Software Defined Network (SDN) Based Intrusion Detection Technique to Enhance Network Performance Using Machine Learning

Lamesgen Asmare Tesema

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Electrical Engineering

Office of Graduate Studies

Adama Science and Technology University

June, 2024
Adama, Ethiopia

Software Defined Network (SDN) Based Flood Intrusion Technique to Enhance Network Performance Using Machine Learning

Lamesgen Asmare Tesema

Advisor: Dr. Nune Sreenivas

A thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Electrical Engineering

Office of Graduate Studies

Adama Science and Technology University

June, 2024

Adama, Ethiopia

DECLARATION

I, Lamesgen Asmare, hereby declare that this Master Thesis entitled “Software Defined Network (SDN) Based Flood Detection Technique to Enhance Network Performance Using Machine Learning” is my original work. That is, it has not been submitted forthe award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Lamesgen Asmare _____ _____

Name of the candidate Signature Date

RECOMMENDATION

I/we, the advisor(s) of this thesis, hereby certify that I/we have read the revised version of the thesis entitled "Software Defined Network (SDN) Based Flood Detection Technique to Enhance Network Performance Using Machine Learning" prepared under my/our guidance by Lamesgen Asmare submitted in partial fulfillment of the requirements for the degree of Mater's of Science in Computer Science and Electrical Engineering. Therefore, I/we recommend the submission of revised version of the thesis to the department following the applicable procedures.

Dr. Nune Sreenivas _____ _____
Major Advisor/Supervisor Signature Date

Approval Page

I/we, the advisors of the thesis entitled "**Software Defined Network (SDN) Based Intrusion Detection Technique to Enhance Network Performance Using Machine Learning**", hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. Nune Sreenivas	_____	_____
Major Advisor	Signature	Date
_____	_____	_____
Co-advisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by **Lamesgen Asmare Tessema**, have read and evaluated the thesis entitled "**Software Defined Network (SDN) Based Intrusion Detection Technique to Enhance Network Performance Using Machine Learning**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in **Computer Science and Engineering**.

_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis are contingent upon the submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____	_____	_____
Department Head	Signature	Date
_____	_____	_____
School Dean	Signature	Date
_____	_____	_____
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGEMENT

Firstly, above all thanks to the almighty God for his help in giving me courage to cope up with complicated situations I have faced for pursuing the study and for his help and encouragement during my whole study time.

Secondly, I would like to pass my gratitude for Nune Sreenivas (PhD) for their support, understanding, care and invaluable comments in my work.

Thirdly, I would also like to thank my friends and family for their love and support during this process. Without them, this journey would not have been possible.

Fourthly, I am grateful to everyone who has supported me throughout this process. Without yourhelp and guidance, this thesis would not have been possible.

Finally, I would like to thank all of the participants in my study for their time and willingness to share their experiences. This work would not have been possible without their contribution

Table of Contents

LIST OF TABLE

LIST OF FIGURE

LIST OF ABBREVIATION

IoT	Internet of Things
API	Application Program Interface
ARP	Address Resolution Protocol
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
FCAPS	Faults, Configuration, Accounting, Performance and Security
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Massage Protocol
ICT	Information Communication Technology
IDS	Intrusion detection system
IPAM	IP address management
KDD	Knowledge Discovery and Data Mining
LAN	Local Area Network
MITM	Man In The Middle
ML	Machine Learning
NetSDN	SDN Intrusion detection
NFV	Network Functions Virtualization
NTP	Network Time Protocol
ONOS	Open Networking Operating System
PoD	Ping of Death
QoS	Quality of Service
R2L	Remote to Local
SDN	Software Define Network
SMTP	Simple Message Transport Protocol
TCP	Transmission Control Protocol
U2R	User to Root
UDP	User Datagram Protocol

TN	True Negative
ANN	Artificial Neural Network
API	Application Programming Interface
CLI	Command Line Interface
CSV	Comma-Separated Values
DR	Detection Rate
FAR	False Alarm Rate
FN	False Negative
FP	False Positive
IDE	Integrated Development Environment
KNN	K-Nearest Neighbor
MLP	Multi-layer Perceptron
ODCA	Open Data Center Alliance
SVM	Support Vector Machine
TNR	True Negative Rate
TP	True Positive

