prediction.
- Handling Missing Values: Can handle missing values in the dataset effectively.

Limitations:

- Less Interpretability: Random Forests can be challenging to interpret compared to a single decision tree.
- Resource Intensive: Constructing multiple trees can demand more computational resources.

5.1.4 Logistic Regression Algorithm

Logistic Regression stands as a cornerstone within the domain of supervised learning, revered for its remarkable performance in binary classification tasks. Despite its misleading name, this algorithm deviates from regression problems, instead focusing on estimating the probability of an instance belonging to a specific class. Its effectiveness stems from its ability to model the relationship between independent features and the probability of an outcome, achieved through the application of a sigmoid function. This transformation of a linear combination of features into a bounded value between 0 and 1 endows Logistic Regression with the capability to offer valuable insights into classification probabilities, thereby facilitating informed decision-making.

The algorithm's simplicity, interpretability, and efficacy in discerning the likelihood of a binary outcome make it a ubiquitous presence across various domains, simplifying prediction endeavours. Its capacity to provide probabilistic outcomes and its adaptability across diverse fields underscore its significance as a foundational choice in predictive modelling and decision support systems. Furthermore, Logistic Regression's versatility, simplicity, and interpretability further solidify its indispensability for understanding and predicting binary outcomes across numerous real-world applications.

The interpretability aspect of Logistic Regression enhances its value, allowing stakeholders to grasp the underlying factors influencing predictions, thereby fostering trust and confidence in the model's outputs. Consequently, Logistic Regression emerges not only as a potent predictive tool but also as a transparent and reliable means of unravelling complex data patterns, essential for informed decision-making across a wide spectrum of industries and applications. Ongoing advancements in regularization techniques and optimization algorithms continue to refine Logistic Regression's capabilities, reinforcing its position as a cornerstone of machine learning methodology. In essence, Logistic Regression embodies a harmonious blend of simplicity, interpretability, and effectiveness, making it an indispensable tool for tackling binary classification tasks and supporting data-driven decision-making processes in today's complex and evolving landscape.

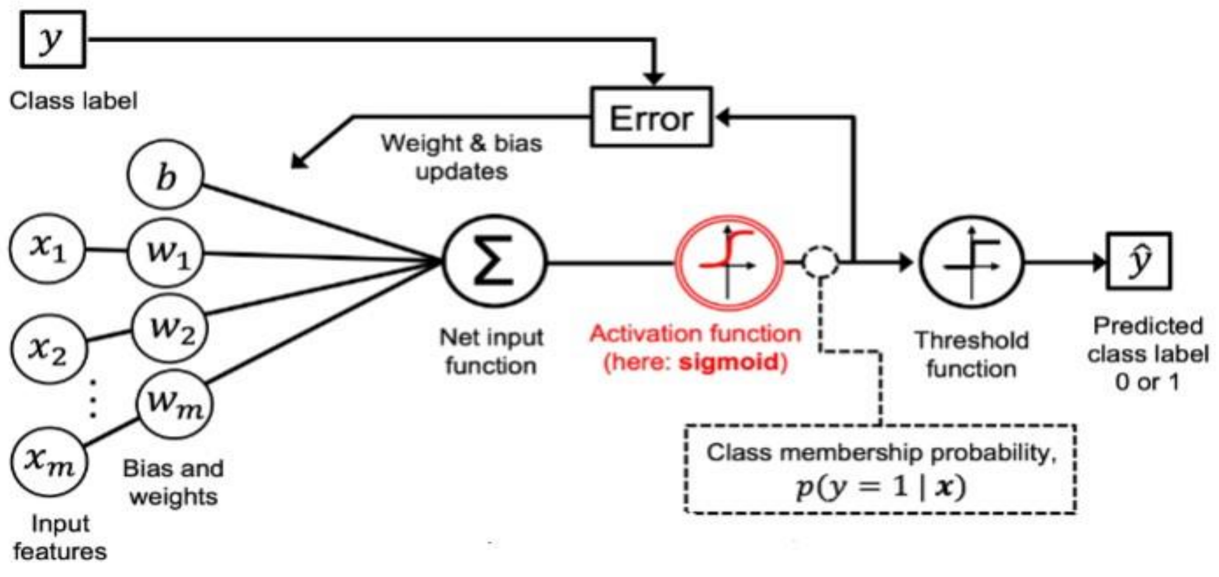


Figure 3.4 Working of Logistic Regression Algorithm

Working steps:

- Linear Model: Logistic Regression constructs a linear model by combining input features with corresponding weights (coefficients) to compute the log-odds of an instance belonging to the positive class.
- Sigmoid Transformation: The linear combination is then transformed using the sigmoid function, which maps the output to a value between 0 and 1:
- $P(y = 1|x) = \frac{1}{1+e^{-z}}$ where z is the linear

  combination of features.

- Probability Prediction: The sigmoid function's output $P(y = 1|x)$ represents the probability that the instance belongs to the positive class.
- Decision Rule: A decision rule is applied to convert predicted probabilities into class labels. For example, if$P(y = 1|x) \geq 0.5$, the instance is classified as positive; otherwise, it's classified as negative.

Advantages:

- Simple and Interpretable: It's straightforward to understand and interpret, especially when compared to more complex models.
- Efficient with Small Datasets: Works well with small datasets and provides good results when the relationship between features and target is linear.

- Probabilistic Interpretation: Provides probabilities for outcomes, aiding decision-making by quantifying uncertainty.

Limitations:

- Limited to Linear Relationships: Assumes a linear relationship between features and the log-odds of the target variable.
- Binary Classification: Originally designed for binary classification and may require modifications for multi-class problems.
- Vulnerable to Outliers: Sensitive to outliers as it maximizes likelihood, making it less robust when outliers are present.

5.1.5 Gradient Classifier

Gradient Boosting, an ensemble learning method, is akin to a team of specialists pooling their expertise to solve a problem collaboratively. Unlike traditional models, it doesn't rely on a single strong learner but harnesses the strength of multiple weaker models, often decision trees, working sequentially to form a robust predictive model. This approach iteratively corrects errors made by the preceding models, placing more focus on instances that were previously misclassified or had higher residuals. By optimizing its predictions through a process akin to gradient descent, Gradient Boosting incrementally refines its models, culminating in a collective prediction that surpasses the individual learners' capabilities. Celebrated for its high accuracy and adaptability across various data types, Gradient Boosting requires careful parameter tuning to mitigate overfitting and manage computational demands. Its ability to integrate insights from multiple weak models renders it a valuable tool for predicting outcomes across diverse domains.

Picture Gradient Boosting as a mentor guiding a team of eager learners—each subsequent model in the ensemble is like a diligent student learning from the mistakes of its predecessors. Through this mentorship, the team collectively polishes its expertise, honing its predictive precision with each iteration. Just as a skilled conductor orchestrates musicians to produce a harmonious melody, Gradient Boosting orchestrates the combination of individual models' predictions, culminating in a harmonized ensemble forecast. This collaborative approach, continually refining predictions through the amalgamation of diverse insights, establishes Gradient Boosting as a versatile and reliable technique for solving intricate prediction challenges in various real-world scenarios.

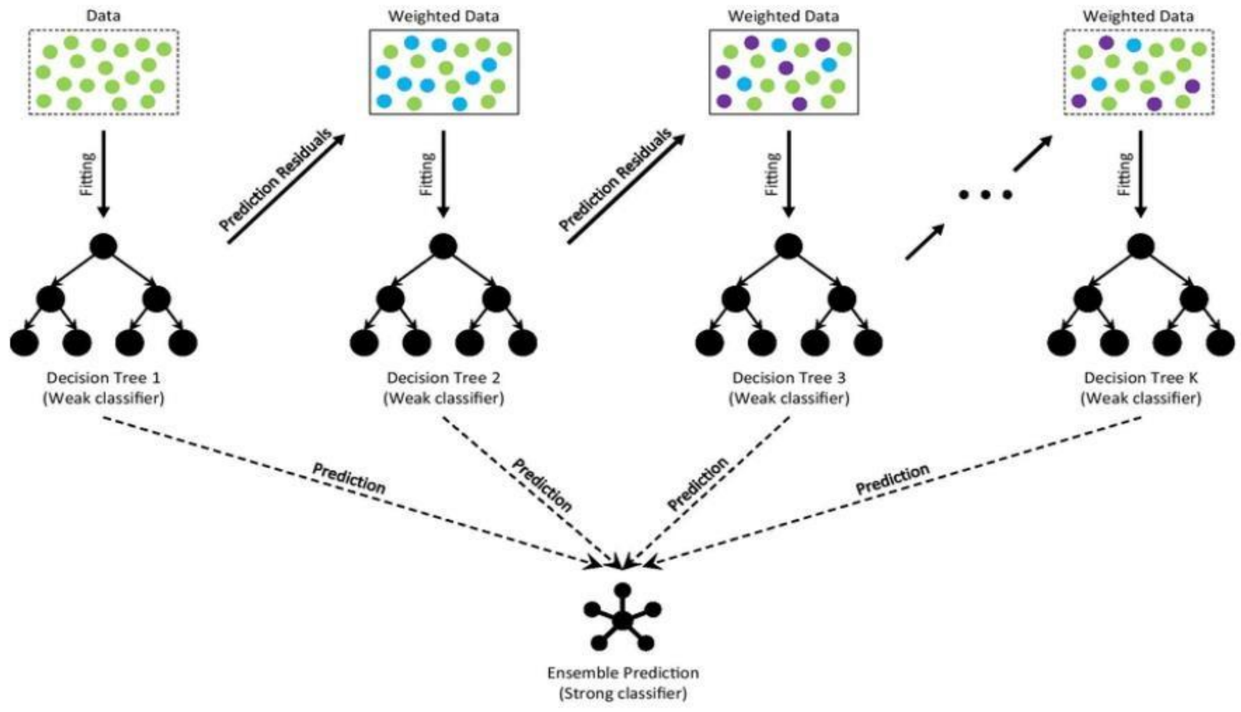


Figure 5.5 Working of Gradient Classifier

Working steps:

- Base Model Creation: Initial Model: The process begins with an initial weak learner (often a decision tree), typically a simple model with limited predictive capability.
- Sequential Improvement: Focus on Residuals: Subsequent models focus on the errors (residuals) made by the previous models. They predict the remaining errors to refine the overall prediction.
- Weighted Training: Instance Weights: Instances that were previously misclassified or had higher residuals are given increased emphasis in subsequent models.
- Gradient Descent: Gradient Descent Optimization: Gradient Boosting employs gradient descentlike techniques to minimize the overall error by adjusting the weights and learning rate in subsequent models.
- Ensemble Prediction: Combined Predictions: Finally, all the individual weak models are combined to produce a strong predictive model by taking a weighted sum or voting of their predictions.

Advantages:

- High Predictive Accuracy: Gradient Boosting often achieves superior predictive performance compared to standalone models due to its ensemble nature.
- Robustness: It handles various types of data and features, including numerical and categorical, effectively.

- Versatility: Suitable for a wide range of problems, from regression to classification, and works well with different loss functions.

Limitations:

- Overfitting Potential: It can be prone to overfitting, especially when the number of trees (iterations) is high or if the data is noisy.
- Computational Intensity: Training multiple trees sequentially can be computationally expensive, especially for large datasets.
- Hyper parameter Sensitivity: Tuning hyper parameters is crucial for optimal performance, which requires careful attention and computational resources.

5.2 Evaluation Parameters

Evaluation parameters in machine learning are pivotal metrics used to gauge the performance and accuracy of models. These metrics are tailored to different types of machine learning tasks, such as classification, regression, and clustering, providing valuable insights into model efficacy. In classification tasks, metrics like accuracy, precision, recall, and F1 score are commonly employed to assess a model's ability to correctly classify instances and manage trade-offs between false positives and false negatives. Accuracy measures the overall correctness of predictions, while precision quantifies the ratio of true positive predictions to all positive predictions, indicating the model's ability to avoid false positives. Conversely, recall assesses the ratio of true positive predictions to all actual positive instances, reflecting the model's sensitivity to detecting positive cases. The F1 score combines precision and recall into a single metric, balancing the trade-off between false positives and false negatives.

5.2.1 Accuracy

Accuracy is a fundamental evaluation metric used in machine learning to measure the performance of classification models. It represents the proportion of correctly predicted instances among the total instances in the dataset.

Formula for Accuracy:

The formula for accuracy is straightforward:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

Mathematically, if we have a binary classification problem (two classes, positive and negative):

$$True\ Positives + True\ Negatives$$

$$Accuracy = \frac{}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives}$$

Where:

True Positives (TP): The number of instances correctly predicted as positive.

True Negatives (TN): The number of instances correctly predicted as negative.

False Positives (FP): The number of instances incorrectly predicted as positive.

False Negatives (FN): The number of instances incorrectly predicted as negative.

