International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Intrusion Detection Using Machine Learning Classification and Regression

Pinaki Shashishekhar Mathan

B.Tech (Hons)
Computer Science & Engineering
OmDayal Group of Institutions, Uluberia, Howrah, West Bengal, India

---

Abstract - An Intrusion Detection System (IDS) is a crucial security mechanism designed to protect computer networks from unauthorized access and cyber threats. With the rapid expansion of Internet-based data transmission, ensuring network security has become increasingly challenging. IDS continuously monitors and analyzes network traffic to detect malicious activities, relying on datasets like KDD Cup 1999 for training and evaluation. Effective IDS development involves preprocessing steps such as feature selection, normalization, and addressing data imbalance to enhance detection accuracy. Various machine learning techniques, including Decision Trees, Support Vector Machines, Neural Networks, Bayesian Networks, and ensemble methods, are employed to classify network traffic as normal or malicious. IDS performance is assessed using accuracy, precision, recall, and F1-score, with cross-validation and hyperparameter tuning improving model robustness. Key challenges include handling dynamic network traffic, achieving real-time scalability, and minimizing false positives and false negatives. As cyber threats continue to evolve, advancements in artificial intelligence and deep learning are driving the development of adaptive IDS capable of detecting and responding to emerging attacks in real time.

Keywords: Network Security, KDD Cup 1999 dataset, Machine Learning, Data Mining, Anomaly Detection, Cybersecurity, Preprocessing, Classification Algorithms, Accuracy Metrics, Model Validation, Ensemble Methods, False Positives, False Negatives, Scalability, Dynamic Network Traffic, Hyperparameter Tuning, Real-time Monitoring, Evolving Intrusion Tactics

1. INTRODUCTION

An Intrusion Detection System (IDS) is a security mechanism that monitors network traffic for malicious activities and potential threats. It detects unauthorized access attempts and alerts administrators or a centralized security system for immediate response. IDS can be classified into signature-based and anomaly-based detection methods, each with its strengths and challenges. While signature-based IDS detects known threats, anomaly-based IDS can identify new and evolving attack patterns. As cyber threats grow in complexity, IDS plays a critical role in ensuring network security by providing real-time monitoring and threat detection capabilities.

Researchers have explored deep learning approaches such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to improve IDS performance. These models can automatically learn complex patterns in network traffic data, reducing the need for manual feature selection. However, computational complexity and the need for extensive training data remain challenges in deploying deep learning-based IDS in real-world environments.

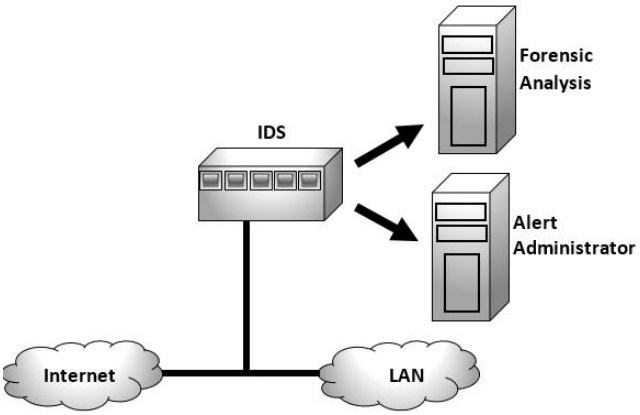


Fig -1: Intrusion Detection System (IDS)

1.1 Importance and Working

The field of intrusion detection has been extensively researched, with numerous studies focusing on improving detection accuracy and minimizing false alarms. Traditional IDS methods rely on signature-based detection and anomaly-based detection. Signature-based IDS identifies known attack patterns but struggles to detect new threats, whereas anomaly-based IDS monitors deviations from normal behavior and can detect previously unknown attacks. However, anomaly-based methods often generate higher false positive rates.

Recent advancements in machine learning have significantly contributed to the development of more effective IDS solutions. Various classification techniques, including Bayesian networks, Naïve Bayes classifiers, decision trees, k-nearest neighbors (KNN), and ensemble methods, have been applied to enhance intrusion detection capabilities. The KDD Cup 1999 dataset remains one of the most widely used benchmarks for IDS evaluation, providing a rich set of network traffic data to train and test machine learning models.

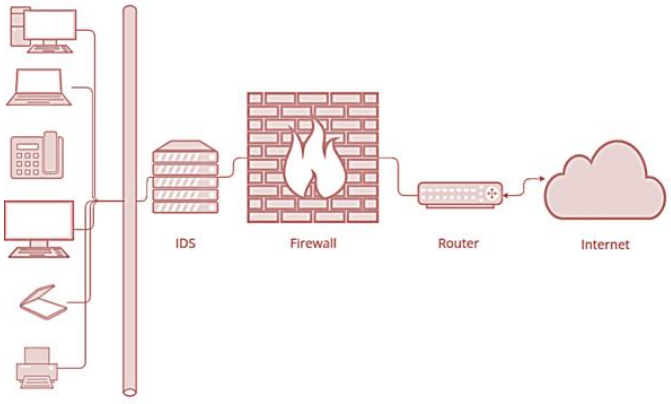


Fig -2: Diagram depicting the functionality of an IDS

1.2 Classification

a. A **Network Intrusion Detection System (NIDS)** monitors network traffic at a strategic point, analyzing data packets and comparing them with known attack patterns. If suspicious activity is detected, it alerts the administrator. For example, it can be placed near a firewall to detect unauthorized access attempts.
b. A **Host Intrusion Detection System (HIDS)** runs on individual devices, monitoring network traffic and system files for unauthorized changes. It takes periodic snapshots of files and alerts administrators if modifications occur. This is useful for critical systems that should remain unchanged.
c. A **Protocol-Based Intrusion Detection System (PIDS)** monitors communication protocols between users and servers, ensuring requests follow expected behavior. It is commonly used on web servers to analyze HTTPS traffic and detect protocol-based attacks.
d. An **Application Protocol-Based Intrusion Detection System (APIDS)** analyzes communication within applications, focusing on specific protocols like SQL queries in web applications. It helps detect attacks such as SQL injection by monitoring application-level interactions.
e. A **Hybrid Intrusion Detection System (Hybrid IDS)** combines multiple IDS methods for enhanced security. By integrating host and network monitoring, it provides a broader detection scope. An example is Prelude, which improves accuracy by using multiple detection techniques.

1.3 Detection Method of IDS Deployment

The **Signature-based IDS** identifies threats by matching network traffic patterns with predefined attack signatures, such as specific byte sequences or malicious instruction sets. It effectively detects known threats but struggles with new, unidentified malware due to the absence of existing signatures. The **Anomaly-based IDS** addresses this limitation by using machine learning to establish a model of normal activity. Any deviation from this model is flagged as suspicious. This approach is more adaptable than signature-based IDS, as it can be trained to detect unknown threats based on system behavior and configuration.

