## ABSTRACT

In the realm of cybersecurity, the rapid proliferation of Internet of Things (IOT) devices has necessitated advanced techniques for the detection and classification of intrusions to ensure the security and integrity of networks. This research presents an Artificial Intelligence (AI) approach designed to enhance the efficacy of intrusion detection and classification in IOT networks. The introduction highlights the exponential growth of IOT devices and the associated increase in security threats, emphasizing the need for robust intrusion detection mechanisms. The conventional system primarily relies on rule-based or signature-based methods, which often fall short in adapting to dynamic and evolving cyber threats. Drawbacks of the existing system include a high rate of false positives, inability to detect novel attacks, and susceptibility to sophisticated intrusion techniques. In response to these limitations, the proposed system leverages advanced AI techniques, such as machine learning algorithms and deep neural networks, to autonomously learn and adapt to emerging threats. The model's ability to analyses complex patterns and anomalous behaviours enables more accurate detection and classification of intrusions. The research showcases promising results in terms of increased accuracy, reduced false positives, and enhanced resilience against novel attack vectors, demonstrating the potential of AI-based solutions for bolstering IOT network security.

**KEY WORDS:** Cybersecurity, Internet of Things (IoT), Intrusion Detection, Intrusion Classification, AI & ML Algorithms, Security Threats, Rule-based Methods, Anomalous Behavior, Accuracy.

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIG NO** | **DESCRIPTION** | **PG NO** |
| Figure 4.1 | Proposed System Model | 20 |
| Figure 4.2 | DTC Algorithm | 23 |
| Figure 5.1 | User Code diagram | 28 |
| Figure 5.2 | Sequence Diagram | 29 |
| Figure 5.3 | Activity Diagram | 30 |

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **DESCRIPTION** | **PG NO** |
| Table 10.1 | Sample dataset | 62 |
| Table 10.2 | Null value status in dataset | 62 |
| Table 10.4 | Dataset after label encoding | 62 |
| Table 10.4 | Classification of existing SVM | 64 |
| Table 10.5 | Classification of report of proposed DTC | 65 |

# LIST OF GRAPHS

|  |  |  |
| --- | --- | --- |
| **GRAPH NO** | **DESCRIPTION** | **PG NO** |
| Graph 3.1 | Support vector machine | 16 |
| Graph 10.1 | Dataset count plot | 63 |
| Graph 10.2 | Confusion matrix of existing SVM | 63 |
| Graph 10.3 | Confusion matrix of proposed DTC | 64 |

**LIST OF CONTENTS**

**ABSTRACT i**

**LIST OF FIGURES ii**

**LIST OF TABLES iii**

**LIST OF GRAPHS iv**

**CH NO DESCRIPTION PG NO**

**CHAPTER 1 INTRODUCTION 1**

1.1 OVERVIEW 1

1.2 MOTIVATION 2

1.3 PROBLEM STATEMENT 4

1.4 OBJECTIVE 5

1.5 REAL TIME APPLICATION 5

**CHAPTER 2 LITERATURE SURVEY 7**

2.1 INTRODUCTION 7

2.2 RELATED WORK 7

2.3 RESEARCH GAPS 15

2.4 SUMMARY 15

**CHAPTER 3 EXISTING MODEL 16**

**CHAPTER 4 PROPOSED MODEL 19**

4.1 INTRODUCTION 19

4.2 DATA PREPROCESSING 21

4.3 DATA SPLITTING 22

4.4 DTC ALGORITHAM 23

4.5 ADVANTAGES OF PROPOSED SYSTEM 25

**CHAPTER 5 UML DIAGRAM 27**

**CHAPTER 6 SOFTWARE ENVIRONMENT 31**

**CHAPTER 7 SYSTEM REQUIREMENTS 37**

**CHAPTER 8 FUNCTIONAL REQUIREMENT 38**

**SOURCE CODE 43**

### RESULT AND DISCUSSION 57

### CONCLUSION AND FUTURE SCOPE 66

### REFERENCES 68

**CHAPTER 1 INTRODUCTION**

## OVERVIEW

An intrusion detection system (IDS) has been an essential component of computer security for a considerable time. The IDS was traced back to the early days of computers when the very first computer viruses were developed, and here is where its origins were found. At that time, the focus was creating antivirus programs to detect and remove viruses from infected systems. The first IDS was developed in the early 1980s by James Anderson. Anderson's IDS, the "Computer Misuse Detection System," was designed to detect unauthorized access to a computer system. The system worked by analysing system logs and network traffic for suspicious activity. Commercial IDS products emerged in the late 1980s and early 1990s [1]. These products were based on Anderson's work and were designed to detect network- and host-based attacks. One of the first commerce IDS products was SRI's Network Security Monitor (NSM), released in 1989. NSM used a signature-based approach to detect attacks by comparing network traffic to a database of known signatures.

In the mid-1990s, an innovative approach to IDS was developed called anomaly detection. Anomaly detection systems monitored network traffic for patterns of activity that deviated from normal behaviour. This approach was more effective than signature-based systems at detecting new and unknown attacks. One of the first anomaly detection systems was the Statistical Anomaly Detector (SAD) [2], developed by Dorothy Denning and Peter Neumann in 1995.

In the late 1990s and early 2000s, IDS technology continued to evolve. Intrusion Prevention Systems (IPS) were developed, which detected attacks and prevented them from being successful. IDS and security system technology were integrated with IPS products to deliver a more all-encompassing security solution. In recent years, advancements in machine learning (ML) [3], deep learning (DL) [4], and artificial intelligence (AI) [5] have each played a role in contributing to the ongoing progress of technology related to IDSs. The ML algorithms were used to analyze enormous volumes of data in-depth and identify patterns of behaviour that suggest an impending

assault. This behaviour-based detection approach is becoming increasingly popular in IDS products.

The history of IDS has been a long and evolving one. From the early days of antivirus software to modern ML-based IDS systems, the technology has continued to improve and adapt to new threats. IDS will continue to be essential to computer security as new threats emerge and technology advances.

## MOTIVATION

The motivation for using AI/ML in IDS is to improve the accuracy and effectiveness of intrusion detection. Traditional IDS systems rely on predefined rules and signatures to detect known attacks, but they are ineffective against new and unknown attacks. Additionally, they can generate many false positives, leading to alert fatigue and reduced effectiveness.

An AI/ML-based IDS may automatically learn the typical behaviour of the system and identify deviations from this behaviour. This can assist in the detection of assaults that were not previously recognized. These systems can also lower the false positives they produce by using sophisticated methods to investigate data and identify irregularities.

Another motivation for using AI/ML in IDS is the increasing complexity and diversity of modern networks and systems. Networks and systems are becoming more complex and dynamic, with various devices and protocols, and traditional IDS systems may need help to keep up with these changes. AI/ML-based IDS can adapt to these changes and learn to detect new threats without requiring constant updates to rules or signatures.

The IDS can function even without the use of AI or ML approaches. At first, the IDS would gather data from various sources, including network traffic, system logs, and application logs. The information gathered throughout this process comprises the IP addresses of the sender and receiver, the ports used, the protocols used, and the payloads of the packets. The data that was obtained is subsequently pre- processed to eliminate noise and information that is not relevant to the study. Techniques such as filtering, normalization, and data reduction are used throughout this step to increase the detection procedure's level of precision.

Signature-based IDSs compare incoming network traffic to a database containing known attack signatures. Signatures of attacks are distinct patterns of network activity linked to specific categories of cyberattacks. If the IDS detects a match between the incoming network traffic and an attack signature in its database, it generates an alert. Anomaly-based IDS works by monitoring network traffic for patterns of activity that deviate from normal behaviour. The IDS builds a baseline of regular activity based on historical network traffic and system logs. It then compares incoming traffic to this baseline and generates an alert if it detects activity significantly different from what is considered normal.

Statistical IDS employs several statistical analysis approaches to identify abnormalities in the network data flow. Calculating statistical measures like the mean and the standard deviation is an integral part of these strategies and variance to detect unusual network activity patterns. Rule-based IDS involves creating rules defining normal and abnormal network traffic behaviour. The IDS compares incoming traffic to these rules and generates an alert if it detects traffic violating them. When the IDS detects potential security breaches, it generates alerts and sends notifications to security analysts or administrators for further investigation. The IDS also generates reports that provide information about the alerts, including the type of attack, the source IP address, and the destination IP address.

So, IDS can operate without using AI or ML techniques by combining signature-based detection, anomaly-based detection, statistical analysis, rule-based detection, and notification and reporting. While AI/ML-based IDS can provide higher accuracy and faster detection, traditional IDS techniques are not widely used. They could detect potential security breaches in the network and system environment more effectively.

Furthermore, AI/ML-based IDS can provide more comprehensive and real- time threat detection, essential for protecting against advanced persistent threats (APTs) and zero-day attacks. These attacks were brutal to detect using traditional IDS systems. Still, AI/ML-based IDS can use advanced techniques such as anomaly detection, behavioural analysis, and DL to detect and mitigate these threats. So, the motivation for using AI/ML in IDS is to improve intrusion detection's accuracy, effectiveness, and scalability and address the limitations of traditional rule-based

systems. By leveraging the power of ML and advanced analytics, AI/ML-based IDS can provide more comprehensive and real-time threat detection, which is essential for protecting against modern cyber threats.

## PROBLEM STATEMENT

The problem statement of implementing an IDS using AI/ML is automatically detecting intrusions and anomalies in network traffic and system logs without relying on predefined signatures or rules. Traditional rule-based IDS have several drawbacks, including a high FPR, a challenging process for upgrading rules, and an inability to identify new and undiscovered threats. The AI and ML-based IDS aim to circumvent these constraints by using ML algorithms to learn the typical behaviour of a network or system and then identify any abnormalities in that behaviour. The most challenging aspect of developing an AI or ML-based IDS is producing dependable models that can identify various intrusions while keeping FPRs to a minimum.

The issue was broken down into a few different sub-issues involving the preparation of data, the engineering of features, the selection of models, the training, and the evaluation. The first thing that must be done is to collect and pre-process the data. This involves filtering and standardizing the raw data and turning it into ML format. The process of choosing and extracting pertinent features from the data, such as packet size, protocol, and time intervals, is referred to as feature engineering. These features were used to capture both regular and malicious traffic characteristics.