Interpretation of Accuracy:

An accuracy score of 1.0 (or 100%) implies perfect predictions, where the model correctly classifies all instances.

An accuracy score of 0.0 (or 0%) indicates the model's complete failure in making correct predictions.

It's important to consider that accuracy alone might not be sufficient, especially for imbalanced datasets where one class dominates the other. For instance, in a dataset where 95% of instances belong to one class, a model that simply predicts the majority class would achieve 95% accuracy, but might not be useful.

5.2.2 Precision

Precision is a critical performance metric in machine learning used to assess the accuracy of positive predictions made by a classifier. It measures the ratio of correctly predicted positive instances to the total instances predicted as positive.

Formula of Precision:

The formula for Precision is straightforward:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

True Positives (TP): Instances correctly predicted as positive by the model.

False Positives (FP): Instances incorrectly predicted as positive when they are actually negative.

Precision quantifies the accuracy of positive predictions made by the model. A high precision score indicates that when the model predicts positive, it is often correct. It is valuable when the cost of False Positives is high, such as in email spam filters or credit card fraud detection.

Interpretation of Precision:

The precision of the model in this case is 0.83 or 83%, indicating that among the instances the model predicted as positive, approximately 83% were indeed positive, while around 17% were false positives.

Precision is a vital metric, especially in scenarios where the focus is on minimizing false positive predictions. However, it should be considered along with other metrics like recall to comprehensively evaluate a model's performance. A balanced model achieves high precision without sacrificing recall, and vice versa.

5.2.3 Recall

In machine learning, recall (also known as Sensitivity or True Positive Rate) is a performance metric used to evaluate the effectiveness of a classification model, particularly in terms of its ability to correctly identify positive instances from the total actual positive instances.

Formula for Recall:

The formula for recall is straightforward:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

True Positives (TP): The number of instances correctly predicted as positive by the model.

False Negatives (FN): The number of instances falsely predicted as negative when they are actually positive.

Recall measures the proportion of actual positive instances that the model correctly identifies as positive. A high recall indicates that the model correctly captures most positive instances. It's beneficial when the cost of missing positive instances (False Negatives) is high, such as in medical diagnoses or fraud detection, where missing positives is critical.

Interpretation of Recall:

In this example, the recall of the model is 0.8 or 80%, indicating that the model correctly identified 80% of the actual positive instances, but it missed 20% of them.

Recall is valuable in scenarios where it's crucial to minimize False Negatives, ensuring that positive instances are correctly identified, even if it means potentially increasing False Positives. However, it should be considered alongside other metrics to get a comprehensive understanding of a model's performance.

CHAPTER 6

DATASET AND METHODS

6.1 Model Overview

An Intrusion Detection System (IDS) employing machine learning leverages algorithms to discern abnormal patterns within network traffic. Initially, relevant data, encompassing packet headers and session details, is gathered and labelled as normal or malicious. Following data pre-processing and feature engineering, the dataset is divided into training and testing sets.

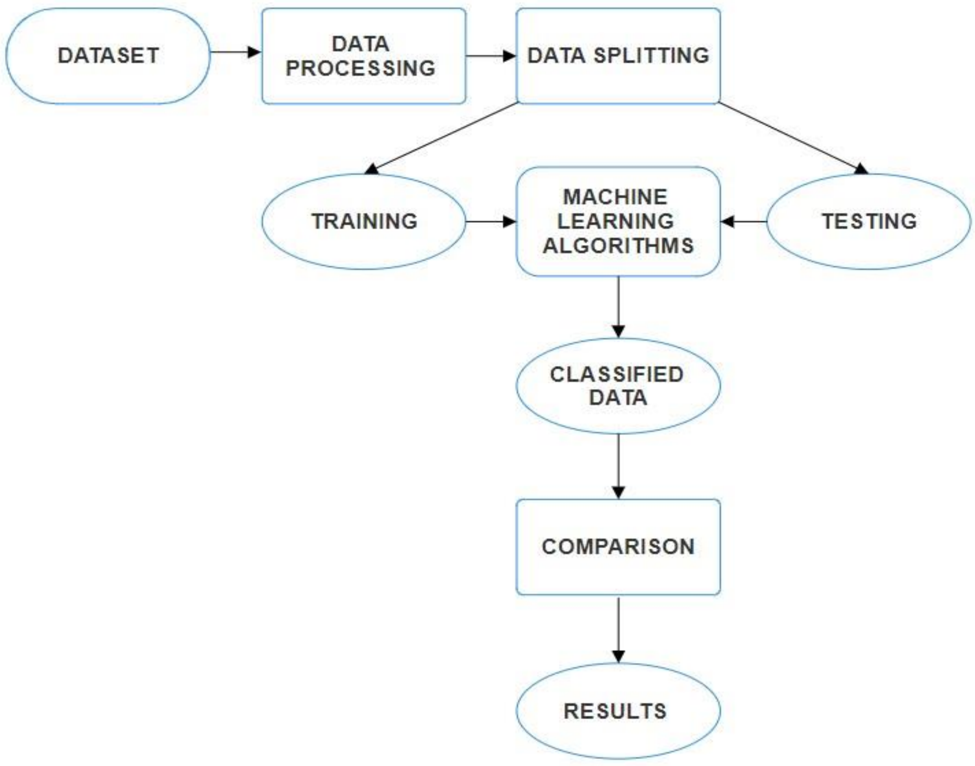


Figure 6.1 Overview Model of IDS

The model selection involves opting for suitable algorithms, such as Gaussian Naïve Bayes, Decision Trees, Random Forest, Logistic Regression or Gradient Classifier. The chosen model undergoes training on the labelled dataset, distinguishing between normal and malicious patterns. Evaluation metrics like accuracy and precision assess the model's performance on the testing set. Upon satisfactory results, the model is deployed to monitor real-time network traffic, with continuous updates and monitoring ensuring adaptability to emerging threats. The system's integration with incident response mechanisms facilitates prompt actions upon detection. This iterative process, including periodic model updates and a feedback loop, ensures the effectiveness of the IDS over time.

6.2 Data Set

Dataset Used: NSL-KDD

NSL-KDD serves as a dataset utilized in network security research for intrusion detection purposes. Originating from "NSL-KDD: A New Intrusion Detection Dataset for the Evaluation of Intrusion Detection Systems," it represents an enhanced iteration of the KDD'99 dataset. While KDD'99 was extensively employed for assessing intrusion detection systems (IDS), it suffered from issues such as redundancy and irrelevant attributes.

To rectify these shortcomings, NSL-KDD was devised. This involved the elimination of duplicate entries, rectification of labelling inaccuracies, and reduction of redundant attributes. Consequently, NSL-KDD furnishes a more authentic and demanding framework for appraising IDS algorithms. Since its inception, NSL-KDD has evolved into a standard benchmark dataset within the intrusion detection domain. It empowers researchers to innovate and assess novel methodologies aimed at detecting and thwarting network intrusions.

Data files

KDDTrain+.ARFF: The full NSL-KDD train set with binary labels in ARFF format

KDDTrain+.TXT: The full NSL-KDD train set including attack-type labels and difficulty level in CSV format

KDDTrain+_20Percent.ARFF: A 20% subset of the KDDTrain+.arff file

KDDTrain+_20Percent.TXT: A 20% subset of the KDDTrain+.txt file

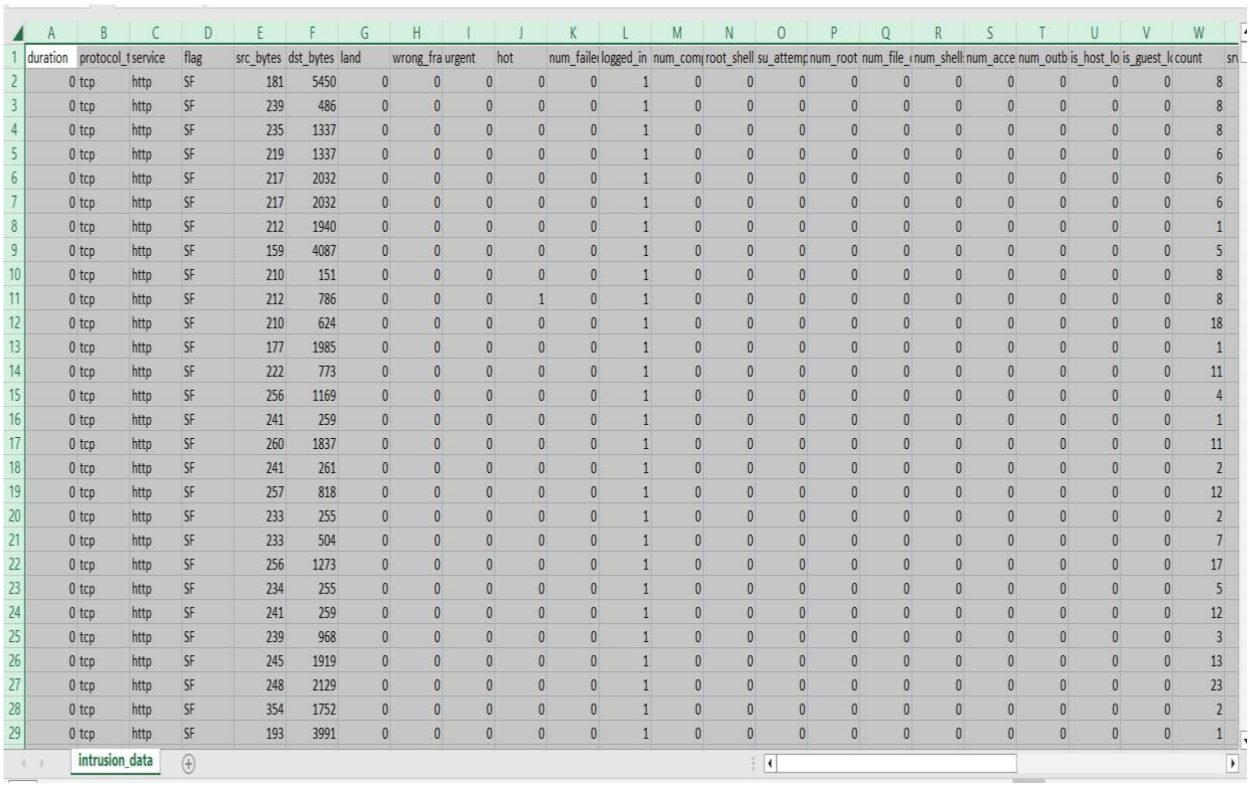
KDDTest+.ARFF: The full NSL-KDD test set with binary labels in ARFF format

KDDTest+.TXT: The full NSL-KDD test set including attack-type labels and difficulty level in CSV format

KDDTest-21. ARFF: A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21

KDDTest-21.TXT: A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21.

When creating the dataset, it's important to maintain the representativeness of real-world scenarios and ensure that the model is exposed to a diverse range of network behaviours. The quality and diversity of the dataset play a crucial role in the effectiveness of the IDS using machine learning.
The input dataset consists of 494021 records.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	duration	protocol_t	service	flag	src_bytes	dst_bytes	land	wrong_fra	urgent	hot	num_faile	logged_in	num_com	root_shell	su_attemp	num_root	num_file_	num_shell	num_acce	num_outb	is_host_lo	is_guest_l	count
2	0	tcp	http	SF	181	5450	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8
3	0	tcp	http	SF	239	486	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8
4	0	tcp	http	SF	235	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8
5	0	tcp	http	SF	219	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6
6	0	tcp	http	SF	217	2032	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6
7	0	tcp	http	SF	217	2032	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6
8	0	tcp	http	SF	212	1940	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
9	0	tcp	http	SF	159	4087	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5
10	0	tcp	http	SF	210	151	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8
11	0	tcp	http	SF	212	786	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	8
12	0	tcp	http	SF	210	624	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	18
13	0	tcp	http	SF	177	1985	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
14	0	tcp	http	SF	222	773	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11
15	0	tcp	http	SF	256	1169	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4
16	0	tcp	http	SF	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
17	0	tcp	http	SF	260	1837	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11
18	0	tcp	http	SF	241	261	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2
19	0	tcp	http	SF	257	818	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	12
20	0	tcp	http	SF	233	255	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2
21	0	tcp	http	SF	233	504	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	7
22	0	tcp	http	SF	256	1273	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	17
23	0	tcp	http	SF	234	255	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5
24	0	tcp	http	SF	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	12
25	0	tcp	http	SF	239	968	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3
26	0	tcp	http	SF	245	1919	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	13
27	0	tcp	http	SF	248	2129	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	23
28	0	tcp	http	SF	354	1752	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2
29	0	tcp	http	SF	193	3991	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

Table 6.1 Data Set Table

6.3 Data Processing

Data pre-processing in an Intrusion Detection System (IDS) involves cleaning and organizing raw network traffic data for machine learning. This includes handling missing values, normalizing numerical features, encoding categorical variables, and selecting relevant features. Balancing the dataset ensures unbiased model training, while proper splitting into training and testing sets facilitates effective evaluation. Time-based features are managed to capture temporal patterns, and redundant features are eliminated for efficiency.