1.4 IDS Evasion Techniques

Intruders use various strategies to bypass intrusion detection systems (IDS), making it difficult to identify malicious activities. One such technique is **fragmentation**, where attackers split data into smaller packets that evade signature detection. These packets are later reassembled at the destination, allowing the attack to go unnoticed.
Another approach is **flooding**, which overwhelms IDS by generating excessive traffic, leading to failure in detection mechanisms. Attackers often exploit protocols like UDP and ICMP to disguise harmful activities within the flood of data.
Obfuscation is another method where attackers modify code or data to make it harder to interpret, reducing the effectiveness of reverse engineering or static analysis. Similarly, **encryption** helps conceal malicious activities by securing data in a way that prevents IDS from analyzing its contents.
Attackers can also manipulate **source routing**, forcing packets to take specific paths that avoid IDS monitoring points. Additionally, **source port manipulation** exploits security loopholes in improperly configured IDS, allowing malicious traffic to pass unchecked through commonly trusted ports like port 80.

2. PROBLEM STATEMENT

Intrusion detection plays a crucial role in cybersecurity, stepping in where firewalls fall short. While preventing unauthorized access is ideal, it is not always feasible, making real-time monitoring essential for identifying vulnerabilities and ongoing attacks. A reliable, accurate, and secure intrusion detection system (IDS) is necessary to ensure robust protection. However, a major challenge with existing IDS technologies is filtering out false alarms, which can overwhelm security teams. IDS continuously monitors and analyzes system or network events to detect security violations or threats. Meanwhile, intrusion prevention goes a step further by actively blocking identified threats.
Intrusion Detection and Prevention Systems (IDPS) not only detect and log incidents but also attempt to mitigate attacks and alert security administrators. Organizations use IDPS for policy enforcement, threat documentation, and deterrence against security breaches. Given the growing complexity of cyber threats, there is a pressing need to develop a stronger and more efficient detection mechanism to enhance network security.

3. LITERATURE SURVEY

Nilamadhab Mishra, Sarojananda Mishra, Engineering, Biju Patnaik University of Technology, Rourkela, Odisha, India [1] A network intrusion occurs when an unauthorized entity gains access to a computer network, posing a threat to data security and system integrity. The primary objective of intrusion detection is to safeguard networks from unauthorized access, including threats from both external and internal users. To achieve this, a local network discovery mechanism is essential to differentiate between normal network activity and potentially harmful intrusions.
Machine learning has emerged as a powerful tool for intrusion detection, enabling automated classification of network traffic. By analyzing patterns in data, machine learning models can effectively distinguish between legitimate connections and suspicious activities. The focus is on developing classification techniques that enhance both training and testing processes, ensuring improved accuracy and efficiency.
Various machine learning algorithms have been explored to identify the most effective model for intrusion detection, considering factors such as detection time and accuracy. By comparing multiple classification methods, it is possible to determine the best-performing approach for securing computer networks. The integration of machine learning into intrusion detection systems enhances their ability to detect and respond to cyber threats in real time, making networks more resilient against attacks.

Bisyron Wahyudi, Kalamullah Ramli, and Hendri Murfi, Universitas Indonesia, Indonesia [2] Intrusion Detection Systems (IDS) play a crucial role in network security by identifying and mitigating cyber threats. Machine learning has been widely adopted to enhance IDS accuracy, making the selection of suitable methods essential.
This research presents an IDS built using a machine-learning approach, utilizing a 28-feature subset from the Knowledge Discovery in Databases (KDD) dataset, excluding content features. The model demonstrates 99.9% accuracy in both binary and multiclass classification. Experimental results confirm its effectiveness in detecting real-world network attacks.

Ahmed M. Mahfouz, Deepak Venugopal, and Sajjan G. Shiva, The University of Memphis, Memphis TN 38152, USA [3] As network-based applications expand rapidly, new security challenges emerge, requiring enhanced mechanisms for speed and accuracy. Despite advancements in security tools, the rise of sophisticated cyber threats continues to pose risks. Intrusion Detection Systems (IDS) play a crucial role in identifying malicious activities within network traffic. Machine learning has become a key approach in IDS by distinguishing between normal and abnormal traffic patterns. However, a thorough evaluation of machine learning algorithms for intrusion detection remains limited.
This study presents a detailed analysis of machine learning classifiers in detecting network intrusions. It explores various aspects such as feature selection, hyperparameter sensitivity, and class imbalance—factors critical to IDS performance. Using the NSL-KDD dataset, we assess multiple classifiers through extensive experimentation to determine their effectiveness in intrusion detection.

Sapna S. Kaushik, Dr. Prof.P.R.Deshmukh, M.E. II Year, Computer Science and Engg., Sipna College of Engg. Amravati, INDIA [4] Intrusion detection involves identifying unauthorized or malicious activities within a network or device. An Intrusion Detection System (IDS) acts as a security layer that continuously monitors network traffic for suspicious patterns and alerts administrators when potential threats are detected. For an IDS to be effective, it must accurately identify threats while handling large volumes of network data efficiently.
Network-based IDS is one of the most widely used detection systems, deployed either as software or a dedicated hardware appliance. Many IDS solutions not only generate real-time alerts but also log detected events for further analysis, allowing security teams to refine policies and mitigate risks. This study examines different types of cyber attacks that can be detected in a simulated network environment, including Probe attacks, Remote-to-Local (R2L) intrusions, Denial-of-Service (DoS) attacks, and User-to-Root (U2R) exploits.

Kajal Rai, M. Syamala Devi, Ajay Guleria, Panjab University, Chandigarh, India [5] An Intrusion Detection System (IDS) plays a crucial role in monitoring computer network activities and identifying unauthorized access attempts that could compromise an organization's data security. With the increasing sophistication of cyber threats, ensuring robust network protection has become a priority for organizations. IDS can generally be categorized into signature-based and anomaly-based systems, each serving a unique role in identifying threats. In this study, a decision tree-based intrusion detection approach is developed using the C4.5 algorithm, which effectively addresses key challenges such as feature selection and split value determination. The model selects the most relevant features through information gain and determines split values to ensure an unbiased classification process. Experimental evaluation is conducted using the NSL-KDD dataset, analyzing the impact of selected features on detection accuracy and model training time. The results demonstrate that the proposed Decision Tree Split (DTS) algorithm effectively enhances signature-based intrusion detection, providing a reliable mechanism for identifying and mitigating network threats.

Yasmeen S. Almutairi, Bader Alhazmi, Amr A. Munshi, Computer Engineering Department, Umm Al-Qura University, Makkah 21961, Saudi Arabia [6] Intrusion Detection Systems (IDS) play a critical role in securing modern communication networks by identifying suspicious activities and preventing potential threats. Traditional IDS primarily relied on predefined signatures and rule-based detection, but with the rise of sophisticated cyberattacks, machine learning and deep learning techniques have emerged as powerful alternatives. These intelligent models can effectively differentiate between normal and anomalous network behavior, enhancing detection capabilities. In this study, the NSL-KDD dataset is utilized to evaluate the performance of various machine learning algorithms, including Support Vector Machine, J48, Random Forest, and Naïve Bayes, in both binary and multi-class classification tasks. The experimental results demonstrate the effectiveness of these approaches in improving network security, with notable advancements over previous methodologies. The findings highlight the potential of machine learning-driven IDS to enhance threat detection and response mechanisms in evolving network environments.