Model selection entails selecting an appropriate ML method that can identify intrusions with an elevated level of accuracy and a low FPR. The composition of the data, the nature of the issue being tackled, and the computational resources at one's disposal are all factors considered while selecting an algorithm. The next step is to train the model of choice using the training data and evaluate its performance using a test dataset kept separate from the training dataset. The assessment of the model considers various performance measures, including accuracy, precision, recall, and the F1 score.

Therefore, the problem statement of IDS using AI/ML is to design an accurate and efficient system that can automatically identify intrusions and abnormalities in network traffic and system logs without depending on predefined signatures or rules.

This is possible without relying on predefined signatures or rules. The solution demands skill in both cybersecurity and ML since it comprises several sub-problems such as data pre-processing, feature engineering, model selection, training, and assessment.

## OBJECTIVES

The main objective is to use decision tree classifier (DTC) which is an ML technique to identify the type of intrusion that is most likely to occur in an IOT network. Train a decision tree to detect various attacks. Achieve high accuracy and low false alarms. Run efficiently on resource-constrained devices. Adapt to evolving threat landscapes. Provide insights into attack patterns and root causes. Minimize manual intervention. Ensure ethical and responsible development and deployment.

## REAL-TIME APPLICATIONS OF IDS

The IDS has numerous real-time applications in the field of cybersecurity. Some of the most common real-time applications of IDS are:

**Network Security:** Real-time detection and prevention of network-based assaults are both possible with the usage of IDS. By monitoring the traffic, IDS can spot abnormalities and unusual behaviour patterns in network traffic. These were signs of an ongoing assault. NIDS can provide complete protection and be installed at several places across the network.

**Host Security:** HIDS was used to monitor the activity of individual hosts in real time. HIDS can detect suspicious behaviour by analysing system logs, calls, and other types of system activity. HIDS can help prevent attacks such as privilege escalation, malware infection, and unauthorized access.

**Cloud Security:** IDS-protected cloud-based systems and applications in real-time. Cloud-based IDS (CIDS) can monitor cloud resources for unauthorized access, data exfiltration, and other types of malicious activity. CIDS was integrated with cloud orchestration and management tools to provide automated responses to detected threats.

**Web Application Security:** IDS-protected web applications in real-time. The online Application Firewall (WAF) is a sort of IDS that can identify and stop assaults directed

at online applications. SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) are all types of threats that WAF detected [6].

**Industrial Control Systems Security:** IDS-protected industrial control systems (ICS) in real-time. ICS-IDS can monitor the activity of critical infrastructure systems to detect any anomalies or suspicious activity. ICS-IDS can help prevent attacks such as DoS, data manipulation, and unauthorized access. So, IDS has numerous real-time applications in the field of cybersecurity. By providing real-time threat detection and prevention, IDS can help protect networks, systems, applications, and critical infrastructure from a wide range of cyber threats.

**CHAPTER 2**

**LITERATURE SURVEY**

## INTRODUCTION

An IDS is a security technology used to monitor the activities of a computer system or network to search for indications of unauthorized access, assaults, or other forms of hostile behavior [7]. The primary goal of an IDS is to identify potential security events and take appropriate action promptly to avoid or limit the negative impacts of a security breach. This was accomplished by preventing or detecting potential threats. A survey of IDS involves evaluating and comparing different IDS technologies [8] and solutions to identify their strengths, weaknesses, and effectiveness in detecting and responding to security incidents. The survey can include a variety of factors, such as the type of IDS technology used (such as NIDS, HIDS, or hybrid), the level of automation and ML capabilities, the accuracy and effectiveness of the detection rules and algorithms, and the ease of deployment and management.

The survey of IDS can help organizations make informed decisions about which IDS solution to implement based on their specific security needs and requirements. By evaluating the different IDS options available in the market [9], organizations can select an IDS solution that provides the best protection against the threats they are most likely to face. The survey of IDS can also assist firms in keeping well-informed of the most recent developments and trends in IDS technology [10], such as the use of AI and ML algorithms for automated threat detection and response. This was accomplished by helping organizations remain current on IDS. Organizations can continuously improve their security posture and protect against emerging threats and attacks by staying informed about the latest IDS developments.

## RELATED WORK

In [11], the authors developed Double Deep Q-Network (DDQN) based IDS. The numerous varieties of IDS, relevant ML approaches, and challenges associated with DDQN are provided as evidence for its significance. This page describes DDQN's wide range of anomaly-detecting uses in detail. The challenges in implementing FLS are also highlighted, providing a glimpse into the potential breadth of future studies.

This work lays the groundwork for further study and concludes with a discussion of practical solutions to the identified problems in implementing a DDQN-based IDS.

In [12], authors used Deep Reinforcement Learning (DRL) for IDS. These algorithms have been put through their paces using datasets generated by DARPA, KDD CUP 99, and KDD to see how well they can classify diverse types of cybercrime. In this study, they investigate various methods for detecting intrusions, ranging from ML to dynamic programming to DRL. The research is a survey with a representative sample conducted in IDS between 2008 and 2020. This research focuses on the approaches that use feature selection in their performance assessment models. In addition, the paper provides an overview of the other IDS datasets and a description of the most current dataset collected by the CIC, which is referred to as IDS-2017. The diverse applications of IDS, the obstacles it now confronts, and prospective future research areas are all covered in this study. The research that has been given has the potential to function as a stepping stone for both seasoned and inexperienced researchers working in network security who are interested in gaining a deeper understanding of and the capacity to develop more effective IDS models.

In [13], authors implemented Markov Decision Process (MDP) based IDS. Over the last several years, a determined effort has been made to lessen the attack surface area in all locations. However, the complexity and variety of attack vectors have grown, and the sophistication of the adversaries' methods for exploiting ecosystems have increased. Exploiting this kind of might lead to the complete inoperability of the system or the exposure of confidential data. To be more specific, the proliferation of the Internet of Things (IOT) has led to the interconnection of a wide variety of previously unconnected devices, many of which are limited in their ability to communicate with one another and access the Internet. These devices' limited processing capacity and resources make it difficult to detect intrusions. This is since SI-based techniques have a reasonable success rate by enhancing many elements of an IDS.

The SI-based strategies have gained much attention, particularly in the recent decade. In addition, this is because SI-based methods have a solid success rate by improving many sections of an IDS. This study aims to present a complete analysis of the state-of-the-art swarm intelligence algorithms implemented on diverse attack

surfaces for intrusion detection in different domains. The research will concentrate on works that have been published between the years 2010 and 2020. These papers were chosen because they provide the most in-depth examination of recent developments in swarm intelligence techniques. This study also categorizes SI approaches based on how well they improve various parts of an intrusion detection process. The capabilities and features of the various datasets utilized in the experiments are also discussed in the study. This is done to help researchers evaluate the strengths and weaknesses of SI algorithms, which is necessary for identifying security dangers and problems that must be met while developing an IDS to detect cyberattacks across several domains. In addition, this will help security staff differentiate SI-based IDS from traditional methods. This means that researchers in swarm intelligence and cyber security benefited from participating in the study. The survey highlights some existing issues and suggests practical solutions. On top of that, fresh avenues for more research are mapped out.

In [14], authors developed a decision-theoretic framework named ADRS- based IDS. ML is effective in pattern recognition when previous methods have failed. Innovative methods like DL were used to improve the safety of IOT devices. In terms of anomaly-based detection, this provides a straightforward alternative. To effectively study entire traffic throughout the IoT, the ADRS-based method to anomaly-based IDSs is presented in this study. The model that has been shown exhibits the capacity to identify intrusions as well as anomalous patterns of traffic. The NIDS Dataset and the BOT-IOT Dataset were used for training and testing, and the results showed an accuracy of 99.51% and 92.85%, respectively.

Kim-Hung Le et al. [15] developed intelligent IMIDS for safeguarding IOT devices. The IMIDS is built on a lightweight DNN model that aims to classify different cyber threats. Combat the issue of inadequate training data; in addition to this, one of their suggestions is to use a conditional generative adversarial network (GAN) to fuel an attack data generator. Through this trial, they demonstrated that IMIDS is superior to its competitors by identifying nine forms of cyber-attacks (including worms, backdoors, and shellcodes) with an average F-measure of 97.22 per cent. Also, after being educated with data from their attack data generator, IMIDS's detection performance was greatly boosted, turning it into a more effective anti-malware

solution. This improvement was made possible by the fact that IMIDS could teach itself. These findings suggest that IMIDS was useful IDS on the IOT context.

Using is something that Nasir and colleagues [16] suggest doing in order to move the FFA processes in binary space. As a result, the V-shaped function transforms into a binary mode, the continuous solution location that the FFA technique employs. This article presents a hybrid technique for classifiers and the BFFA as a fast and reliable IDS. The KDD and UNSW-NB15 IDS datasets are used to ascertain whether the proposed approach yields reliable results. Different classifiers, including KNN, SVM, DTC, RFC, and Ad boost, are assessed and compared to one another on these datasets regarding accuracy, precision, recall, and F1 Score.

Le, Thi-Thu-Huong, et al. [17] developed the DTC and RFC-based ensemble trees approach for the ML IDS problem. Both classifiers were used to train models using just a few computing resources. An experimental evaluation of the proposed method is carried out by making use of the feature set of the net flow meter in conjunction with two substantial datasets called NF-BoT-IoT-v2 and NF-ToN-IoT- v2, both of which are updated versions of the BoT-IoT and ToN-IoT datasets, respectively. These datasets are used in conjunction with one another for evaluation. Because of this, the evidence was presented more compellingly. In the context of lab research, the IoTDS20 dataset is also employed. In addition, the judgments about categorization made by the DTC and RFC models are examined and explained using the Shapley additive explanations to the explainable AI (XAI) technique. Not only does this technique aid cybersecurity professionals in fast optimizing and grading their judgements based on the explanations of the data, but it also interprets the ultimate choice of the ensemble tree approach correctly.

Alzaqebah et al. [18] developed the IDS using ML with the UNSWNB-15 dataset. The primary objective of this research was to identify specific instances of generic assaults in network traffic. This was since the generic form of attack predominated in the sample data. The new model surpassed the previously used approaches, bringing the crossover error rate and the FPR down to percentages lower than 30%. The accuracy, F1-score, and G-mean results were 81%, 78%, and 84%, respectively, making it the most successful choice.

Yu and Jing et al. [19] developed IDS for network security communication, and it was constructed utilizing multi-scale DNNs. After then, the appropriate tests were conducted on sets of data that were accessible to the public. A multi-scale DNN system beats systems based on the Add-boost model and the recurrent neural network (RNN) model in terms of convergence speed, average error ADR, and average accuracy, as shown by the findings of the experiments. The findings are that the multi- scale DNN arrangement produces more accurate predictions. According to the results, the detection accuracy of the IDS based on a multi-scale convolution neural network is relatively high. Due to the specific nature of this situation, a reference is required.