Data pre-processing from the NSL-KDD dataset is a pivotal stage aimed at refining and organizing the collected data to prepare it for effective analysis. The NSL-KDD dataset, renowned for its preprocessed and labelled network traffic data, undergoes meticulous pre-processing steps to enhance its suitability for training ML models. During this phase, various tasks such as data cleaning, normalization, and feature engineering are performed to ensure the integrity and quality of the dataset. This may involve handling missing values, encoding categorical variables, and scaling numerical features to bring the data into a standardized format conducive to ML analysis. By meticulously preparing the dataset through pre-processing, potential inconsistencies and noise within the data are addressed, enabling more accurate and reliable detection of security threats within computer networks. The resulting pre-processed dataset serves as a robust foundation for subsequent model training and evaluation, ultimately contributing to the development of effective intrusion detection capabilities.

6.4 Feature Extraction

Feature extraction in an Intrusion Detection System (IDS) involves distilling relevant information from raw network traffic data to create a focused set of features for machine learning. This process includes extracting essential details from packet headers and session information, computing statistical measures for various parameters, and analysing frequency distributions of protocols and ports. Time-based features capture temporal patterns, while aggregate features summarize information over specific time windows. Dimensionality reduction techniques, such as Principal Component Analysis, may be applied to enhance efficiency. Domain-specific features cater to the unique characteristics of the network environment, and correlation analysis helps identify redundant variables. The resulting feature set is scaled and structured to empower the machine learning model to discern abnormal patterns and potential security threats effectively.

It plays a vital role in safeguarding network security by detecting and mitigating malicious activities. Their primary function is to identify and respond to unauthorized access attempts, data breaches, and other suspicious behaviours occurring within a network. Achieving this requires IDS to utilize a diverse range of features that provide insight into network traffic and behaviour.

These features serve as the foundation for detecting anomalies and potential security threats within the network. By analysing various aspects of network communication, such as protocol types, source and destination addresses, packet sizes, and duration of connections, IDS can identify patterns indicative of malicious activities. Additionally, IDS may track the frequency of failed login attempts, examine the volume of data transferred, and assess the nature of network services being accessed to further enhance threat detection capabilities.

In essence, the features utilized by IDS act as the eyes and ears of network security, continuously monitoring and analysing network traffic to identify any deviations from normal behaviour. By leveraging these features effectively, IDS can proactively detect and respond to security incidents, helping organizations mitigate the risks associated with cyber threats.

Basic features of individual TCP connections.

Feature Name	Description	Type
duration	length (number of seconds) of the connection	continuous
protocol type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete

wrong_fragment	number of “wrong” fragments	continuous
urgent	number of urgent packets	continuous

Table 6.2: Basic features of individual TCP connections.

Content features within a connection suggested by domain knowledge.

Feature Name	Description	Type
hot	Number of “hot” indicators	Continuous
num_failed_logins	Number of failed login attempts	Continuous
logged_in	1 if successfully logged in; 0 otherwise	Discrete
num_compromised	Number of “compromised” conditions	Continuous
root_shell	1 if root shell is obtained; 0 otherwise	Discrete
su_attempted	1 if “su root” command attempted; 0 otherwise	Discrete
num_root	Number of “root” accesses	Continuous
num_file_creations	Number of file creation operations	Continuous
num_shells	Number of shell prompts	Continuous
num_access_files	Number of operations on access control files	Continuous
num_outbound_cmds	Number of outbound commands in an ftp session	Continuous
is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise	Discrete
is_guest_login	1 if the login is a “guest”login; 0 otherwise	Discrete

Table 6.3: Content features within a connection suggested by domain knowledge.

Traffic features computed using a two-second time window.

Feature Name	Description	Type
count	Number of connections to the same host as the current connection in the past two seconds	Continuous
serror_rate	% of connections that have “SYN” errors	Continuous
rerror_rate	% of connections that have “REJ” errors	Continuous
same_srv_rate	% of connections to the same service	Continuous
diff_srv_rate	% of connections to different services	Continuous

srv_count	Number of connections to the same service as the current connection in the past two seconds	Continuous
srv_serror_rate	% of connections that have “SYN” errors	Continuous
srv_rerror_rate	% of connections that have “REJ” errors	Continuous
srv_diff_host_rate	% Of connections to different hosts	Continuous

Table 6.4 Traffic features computed using a two-second time window

Feature Set Used

Feature Name	Description	Type
diff_srv_rate	% of connections to different services	Continuous
dst_host_srv_diff_host_rate	The rate of different destination hosts communicating with a specific service,	Continuous
dst_host_same_src_port_rate	The rate of connections from the same source port to a particular service on the destination host	Continuous
srv_count	Number of connections to the same service as the current connection in the past two seconds	Continuous
protocol_type	Type of the protocol, e.g. Tcp, udp, etc.	Discrete
dst_host_count	Total number of connections made to the destination host in a network dataset.	Discrete
logged_in	1 if successfully logged in; 0 otherwise	Discrete
dst_bytes	Number of data bytes from destination to source	Continuous
count	Number of connections to the same host as the current connection in the past two seconds	Continuous

Table 6.5: Feature Set

6.5 Classification Technique

Classification techniques in the context of an Intrusion Detection System (IDS) involve using machine learning algorithms to categorize network activities as either normal or malicious. Various supervised learning algorithms are commonly employed for this purpose.

- Gaussian Naive Bayes is a probabilistic algorithm based on Bayes' theorem. It assumes independence between features and is particularly efficient for text classification.
- Decision trees partition the data based on features, creating a tree-like structure of decisions. They are interpretable and can handle both numerical and categorical data.

- Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions. It often improves accuracy and generalization.
- Logistic Regression models the probability of an instance belonging to a particular class. It is a simple yet effective algorithm for binary classification.
- Gradient Classifier an efficient and scalable gradient boosting library that is widely used in classification tasks, including IDS. It combines the strengths of tree-based models with regularization techniques.

6.6 Code

```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
model1 = GaussianNB()
model1.fit(X_train, Y_train.values.ravel())
Y_test_pred1 = model1.predict(X_test)
print("Train score is:", model1.score(X_train, Y_train))
print("Test score is:",model1.score(X_test,Y_test))

#Decision Tree
from sklearn.tree import DecisionTreeClassifier
model2 = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
model2.fit(X_train, Y_train.values.ravel())
Y_test_pred2 = model2.predict(X_test)
Y_train_pred2 = model2.predict(X_train)
print('test accuracy:',accuracy_score(Y_test,Y_test_pred2))
print('train accuracy:',accuracy_score(Y_train,Y_train_pred2))

from sklearn.ensemble import RandomForestClassifier
model3 = RandomForestClassifier(n_estimators=30)
model3.fit(X_train, Y_train.values.ravel())
Y_test_pred3 = model3.predict(X_test)
print("Train score is:", model3.score(X_train, Y_train))
print("Test score is:",model3.score(X_test,Y_test))
```

```
from sklearn.linear_model import LogisticRegression
model5 = LogisticRegression(max_iter=1200000, solver='lbfgs')
model5.fit(X_train, Y_train.values.ravel())
Y_test_pred5 = model5.predict(X_test)
print("Train score is:", model5.score(X_train, Y_train))
print("Test score is:",model5.score(X_test,Y_test))

from sklearn.ensemble import GradientBoostingClassifier
model6 = GradientBoostingClassifier(random_state=42)
model6.fit(X_train, Y_train.values.ravel())
Y_test_pred6 = model6.predict(X_test)
print("Train score is:", model6.score(X_train, Y_train))
print("Test score is:", model6.score(X_test,Y_test))

names = ['NB','DT','RF','SVM','LR','GB','ANN']
values = [87.95,99.06,99.99,99.86,99.35,99.79,99.91]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.ylim(80,102)
plt.bar(names,values)

names = ['NB','DT','RF','SVM','LR','GB','ANN']
values = [87.903,99.052,99.969,99.879,99.352,99.771,99.886]
f = plt.figure(figsize=(15,3),num=10)
plt.subplot(131)
plt.ylim(80,102)
plt.bar(names,values)

from sklearn.ensemble import RandomForestClassifier
import pandas as pd
# Assuming you have already defined and loaded your X_train and Y_train
# Create a RandomForestClassifier instance
rfc = RandomForestClassifier(random_state=42)
```

```
# Fit the model to the training data
rfc.fit(X_train, Y_train.values.ravel())
# Get feature importance scores
feature_importances = rfc.feature_importances_
feature_importances
# Convert X_train (numpy array) to a DataFrame
# X_train_df = pd.DataFrame(X_train, columns=X.columns)
# Create a DataFrame with feature names and their corresponding importances
# imp_feature = pd.DataFrame({'feature importance': feature_importances}, index=X_train_df.columns)
imp_feature=pd.DataFrame(index=X.columns,data=feature_importances,columns=['feature
importance'])
imp_feature
# Display the DataFrame
print(imp_feature)

from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(random_state=42)
rfc.fit(X_train,y_train.values.ravel())
print('*'*20)

# prediction
train_prediction= rfc.predict(X_train)
test_prediction= rfc.predict(X_test)
print('*'*20)

# evaluation
from sklearn.metrics import accuracy_score
print('test accuracy:',accuracy_score(y_test,test_prediction))
print('train accuracy:',accuracy_score(y_train,train_prediction))
print('*'*20)

#  prediction Summary by species
print(classification_report(y_test, test_prediction))
print('*'*20)
```

CHAPTER 7

EXPERIMENTAL RESULTS

7.1 Home Page

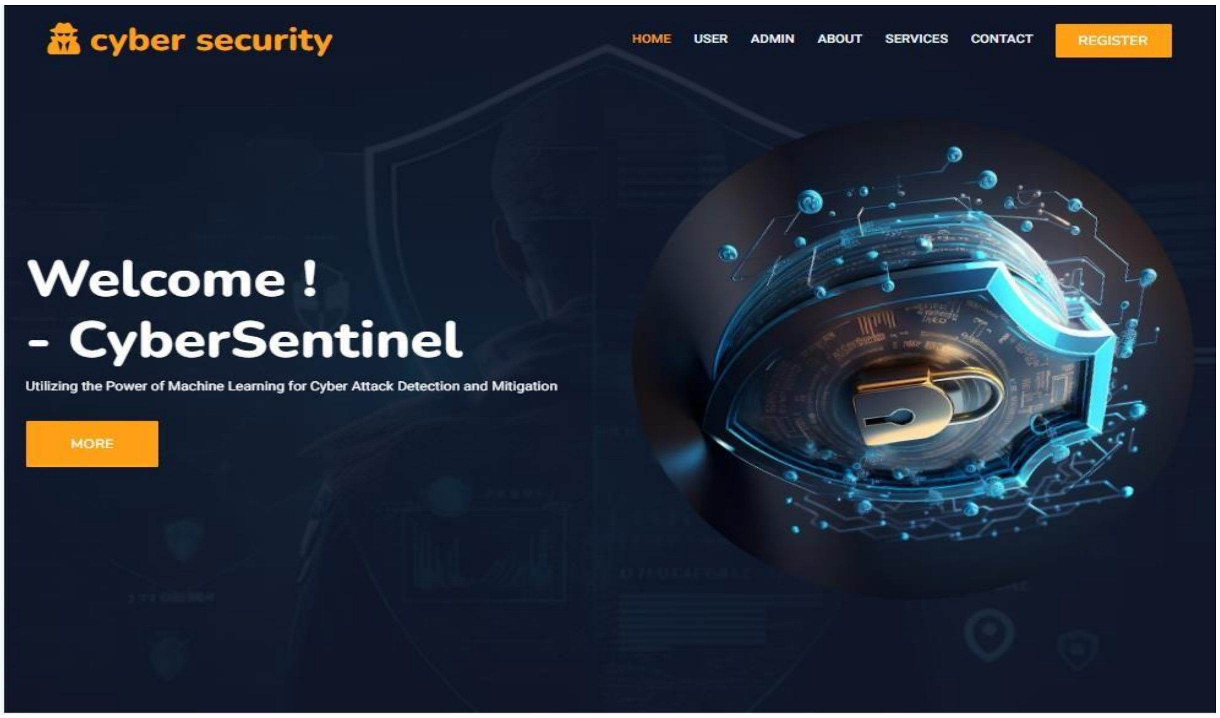


Figure 7.1 Home Page Layout

7.2 Admin Login

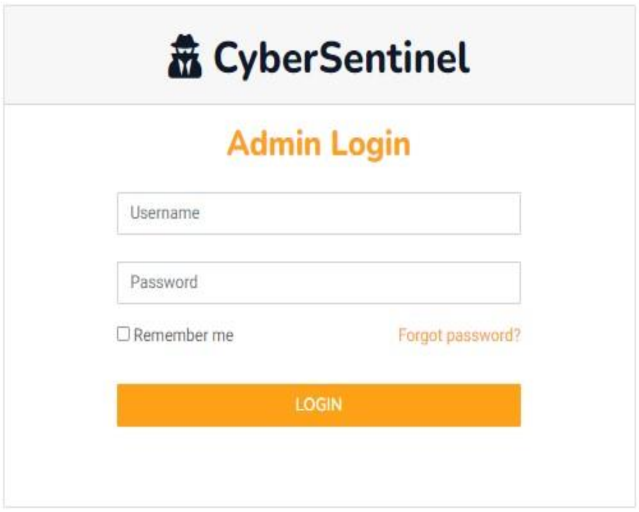


Figure 7.2 Login Page (Admin)