ABSTRACT

The internet has brought about a significant transformation in the world. It serves as a tool for individuals to maintain their social connections and reach out to others within their networks for support. However, the act of sharing personal and professional information online exposes individuals and organizations to various risks. Given the integral role of the internet in our daily lives, the security of our data is constantly under threat. Consequently, the role of Intrusion Detection Systems (IDS) in safeguarding internet users against malicious network attacks is paramount. An Intrusion Detection System (IDS) is a security mechanism that continuously monitors network traffic for any suspicious activities and promptly alerts users upon detecting such anomalies. This paper aims to explore three distinct classifications, focusing on machine learning algorithms such as Artificial Neural Network (ANN), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). These algorithms will be employed to determine the most accurate approach using the SDN Intrusion Dataset in the initial phase. Subsequently, based on the outcomes of the first stage, the most effective algorithm will be applied to process our database. The goal of this work is to detect suspicious attacks on the SDN network by using supervised machine learning methods. A flood attack is an attack that has a huge impact on the performance of the network because it can prevent the victim from reacting. Response, preventing him from providing services and wasting network resources. To maintain the performance of the network, these attacks must be detected. So we proposed a supervising machine learning model. The model SVM, KNN, and ANN received performance evaluations of 93.78, 96.56, and 93.63, respectively. Based on the result KNN model yielded the best accuracy results, KNN classifier strategies achieved the desired results.

***Key Words**: SDN network, DDoS attack, Intrusion detection, controller, NetSDN.*

CHAPTER ONE

NTRODUCTION

1.1. Background of the study

Software Defined Networking (SDN) is a widely employed dynamic and cost-effective interconnected framework, making it well-suited for diverse devices and network projects. This architecture segregates network control and advancement activities, allowing for straightforward programmability of network control and preoccupation of framework components for applications and organizational services. Constructing an Intrusion Detection System (IDS) based on network to detect DDoS attacks using information from network traffic flows is crucial in SDN. In SDN, the control planes (CP) are separated by network devices, distinguishing it from traditional network design, which lacks dynamic adjustability due to its fixed link nature.

Network managers may coordinate and regulate a wide range of network components, such as service, topology, traffic flow, and network rules in a single or multiple tenant environment, by utilizing high-level languages and application programming interfaces (APIs) with software define network(SDN). SDN aims to improve network control by enabling companies or service providers to react swiftly and effortlessly to evolving business requirements. In software-defined networks, network engineers or administrators can manage traffic from a single console using a console-based console. The centralized SDN controller will instruct the switches to supply network services as needed, regardless of the precise connection type between a server and a device (Amin, Reisslein, & Shah, 2018).

Conventional networks can be replaced by SDN. The abstraction of the control and data plane serves as its foundation. The basic concept is to produce fewer intricate data plane components, such as switches, that solely forward traffic in accordance with a predetermined set of software-defined plane control rules. This removes the need to distinguish between proprietary interfaces on different devices and gives network administration autonomy over data plane device suppliers. Through the SDN control plane, SDN enables network applications and services to view the network as a single logical unit and to provide all devices with uniform access. This allows software to control network traffic flow to access the network's top layer. The goal of software-defined networking (SDN) is to increase the agility and flexibility of network management. A collection of streams, each containing a set of packets, makes up network traffic. Every packet that arrives over the network in an SDN switch is matched in SDN. The SDN controller will only receive packet headers. Stream header fields are referred to as tuples in terminology (Zhu et al., 2020).

After conducting a thorough and comprehensive analysis of the networking trends, we have decided to focus our efforts on the SDN-based intrusion detection. Despite the numerous benefits of this technology and our initial implementation of for improved performance, we believe that security is a major concern. While the centralized control of SDN offers significant advantages, the entire network could be compromised if the controller is targeted in an attack. Therefore, we have chosen to launch a DDoS attack on the control layer in the SDN-based network. It is important to note that is vulnerable to various threat vectors, with DoS and DDoS attacks being a common concern. In the literature, there are numerous methods for detecting DDoS attacks.