Ansam Khraisat, Iqbal Gondal, Peter Vamplew and Joarder Kamruzzaman [7] With the increasing complexity of cyber-attacks, detecting intrusions accurately has become a significant challenge. Failure to identify and prevent these attacks can compromise essential security services such as data confidentiality, integrity, and availability. To address these threats, various intrusion detection methods have been developed, primarily categorized as Signature-based Intrusion Detection Systems (SIDS) and Anomaly-based Intrusion Detection Systems (AIDS). SIDS rely on predefined attack signatures, while AIDS detect deviations from normal behavior, making them more effective against unknown threats. This study provides an in-depth analysis of modern IDS, reviewing recent advancements and widely used evaluation datasets. It also explores evasion strategies employed by attackers to bypass detection and highlights future research directions to strengthen security mechanisms. By enhancing detection techniques and countering evasion tactics, IDS can play a crucial role in safeguarding computer systems against evolving cyber threats.

Rachid Tahri, Youssef Balouki, Abdessamad Jarrar, and Abdellatif Lasbahani, National School of Applied Sciences, Sultan Moulay Slimane University, Bni Mellale [8] The internet has revolutionized the way people interact, offering seamless communication and connectivity. While it enables individuals to maintain social and professional networks, it also exposes them to security risks that can compromise personal and organizational data. As digital dependency continues to grow, safeguarding sensitive information becomes increasingly critical. Intrusion Detection Systems (IDS) serve as a crucial defense mechanism against cyber threats by continuously monitoring network traffic for suspicious activities and issuing alerts when potential attacks are detected. This study explores the effectiveness of machine learning algorithms in enhancing IDS performance, focusing on Naïve Bayes (NB), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Initially, the accuracy of these algorithms is evaluated using the UNSW-NB15 dataset to determine the most effective model. The selected algorithm is then further analyzed using additional datasets, including NSL-KDD and UNSW-NB15, to validate its reliability. By comparing multiple datasets and refining the

detection model, this research aims to improve intrusion detection accuracy and strengthen cybersecurity defenses.

Qadeer, Mohammed & Iqbal, Arshad & Zahid, Mohammad & Siddiqui, Misbahur, Communication Software and Networks, International Conference [9] A packet sniffer is a specialized software tool designed to capture, analyze, and log network traffic as it traverses a digital network. It functions by setting the Network Interface Card (NIC) into promiscuous mode, allowing it to intercept data packets flowing through the network. Once captured, these packets are decoded to extract valuable information, which can be utilized for various purposes, depending on the user's intent. The extent of network traffic that can be intercepted varies based on the network's architecture, as some configurations allow access to all packets, while others restrict visibility to a subset of traffic.

In certain cases, network switches may limit packet sniffing capabilities, but techniques exist to bypass these restrictions and gain access to data from multiple systems. This research focuses on the working principles of packet sniffers, their development on a Linux platform, and their application in Intrusion Detection Systems (IDS). Additionally, methods for detecting unauthorized sniffing activities and mitigating their impact are explored. A self-developed packet sniffer has been designed to analyze network performance, identify bottlenecks, and enhance security monitoring.

Before creating this tool, an in-depth analysis of existing sniffing software such as Wireshark, tcpdump, and Snort was conducted to understand their functionality and limitations. The libpcap library has been utilized for packet capturing, providing a foundation for efficient data collection. The development of this packet sniffer presents an opportunity to integrate additional security features beyond those available in conventional sniffing tools, improving network analysis and cybersecurity measures.

3.METHODOLOGY

3.1 Proposed System

A smart Intrusion Detection System (IDS) plays a crucial role in safeguarding networks from external threats by identifying malicious activities in real time. Traditional IDS models often struggle with detecting new attack patterns and suffer from high computational overhead when processing large volumes of audit data. To overcome these challenges, machine learning techniques have emerged as powerful tools for improving detection accuracy and efficiency.

The proposed IDS leverages Decision Tree, Logistic Regression, Random Forest, and K-Nearest Neighbors (KNN) algorithms to enhance pattern recognition and intrusion detection capabilities. These models are designed to analyze network traffic, classify anomalies, and distinguish between legitimate and malicious activities with greater precision. Unlike conventional classification methods, these machine learning techniques offer superior performance in handling small sample sizes, nonlinear patterns, and high-dimensional data.

By applying these algorithms, the system achieves a balance between accuracy and computational efficiency, making it suitable for real-time security monitoring. The integration of multiple classification techniques ensures robustness against evolving cyber threats while minimizing false positives. This approach provides an effective solution for improving network security, enhancing threat detection, and reducing the limitations faced by traditional IDS models.

3.2 Significance of IDSs

To protect against cyber threats, security systems typically rely on tools such as firewalls, access control mechanisms, and other protective measures. However, past incidents, such as the spread of internet worms and malware like "I Love You" or policy exploits, have demonstrated that existing security frameworks are not entirely foolproof. Despite the deployment of advanced safety measures, vulnerabilities persist, making systems susceptible to potential breaches.

Implementing robust security mechanisms with strong cryptographic techniques can significantly mitigate risks, but complete prevention of intrusions remains unrealistic. Achieving an entirely secure system requires software to be free of flaws and administrators to continuously refine security policies for every process executed within the system. Even with the best security tools, human factors remain a critical weak link. A highly encrypted system can be compromised if access credentials are carelessly stored, and insiders with legitimate access may exploit their privileges. Moreover, stringent security protocols often come at the cost of system efficiency. Lengthy passwords may delay user access, and complex encryption algorithms can slow down system performance. Studies have shown that even well-documented vulnerabilities remain exploitable for long periods after patches are released.

Given these challenges, intrusions are possible even in highly secure environments. When a breach occurs, a resilient system must be capable of responding swiftly by capturing audit data related to the attack. This information is crucial for preventing similar future threats and identifying potential attackers. While Intrusion Detection Systems (IDS) play a vital role in identifying security breaches, they often focus on monitoring rather than taking immediate corrective actions. Strengthening detection and response mechanisms is essential to ensuring continuous protection against evolving cyber threats.

3.3 Methods of Intrusion Detection Systems

Intrusion Detection Systems (IDS) can be categorized based on their detection methods, primarily into misuse detection, anomaly detection, and hybrid approaches. Each of these methods operates differently, offering various advantages and limitations in identifying potential security breaches.

Misuse detection works by recognizing specific attack patterns or vulnerabilities that have been previously identified. This approach relies on predefined signatures, which are constructed by analyzing known system weaknesses and common attack methods. Security analysts gather data from different sources to build these signatures, ensuring that IDS can efficiently detect recognized threats. However, one of the primary challenges of misuse detection is its reactive nature—it cannot identify novel attacks until a corresponding signature is developed. This means an intrusion could occur, and the attacker could accomplish their goal before the IDS recognizes

the threat. While signature-based systems are highly reliable in detecting known attacks, they fall short in identifying new or evolving threats.

On the other hand, anomaly detection focuses on establishing a baseline of normal system behavior and then identifying deviations from this norm. By continuously monitoring network activity or system operations, anomaly-based IDS can detect unusual patterns that may indicate an intrusion. This method is particularly effective in uncovering unknown attacks, as it does not rely on predefined signatures. However, one major limitation is its inability to precisely classify the type of attack occurring. Unlike misuse detection, which directly links an attack to a signature, anomaly-based systems only alert security personnel to irregular activity, requiring further investigation to determine the nature of the threat.