7.2.1 Admin Dashboard

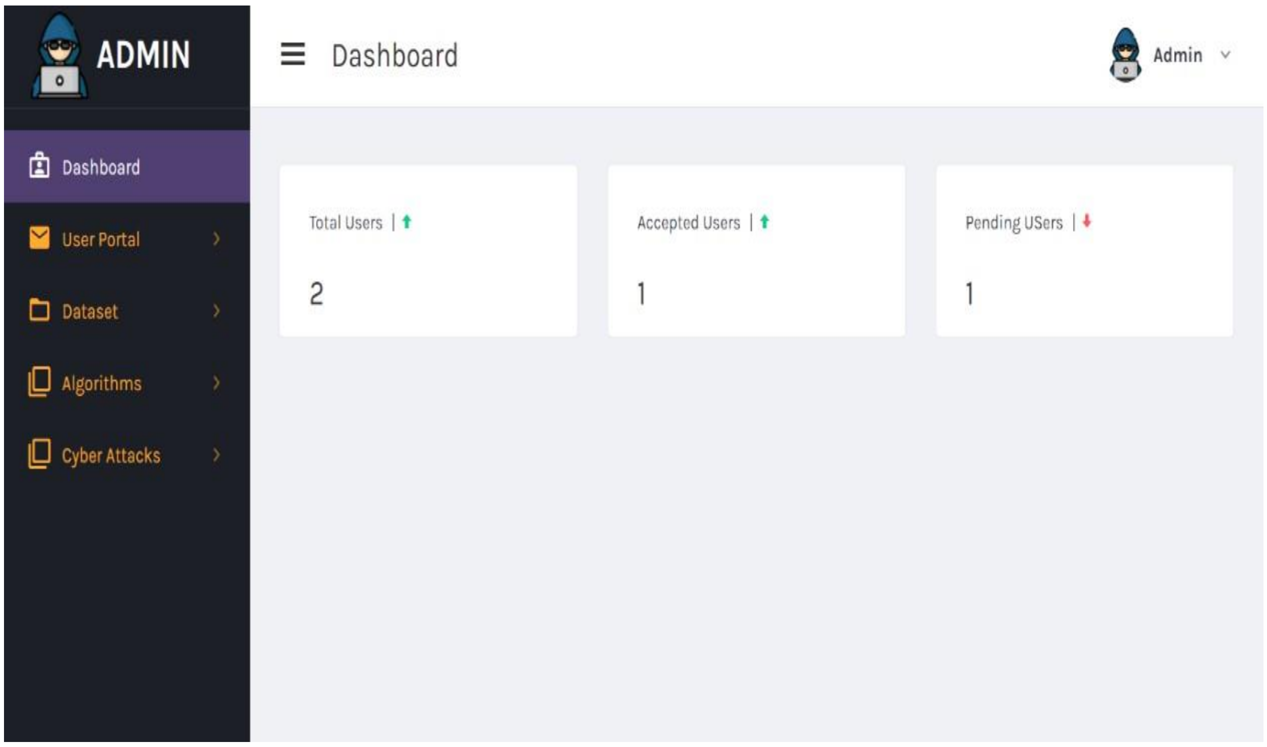
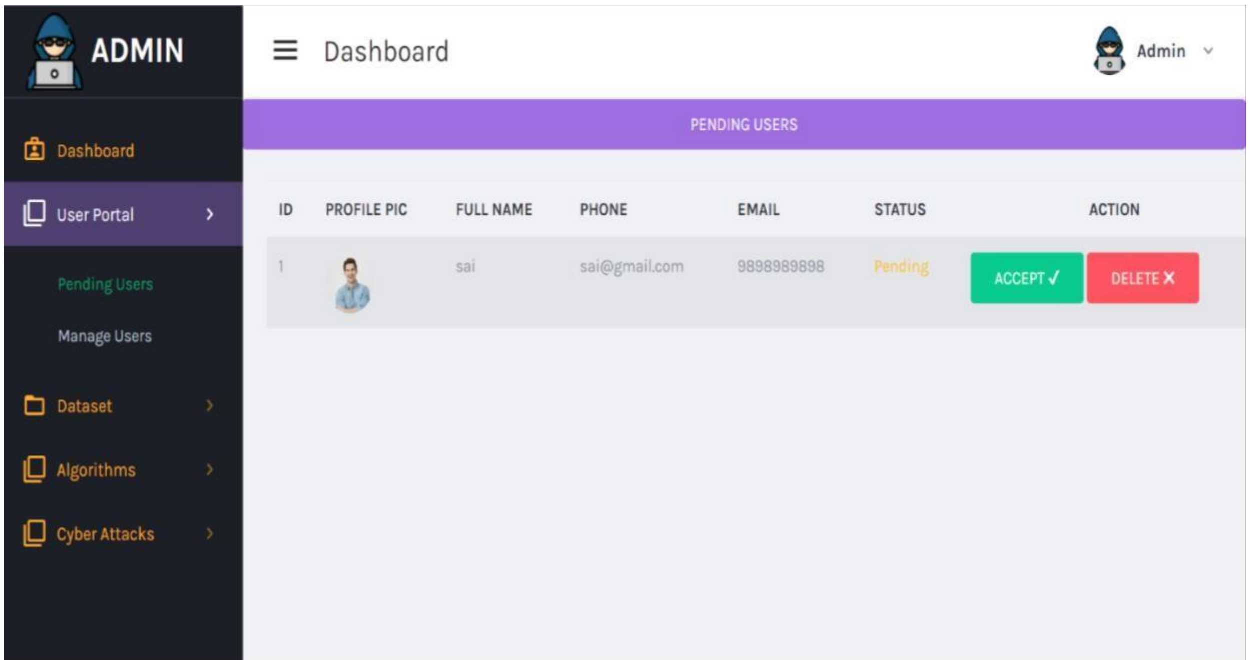


Figure 7.3 Dashboard of Admin

7.2.2 User Manage Portal



ID	PROFILE PIC	FULL NAME	PHONE	EMAIL	STATUS	ACTION
1		sai	sai@gmail.com	9898989898	Pending	ACCEPT ✓ DELETE ✕

Figure 7.4 User Portal of Admin

7.2.3 Upload Dataset

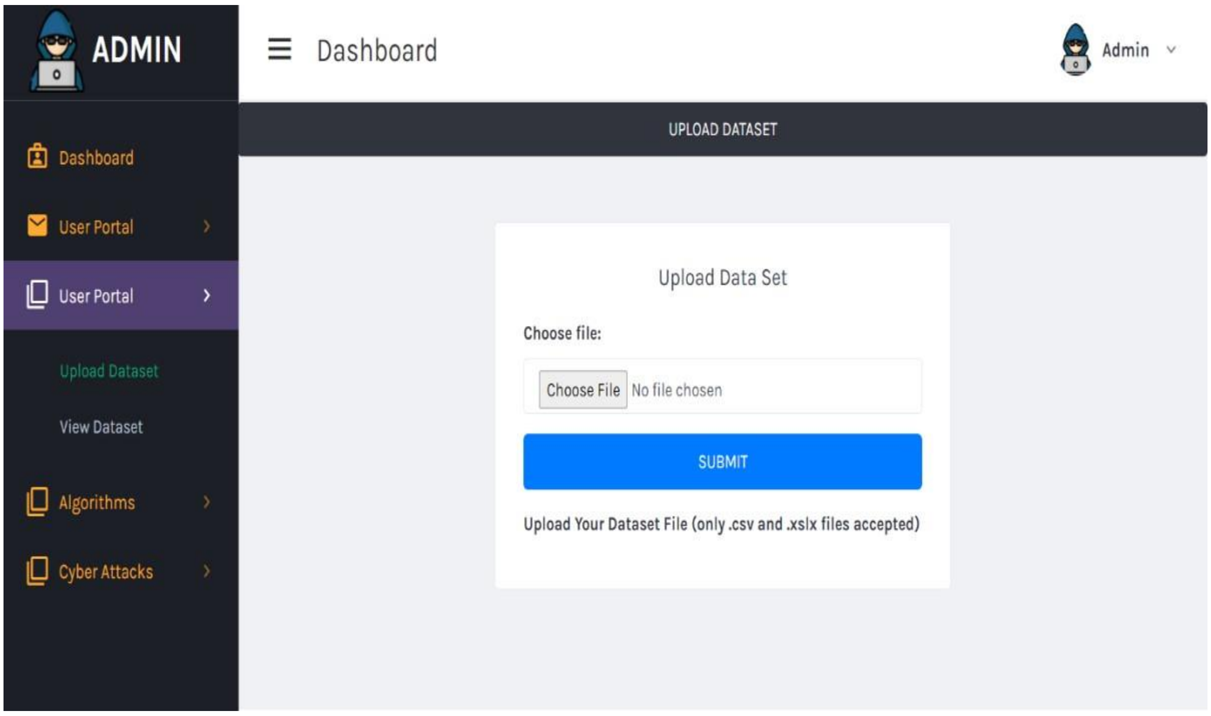
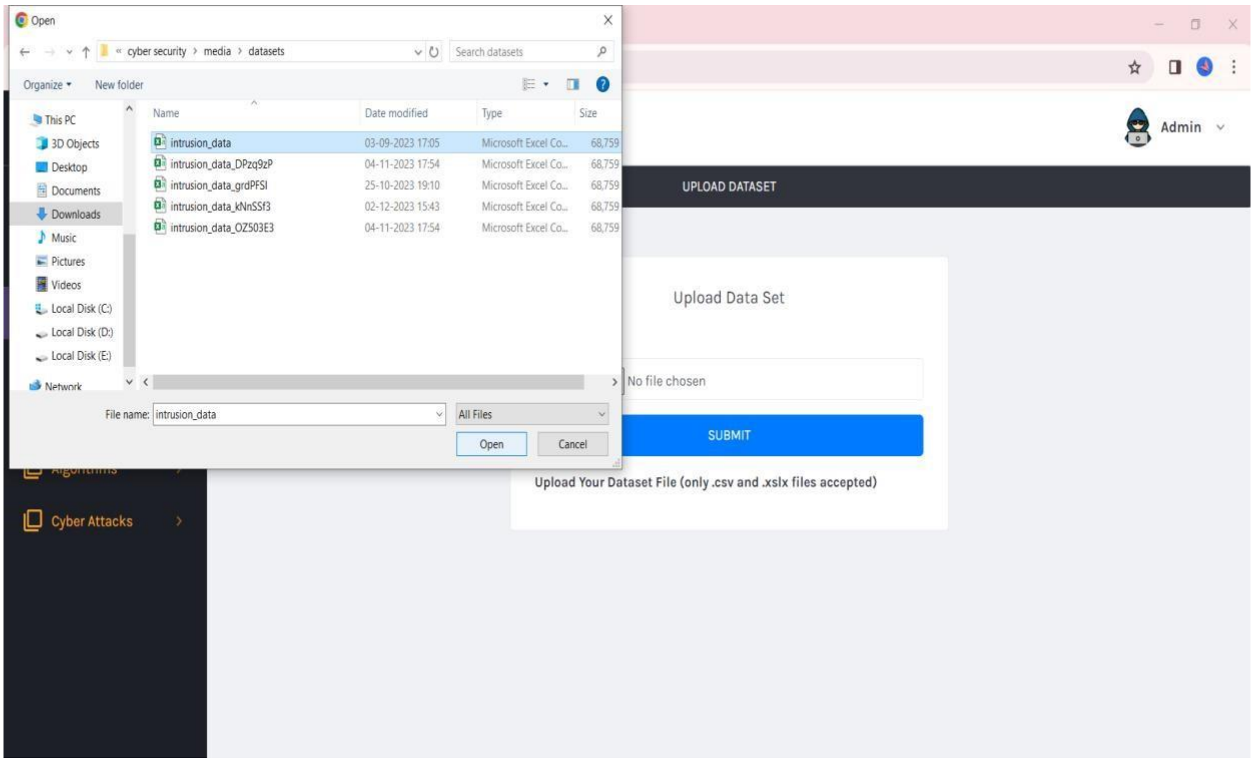
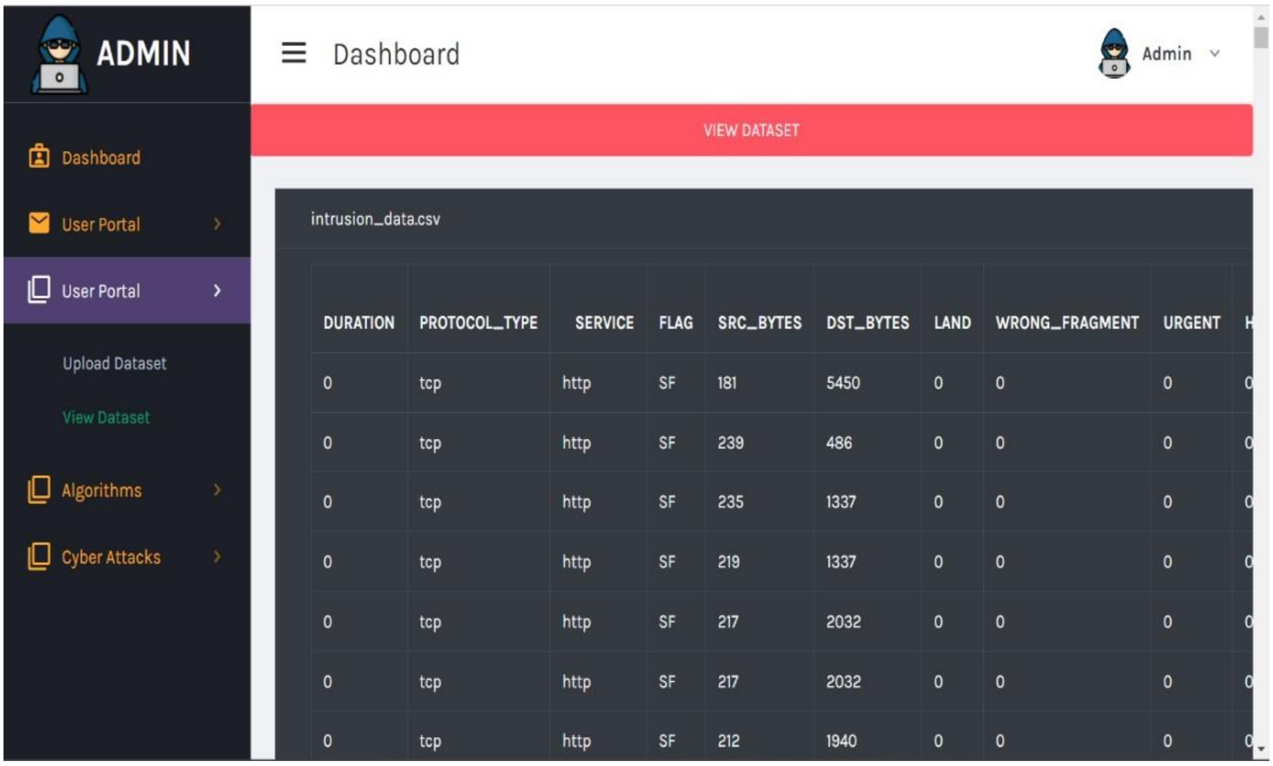