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks disrupt machine or network resources for legitimate users by overwhelming network resources with excessive traffic, causing bandwidth and memory consumption. These attacks can utilize various protocols, including UDP, TCP, and HTTP, as documented in (Corsetti et al., 2021), and have significantly increased in frequency in recent years. Extensive research, including(Anusuya, 2023), has been conducted on these attacks, with studies like (Dehkordi et al., n.d.), demonstrating the use of machine learning (ML) classifiers for detection. While some studies have achieved success with classifiers like Random Forests (RF), this research aims to highlight the potential for accuracy improvements through the implementation of advanced classification algorithms, such as of k-nearest neighbor, to detect DoS/DDoS type attacks. Furthermore, utilizing a diverse and high-variance dataset could enhance the robustness of these algorithms.

1.2. Statement of the problem

Software Defined Networks (SDN) face a multitude of cyberattack vulnerabilities due to their distinctive architecture. In an SDN, the controller is responsible for making every decision and serves as the central hub for all traffic. While centralization offers advantages, if compromised, it can result in the entire network being disrupted, making it an attractive target for attackers. Additionally, the OpenFlow Protocol used in SDN networks, while essential for controller-device communication, can be exploited if not adequately secured. Attackers may exploit weaknesses in the protocol to gain control of network elements. Consequently, these security threats render SDNs susceptible to various types of cyberattacks.

Many prepared methods are studied and evaluated as they become available in the literature. However, these methods can reduce the impact of flood attacks, but not when the amount of attack traffic is very large and each study has a single point of failure.

Therefore, the research this study was eliminated a single point of failure in the SDN network

to forward the packet directly to the destination host and detect the large number of packets generated by the attacker. The relative usefulness of predictive tools and others functions would investigated using computational intelligence techniques such as global sensitivity analysis and parallel coordinates.

1.3. Research questions

This study aimed to addresses the problem by answering the following question:

Q1. How to select more relevant features to detect an attack on SDN network traffic?

Q2. Which algorithms are more preferable to detect malicious activities in software define network?

Q3. How to measure the performance of our model?

1.4. Objectives of the Study

1.4.1. General Objective

The general objective of this thesis work is to improve the detect rate of intrusion malicious attack that to be caused to DDoS attack to improve SDN based LAN network performance using machine learning.

1.4.2. Specific Objective

The specific objectives we have followed to accomplish the general objectives are:

- To select dataset
- To identify the most significant attributes to reduce false alarm rate and improve detection accuracy.
- To specify an algorithm that implements detection of flooding attack
- To evaluate the performance of proposed model result.

1.5. Scope of the Study

The scope of this research work is only identify, detected and classified the intrusion malicious traffic that to be caused to DDoS attack in SDN network architecture using supervised machine learning model.

1.6. Significance of the study

This research, SDN based intrusion detection security mechanism has demonstrated that integrating intrusion detection methods into books has a consistently positive impact on students. This impact is increased student motivation, enhanced participation, and a higher level of satisfaction. This highlights the potential for an improved educational system that combines teaching methods with cyberattack to significantly benefit the educational progress of students. And also use as references related to the topic, students can greatly enhance their understanding. Additionally, this approach helps them save time by reducing the need to

conduct research from unfamiliar or unreliable sources.

1.7. Limitation of the study

- To address this issue, existing predetermined datasets available online could not be up to date, creating a up to date dataset for the machine learning algorithms posed a significant challenge due to lack of resource.
- . Instead, a new dataset had to be generated, comprising a substantial amount of data to facilitate effective learning by the algorithms. The training process involved using this large dataset to ensure improved accuracy over time.
- Training multiple models was necessary to identify the model with the highest accuracy. Each model required a considerable amount of time to train due to the dataset's size.

Terminology

Forwarding Plane: - The forwarding plane is the lowest layer of a software-defined network that contains all of the physical or virtual forwarding devices.

Control Plane: - The SDN Controller and Network Operating System are part of the middle layer of SDN.

Application plane: - The upper most stratum of SDN, encompassing Applications.

Southbound interface: - The interface that connects the data plane forwarding devices and the controller is known as the southbound interface.

Northbound Interface: - The link that connects the application plane and the controller is known as the "Northbound Interface."