Imran, Muhammad, et al. [20] developed deep auto-encoder (DAE) based NIDS. However, they face substantial challenges because of the constant emergence of new threats that the current systems need help identifying. Here, they introduce a deep-learning-powered NIDS that is smart and fast in finding intrusions into networks. The remarkable success of DL in a wide variety of detection and identification tasks is what inspired us to take this step. This research presents a non-symmetric DAE to overcome obstacles in network intrusion detection and demonstrate its robust capabilities and performance. To ensure the new NIDS is dependable and practical, they put it through its paces on the industry-standard KDD CUP'99 dataset. The Tensor Flow library and the GPU framework also use their DL-based method, capable of

99.65 per cent accuracy. The method described here was used in DL-based detection and classification systems and network security research.

Mendonca et al. [21] created parameters for the SET-based prediction model for IDS. After analyzing these parameters, the model's performance is analysed and compared to the performance of other methodologies that are now considered to be state-of-the-art. The SET-based model that was advised to be used had an average testing duration of 2.29 milliseconds, which resulted in an accuracy of 0.99%. In a real-world scenario of IoT security in Industry 4.0, the recommended model increases ADR by an average of 6.25 percentage points compared to current state-of-the-art ML models. These findings are based on the research that was conducted. This is the conclusion drawn from analyzing how well these models detect attacks. This was established by comparing the proposed model to other existing ML models. This conclusion was reached as a direct consequence of the findings obtained by comparing the recommended model with the alternatives.

Whelan, Jason, et al. [22] identified attacks using one-class classifiers and principal component analysis (PCA). Flight records is used as a data source, resulting in a flexible and comprehensive approach. MAVIDS is the final IDS, which incorporates the suggested detection approach. The UAV comes outfitted with a resource-constrained agent device that contains an IDS. As a result, it can identify assaults and halt them, even if the connection with the ground control station is disrupted by jamming. With macro-averaged F1 scores of 90.57 per cent and 94.3 per cent against GPS spoofing and jamming, respectively, the method is effective against both threats.

Yadav, Neha, et al. [23] suggested an approach for identifying invasions that would make it easier to identify actual global invaders. In identifying attacks, a neural network works quite well. Furthermore, 5G networks must gather, process, and analyse vast data traffic and network connections to meet the rising need for user- centric cybersecurity solutions. Prevent attacks on 5G networks; this is essential. Extensive testing has validated the auto encoder model's claims of superiority due to its ability to reduce detection time while improving detection accuracy. In practice, the proposed approach had a 99.76% success rate.

Mushtaq, Earum, et al. [24] presented DAE with Long-Short Term Memory (LSTM) plus bidirectional LSTM (Bi-LSTM) as a hybrid framework for IDS. Categorizing samples as typical or abnormal, this framework first obtains optimum features using the DAE. Error measures like precision, recall, F-score, accuracy, ADR, and FAR are used to determine how successful the presented models are when applied to the well-known dataset KDD. Results from different models were compared using these measures. Experimental findings show that the proposed AE-LSTM outperforms established and novel deep and shallow ML algorithms regarding prediction error. When compared to experimental results, this is the case.

Thakkar et al. [25] developed a unique technique for choosing features that considers the statistical importance of the Standard Deviation, the Difference of Mean, and the Median before choosing features using this fusion. With this method, characteristics are picked. The proposed technique filters out irrelevant features based on their relative ranking, calculated by combining statistically noteworthy features. Furthermore, the fusion of statistically noteworthy features aims to provide beneficial

characteristics with high discernibility and deviation, which aids in enhanced data learning. They conduct an analysis of the proposed method using three different intrusion detection datasets: the KDD, the UNSW NB-15, and the CIC-IDS-2017. Within the context of the performance study, some assessment measures used were accuracy, precision, recall, f-score, and FPR, which are then compared to numerous existing feature selection methodologies. The time required to complete the job is also compared between the two to provide more context to the performance metrics. The results are then statistically analyzed using the Wilcoxon Signed Rank test.

Qiu et al. [26] provided an approach for detecting intrusions that use the Dempster-Shafer Theory in combination with packet-based and flow-based methods (DST). This method is intended to aid in the prevention of malicious activity. The composition of DST-IDS is arranged in a manner comparable to that of an ensemble. DST is used to merge the predictions of flow-based IDS with packet-based IDS to provide a conclusive detection result. The former seeks to predict traffic flows, whereas the latter looks for attacks in the packets carried by the traffic. The goal of flow-based IDS, on the other hand, is to anticipate the flow of traffic, whereas packet- based IDS aims to identify malicious activity in data transfers. In addition to this, they design and construct an innovative data collection and processing instrument for DST- IDS. This is done in order to assist in lowering the amount of data necessary for intrusion detection, hence making early detection easier. In addition, the DST-IDS was developed to function well with a wide variety of data distribution methods. This suggests that the distribution of the data in the training dataset and the distribution in the actual world will not necessarily be the same.

On the other hand, DST-IDS was developed to work well even with diverse types of data distribution. The incorporation of this function results in a considerable enhancement of the DST application process. IDS's Experiments are run on live networks and public data sets so that they can determine whether the proposed approach is practical. DST-IDS can manage the distribution of various data types and provide real-time network detection. The experimental findings demonstrate that DST-IDS is superior to the most effective state-of-the-art benchmarks regarding accuracy and speed when identifying intrusions.

Azeroual et al. [27] used Apache Spark as the backbone of their big data analytics engine to investigate security implications. To discover anomalies in data using ML, a prototype of an IDS is built utilizing the k-means approach for clustering analysis available in Sparks MLlib. The system aims to identify any vulnerabilities in the safety measures in place. As the problem of detecting anomalies has yet to be contained, not painstakingly reviewed, extracting fees that identify them is challenging. Using relevant data currently stored by corporations and scientific institutions was helpful. Their interpretation and subsequent processing in a continuous manner may contribute significantly to the detection of abnormalities and intrusions.

Chang et al. [28] provide a concise summary of the present status of IDS and the underlying concepts. Functionalities and methodologies for building these protective mechanisms, as well as actual products employed in operational systems, are considered in this research. In conclusion, these are nagging problems that this area has.

Islamists et al. [29] provided an IDS suitable for Apache web servers. The MBC ML algorithm is used to train using the suggested strategy. The dataset for the training that is utilized is provided by IEEE. The suggested system has an extremely high degree of accuracy (98.6%) regarding cross-validation.

Friha et al. [30] created a federated learning-based IDS named FELIDS to safeguard IoT infrastructures in agriculture. More specifically, localized learning is used by the FELIDS system to protect user information. In this approach, devices improve their detection model by learning from their contemporaries and only communicating model changes to a centralized server. The FELIDS system employs three distinct DL classifiers to shield Agricultural IoTs against cyberattacks: Deep Neural Networks (DNN), DNNs, and RNNs. When they evaluate the suggested IDS on three different data sets—CSE-CIC-IDS2018, MQTTset, and InSD. They discover that it works well on all these different data sets. The findings demonstrate that the FELIDS system beats more traditional and centralized versions of ML (non-federated learning) regarding safeguarding the privacy of data collected from IoT devices and achieving the maximum possible accuracy when identifying malicious behaviour.

## RESEARCH GAPS

Several research gaps in the field of IDS need to be addressed. Some of these research gaps include:

* + - Real-time detection: Most IDS systems are reactive and detect threats after they have occurred. There is a need for IDS systems that can detect threats in real-time and respond to them immediately.
    - Accuracy and false positives: IDS systems often produce many false positives, which may make it difficult for security managers to recognize actual dangers. It is necessary to research to enhance IDS systems' precision and lower the overall number of false positives.
    - ML and AI: ML and AI are increasingly used in IDS systems to improve threat detection. However, there is a need for more research to determine the effectiveness of these technologies in detecting threats and reducing false positives.
    - Privacy concerns: IDS systems collect large amounts of data, which can raise privacy concerns. Research is needed to develop IDS systems that can effectively monitor and detect threats while respecting user privacy.
    - Cost-effectiveness: IDS systems were expensive to deploy and maintain. There is a need for research to develop cost-effective IDS systems that are accessible to organizations of all sizes.

## SUMMARY

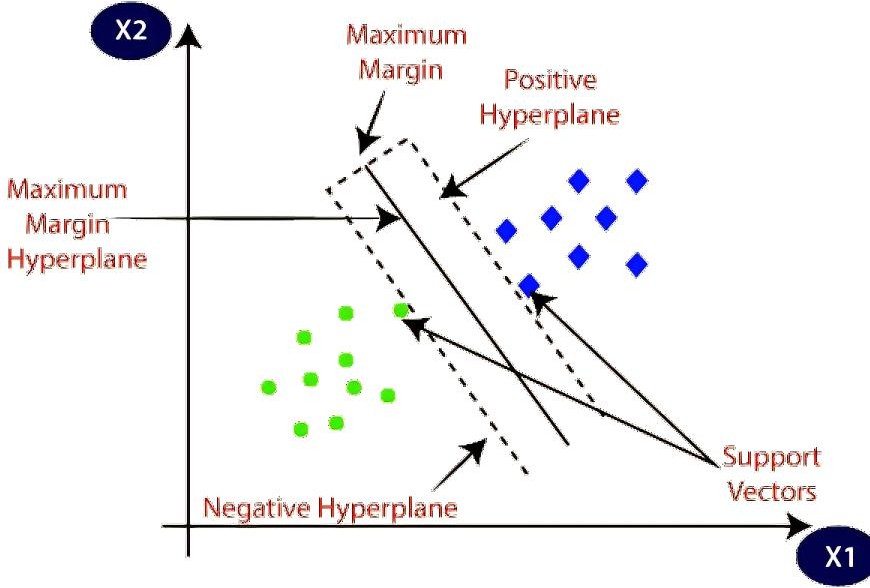
An IDS is a security technology that monitors the activities of a system and the traffic on a network to detect and react to possible security breaches. There are primarily two categories of IDS, which are NIDS and HIDS. Nevertheless, several research gaps in IDS need to be filled to increase real-time detection, accuracy and false positives, ML and AI, IoT and cloud settings, privacy issues, and cost- effectiveness. Addressing these research gaps will help organizations protect their networks and systems from cyber threats.