Figure 7.5 Uploading Dataset



Name	Date modified	Type	Size
intrusion_data	03-09-2023 17:05	Microsoft Excel Co...	68,759
intrusion_data_DPzq9zP	04-11-2023 17:54	Microsoft Excel Co...	68,759
intrusion_data_grdPFSI	25-10-2023 19:10	Microsoft Excel Co...	68,759
intrusion_data_kNnSSf3	02-12-2023 15:43	Microsoft Excel Co...	68,759
intrusion_data_OZ503E3	04-11-2023 17:54	Microsoft Excel Co...	68,759

Figure 7.6 Choosing File to Upload



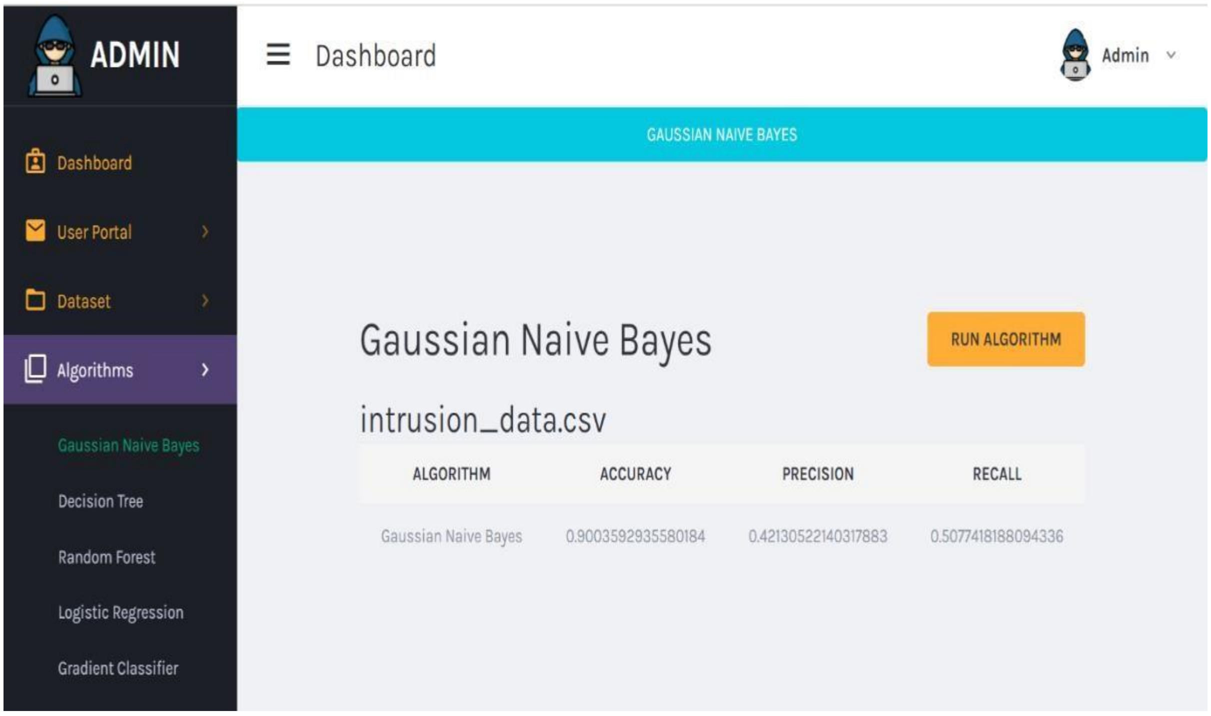
intrusion_data.csv

DURATION	PROTOCOL_TYPE	SERVICE	FLAG	SRC_BYTES	DST_BYTES	LAND	WRONG_FRAGMENT	URGENT
0	tcp	http	SF	181	5450	0	0	0
0	tcp	http	SF	239	486	0	0	0
0	tcp	http	SF	235	1337	0	0	0
0	tcp	http	SF	219	1337	0	0	0
0	tcp	http	SF	217	2032	0	0	0
0	tcp	http	SF	217	2032	0	0	0
0	tcp	http	SF	212	1940	0	0	0

Figure 7.7 View Dataset

7.2.4 Performance Analysis:

Gaussian Naïve Bayes Algorithm



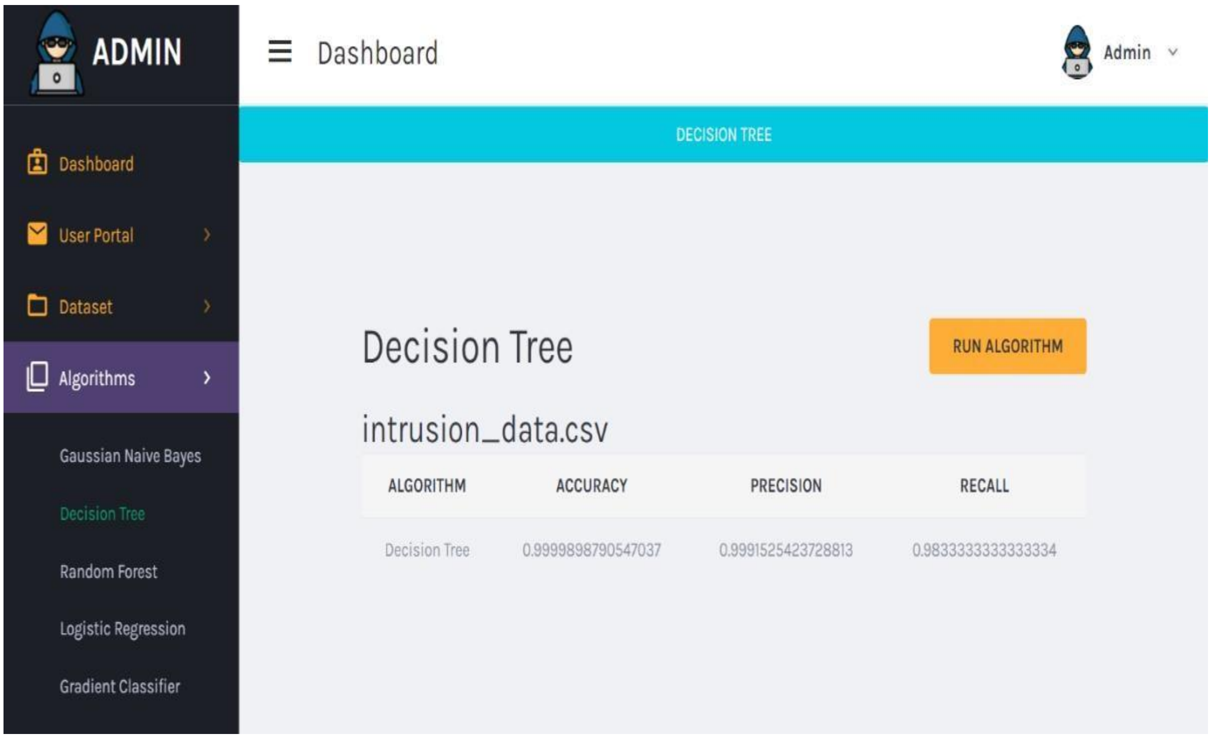
Gaussian Naive Bayes

intrusion_data.csv

ALGORITHM	ACCURACY	PRECISION	RECALL
Gaussian Naive Bayes	0.9003592935580184	0.42130522140317883	0.5077418188094336

Figure 7.8 Performance Analysis of GNB

Decision Tree Algorithm



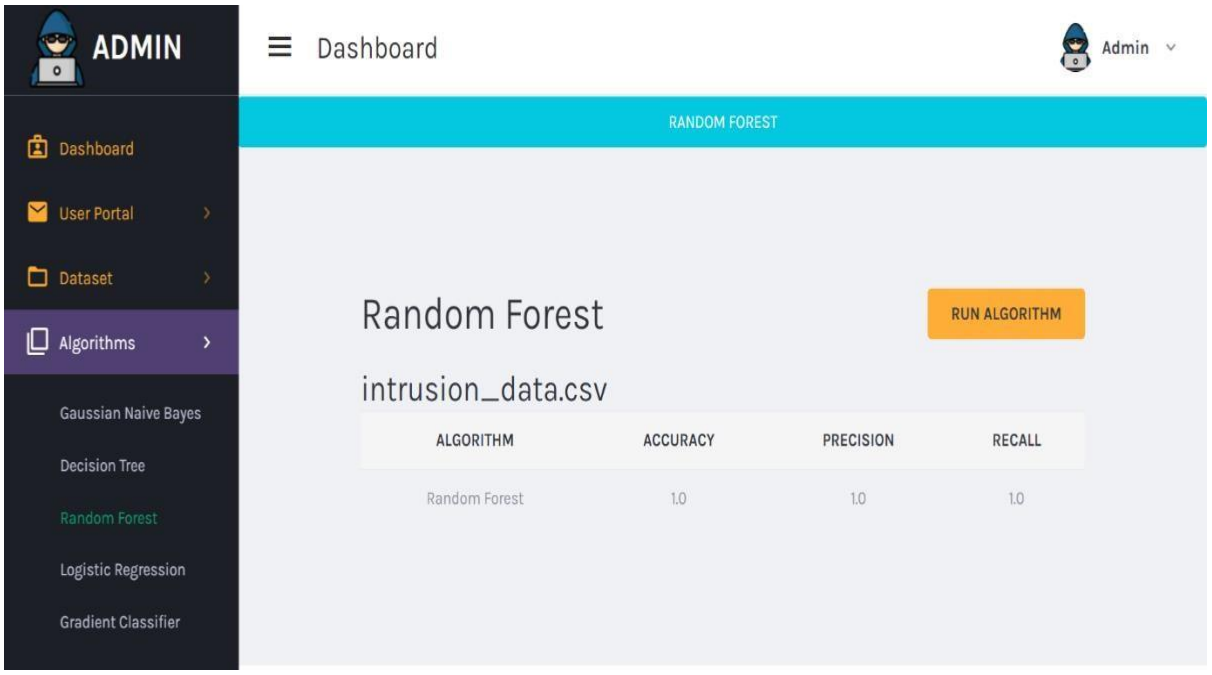
Decision Tree

intrusion_data.csv

ALGORITHM	ACCURACY	PRECISION	RECALL
Decision Tree	0.9999898790547037	0.9991525423728813	0.9833333333333334