A significant challenge in intrusion detection is dealing with errors in classification, particularly false positives and false negatives. A false positive occurs when an IDS mistakenly flags legitimate activity as a threat. For example, if a user mistypes their password multiple times, an anomaly-based system might interpret this as a brute-force attack, triggering an alert unnecessarily. While signature-based IDS generally produce fewer false positives, anomaly-based systems are more prone to such errors because they flag deviations from normal behavior, even if those deviations are harmless.

Conversely, a false negative happens when an IDS fails to detect an actual attack. This can occur when an attacker carefully operates below a detection threshold. For instance, if an IDS is set to detect brute-force attempts after ten failed login attempts within a minute, but an attacker attempts only nine, the system might not recognize the attack. False negatives are particularly dangerous because they allow intrusions to go unnoticed.

To measure the effectiveness of an IDS, four key scenarios are considered. A true positive occurs when the IDS correctly identifies a real attack, allowing security personnel to respond appropriately. A false positive happens when an IDS incorrectly triggers an alert for benign activity. A false negative is when an actual attack goes undetected, leaving the system vulnerable. Finally, a true negative refers to a scenario where no attack occurs, and the IDS correctly refrains from raising an alarm.

	True	False
Positive	*True-Positive* An alarm was generated, and a present condition should be alarmed	*False-Positive* An alarm was generated, and no condition was present to generate it
Negative	*True-Negative* An alarm was not generated, and there is no present condition that should be alarmed	*False-Negative* An alarm was not generated, and a condition was present that should be alarmed

Fig-3: False Positives and False Negatives

3.4 Architecture of the System

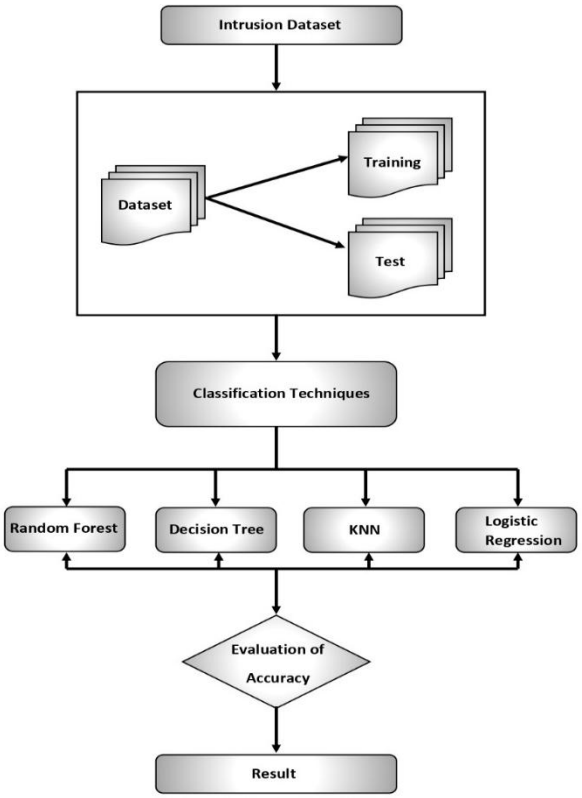


Fig-4: System Architecture

3.5 Implementation of the System

The implementation of the proposed intrusion detection system is carried out using Python 3.11 and Jupyter Lab. Several libraries, including scikit-learn, pandas, and matplotlib, are utilized along with other necessary libraries to facilitate data processing, visualization, and model training. The dataset for this study is obtained from the KDD dataset repository available at kdd.ics.uci.edu. This dataset consists of separate train and test sets, each containing four distinct classes of intrusions. The training dataset is used to train the model, while the test dataset is utilized for evaluation.

In the initial phase, data collection plays a crucial role in ensuring the availability of high-quality data for analysis. The dataset selected from the KDD repository is well-suited for implementing machine learning models to detect intrusions. A significant responsibility in data analysis is to identify appropriate sources of data, gather relevant information, interpret findings, and apply statistical techniques to extract meaningful insights.

To make complex data more comprehensible, data visualization techniques are employed. Representing large volumes of data through graphical means simplifies the interpretation process. In this approach, the detection rates of intrusions are illustrated using various visualization methods, making it easier to analyze the system's performance.

Before applying machine learning techniques, the dataset undergoes preprocessing. Raw data must be transformed into a structured format to ensure accurate predictions. Data preprocessing involves cleaning, formatting, and sampling, which eliminates inconsistencies and enhances the model's

effectiveness. Well-prepared data leads to better performance of the learning algorithm.

A crucial step in machine learning is dataset splitting, which divides the data into separate subsets to train and evaluate the model effectively. The dataset is partitioned into training, testing, and validation sets. The training set is used to train the model and optimize its parameters. The test set is essential for assessing the model's performance and its ability to generalize to new data. Separating training and testing data prevents overfitting, which occurs when a model performs well on training data but fails to recognize patterns in unseen data.

Once the data is prepared and split, model training is initiated. The machine learning models used in this approach include decision tree, regression, random forest, and KNN. During training, the selected algorithm processes the training data to learn patterns and generate a model capable of predicting target values. This step is crucial in developing an intrusion detection system that can identify potential threats accurately.

The final stage of the process is model evaluation and testing. The objective is to create an efficient model that can quickly and accurately identify intrusions. Model tuning is performed to optimize parameters and enhance performance. It is important to note that the test data differs from the training data in terms of probability distribution and contains specific attack types that are not present in the training set. This variation makes the detection task more realistic. Experts suggest that many novel attacks are modified versions of known intrusions, meaning that recognizing the signature of known attacks can help identify new threats. The dataset consists of 24 attack types in the training set, with an additional 14 attack types exclusive to the test set.

By implementing machine learning models and systematically analyzing intrusion patterns, this approach aims to develop an effective intrusion detection system capable of identifying both known and previously unseen threats.

Feature Name	Description	Type
duration	The total time (in seconds) for which the connection remained active.	Continuous
protocol_type	Specifies the communication protocol used, such as TCP, UDP, or ICMP.	Discrete
service	Represents the type of network service requested at the destination, such as HTTP or FTP.	Discrete
src_bytes	The amount of data (in bytes) sent from the source to the destination.	Continuous
dst_bytes	The amount of data (in bytes) received at the source from the destination.	Continuous
flag	Describes the connection's status, indicating whether it is normal or has errors.	Discrete
land	Identifies whether the connection originates and terminates on the same host and port (1 for yes, 0 for no).	Discrete
wrong_fragment	Counts the number of incorrectly fragmented packets in the connection.	Continuous
urgent	Represents the count of urgent packets sent during the connection.	Continuous

Table-1: Network Connection Features

Feature Name	Description	Type
hot	Total count of "hot" indicators, representing potentially suspicious actions.	Continuous
num_failed_logins	Number of unsuccessful login attempts recorded.	Continuous
logged_in	Indicates whether the user successfully logged in (1 for yes, 0 for no).	Discrete
num_compromised	Total number of conditions where a system compromise has been detected.	Continuous
root_shell	Indicates whether a root shell was obtained during the session (1 for yes, 0 for no).	Discrete
su_attempted	Shows if the "su root" command was attempted (1 for yes, 0 for no).	Discrete
num_root	Total number of times root-level access was used.	Continuous
num_file_creations	Number of times a file was created during the session.	Continuous
num_shells	Total count of shell prompts opened.	Continuous
num_access_files	Number of operations performed on access control files.	Continuous
num_outbound_cmds	Count of outbound commands executed in an FTP session.	Continuous
is_hot_login	Indicates if the login belongs to a predefined "hot"	Discrete

	list of users (1 for yes, 0 for no).	
is_guest_login	Identifies whether the login was performed using a guest account (1 for yes, 0 for no).	Discrete