**CHAPTER 3 EXISTING METHODOLOGY**

## SUPPORT VECTOR MACHINE

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

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



Graph 3.1 Support Vector Machine

Machine learning involves predicting and classifying data and to do so we employ various machine learning algorithms according to the dataset. SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the

data into classes. In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification. As a simple example, for a classification task with only two features (like the image above), you can think of a hyperplane as a line that linearly separates and classifies a set of data.

* Intuitively, the further from the hyperplane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyperplane as possible, while still being on the correct side of it.
* So, when new testing data is added, whatever side of the hyperplane it lands will decide the class that we assign to it.

#### The operation of SVM is illustrated as follows:

* **Model Training:** Choose a suitable kernel function based on the nature of your data. Common kernels include Linear, Polynomial, Radial Basis Function (RBF), and Sigmoid. The choice of kernel can significantly impact the SVM's performance. The training process involves finding the optimal hyperplane (decision boundary) that best separates the data points of different classes while maximizing the margin (distance) between the hyperplane and the nearest data points (support vectors).
* **Optimizing Parameters:** Tune hyperparameters, such as the regularization parameter (C) and kernel-specific parameters, to achieve the best classification performance. Grid search or random search can be used for hyperparameter optimization.
* **Handling Imbalanced Data:** Address class imbalance issues if present in the dataset by using techniques like class weighting or resampling.
* **Model Representation:** The trained SVM model is represented by the support vectors (data points closest to the decision boundary) and the associated coefficients. After training, evaluate the SVM model using the test dataset. Common evaluation metrics include accuracy, precision, recall, F1-score, ROC-AUC, and the confusion matrix.
* **Making Predictions:** Once the SVM model is trained and evaluated, it can be used to make predictions on new, unseen data points. To make a prediction for a new data point:
  + Apply the same feature extraction techniques used during training to preprocess the new data point.
  + Use the trained SVM model to calculate the decision function, which assigns the data point to one class or another based on the sign of the function's output.
* **Decision Function:** The decision function of an SVM calculates the signed distance of a data point from the decision boundary (hyperplane). This distance is also known as the margin. The sign of the distance determines the predicted class label. If the distance is positive, the point is classified as one class; if it's negative, it's classified as the other class.
* **Margin and Support Vectors:** Support vectors are data points that are closest to the decision boundary and have non-zero coefficients. They are crucial in defining the decision boundary and maximizing the margin. The margin is the distance between the decision boundary and the support vectors. SVM aims to maximize this margin during training.
* **Handling Non-Linearity:** For non-linearly separable data, SVM can use kernel functions to map the data into a higher-dimensional space where linear separation is possible. The decision boundary in this transformed space corresponds to a non-linear decision boundary in the original feature space.

**CHAPTER 4 PROPOSED METHODOLOGY**

## INTRODUCTION

Because of the tremendous advancements in internet and communication technologies, the number of data being stored is growing at an alarming rate. This has resulted in a rise in traffic moving across networks from locations all over the globe. Cyberattacks are fast rising due to the development of novel attacks and mutations of previous ones, which is a direct result of the rise in network traffic. As more and more things connect to the Internet, we may anticipate an increase in the number of assaults that take advantage of vulnerabilities in computer systems. As a result, protecting data and networks from unwanted users and attackers has become an essential study topic in recent years. Several different security solutions have been made available to protect networks from external and internal assaults. Despite this, preventing such attacks remains difficult owing to the inherent limits of security rules, firewalls, access control schemes, and antivirus software.

Conventional tactics, on the other hand, have their limits when detecting new and undiscovered threats, and they are ineffective when managing the large-scale and dynamic nature of today's networks. Recently, machine learning and deep learning methods have emerged as potentially effective tools to solve these challenges. This contrasts current statistical methodologies and knowledge-based expert systems, which have existed for some time. Algorithms that use machine learning can learn from the network traffic data to discover patterns and anomalies that hint at the probability of an attack being launched against the network. Algorithms that use machine learning can learn and adapt to new attack patterns and identify attacks that have never been seen before. When used to network intrusion detection, machine learning can enable real-time detection and reaction to possible assaults, improving the overall security of computer networks.

#### System Model

The suggested model of the system is shown in Figure 4.1. The KDD dataset is used in the technique that has been suggested. This dataset, which is often used for

research on intrusion detection, provides data on network traffic, including various assaults in addition to routine activity. The first step is to split the KDD dataset into two parts: an 80% portion for training the classifiers and a 20% portion for testing the classifiers. In machine learning, it is standard practice to train the models on a portion of the data and assess how well they perform on data they have not previously seen. Before training the classifiers, dataset preprocessing is performed. This step involves various operations, one of which is normalization. Normalization scales the features in the dataset to a similar range, usually between 0 and 1, which helps improve the performance of some machine learning algorithms.

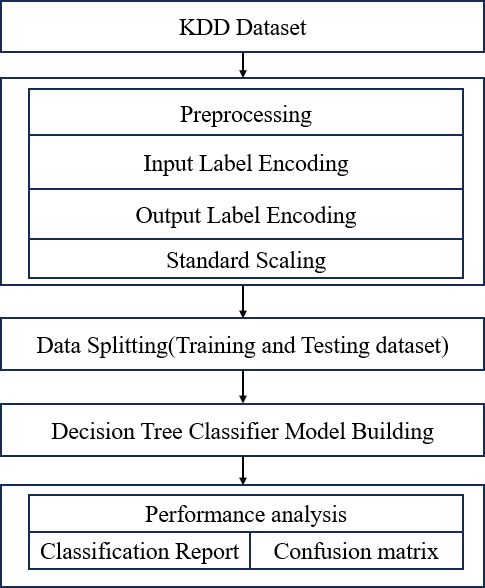


Figure 4.1 Proposed System Model

The suggested technique predicts assaults based on the test samples using three distinct classifiers: the support vector machine (SVM), and the Decision Tree Classifier (DTC). In this scenario, the Support Vector Machine, or SVM, is a technique for supervised machine learning that has shown to be both successful and popular. Both linear and non-linear classification problems are well within their capabilities to

solve thanks to their strong performance. To provide more precise predictions, the Random Forest method of ensemble learning incorporates the findings of a wide variety of decision trees. It is resistant to overfitting and typically performs well in various areas. DTC is a specific kind of artificial neural network that consists of many hidden layers. It can learn complex patterns and representations from data, making it suitable for tasks with high-dimensional inputs like intrusion detection.

After training the SVM, and DTC classifiers on the training data, they predict attacks on the test samples. Each classifier will generate its predictions based on the features of the test samples. An assessment of performance is carried out so that a comparison of the classifiers' effectiveness was made. At this stage, we will evaluate the accuracy of the classifiers' predictions based on the test data. The performance assessment aims to show that the suggested DTC approach is better than the other classifiers (namely, SVM) when it comes to identifying intrusions.

## DATA PREPROCESSING

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So, for this, we use data pre-processing tasks. Real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing requires tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

#### Standardization

Standard Scaler is applied to scale numeric features, ensuring that they have a mean of 0 and a standard deviation of 1. The 'Standard Scaler' from scikit-learn is used to standardize specific numeric features. Standardization is a common preprocessing step to bring features to a similar scale, which can improve the performance of some machine learning algorithms. This transformation is important for several reasons:

* + - 1. Equal Scaling: StandardScaler scales each feature to have the same scale. This is crucial for algorithms that are sensitive to the scale of features, such as gradient-based optimization algorithms (e.g., in neural networks) and distance- based algorithms (e.g., k-means clustering).
      2. Mean Centring: By subtracting the mean from each data point, StandardScaler centers the data around zero. This can help algorithms converge faster during training and improve their performance.
      3. Normalization: Scaling by the standard deviation normalizes the data, ensuring that features have comparable variances. This can prevent certain features from dominating others in the modelling process.

Interpretability: Standardized data is more interpretable because it puts all features on a common scale, making it easier to compare the relative importance of features

## DATASET SPLITTING

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.

Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

**Training Set**: A subset of dataset to train the machine learning model, and we already know the output.

**Test set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

## DTC ALGORITHM

DTC is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "DTC is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the DTC takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

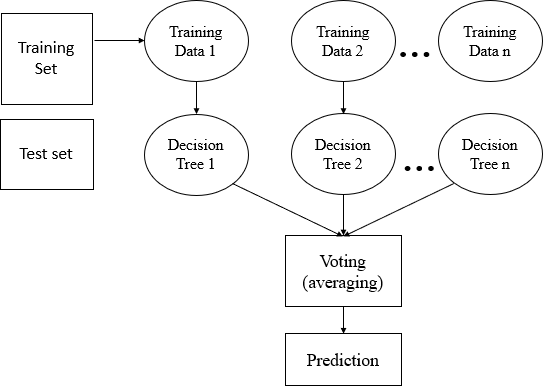


Fig. 4.2: DTC algorithm.

#### DTC algorithm

Step 1: In DTC n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample. Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

#### Important Features of DTC

* **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
* **Immune to the curse of dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
* **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build DTCs.
* **Train-Test split**- In a DTC we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
* **Stability**- Stability arises because the result is based on majority voting/ averaging.liter

#### Assumptions for DTC

Since the DTC combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better DTC classifier:

* There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
* The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the DTC algorithm

* It takes less training time as compared to other algorithms.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing.

#### Types of Ensembles

Before understanding the working of the DTC, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

**Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, DTC. Bagging, also known as Bootstrap Aggregation is the ensemble technique used by DTC. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

**Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

## ADVANTAGES OF PROPOSED SYSTEM

Decision trees are a popular machine learning algorithm with several advantages that make them useful for various tasks, including classification and regression. Here are some of the key advantages of decision trees:

* **No Assumptions about Data:** Decision trees do not require any assumptions about the data distribution or relationships between variables. They can handle both numerical and categorical data, making them versatile for different types of datasets.
* **Handling Non-Linear Relationships:** Decision trees can model complex, non-linear relationships in the data effectively. They can capture interactions between features without the need for feature engineering or manual transformation.
* **Feature Importance:** Decision trees provide a built-in feature importance ranking. By examining the splits in the tree and how much they improve the model's performance, you can identify which features are most relevant for making predictions. This information can guide feature selection and data understanding.
* **Scalability:** Decision trees are relatively computationally efficient, especially when compared to more complex algorithms like deep neural networks. They can handle large datasets with many features without significant computational overhead.
* **Handling Missing Values:** Decision trees can naturally handle missing values by making decisions based on the available information. They do not require imputation or preprocessing steps to handle missing data.
* **Robustness to Outliers:** Decision trees are robust to outliers in the data. Outliers may affect individual splits in the tree, but they are less likely to have a significant impact on the overall model's performance.
* **Ensemble Learning:** Decision trees can be used as building blocks for ensemble methods like Random Forests and Gradient Boosting, which often outperform single decision trees by reducing overfitting and improving generalization.
* **Non-Parametric:** Decision trees are non-parametric models, meaning they do not make assumptions about the underlying data distribution. This makes them flexible and adaptable to various types of data.
* **Easy to Implement:** Decision trees are relatively simple to implement from scratch, making them accessible to those with basic programming skills. Additionally, there are numerous libraries and software packages that provide decision tree implementations.