Figure 7.9 Performance Analysis of Decision Tree

Randon Forest Algorithm



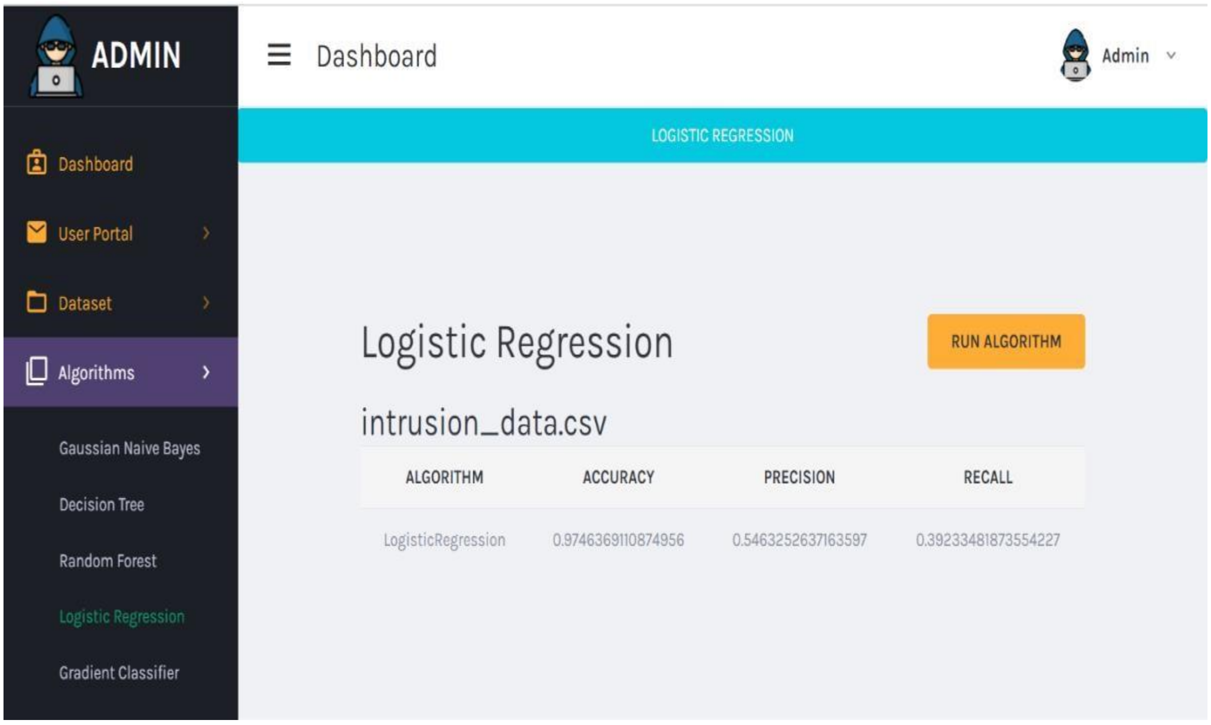
Random Forest

intrusion_data.csv

ALGORITHM	ACCURACY	PRECISION	RECALL
Random Forest	1.0	1.0	1.0

Figure 7.10 Performance Analysis of Random Forest

Logistic Regression Algorithm



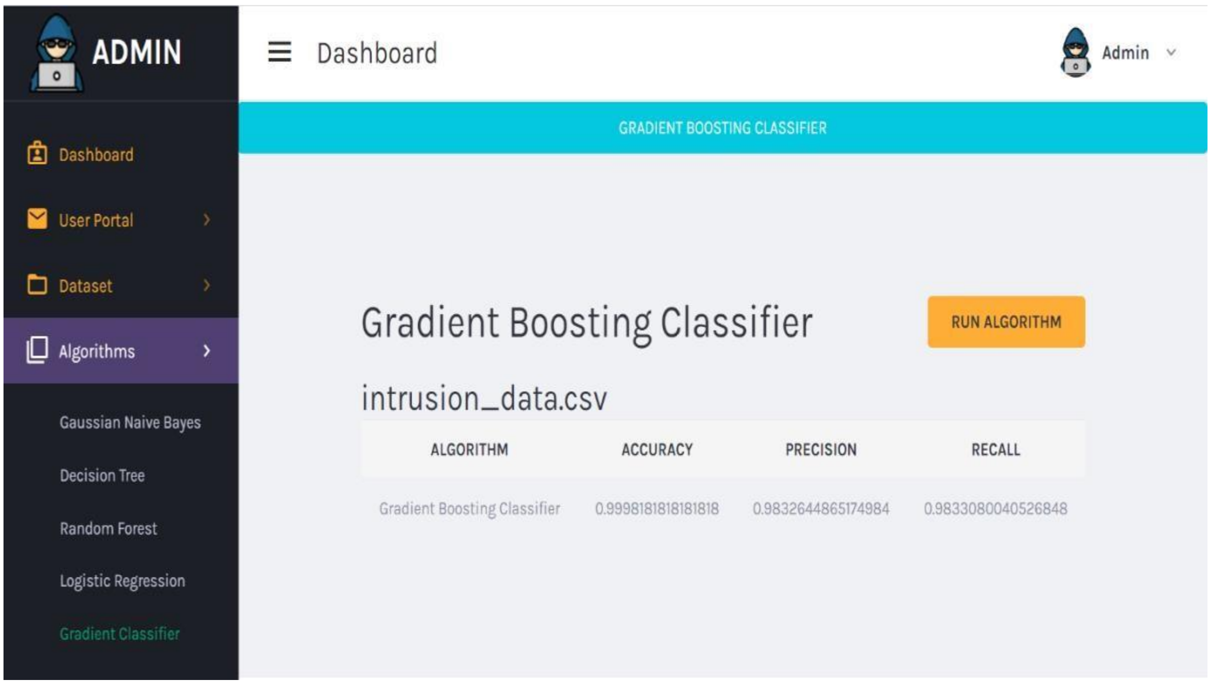
Logistic Regression

intrusion_data.csv

ALGORITHM	ACCURACY	PRECISION	RECALL
LogisticRegression	0.9746369110874956	0.5463252637163597	0.39233481873554227

Figure 7.11 Performance Analysis of Logistic Regression

Gradient Classifier Algorithm



Gradient Boosting Classifier

intrusion_data.csv

ALGORITHM	ACCURACY	PRECISION	RECALL
Gradient Boosting Classifier	0.9998181818181818	0.9832644865174984	0.9833080040526848

Figure 7.12 Performance Analysis of Gradient Classifier

7.3 User Login

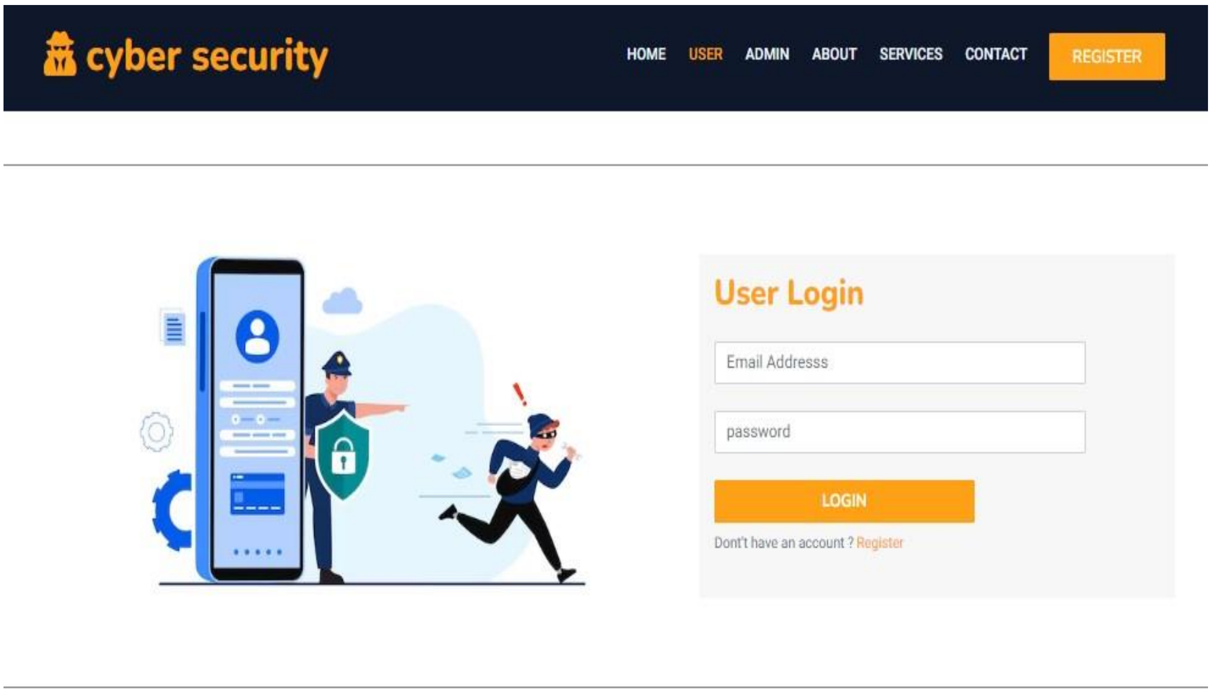


Figure 7.13 Login Page (User)