Table-2: User Behavior and Authentication Features

Feature Name	Description	Type
count	Represents the number of connections made to the same host as the current one within the past two seconds.	Continuous
serror_rate	Indicates the percentage of connections that have encountered "SYN" errors.	Continuous
rerror_rate	Shows the percentage of connections that resulted in "REJ" errors.	Continuous
same_srv_rate	Represents the percentage of connections made to the same service.	Continuous
diff_srv_rate	Indicates the percentage of connections directed to different services.	Continuous
srv_count	Refers to the number of connections made to the same service as the current one within the past two seconds.	Continuous
srv_serror_rate	Displays the percentage of connections that have experienced "SYN" errors in same-service connections.	Continuous
srv_rerror_rate	Represents the percentage of same-service connections that resulted in "REJ" errors.	Continuous
srv_diff_host_rate	Indicates the percentage of connections that were directed to different hosts.	Continuous

Table-3: Traffic Features Based on a Two-Second Time Window

3.6 Implementation of Machine Learning Algorithms

A. Logistic Regression Algorithm

Logistic Regression is a widely used technique in predictive modeling that helps classify data into distinct categories. It is a powerful yet simple supervised learning algorithm that deals with classification problems. Unlike linear regression, which predicts continuous values, logistic regression is specifically designed to predict the probability of an outcome that falls into one of two categories.

This algorithm is commonly applied in situations where the output variable is discrete, meaning it can take only specific values such as "yes" or "no," "true" or "false," or "0" and "1." Approximately 60% of classification problems worldwide can be effectively addressed using logistic regression. The core idea behind this algorithm is to establish a relationship between input features and a probability score, which helps determine the most likely category for a given data point.

One of the primary applications of logistic regression is binary classification, where it estimates the probability of an event occurring using a mathematical function known as the logit function. This makes logistic regression a special type of linear regression, except that it applies a transformation using the log function to limit the predicted values within a range of 0 to 1.
In a simpler way, linear regression estimates a continuous value based on an input variable, and this input variable is known as the predictor variable. The outcome being predicted is referred to as the criterion variable. When there is only one predictor variable, the method used is called simple regression. Logistic regression, however, extends this idea by predicting probabilities rather than direct numerical values.

To convert the predicted probabilities into categorical values, an activation function known as the sigmoid function is used. This function maps any real-valued number into a range between 0 and 1, forming an S-shaped curve. The sigmoid function plays a crucial role in determining the classification outcome. If the probability score is greater than 0.5, the data point is classified into one category (often labeled as Class 1 or the Positive Class), while if it is less than 0.5, the data point is classified into the other category (Class 0 or the Negative Class).

There are numerous practical applications of logistic regression across different fields. It is extensively used for fraud detection, where it helps identify fraudulent transactions based on patterns in transaction data. In spam detection, logistic regression is used to filter out unwanted emails by analyzing email content and sender behavior. It is also valuable in medical diagnosis, such as cancer detection, where it helps predict the likelihood of a disease based on medical test results.

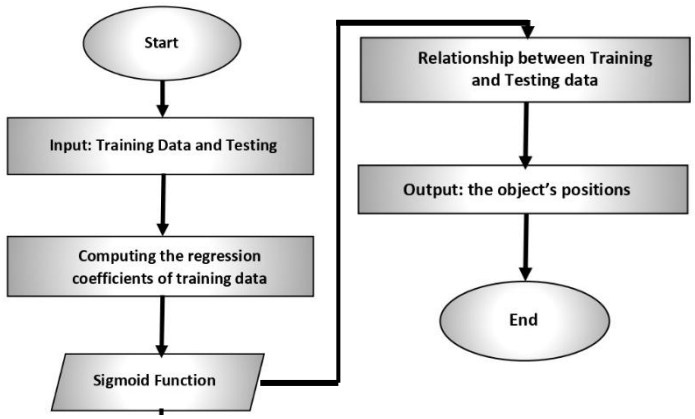


Fig-5: Flow chart of Logistic Regression algorithm

B. Decision Tree Algorithm

A decision tree is a supervised learning algorithm that is primarily used for classification tasks, but it can also handle regression problems. This method is effective for both

categorical and continuous input and output variables. The fundamental concept behind a decision tree is to break down a dataset into smaller and more homogeneous subsets based on the most important attribute, creating a structured flow of decision-making.

The decision tree consists of different elements, each playing a crucial role in classification. The internal nodes represent conditions or tests performed on specific attributes, the branches illustrate possible outcomes of those tests, and the leaf nodes signify the final classification or prediction. The primary objective of constructing a decision tree is to develop a model that can classify new data or predict values based on patterns identified from previous training data.

One of the key advantages of a decision tree is its simplicity and ease of understanding when compared to other classification algorithms. It organizes the data in a hierarchical tree-like structure, making it easier to interpret and analyze decision-making patterns. The tree is constructed by following a step-by-step process that involves selecting the most influential attribute and splitting the dataset accordingly.
To build a decision tree, the process begins by identifying the most significant attribute from the dataset and placing it at the root. The dataset is then divided into smaller subsets in such a way that each subset consists of data points with a common attribute value. This splitting process continues recursively for each subset until all branches of the tree reach leaf nodes, which represent final classifications.

When using a decision tree for classification, the process starts at the root node, where the algorithm compares the attribute value from the input data with the conditions in the tree. Based on the comparison, the data follows the appropriate branch and moves to the next node. This process is repeated until a leaf node is reached, where the final classification is determined.

Decision trees are widely used across various applications due to their interpretability and effectiveness. They play a crucial role in areas such as medical diagnosis, financial risk assessment, and fraud detection. Their ability to provide clear decision-making paths makes them an essential tool in machine learning and data analysis.

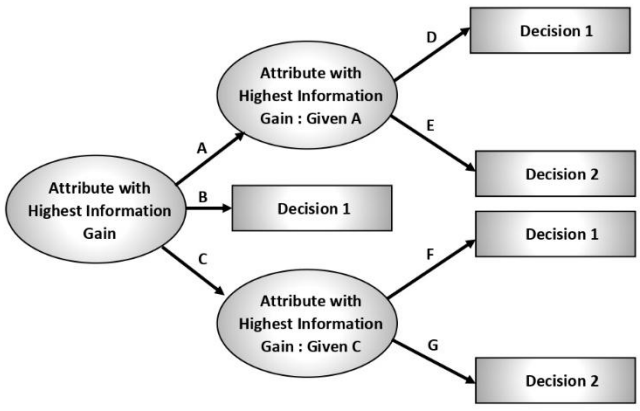


Fig-6: Flow chart Decision Tree algorithm

C. Random Forest Algorithm

The Random Forest algorithm is an advanced ensemble learning technique used for classification and regression problems. It is built on the concept of bagging, where multiple decision trees are trained on random subsets of the dataset to improve accuracy and reduce overfitting. Given a dataset with n instances, multiple sub-samples are selected randomly with replacement, ensuring diversity in training data for different trees in the forest.