**CHAPTER 5 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general- purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

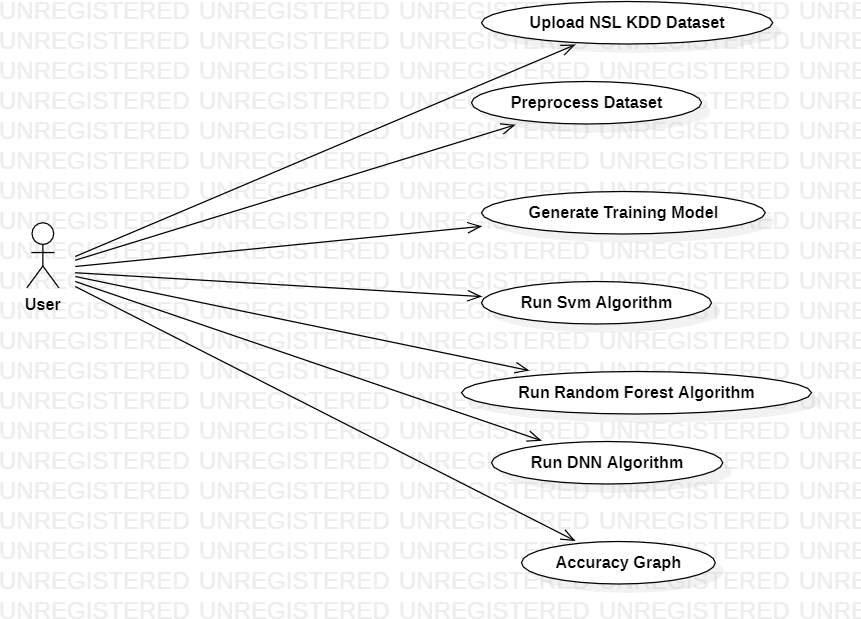
### GOALS:

The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

### USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

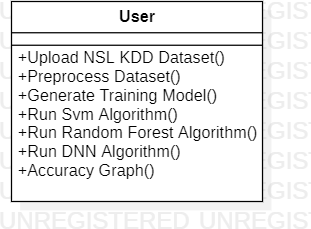


Figures 5.1 User Case Diagram

### CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing

the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



### SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

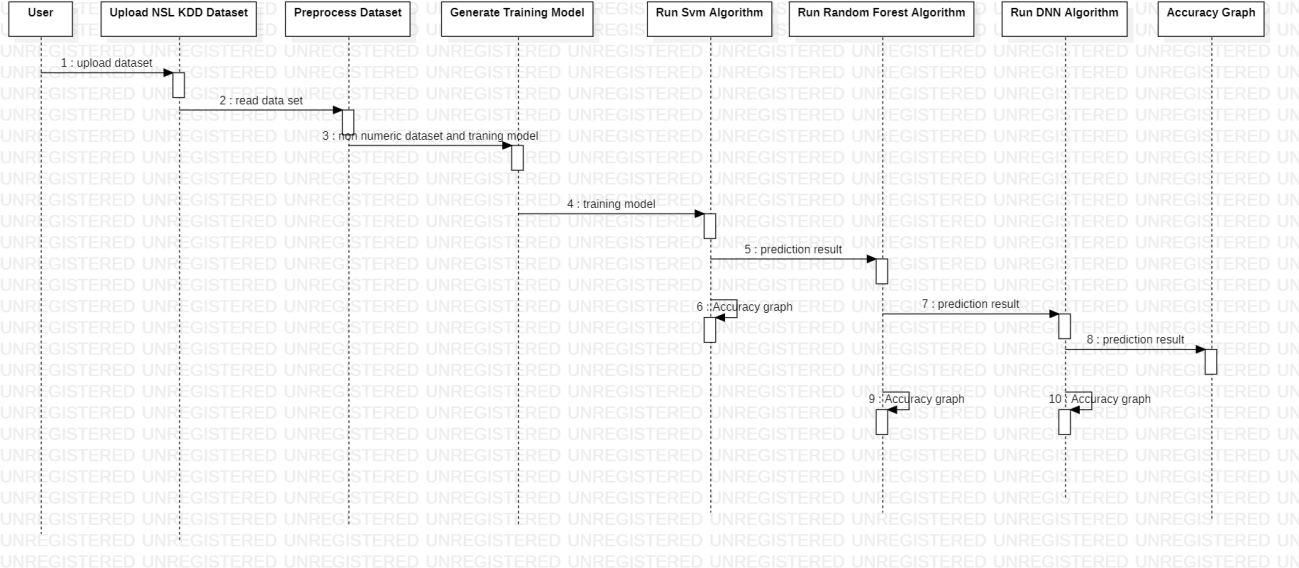
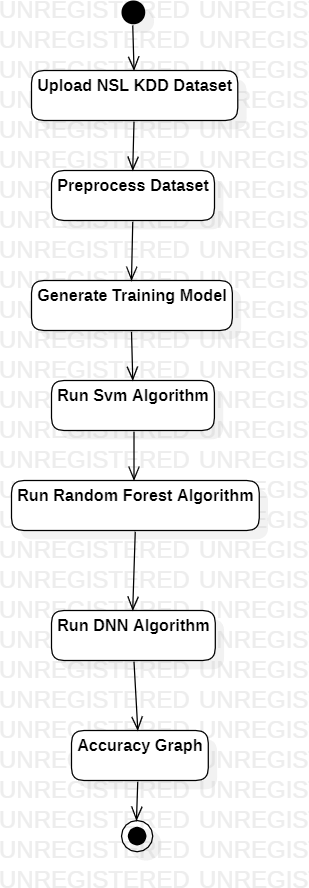


Figure 5.2 Sequence Diagram

### ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



Figures 5.3 Activity Diagram

**CHAPTER 6 SOFTWARE ENVIRONMENT**

#### What is Python?

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

#### Advantages of Python

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

1. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

1. Simple and Easy

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

1. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

1. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

1. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

#### Disadvantages of Python

1. **Speed Limitations:** We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.
2. **Weak in Mobile Computing and Browsers:** While it serves as an excellent server- side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

1. **Design Restrictions:** As you know, Python is dynamically-typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck- typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.
2. **Underdeveloped Database Access Layers:** Compared to more widely used technologies like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

This was all about the Advantages and Disadvantages of Python Programming Language.

#### Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

* + Print is now a function.
  + Views and iterators instead of lists
  + The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
  + There is only one integer type left, i.e., int. long is int as well.
  + The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
  + Text Vs. Data Instead of Unicode Vs. 8-bit

#### Purpose

We demonstrated that our approach enables successful segmentation of intra- retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

#### Modules Used in Project

#### TensorFlow:

TensorFlow is an open-source software library for numerical computation using data flow graphs. It's widely used for machine learning, deep learning, and artificial intelligence.

Key Features:

* + Building and deploying complex neural networks.
  + Efficient numerical computation on various devices (CPUs, GPUs, TPUs).
  + A large community and extensive documentation.

Use Cases: Image recognition, natural language processing, recommender systems, time series forecasting, and more.

#### NumPy:

NumPy is a fundamental library for scientific computing in Python. It provides high-performance multidimensional arrays and tools for efficient data manipulation and mathematical operations.

Key Features:

* + Efficient storage and manipulation of large datasets.
  + Array-based operations for vectorization and performance optimization.
  + Integration with other scientific libraries like SciPy and pandas.

Use Cases: Data analysis, scientific computing, image processing, signal processing, and more.

#### Pandas:

Pandas is a library for data analysis and manipulation in Python. It provides powerful data structures and tools for importing, cleaning, exploring, and analyzing data.

Key Features:

* + Efficient data structures like DataFrames and Series for representing and manipulating data.
  + Flexible data cleaning and transformation tools.
  + Advanced data analysis functionalities like pivoting, grouping, and merging.

Use Cases: Data cleaning, preprocessing, exploratory data analysis, feature engineering, and time series analysis.

#### Matplotlib:

Matplotlib is a library for creating various types of visualizations in Python. It allows you to create charts, graphs, plots, and other visual representations of your data.

Key Features:

* + Extensive variety of plot types and styles.
  + Flexible customization options for axes, labels, and annotations.
  + Integration with other libraries like NumPy and pandas.

Use Cases: Data exploration, communicating results, presentations, publications, and visual storytelling with data.

#### Scikit-learn:

Scikit-learn is a comprehensive library for machine learning in Python. It provides various algorithms and tools for building, evaluating, and deploying machine learning models.

Key Features:

* + Wide range of supervised and unsupervised learning algorithms.
  + Feature scaling, dimensionality reduction, and model selection tools.
  + Evaluation metrics and visualizations for model performance analysis.

Use Cases: Classification, regression, clustering, dimensionality reduction, feature engineering, and more.

**CHAPTER 7**

**SYSTEM REQUIREMENTS SPECIFICATIONS**

#### Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* + Python IDLE 3.7 version (or)
  + Anaconda 3.7 (or)
  + Jupiter (or)
  + Google colab

#### Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system : Windows, Linux

Processor : minimum AMD ryzen 5

Ram : minimum 8 GB

Hard disk : minimum 256GB

**CHAPTER 8 FUNCTIONAL REQUIREMENTS**

#### Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* + External Outputs, whose destination is outside the organization
  + Internal Outputs whose destination is within organization and they are the
  + User’s main interface with the computer.
  + Operational outputs whose use is purely within the computer department.
  + Interface outputs, which involve the user in communicating directly.

#### Output Definition

The outputs should be defined in terms of the following points:

* + Type of the output
  + Content of the output
  + Format of the output
  + Location of the output
  + Frequency of the output
  + Volume of the output
  + Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

#### Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

* + To produce a cost-effective method of input.
  + To achieve the highest possible level of accuracy.
  + To ensure that the input is acceptable and understood by the user.