API: - Application Programmable Interface, or API for short. Organize software and system interaction.

SDN forwarding switches: - The forwarding switch utilized in the data plane is an SDN forwarding switch.

Controller of SDN: - The forwarding plane is controlled by the core SDN network control mechanisms.

Flow Rules: - Activities designated for the flow

Flow tables: - The tables on the switch that manage traffic flow are called flow tables.

Open Flow: - Southbound Interface API, which facilitates the exchange of data between SDN Controller and SDN Data Plane apparatuses.

Thesis work contributions are follows

- Our research involved an analysis of various intrusion detection systems and machine learning algorithms to gain insights into their current capabilities. This led to a comprehensive classification of intrusion detection systems based on their

deployment and detection methods, as well as a similar categorization for machine learning algorithms.

- We specifically focused on identifying algorithms that excelled in anomaly detection and classification, using performance metrics such as accuracy, detection rate, true positive rate, false positive rate, precision, and recall to make our selections.
- Furthermore, we assessed the performance of these chosen algorithms by testing them on a dataset containing multiple anomalies.

1.8. Brief Thesis Outline

There are six chapters in the format of this thesis. The document's introduction is covered in the first chapter, research purpose, problem description, and SDN terminology. The state of art of SDN architecture, SDN model, SDN approach, SDN features, network penetration, and related work are covered in Chapter two. The third chapter presents research methods such as document review, data collection, and tools. Chapter four covers proposed solution, data description, data preprocessing and data cleaning, feature selection, data transformation, and evaluation metrics. Chapter five presents experimental results, performance analysis and model evaluation. The sixth chapter presents conclusions and future work.

CHAPTER TWO

LITERATURE REVIEW AND RELATED WORKS

1.9. Introduction

This section presents papers that offer background information and discuss the problem mentioned earlier. Simultaneously, we seek inspiring examples to aid in designing an intrusion detection system. By studying literature surveys and previous works, we aim to propose the framework we are seeking. Our research process involved reviewing various approaches and identifying the most relevant articles that we believe can offer a method to solve our problem.

Our approach involves examining SDN from a systems perspective, focusing on design principles that guide the development of software-defined networks rather than treating it as a single solution. Software-Defined Networking (SDN) is an emerging computer network architecture that distinguishes itself by separating the data plane and control plane in routers and switches. In this architecture, control is decoupled from hardware and implemented in software, while the data plane is implemented inside the hardware or network device (Nakazawa, 2017).

1.10. SDN architecture

The three levels of a standard SDN architecture consist of the application, control, and infrastructure layers. These layers utilize northbound and southbound application programming interfaces (APIs) for communication (Thirupathi, Sandeep, Kumar, & Kumar, 2019).

Application layer: This layer houses typical network applications or functions commonly used by enterprises. Examples include firewalls, load balancers, and intrusion detection systems. In a software-defined network, an application that utilizes a controller to manage behavior acts as a dedicated device, like a firewall or load balancer in traditional networks, influencing the data planes' behavior(Jawad, 2018).

Control layer: Positioned at the center of the software-defined network, the control layer represents the SDN control software, acting as the network's central intelligence. This controller manages network traffic flows and regulations from its server-based location(Engineering, 2004).

Infrastructure layer: The infrastructure layer includes the network's hardware switches responsible for routing network traffic to its designated destinations