When constructing a decision tree within the Random Forest model, a predefined number of m features are selected randomly out of k total features at each node, ensuring m is smaller than k. Among these selected features, the best possible split is chosen to create a node. This process is repeated consistently throughout the growth of the forest while keeping m unchanged. Each decision tree in the Random Forest is allowed to grow fully without any pruning, leading to deep trees capable of capturing complex patterns in the data.

For classification, a new object is predicted by aggregating the outputs from all trees in the ensemble, where the majority class label determines the final decision. This process reduces variance and improves the model's generalization ability compared to individual decision trees. The primary distinction between a single decision tree and a Random Forest lies in the randomness introduced through both bootstrap sampling and feature selection, which minimizes correlation among trees and enhances predictive performance.

In the context of intrusion detection, the dataset attributes play a significant role in determining the classification results. Since different attributes contribute variably to model accuracy, a method is used to assess their importance, increasing the likelihood of selecting the most influential attributes. Traditional methods of measuring attribute importance may not be optimal, as they often remove attributes directly, leading to information loss. Instead, hierarchical sampling is employed, where the original dataset is sampled in proportion to maintain data balance.

Given that the dataset contains continuous attributes, the data undergoes discretization using equal distance dispersion or equal frequency dispersion techniques. This transformation converts continuous values into discrete categories, making it easier for the decision trees to process the data. Self-service sampling is then performed on the preprocessed dataset, generating N subsets and corresponding out-of-bag data. These subsets serve as training samples for different trees in the forest. To improve attribute selection, an evaluation function based on Decision Boundary Entropy (DBE) is employed, with attributes categorized into positive, boundary, or negative domains based on predefined threshold values (α, β). This technique refines the feature selection process by prioritizing more informative attributes. For each tree, the square root of the total number of features ($\sqrt{k}$) is chosen according to three attribute selection rules.

The Gini index serves as the splitting criterion for node division, helping construct highly efficient decision trees. After all N trees are trained, the final classification result is obtained through majority voting, ensuring robustness against overfitting and noise. To validate the effectiveness of the Random Forest model in intrusion detection, test data is introduced into the integrated model, verifying its accuracy and reliability in detecting anomalies within network traffic.

D. K Nearest Neighbor (KNN) Algorithm

The K-Nearest Neighbors (KNN) algorithm is a widely used machine learning technique that determines the similarity between a query instance and existing instances in a dataset. The similarity between instances is measured using a distance function, which quantifies how close or far two points are in a given feature space. Given two instances, each composed of *N* features, the distance between them is computed using a chosen metric.

One of the most common distance metrics is Euclidean distance, which calculates the straight-line distance between two points in an *N*-dimensional space. Another common metric is absolute distance, also known as Manhattan distance, which sums the absolute differences between corresponding feature values. The choice of distance function depends on the nature of the data and the problem being solved. For numerical data, Euclidean distance is typically preferred, whereas for categorical or binary data, Hamming distance is often used.

Unlike traditional machine learning models that learn patterns from data, KNN does not build an explicit model during training. Instead, it memorizes the entire dataset and uses it to make predictions. When an unseen data instance needs to be classified or assigned a value, KNN searches the dataset for the *k* most similar instances based on the chosen distance function. Once the *k* nearest neighbors are identified, the final prediction is made by summarizing their attributes. For classification tasks, the most frequently occurring class among the *k* neighbors is assigned to the new instance.

KNN belongs to the category of instance-based learning algorithms, where predictions are made directly using training instances without building a generalized model. Since it stores all training data, KNN is considered a memory-intensive algorithm. It also falls under the category of competitive learning algorithms, as each data instance competes with others to be recognized as the most similar to a new instance.

Due to its simplicity and effectiveness, KNN is widely used in various applications such as pattern recognition, recommendation systems, and anomaly detection. However, its computational complexity increases with large datasets, making optimizations such as indexing structures and dimensionality reduction techniques essential for improving efficiency.

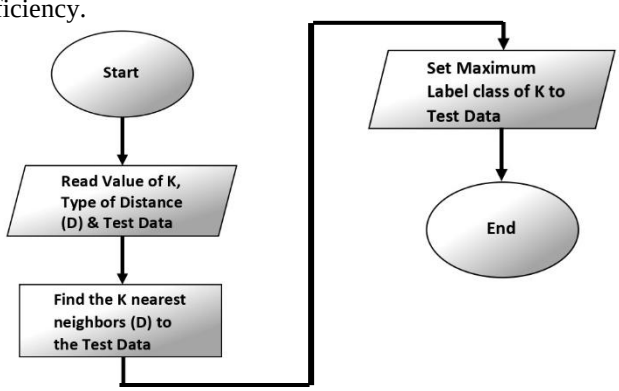


Fig-7: Flow chart of KNN Algorithm

4. RESULTS AND ANALYSIS

A. Software Requirements

The software requirements for the project include a combination of essential tools and frameworks that facilitate data analysis, machine learning, and network intrusion detection. To begin with, Python serves as the primary programming language due to its versatility and extensive library support. It enables efficient data processing, model training, and seamless integration with other software tools.

The Intrusion Detection System (IDS) is designed to operate on a user device running Windows 7 or a higher version. It utilizes Python 3.11 along with JupyterLab as the development environment, enabling efficient coding and implementation of security features. Within this setup, the system employs Scikit-learn, a powerful machine-learning library, to enhance threat detection capabilities through intelligent classification and anomaly detection techniques.

To monitor network activities, Snort is integrated as the primary intrusion detection tool. This open-source network-based IDS inspects packets transmitted over the network and identifies potential threats based on predefined rules and signatures. Snort captures incoming and outgoing network traffic through the network interface and forwards the data to the detection engine. The detection engine analyzes the captured packets, checking for malicious patterns and unusual behaviors that may indicate security threats.

Upon detecting any suspicious activity, the system records detailed logs and triggers real-time alerts to notify administrators or users about potential security breaches. These alerts ensure timely action can be taken to prevent unauthorized access or mitigate possible cyberattacks. The combination of machine learning models and traditional rule-based detection enhances the accuracy and efficiency of the intrusion detection mechanism, reducing false positives and improving response time.

JupyterLab is another crucial component, providing an interactive environment for writing and executing Python code. It allows for real-time data visualization, easy debugging, and better code organization, which enhances productivity in data science tasks.

Scikit-learn plays a vital role in machine learning applications. This library offers a wide range of algorithms for classification, regression, and clustering. It simplifies the process of building and evaluating models, making it easier to implement machine learning techniques effectively.

Snort, a network intrusion detection system, is required for monitoring network traffic and detecting potential security threats. It helps in analyzing packets, identifying malicious activity, and generating alerts based on predefined rules. Its integration enhances the security aspect of the project, ensuring robust protection against cyber threats.

B. Hardware Requirements

The hardware requirements for setting up an Intrusion Detection System (IDS) include a processor with a speed of at least 500 MHz to ensure smooth operation. A minimum of 4 GB RAM is necessary for handling network traffic efficiently, and a hard disk with at least 4 GB of storage capacity is required to store logs and system files. Input devices such as a standard keyboard and mouse will be needed for configuring and managing the system, while output devices should include a

VGA or high-resolution monitor to display alerts and system status effectively.