#### Input Stages

The main input stages can be listed as below:

* + Data recording
  + Data transcription
  + Data conversion
  + Data verification
  + Data control
  + Data transmission
  + Data validation
  + Data correction

#### Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* + External inputs, which are prime inputs for the system.
  + Internal inputs, which are user communications with the system.
  + Operational, which are computer department’s communications to the system?
  + Interactive, which are inputs entered during a dialogue.

#### Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* + Type of input
  + Flexibility of format
  + Speed
  + Accuracy
  + Verification methods
  + Rejection rates
  + Ease of correction
  + Storage and handling requirements
  + Security
  + Easy to use
  + Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

#### Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

#### Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

#### Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

#### User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

#### User Interface Systems Can Be Broadly Classified As:

* + User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
  + Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

#### User Initiated Interfaces

User initiated interfaces fall into two approximate classes:

* + Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
  + Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

#### Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

* + The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
  + Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

#### Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

#### Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* + The system should be able to interface with the existing system
  + The system should be accurate
  + The system should be better than the existing system
  + The existing system is completely dependent on the user to perform all the duties.

**SOURCE CODE**

# In[1]:

import numpy as np import pandas as pd

from pandas import Series,DataFrame

from sklearn.preprocessing import LabelEncoder import sklearn as sk

dataset=pd.read\_csv(r"C:\Users\narla\OneDrive\Desktop\python\KDD.csv") dataset

# In[2]:

dataset.info() # In[3]:

dataset.isnull() # In[4]:

dataset.fillna(0,inplace=False) # In[5]:

dataset['label'] # In[6]:

x=dataset.drop('label',axis=1) x

# In[7]:

lab=LabelEncoder()

# In[9]:

dataset['service']=lab.fit\_transform(dataset['service']) dataset['protocol\_type']=lab.fit\_transform(dataset['protocol\_type']) dataset['flag']=lab.fit\_transform(dataset['flag']) dataset['label']=lab.fit\_transform(dataset['label'])

dataset

# In[10]:

import seaborn as sns

import matplotlib.pyplot as plt plot=sns.countplot(data=dataset,x="label") value\_counts=dataset["label"].value\_counts() for i,count in enumerate(value\_counts):

plot.text(x=i,y=count+i,s=str(count),ha="center") plot.set\_xlabel("Categories") plot.set\_ylabel("Count")

plot.set\_title("Count Plot of Labels") plt.show()

# In[11]:

x=dataset.iloc[:,0:41].values x

# In[12]:

y=dataset.iloc[:,41].values y

# In[13]:

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=0) print("x shape is",x.shape)

print("y shape is",y.shape) print("x\_test shape is",x\_test.shape) print("x\_train shape is",x\_train.shape) print("y\_train shape is",y\_train.shape) print("y\_test shape is",y\_test.shape)

# In[14]:

labels=set(dataset['label']) labels

# In[15]:

from sklearn.tree import DecisionTreeClassifier from sklearn.metrics import confusion\_matrix from sklearn.metrics import classification\_report from sklearn.metrics import accuracy\_score from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score import seaborn as sns

dt = DecisionTreeClassifier() dt.fit(x\_train,y\_train)

y\_pred = dt.predict(x\_test) print("Original y\_test values are ",y\_test)

print("predicted y\_test values are",y\_pred) accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted') recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted') print(f'Accuracy: {accuracy:.2f}') print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues', xticklabels=labels,yticklabels=labels)

plt.xlabel('Predicted') plt.ylabel('True') plt.title('Confusion Matrix of DTC') plt.show()

class\_report = classification\_report(y\_test, y\_pred) print('Classification Report of DTC:')

print(class\_report)

# In[16]:

from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report from sklearn.metrics import accuracy\_score from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score import seaborn as sns

rf = RandomForestClassifier() rf.fit(x\_train,y\_train)

y\_pred = rf.predict(x\_test) print("Original y\_test values are ",y\_test)

print("predicted y\_test values are",y\_pred) accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted') recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted') print(f'Accuracy: {accuracy:.2f}') print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=labels,ytickl abels=labels)

plt.xlabel('Predicted') plt.ylabel('True') plt.title('Confusion Matrix of RFC') plt.show()

class\_report = classification\_report(y\_test, y\_pred) print('Classification Report of RFC:') print(class\_report)

# In[17]:

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix from sklearn.metrics import classification\_report from sklearn.metrics import accuracy\_score from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score import seaborn as sns

svm = SVC() svm.fit(x\_train,y\_train) y\_pred = svm.predict(x\_test)

print("Original y\_test values are ",y\_test)

print("predicted y\_test values are",y\_pred) accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted') recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted') print(f'Accuracy: {accuracy:.2f}') print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=labels,ytickl abels=labels)

plt.xlabel('Predicted') plt.ylabel('True')

plt.title('Confusion Matrix of SVM') plt.show()

class\_report = classification\_report(y\_test, y\_pred) print('Classification Report of SVM:') print(class\_report)

# In[18]:

from sklearn.neighbors import KNeighborsClassifier from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report from sklearn.metrics import accuracy\_score from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score import seaborn as sns

knn = KNeighborsClassifier() knn.fit(x\_train,y\_train) y\_pred = knn.predict(x\_test)

print("Original y\_test values are ",y\_test) print("predicted y\_test values are",y\_pred) accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted') recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted') print(f'Accuracy: {accuracy:.2f}') print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=labels,ytickl abels=labels)

plt.xlabel('Predicted') plt.ylabel('True')

plt.title('Confusion Matrix of KNC') plt.show()

class\_report = classification\_report(y\_test, y\_pred) print('Classification Report of KNC:') print(class\_report)

# In[19]:

from sklearn.neural\_network import MLPClassifier from sklearn.metrics import confusion\_matrix from sklearn.metrics import classification\_report from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score from sklearn.metrics import f1\_score import seaborn as sns

mlp = MLPClassifier() mlp.fit(x\_train,y\_train) y\_pred = mlp.predict(x\_test)

print("Original y\_test values are ",y\_test) print("predicted y\_test values are",y\_pred) accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted') recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted')

print(f'Accuracy: {accuracy:.2f}') print(f'Precision: {precision:.2f}') print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=labels,ytickl abels=labels)

plt.xlabel('Predicted') plt.ylabel('True')

plt.title('Confusion Matrix of MLPC') plt.show()

class\_report = classification\_report(y\_test, y\_pred) print('Classification Report of MLPC:') print(class\_report)

# In[20]:

from sklearn.linear\_model import LinearRegression from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report from sklearn.metrics import accuracy\_score from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score import seaborn as sns

lin\_reg = LinearRegression() lin\_reg.fit(x\_train,y\_train) y\_pred = lin\_reg.predict(x\_test) y\_pred=np.round(y\_pred)

print("Original y\_test values are ",y\_test) print("predicted y\_test values are",y\_pred) accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted') recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted') print(f'Accuracy: {accuracy:.2f}') print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=labels,ytickl abels=labels)

plt.xlabel('Predicted') plt.ylabel('True') plt.title('Confusion Matrix of LR') plt.show()

class\_report = classification\_report(y\_test, y\_pred) print('Classification Report of LR:') print(class\_report)

# In[21]:

from sklearn.ensemble import AdaBoostClassifier from sklearn.metrics import confusion\_matrix from sklearn.metrics import classification\_report from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score from sklearn.metrics import f1\_score import seaborn as sns

ada\_boost = AdaBoostClassifier() ada\_boost.fit(x\_train,y\_train) y\_pred = ada\_boost.predict(x\_test)

print("Original y\_test values are ",y\_test) print("predicted y\_test values are",y\_pred) accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted') recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted') print(f'Accuracy: {accuracy:.2f}') print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=labels,ytickl abels=labels)

plt.xlabel('Predicted') plt.ylabel('True') plt.title('Confusion Matrix of ABC') plt.show()

class\_report = classification\_report(y\_test, y\_pred) print('Classification Report of ABC:') print(class\_report)

# In[22]:

from sklearn.ensemble import ExtraTreesClassifier from sklearn.metrics import confusion\_matrix from sklearn.metrics import classification\_report from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score from sklearn.metrics import recall\_score from sklearn.metrics import f1\_score import seaborn as sns

extra\_tree = DecisionTreeClassifier() extra\_tree.fit(x\_train,y\_train)

y\_pred = extra\_tree.predict(x\_test) print("Original y\_test values are ",y\_test) print("predicted y\_test values are",y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted') recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted') print(f'Accuracy: {accuracy:.2f}') print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1 Score: {f1:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix,annot=True,fmt='d',cmap='Blues',xticklabels=labels,ytickl abels=labels)

plt.xlabel('Predicted') plt.ylabel('True') plt.title('Confusion Matrix of ETC') plt.show()

class\_report = classification\_report(y\_test, y\_pred) print('Classification Report of ETC:') print(class\_report)

**RESULTS AND DISCUSSION**

## DATASET

The KDD dataset is a benchmark dataset often used to assess machine learning–based IDS. This is because intrusion detection makes use of the KDD dataset. The reason for this was seen in the previous sentence. In 1999, for the DARPA Intrusion Detection Evaluation Program, the first version of the dataset was compiled, and this version was utilized in the KDD Cup in 1999. This dataset is an upgraded version of the one given for the first time at the KDD Cup 1999. There were many ways in which the original KDD Cup dataset might have been enhanced, including its high level of duplication and its inadequate representation of the network traffic that exists today.

As a consequence of these problems, the KDD dataset was produced as a remedy to overcome the shortcomings of the dataset that was originally used. The KDD dataset collects network traffic characteristics compiled from various network assaults and typical network traffic. The creation of the dataset required the collection of certain characteristics of network traffic. The dataset has forty-one properties, some of which are statistical aspects such as the number of unsuccessful login attempts and connections to the same server. In addition to that, the dataset contains information on the total number of connections made to the same host. The sort of protocol that is being used, the kind of service that is being utilized, the IP address of the source, as well as the IP address of the destination are all additional elements.

The data set consists of not just one but two separate sets of information, which are referred to as the training set and the test set, respectively. The total number of samples in the training set is substantially more than in the test set, which is lower. The total number of samples in the training set is 125,973. The samples included in the dataset were either categorized as standard or belonging to one of the following four types of attacks: DoS, probe, U2R, or R2L all come to mind. The standard samples have also been included in the d. One of the most notable aspects of the KDD dataset is that it includes attacks that are already known about and those that have yet to be found. Because of this, it is a more accurate portrayal of contemporary network traffic, which contains a wide variety of assaults, some of which were unknown in the

past. Because of this, it is much more difficult for ML-based IDS models since they need to be able to identify both known and new threats.