7.3.1 Normal Attack Detection

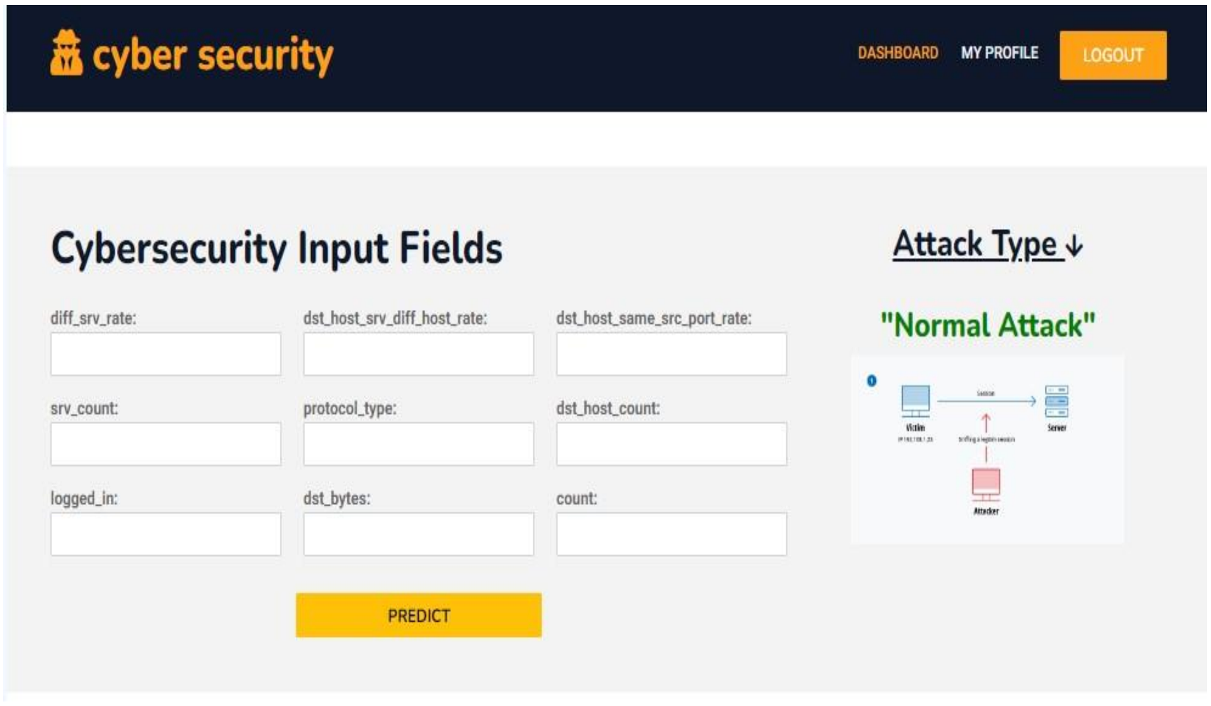
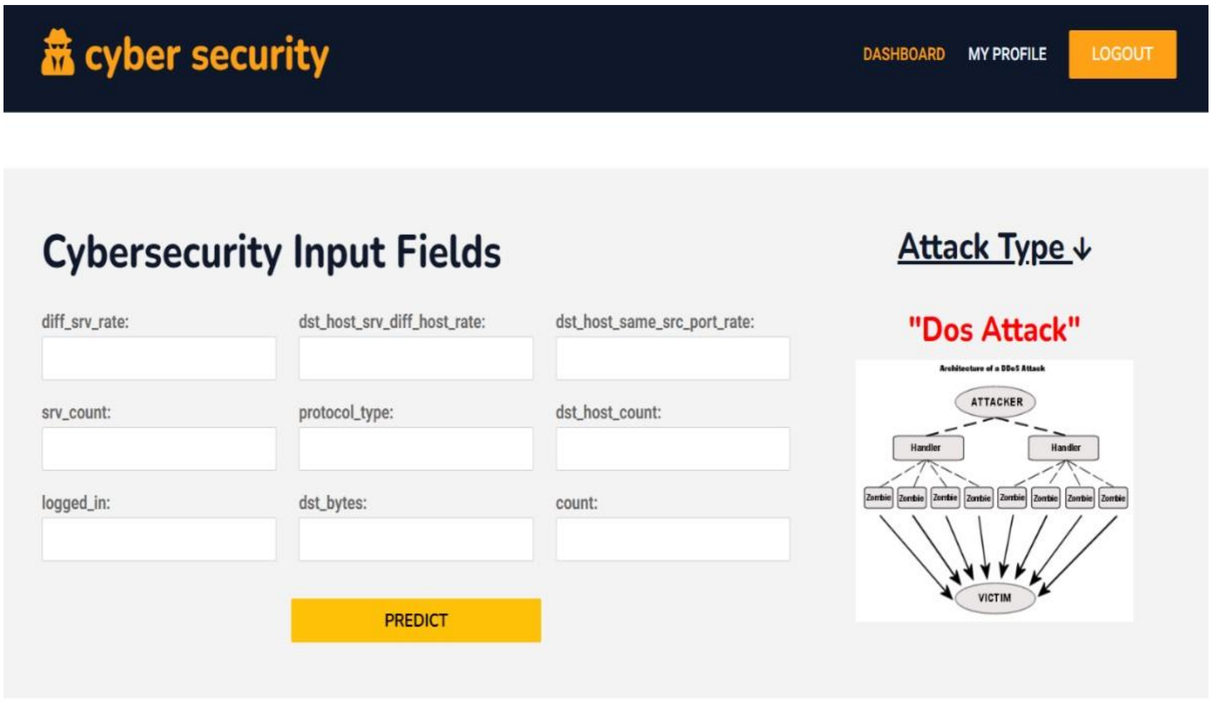


Figure 7.14 Normal Attack prediction

7.3.2 Dos Attack Detection



Cybersecurity Input Fields

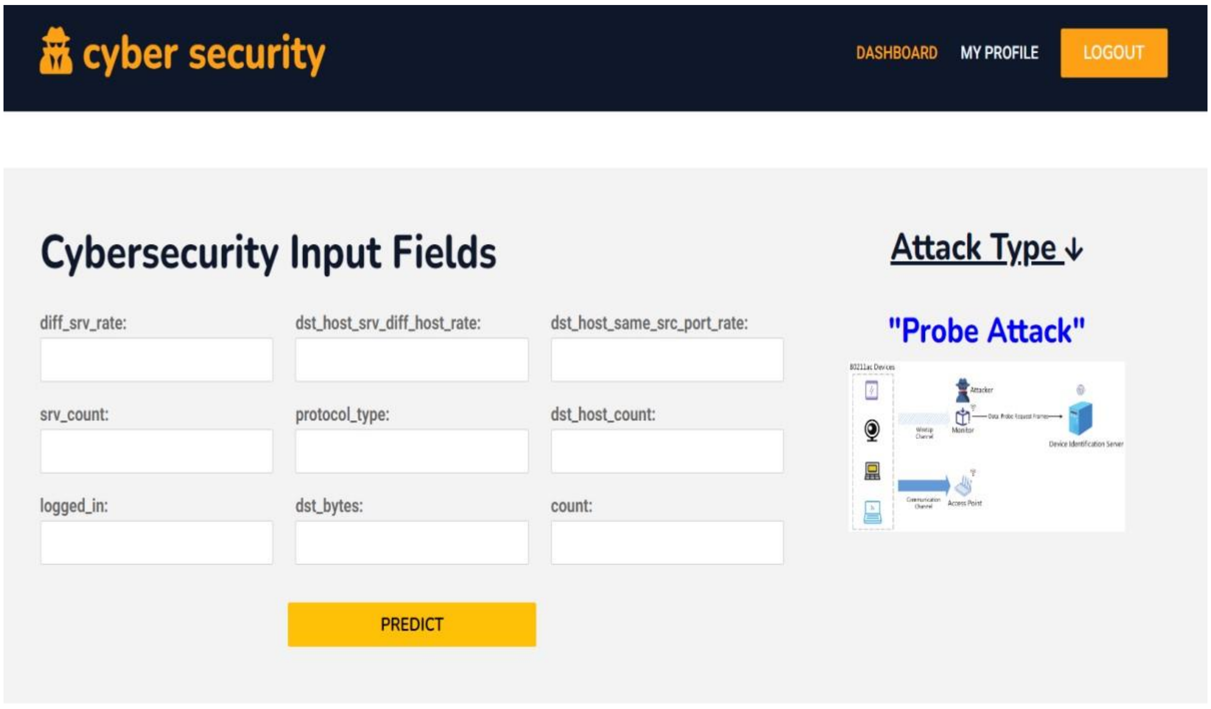
diff_srv_rate:	dst_host_srv_diff_host_rate:	dst_host_same_src_port_rate:
srv_count:	protocol_type:	dst_host_count:
logged_in:	dst_bytes:	count:

Attack Type ↓

"Dos Attack"

Figure 7.15 Dos Attack Prediction

7.3.3 Probe Attack Detection



Cybersecurity Input Fields

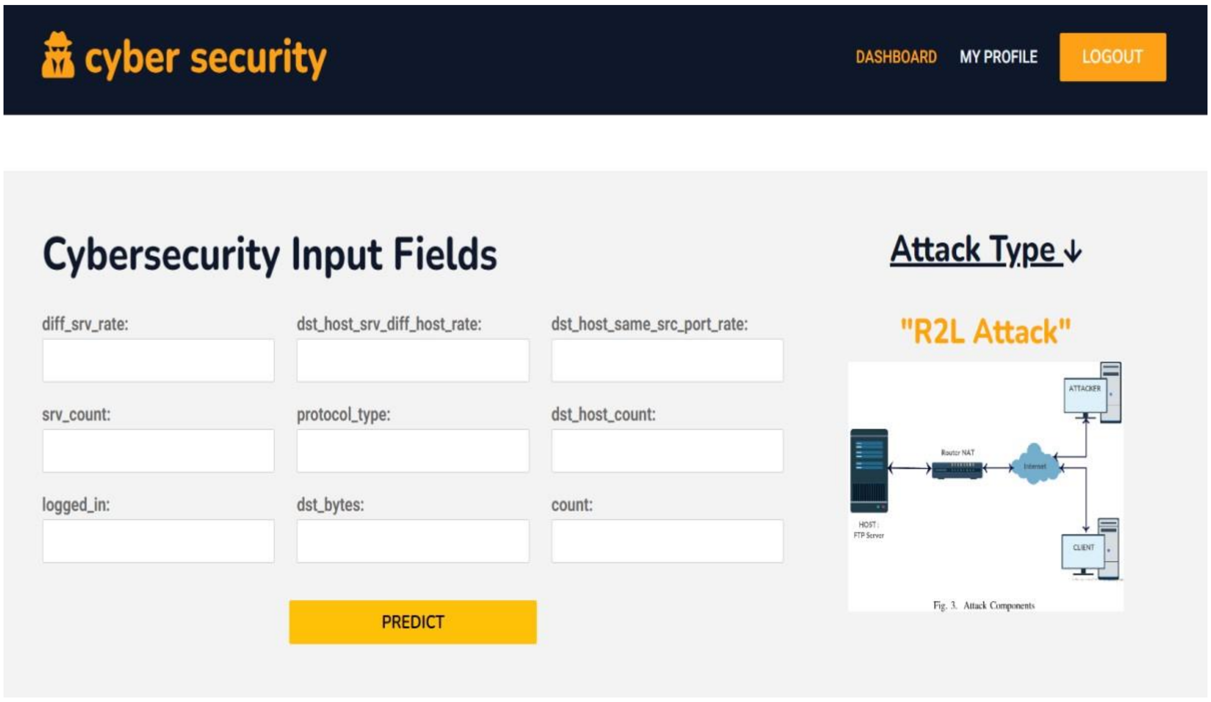
diff_srv_rate:	dst_host_srv_diff_host_rate:	dst_host_same_src_port_rate:
srv_count:	protocol_type:	dst_host_count:
logged_in:	dst_bytes:	count:

Attack Type ↓

"Probe Attack"

Figure 7.16 Probe Attack Prediction

7.3.4 R2L Attack Detection



Cybersecurity Input Fields

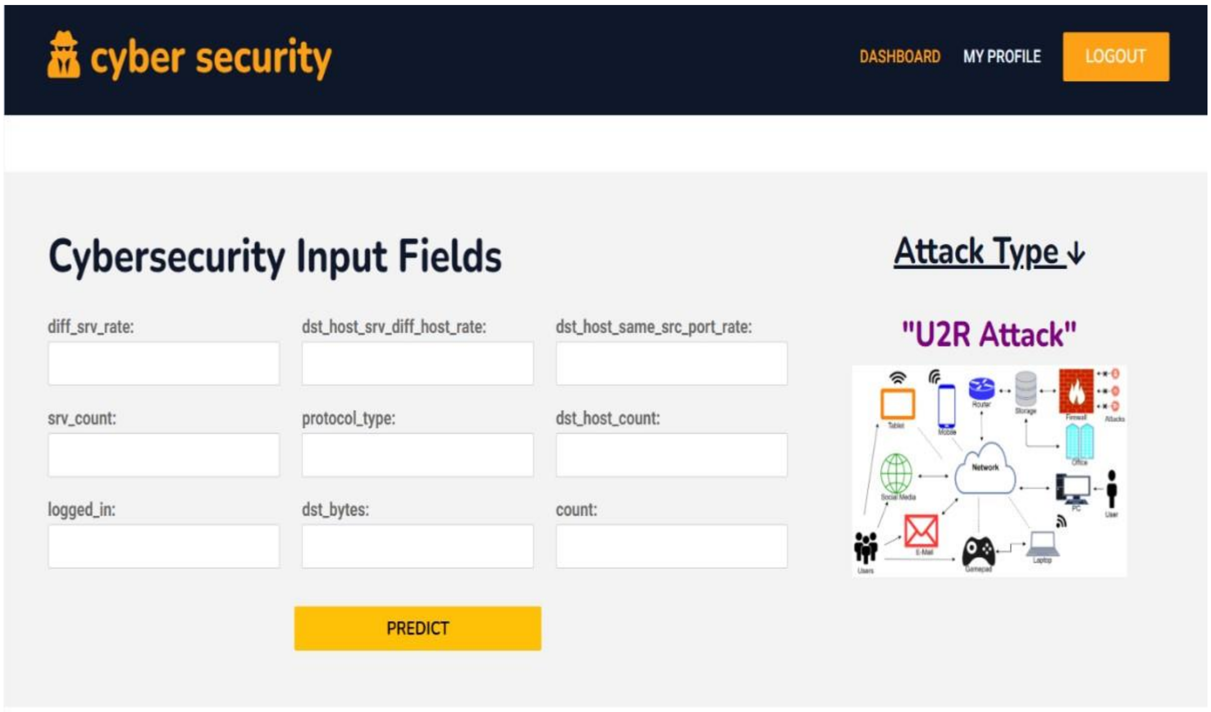
diff_srv_rate:	dst_host_srv_diff_host_rate:	dst_host_same_src_port_rate:
srv_count:	protocol_type:	dst_host_count:
logged_in:	dst_bytes:	count:

Attack Type ↓

"R2L Attack"

Figure 7.17 R2L Attack Prediction

7.3.5 U2R Attack Detection



Cybersecurity Input Fields

diff_srv_rate:	dst_host_srv_diff_host_rate:	dst_host_same_src_port_rate:
srv_count:	protocol_type:	dst_host_count:
logged_in:	dst_bytes:	count:

Attack Type ↓

"U2R Attack"

Figure 7.18 U2R Attack Prediction

7.4 Attack Analysis

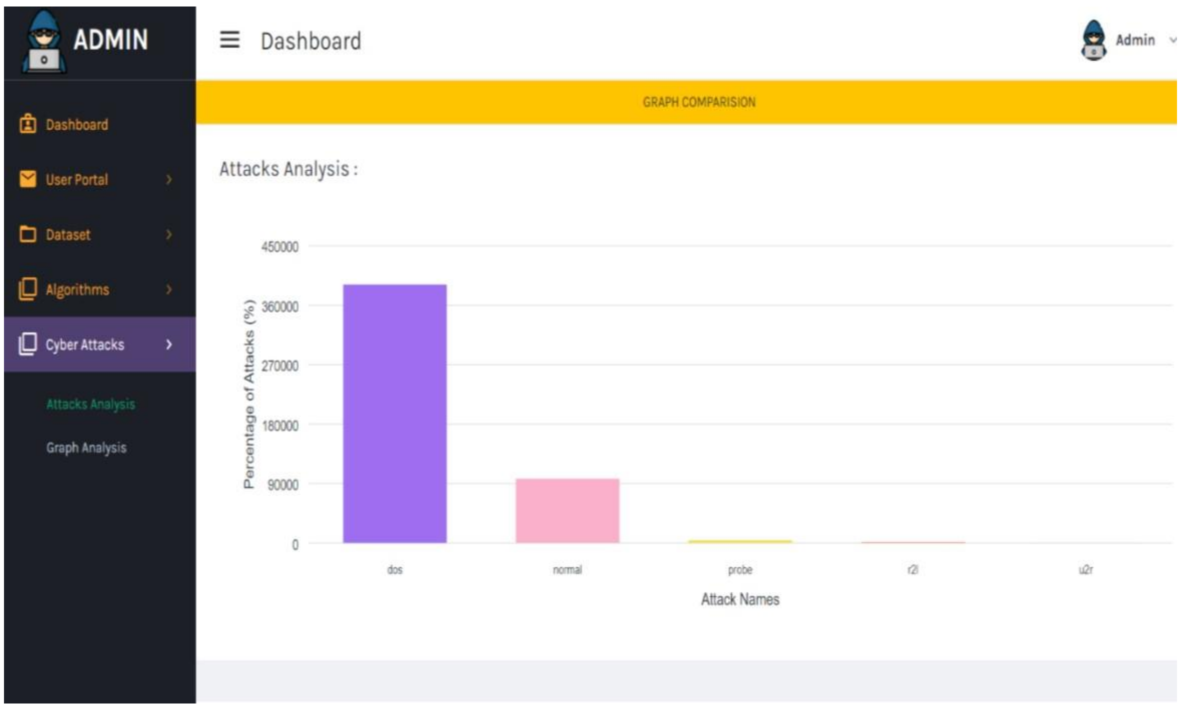


Figure 7.19 Attack Analysis

7.5 Algorithm Comparison

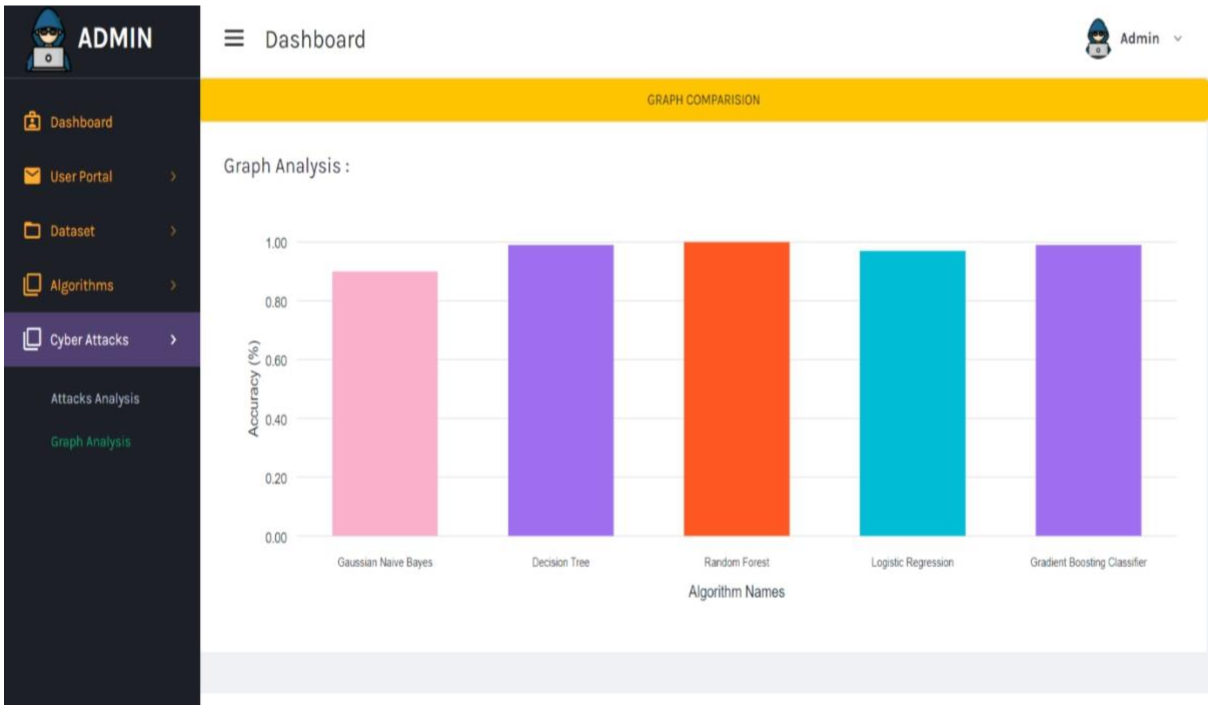


Figure 7.20 Accuracy Analysis of the Machine Learning Algorithms

CHAPTER 8

TESTING

8.1 Test Cases

Testing is a critical phase in the development cycle aimed at identifying and rectifying faults and errors within a system or work product. This multifaceted process scrutinizes the functionalities of individual components to ascertain that they operate as intended and adhere to specified requirements without failure during execution.

Various types of tests are employed, each tailored to address specific testing requirements. Unit testing involves the meticulous design of test cases to validate the internal logic of programs, ensuring that they function correctly and generate accurate outputs in response to given inputs.

Test cases: These set of test cases encompasses various implementation scenarios.

Test Case ID	Test Scenario	User Action	Expected Result	Actual Result	Remarks
1	Login (Admin)	Enter Credentials.	Login into the System.	Successfully logged in.	Pass
2	Upload Dataset (Admin)	Upload the CSV File.	Upload the File.	Successfully Uploaded the Dataset.	Pass
3	Run Algorithms (Admin)	Check Algorithm's Performance	Display Evaluation parameters.	Successfully Displayed Evaluation Results.	Pass
4	Registration (User)	User Registration into the system.	Register into the system.	Successfully registered.	Pass
5	Login (User)	Enter Credentials.	Login into the System.	Successfully logged in.	Pass
6	Detect Intrusion (User)	Enter the Feature Set values.	Detect the type of Intrusion.	Successfully Detected the type of Intrusion.	Pass

Table 8.1 Test Cases

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 Conclusion

This project represents a significant stride towards fortifying network security through the development of an intelligent Intrusion Detection System (IDS). Leveraging sophisticated machine learning techniques including Gaussian Naive Bayes, Decision Trees, Logistic Regression, Random Forest, and Gradient Classifier, our IDS seeks to address the limitations of existing systems in identifying novel threats and handling complex data efficiently.

The emphasis on employing the Random Forest algorithm within our proposed IDS showcases its prowess in handling high-dimensional datasets and noisy data, making it a compelling choice for intrusion detection in dynamic network environments. By harnessing ensemble learning, this system is designed to analyse diverse network traffic, distinguishing normal behaviour from potential intrusions with enhanced accuracy.

Our envisioned IDS involves a comprehensive approach encompassing data pre-processing, feature engineering, model development, and real-time implementation. Through rigorous model training, parameter optimization, and continuous monitoring, our system aspires to establish a robust defence mechanism adept at detecting and mitigating diverse intrusion attempts.

The integration of Random Forest into our IDS aims to elevate detection precision, minimize false alarms, and strengthen network security against evolving cyber threats. Our research endeavours to not only evaluate the efficacy of the Random Forest algorithm in handling complex network data but also optimize the IDS for scalability, computational efficiency, and real-time detection in dynamic network settings.

Ultimately, this project represents a step forward in the development of an adaptive and reliable IDS, contributing to the ongoing efforts to safeguard networks from evolving cyber threats.
Through the integration of advanced machine learning techniques, particularly the robustness of Random Forest, our IDS stands poised to proactively combat emerging cyber threats, ensuring the resilience and security of modern network infrastructures.

9.2 Future Enhancement

The proposed system's reliability and efficiency can indeed be significantly enhanced by integrating additional machine learning algorithms alongside the existing ones. This expansion would not only facilitate easier detection of intrusions but also broaden the classification spectrum to encompass various types of attacks beyond the current scope. By incorporating diverse algorithms and categorizing different attack types as intrusion classes, the system can effectively identify a wider array of threats, thereby bolstering security measures and reliability. Consequently, further advancements in the system's development hold the potential to elevate detection rates significantly while concurrently reducing the occurrence of false positives, thus amplifying overall effectiveness in safeguarding against cyber threats.

This integration of multiple machine learning algorithms not only enhances the system's capability to adapt to evolving threats but also ensures a more comprehensive approach to intrusion detection, ultimately fortifying the defence mechanisms against sophisticated attacks in the ever-changing cybersecurity landscape.

Moreover, the integration of additional machine learning algorithms can contribute to the system's resilience against adversarial attacks and evasion techniques employed by cybercriminals. By leveraging a diverse range of algorithms, the system can detect anomalies and patterns that may go unnoticed by traditional methods, thereby staying ahead of emerging threats. Furthermore, continuous learning and refinement of the algorithms through feedback mechanisms enable the system to evolve dynamically, adapting to new attack vectors and improving its efficacy over time. This iterative process of enhancement ensures that the IDS remains robust and effective in safeguarding sensitive data and critical infrastructure against the ever-evolving cybersecurity landscape.

BIBLIOGRAPHY

[1] Chen, L.; Kuang, X.; Xu, A.; Suo, S.; Yang, Y. A Novel Network Intrusion Detection System Based on CNN. In Proceedings of the 2020 Eighth International Conference on Advanced Cloud and Big Data (CBD), Taiyuan, China, 5–6 December **2020**; pp. 243–247.

[2] Chen, Y.; Yuan, F. Dynamic detection of malicious intrusion in wireless network based on improved random forest algorithm. In Proceedings of the 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, 14–16 April **2022**; pp. 27–
32.

[3] Kurniawan, Y.; Razi, F.; Nofiyati, N.; Wijayanto, B.; Hidayat, M. Naive Bayes modification for intrusion detection system classification with zero probability. *Bull. Electr. Eng. Inform.* **2021**, *10*, 2751–2758.

[4] Gu, J.; Lu, S. An effective intrusion detection approach using SVM with naïve Bayes feature embedding. *Comput. Secur.* **2021**, *103*, 102158.

[5] Wisanwanichthan, T.; Thammawichai, M. A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM. *IEEE Access* **2021**, *9*, 138432–138450.

[6] Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Xu, M. A Survey on Machine Learning Techniques for Cyber Security in the Last Decade. *IEEE Access* **2020**, *8*, 222310–222354.

[7] Arora, P.; Kaur, B.; Teixeira, M.A. Evaluation of Machine Learning Algorithms Used on Attacks Detection in Industrial Control Systems. *J. Inst. Eng. (India) Ser. B* **2021**, *102*, 605–616.

[8] Md Nasimuzzaman Chowdhury and Ken Ferens, Mike Ferens1Department of Electrical and Computer Engineering University of Manitoba Winnipeg, Manitoba,Canada February **2020.**

[9] Jiyeon Kim , Jiwon Kim , Hyunjung Kim Minsun Shim and Eunjung Choi. 1 June **2020**.

[10] Prashanth, S.K.; Shitharth, S.; Praveen Kumar, B.; Subedha, V.; Sangeetha, K. Optimal Feature Selection Based on Evolutionary Algorithm for Intrusion Detection. *SN Comput. Sci.* **2022**, *3*, 439.

[11] Khan S., Sivaraman E., Honnavalli P.B. Performance evaluation of advanced machine learning algorithms for network intrusion detection system.
Dutta M., Krishna C.R., Kumar R., Kalra M. (Eds.), Proc. Int. Conf. IoT Incl. Life, Springer
Singapore, Singapore, NITTTR Chandigarh, India (**2020**), pp. 51-59