When setting up the network, it is important to carefully determine the placement of the IDS to ensure it monitors critical points in the network. Positioning it behind the firewall or in key locations where traffic flows through the network is essential. The system should be capable of analyzing all relevant network traffic without becoming a bottleneck that slows down data transfer.

The configuration of the IDS involves setting up rule sets and signatures that help detect potential intrusions. These rules can either be custom-made or downloaded from reliable sources. Threshold levels should be carefully adjusted to minimize false alerts while ensuring that real threats are detected. Proper logging is another crucial aspect, as it enables the system to record all significant events for later analysis. Additionally, deciding on response actions in advance, whether they are automated or manual, will help in effectively dealing with detected threats.

Monitoring and maintenance are essential for the long-term effectiveness of an IDS. Setting up a dedicated monitoring console ensures that alerts and system health can be checked in real time. Keeping IDS software, rules, and signatures up to date is crucial to detect the latest security threats effectively. Regular testing should also be conducted to verify that the IDS is functioning as expected and can respond appropriately to potential security incidents.

C. Source Code with Result

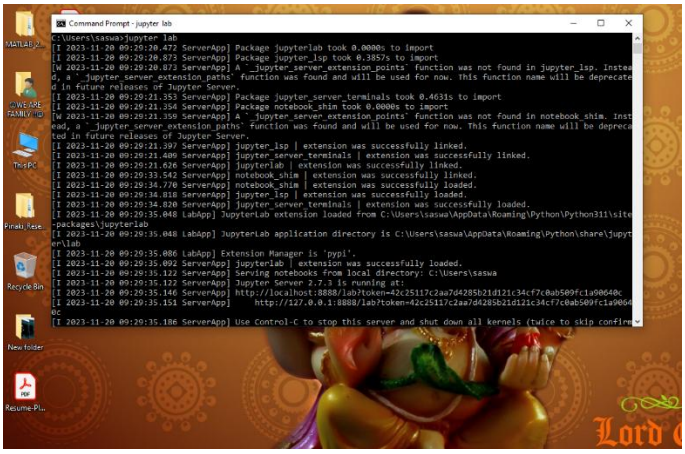


Fig-8: Command Prompt

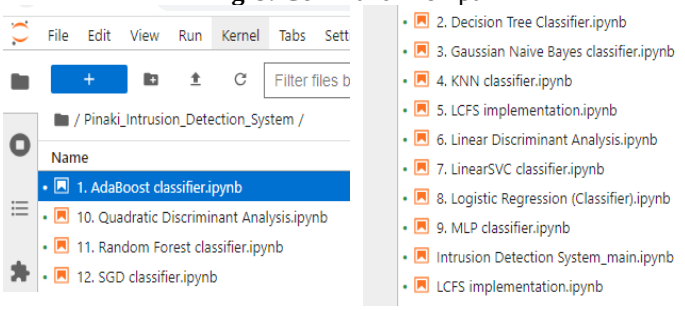


Fig-9: List of Programs

Experimental Analysis / Testing

```
#Training a classifier
clf = AdaBoostClassifier()
t0 = time()
clf.fit(features, labels)
tt = time() - t0
print ("Classifier trained in {} seconds.".format(round(tt, 3)))
```

```
Classifier trained in 156.646 seconds.
```

```
#Predictions on the test data
clf.fit(features, labels)
t0 = time()
pred = clf.predict(features_test)
tt = time() - t0
print ("Predicted in {} seconds".format(round(tt,3)))
```

```
Predicted in 1.756 seconds
```

```
#Calculating out the accuracy
from sklearn.metrics import accuracy_score
acc = accuracy_score(pred, labels_test)
print ("Accuracy is {}.".format(round(acc,4)))
```

```
Accuracy is 0.9239.
```

Fig-10: Program using AdaBoost classifier

```
#Training a classifier
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0)
t0 = time()
clf.fit(features, labels)
tt = time() - t0
print ("Classifier trained in {} seconds.".format(round(tt, 3)))
```

```
Classifier trained in 6.585 seconds.
```

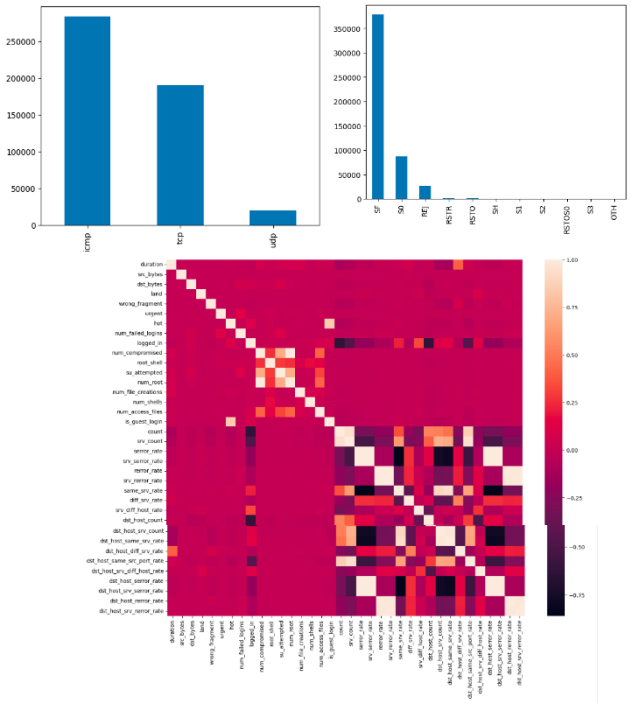
```
#Predictions on the test data
clf.fit(features, labels)
t0 = time()
pred = clf.predict(features_test)
tt = time() - t0
print ("Predicted in {} seconds".format(round(tt,3)))
```

```
Predicted in 0.067 seconds
```

```
#Calculating out the accuracy
from sklearn.metrics import accuracy_score
acc = accuracy_score(pred, labels_test)
print ("Accuracy is {}.".format(round(acc,4)))
```

```
Accuracy is 0.9308.
```

Fig-11: Program using Decision Tree classifier



The implementation of the proposed work is carried out using Jupyter Lab, utilizing essential libraries such as scikit-learn, pandas, and matplotlib, along with other necessary libraries for data analysis and visualization. The dataset used for this study is the KDD Cup 1999 dataset, which is one of the most widely

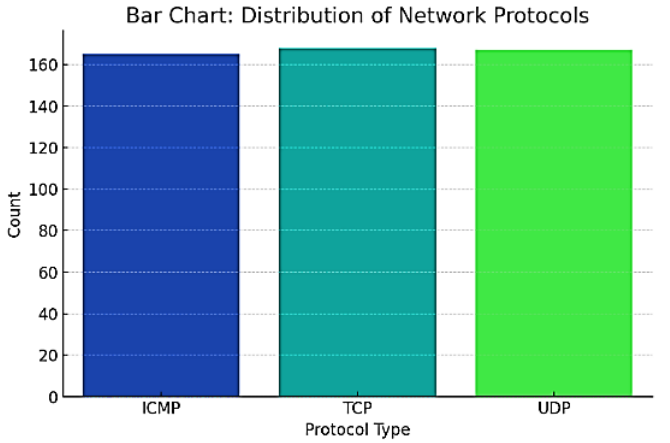
recognized datasets in intrusion detection and network security research. It contains detailed network traffic data, making it a valuable resource for analyzing and detecting network anomalies.