#### Columns/features of the dataset

The attributes of the dataset as follows

* **duration**: Duration of the connection in seconds. Represents the time elapsed for the connection.
* **protocol\_type**: The transport layer protocol used in the connection, such as TCP (Transmission Control Protocol), UDP (User Datagram Protocol), ICMP (Internet Control Message Protocol), etc.
* **service**: The network service on the destination, indicating the type of service being accessed (e.g., HTTP, FTP, Telnet).
* **flag**: Flags associated with the connection, indicating the status or state of the connection (e.g., FIN, SYN, RST).
* **src\_bytes**: The number of source bytes transmitted in the connection.
* **dst\_bytes**: The number of destination bytes transmitted in the connection.
* **land**: Binary indicator of whether the connection is from/to the same host/port. 1 if the connection is from/to the same host/port, 0 otherwise.
* **wrong\_fragment**: The number of "wrong" fragments in the connection, indicating irregularities in fragment handling.
* **urgent**: Urgent packet count. Represents the number of packets with the urgent flag set.
* **hot**: Hot indicator, suggesting activity in the connection. This feature is often used to detect suspicious behavior.
* **num\_failed\_logins**: Number of failed login attempts during the connection.
* **logged\_in**: Binary indicator of whether a user is logged in. 1 if a user is logged in, 0 otherwise.
* **num\_compromised**: Number of compromised conditions. Indicates the number of security compromises.
* **root\_shell**: Binary indicator of whether a root shell is obtained. 1 if a root shell is obtained, 0 otherwise.
* **su\_attempted**: Binary indicator of whether the **su** command was attempted. 1 if **su** command attempted, 0 otherwise.
* **num\_root**: Number of root accesses. Represents the number of times the root user is accessed.
* **num\_file\_creations**: Number of file creations. Indicates the number of new files created during the connection.
* **num\_shells**: Number of shell prompts. Represents the number of shell prompts (command line interfaces) during the connection.
* **num\_access\_files**: Number of access control files. Indicates the number of times access control files are accessed.
* **num\_outbound\_cmds**: Number of outbound commands. Represents the number of outbound commands executed.
* **is\_host\_login**: Binary indicator of whether the login belongs to the "host list" (root privileges). 1 if the login is on the host list, 0 otherwise.
* **is\_guest\_login**: Binary indicator of whether the login is a guest login. 1 if it's a guest login, 0 otherwise.
* **count**: Number of connections to the same host as the current connection in the past two seconds.
* **srv\_count**: Number of connections to the same service as the current connection in the past two seconds.
* **serror\_rate**: Error rate for connections. Represents the percentage of connections that have "SYN" errors.
* **srv\_serror\_rate**: Error rate for service-related connections. Represents the percentage of service-related connections that have "SYN" errors.
* **rerror\_rate**: Error rate for rejected connections. Represents the percentage of rejected connections.
* **srv\_rerror\_rate**: Error rate for service-related rejected connections. Represents the percentage of service-related rejected connections.
* **same\_srv\_rate**: Percentage of connections to the same service.
* **diff\_srv\_rate**: Percentage of connections to different services.
* **srv\_diff\_host\_rate**: Percentage of connections to different hosts for the same service.
* **dst\_host\_count**: Number of connections to the same host as the current connection.
* **dst\_host\_srv\_count**: Number of connections to the same service as the current connection on the destination host.
* **dst\_host\_same\_srv\_rate**: Percentage of connections to the same service on the destination host.
* **dst\_host\_diff\_srv\_rate**: Percentage of connections to different services on the destination host.
* **dst\_host\_same\_src\_port\_rate**: Percentage of connections from the same source port on the destination host.
* **dst\_host\_srv\_diff\_host\_rate**: Percentage of connections to different hosts for the same service on the destination host.
* **dst\_host\_serror\_rate**: Error rate for connections to the destination host.
* **dst\_host\_srv\_serror\_rate**: Error rate for service-related connections to the destination host.
* **dst\_host\_rerror\_rate**: Error rate for rejected connections to the destination host.
* **dst\_host\_srv\_rerror\_rate**: Error rate for service-related rejected connections to the destination host.
* **label**: Binary label indicating normal (0) or malicious (1) activity in the network connection.

## PERFORMANCE MEASURES

It is common practice to employ performance metrics based on confusion matrices when assessing the efficacy of an intrusion detection system (IDS) model. True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) are the four metrics that are included in the confusion matrix, which is a tabular representation of the anticipated and actual classification results. The confusion matrix was found in the following format: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). It was discovered that the confusion matrix has the following structure: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

The variable TP shows the total number of real positive events the model has accurately categorized. The variable FP represents the total number of instances in which the model incorrectly identified a negative event as a positive. The total number of real-world scenarios the algorithm correctly detected as having a negative consequence is denoted by the TN symbol. FN represents the number of genuine positive examples that are, for some reason, given a negative interpretation by the model. Calculation of the following performance measures is possible based on these four indicators.

**Accuracy:** Accuracy is the ratio of the number of correct predictions to the total number of predictions made by the model. It is calculated as (TP + TN) / (TP + FP + TN + FN).

**Precision:** Precision is the ratio of the number of true positives to the total number of optimistic predictions the model makes. It is calculated as TP / (TP + FP).

**Recall:** Recall is the ratio of the number of true positives to the total number of actual positive instances. It is calculated as TP / (TP + FN).

**F1-Score:** The F1-score is determined by finding the harmonic mean of the values for accuracy and recall. It provides a measurement that is accurate in terms of recall as well as accuracy at the same time. It is calculated by using the formula 2 times (precision times recall) divided by (precision plus recall).

## Simulation results

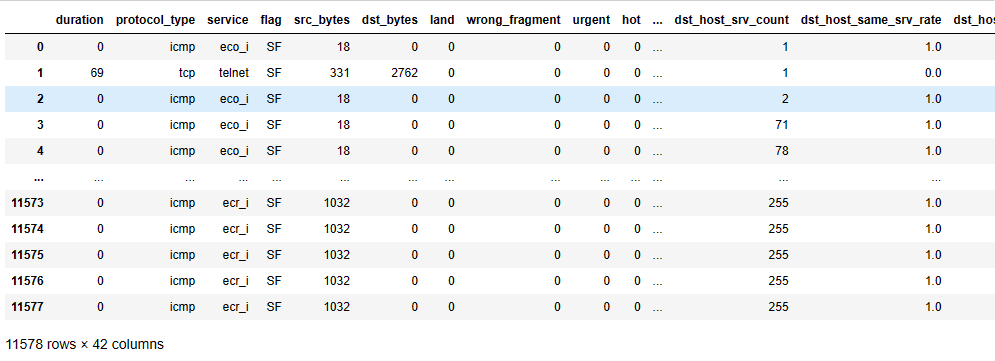


Table 10.1. Sample dataset.



Table 10.2. Null values status in dataset.

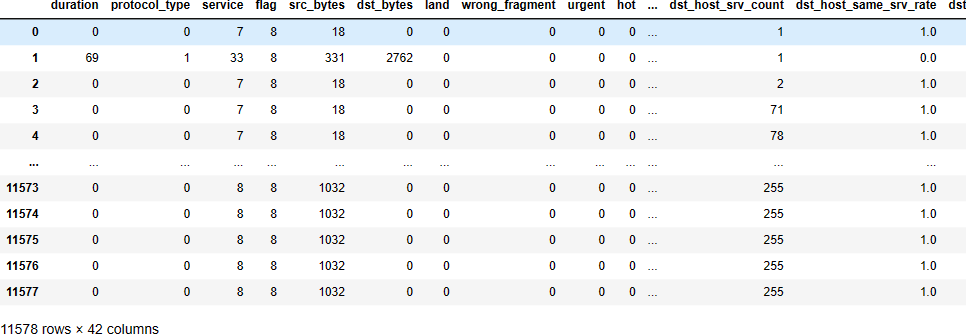
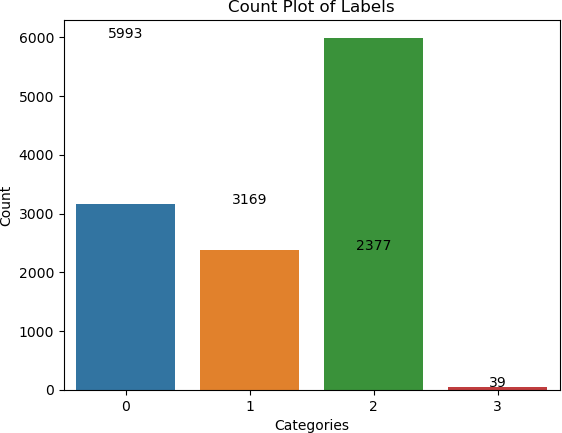
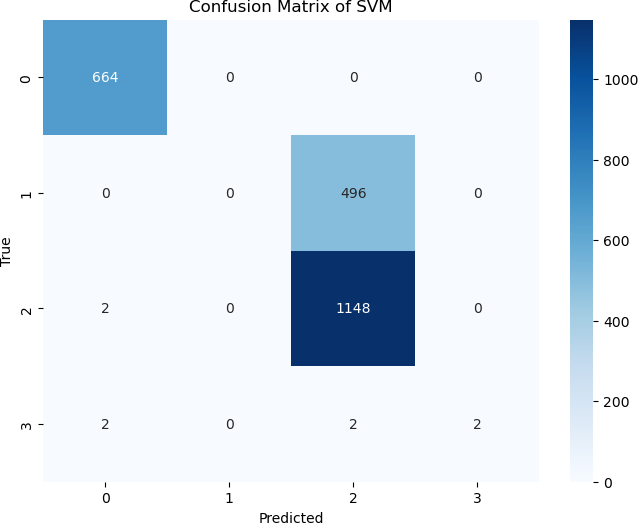


Table 10.3. Dataset after label encoding.



Graph 10.1 Dataset count plot.



Graph 10.2 Confusion matrix of existing SVM.

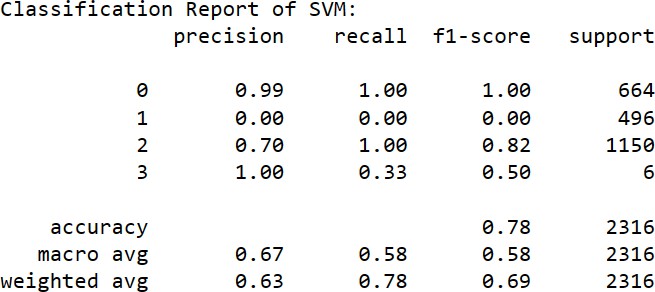
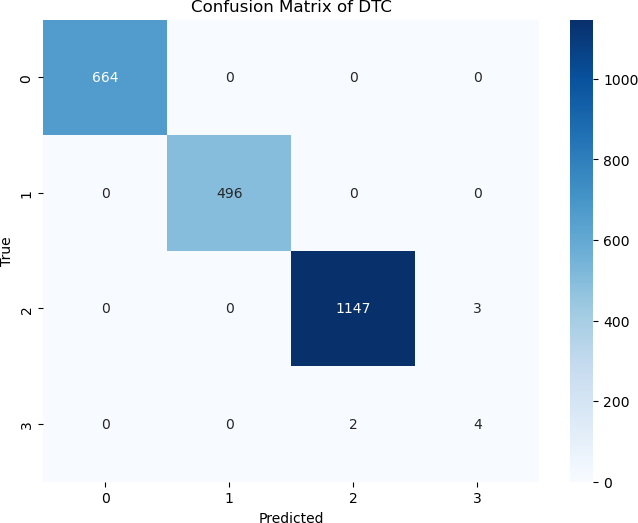


Table 10.4 Classification report of existing SVM.



Graph 10.3 Confusion matrix of proposed DTC.

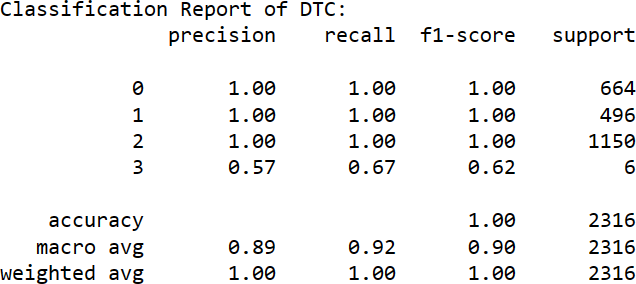


Table 10.5 Classification report of proposed DTC.

## Conclusion

**CONCLUSION AND FUTURE SCOPE**

This study presents a proposed method for attack prediction using the KDD dataset. Its performance is compared against traditional classifiers like SVM, focusing on demonstrating the superiority of the proposed DTC approach. The methodology begins with splitting the KDD dataset into two portions: 80% is used for training the models, while the remaining 20% is dedicated to testing and evaluating their performance. Before training, the dataset undergoes preprocessing operations to normalize the data, ensuring that all features have a consistent scale, which is essential for effective machine learning. Three different classifiers are employed to predict attacks from the test samples: SVM, and DTC. Here, SVM is known for its effectiveness in binary classification tasks, Random Forest is a powerful ensemble learning method, and DTC have gained popularity due to their ability to automatically learn complex patterns in data. After training the classifiers and obtaining predictions on the test set, performance evaluation metrics are used to compare their effectiveness. The proposed DTC method is specifically focused on, and its results are compared against the SVM and Random Forest classifiers. The performance evaluation results provide evidence that the proposed DTC approach outperforms the traditional SVM and Random Forest methods in predicting attacks. This suggests that the DTC classifier can better generalize and capture intricate patterns within the dataset, leading to more accurate and reliable predictions.

## Future scope

Looking ahead, the future holds promising opportunities for attack prediction and cybersecurity. As technology advances, there are several key areas of future scope that researchers and practitioners can explore to further enhance the effectiveness of attack prediction methods. With ongoing advancements in deep learning algorithms, developing even more sophisticated and efficient neural network architectures for attack prediction is possible. Researchers can explore novel architectures, explore transfer learning techniques, and investigate methods to optimize hyperparameters for improved performance.

Feature engineering plays a crucial role in the performance of machine learning models. In the future, researchers can focus on identifying and selecting relevant features to improve the efficiency and accuracy of attack prediction algorithms. Automated feature selection techniques and domain-specific feature engineering approaches can be explored to streamline this process. As cyber threats evolve rapidly, the ability to predict attacks in real time becomes increasingly important. Future research can centre on developing real-time and streaming data analysis methods that can efficiently detect and predict attacks as they happen, enabling proactive responses to potential threats.

In cybersecurity datasets, the occurrence of attacks is often significantly lower than in normal instances, leading to imbalanced data. Future research may focus on resolving concerns of class imbalance by using methods such as oversampling, under- sampling, or sophisticated algorithms developed expressly for use with unbalanced data. This will hopefully result in more accurate and dependable predictions. Ensemble methods combine the predictions of multiple classifiers to enhance overall performance. The future scope lies in exploring ensemble learning techniques to combine the strengths of various classifiers, such as SVM, Random Forest, and DNN, to create more robust and accurate attack prediction models.

**REFERENCES**

[1]. Logeswari, G., S. Bose, and T. Anitha. "An Intrusion Detection System for SDN Using Machine Learning." Intelligent Automation & Soft Computing 35.1 (2023).

[2]. Talukder, Md Alamin, et al. "A dependable hybrid machine learning model for network intrusion detection." Journal of Information Security and Applications 72 (2023): 103405.

[3]. Rizvi, Syed, et al. "Deep learning based network intrusion detection system for resource-constrained environments." Springer. 2023.

[4]. Mahadik, Shalaka, Pranav M. Pawar, and Raja Muthalagu. "Efficient Intelligent Intrusion Detection System for Heterogeneous Internet of Things (HetIoT)." Journal of Network and Systems Management 31.1 (2023): 2.

[5]. Yang, Zhen, et al. "A systematic literature review of methods and datasets for anomaly-based network intrusion detection." Computers & Security (2022): 102675.

[6]. Sengan, Sudhakar, et al. "Secured and privacy-based IDS for healthcare systems on E-medical data using machine learning approach." International Journal of Reliable and Quality E-Healthcare (IJRQEH) 11.3 (2022): 1-11.

[7]. Liu, Gaoyuan, et al. "An enhanced intrusion detection model based on improved kNN in WSNs." Sensors 22.4 (2022): 1407.

[8]. Otoum, Yazan, Dandan Liu, and Amiya Nayak. "DL‐IDS: a deep learning– based intrusion detection framework for securing IoT." Transactions on Emerging Telecommunications Technologies 33.3 (2022): e3803.

[9]. Khraisat, Ansam, et al. "Survey of intrusion detection systems: techniques, datasets and challenges." Cybersecurity 2.1 (2019): 1-22.

[10]. Yulianto, Arif, Parman Sukarno, and Novian Anggis Suwastika. "Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset." Journal of Physics: Conference Series. Vol. 1192. No. 1. IOP Publishing, 2019.

[11]. Agrawal, Shaashwat, et al. "Federated learning for intrusion detection system: Concepts, challenges and future directions." Computer Communications (2022).

[12]. Thakkar, Ankit, and Ritika Lohiya. "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions." Artificial Intelligence Review 55.1 (2022): 453-563.

[13]. Nasir, Muhammad Hassan, et al. "Swarm intelligence inspired intrusion detection systems—a systematic literature review." Computer Networks (2022): 108708.

[14]. Saba, Tanzila, et al. "Anomaly-based intrusion detection system for IoT networks through deep learning model." Computers and Electrical Engineering 99 (2022): 107810.

[15]. Le, Kim-Hung, et al. "IMIDS: An intelligent intrusion detection system against cyber threats in IoT." Electronics 11.4 (2022): 524.

[16]. Naseri, Touraj Sattari, and Farhad Soleimanian Gharehchopogh. "A feature selection based on the farmland fertility algorithm for improved intrusion detection systems." Journal of Network and Systems Management 30.3 (2022): 40.

[17]. Le, Thi-Thu-Huong, et al. "Classification and explanation for intrusion detection system based on ensemble trees and SHAP method." Sensors 22.3 (2022): 1154.

[18]. Alzaqebah, Abdullah, et al. "A modified grey wolf optimization algorithm for an intrusion detection system." Mathematics 10.6 (2022): 999.

[19]. Yu, Jing, Xiaojun Ye, and Hongbo Li. "A high precision intrusion detection system for network security communication based on multi-scale convolutional neural network." Future Generation Computer Systems 129 (2022): 399-406.

[20]. Imran, Muhammad, et al. "An intelligent and efficient network intrusion detection system using deep learning." Computers and Electrical Engineering 99 (2022): 107764.

[21]. Mendonça, Robson V., et al. "A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms." Expert Systems

39.5 (2022): e12917.

[22]. Whelan, Jason, Abdulaziz Almehmadi, and Khalil El-Khatib. "Artificial intelligence for intrusion detection systems in unmanned aerial vehicles." Computers and Electrical Engineering 99 (2022): 107784.

[23]. Yadav, Neha, et al. "Intrusion detection system on IoT with 5G network using deep learning." Wireless Communications and Mobile Computing 2022 (2022): 1-13.

[24]. Mushtaq, Earum, et al. "A two-stage intrusion detection system with auto-encoder and LSTMs." Applied Soft Computing 121 (2022): 108768.

[25]. Thakkar, Ankit, and Ritika Lohiya. "Fusion of statistical importance for feature selection in Deep Neural Network-based Intrusion Detection System." Information Fusion 90 (2023): 353-363.

[26]. Qiu, Weicheng, et al. "Hybrid intrusion detection system based on Dempster- Shafer evidence theory." Computers & Security 117 (2022): 102709.

[27]. Azeroual, Otmane, and Anastasija Nikiforova. "Apache spark and mllib-based intrusion detection system or how the big data technologies can secure the data." Information 13.2 (2022): 58.

[28]. Chang, Victor, et al. "A survey on intrusion detection systems for fog and cloud computing." Future Internet 14.3 (2022): 89.

[29]. Muhammad, M. U. U. A. H., and A. M. S. F. M. Saleem. "Intelligent Intrusion Detection System for Apache Web Server Empowered with Machine Learning Approaches." International Journal of Computational and Innovative Sciences 1.1 (2022): 1-8.

[30]. Friha, Othmane, et al. "FELIDS: Federated learning-based intrusion detection system for agricultural Internet of Things." Journal of Parallel and Distributed Computing 165 (2022): 17-31.