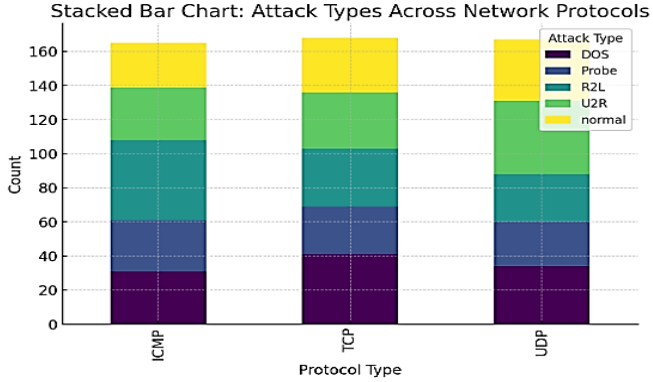
This dataset consists of 41 different features, each representing specific attributes related to network connections. These features provide information about various aspects such as the duration of the connection, the type of protocol used, the type of service requested, connection flags, and the source and destination IP addresses. Each row in the dataset corresponds to an individual network connection, offering insights into different network activities and potential threats.

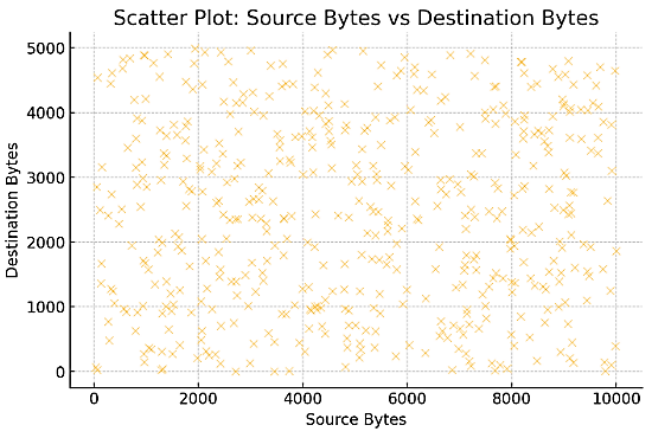
The original dataset contains approximately 4.9 million records, making it extensive and complex. Due to its large size, it is highly useful for testing different machine learning models and techniques for intrusion detection. However, because of computational constraints, researchers often work with smaller subsets of the dataset, focusing on specific types of attacks or selected features to conduct targeted experiments. This approach helps in effectively analyzing network security threats while optimizing the use of computational resources.

Algorithm	Accuracy	Time (seconds)	
		Classifier Trained	Predicted
AdaBoost classifier	0.9239	156.646	1.756
Decision Tree Classifier	0.9308	6.585	0.067
Gaussian Naive Bayes classifier	0.8758	4.377	0.211
KNN classifier	0.9253	28.937	1177.354
Linear Discriminant Analysis	0.8106	21.019	0.033
Linear SVC classifier	0.8999	428.513	0.066
Logistic Regression (Classifier)	0.1894	53.196	0.037
MLP classifier	0.9221	1343.741	0.248
Quadratic Discriminant Analysis	0.8091	16.003	0.315
Random Forest classifier	0.927	100.938	1.348
SGD classifier	0.0837	7.752	0.083

Table-4: Experimental Analysis Result







5. CONCLUSIONS

An Intrusion Detection System (IDS) is designed to identify and prevent attacks and unauthorized activities within a network while minimizing false alarms. By integrating machine learning algorithms, the accuracy and reliability of the IDS are significantly enhanced, ensuring more advanced detection of security threats. This system also evaluates and displays the accuracy rate of detected attacks, which varies depending on the machine learning models applied.

As technology continues to evolve, the volume of data generated has increased substantially, requiring secure storage and processing methods to protect users' information. Security plays a vital role in maintaining user trust, as a well-secured system ensures greater privacy and reliability. A robust IDS enhances the overall security of a network, making it more resilient against potential threats and intrusions.

The KDD Cup 1999 dataset has been a key resource in the development and improvement of intrusion detection models. It has provided a foundation for researchers and security experts to experiment with various machine learning techniques and benchmark their effectiveness. However, with the rapid evolution of cyber threats, it is essential to continuously refine IDS models by incorporating modern datasets and utilizing advanced methodologies. This ongoing adaptation is crucial in ensuring that intrusion detection systems remain effective in identifying and mitigating emerging security risks.

REFERENCES

[1] Intrusion Detection in Computer Networks Using Machine Learning, Nilamadhab Mishra, Sarojananda Mishra, Engineering, Biju Patnaik University of Technology, Rourkela, Odisha, India, DOI: - 10.48047/ecb/2023.12.si5a.054, Eur. Chem. Bull. 2023, 12(Special Issue 5), 1756 – 1767

[2] Implementation and Analysis of Combined Machine Learning Method for Intrusion Detection System, Bisyron Wahyudi, Kalamullah Ramli, and Hendri Murfi, Universitas Indonesia, Indonesia, Vol. 10, No. 2, August 2018, IJCNIS

[3] Comparative Analysis of ML Classifiers for Network Intrusion Detection, Ahmed M. Mahfouz, Deepak Venugopal, and Sajjan G. Shiva, The University of Memphis, Memphis TN 38152, USA

[4] Detection of Attacks in an Intrusion Detection System, Sapna S. Kaushik#1, Dr. Prof.P.R.Deshmukh, M.E. II Year, Computer Science and Engg., Sipna College of Engg. Amravati, INDIA

[5] Decision Tree Based Algorithm for Intrusion Detection, Kajal Rai, M. Syamala Devi, Ajay Guleria, Panjab University, Chandigarh, India, Int. J. Advanced Networking and Applications, Volume: 07 Issue: 04 Pages: 2828-2834 (2016) ISSN: 0975-0290

[6] Network Intrusion Detection Using Machine Learning Techniques, Yasmeen S. Almutairi, Bader Alhazmi, Amr A. Munshi, Computer Engineering Department, Umm Al-Qura University, Makkah 21961, Saudi Arabia, Advances in Science and Technology Research Journal 2022, 16(3), 193–206, ISSN 2299–8624, License CC-BY 4.0

[7] Survey of intrusion detection systems: techniques, datasets and challenges, Ansam Khraisat, Iqbal Gondal, Peter Vamplew and Joarder Kamruzzaman, Khraisat et al. Cybersecurity (2019) 2:20

[8] Intrusion Detection System Using machine learning Algorithms, Rachid Tahri, Youssef Balouki, Abdessamad Jarrar, and Abdellatif Lasbahani, National School of Applied Sciences, Sultan Moulay Slimane University, Bni Mellale, Morocco, ITM Web of Conferences 46, 0 (2022), ICEAS'22

[9] Network Traffic Analysis and Intrusion Detection Using Packet Sniffer, Qadeer, Mohammed & Iqbal, Arshad & Zahid, Mohammad & Siddiqui, Misbahur, Communication Software and Networks, International Conference on. 313-317. 10.1109/ICCSN.2010.104.

BIOGRAPHY

Pinaki Shashishekhar Mathan
B.Tech (Hons)
Computer Science & Engineering
OmDayal Group of Institutions,
Uluberia, Howrah, West Bengal