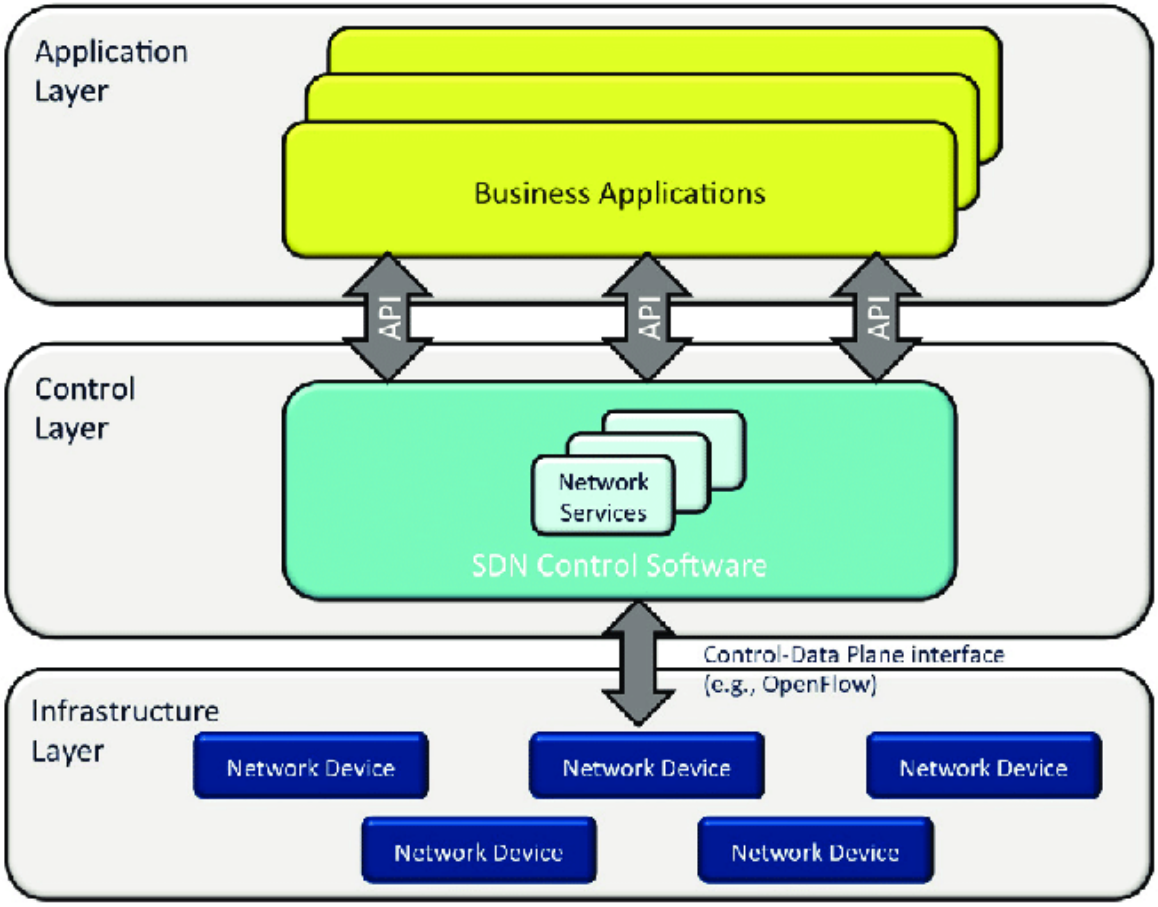

Figure 1: software defined network architecture (Bundy, 2013)

The SDN API comes in two types: the north interface and the south interface. The north interface is the bridge between the controller and the applications and the south interface is the bridge between the controller and the physical hardware devices in the network because SDN is a virtualization architecture in which the Network components do not need to be located in the same location. The controller layer is the main area where the SDN controller logic will reside to control the network infrastructure. This is an area where each vendor comes up with their own logic for building products that can work with controllers and frameworks (Fawcett, Scott-Hayward, Broadbent, Wright, & Race, 2018).

1.11. Different models of SDN

Although the principle of centralized data flow control in switches and routers applies to all software-defined networks, there are still different SDN models (Fawcett et al., 2018).

Open SDN: - Network managers can regulate the actions of both physical and virtual switches at the data plane level by using a protocol such as OpenFlow.

SDN using APIs: - The application programming interface (API) regulates how data moves across the network on each device, as opposed to employing an open protocol.

SDN Overlay Model: - Another kind of SDN builds dynamic tunnels to different on-site and remote data centers by operating a virtual network atop the hardware architecture already

in place. Virtual networks preserve the actual network by distributing bandwidth among multiple channels and assigning devices to each channel.

Hybrid SDN: - In order to enable various network functions, this paradigm integrates software-defined networking with conventional network protocols in a single environment. Network managers can gradually integrate SDN into the current environment by having certain traffic handled by standard network protocols and other traffic handled by SDN.

1.12. The SDN Approach

An introduction to SDN and an example of how it is made to adapt to changing network needs are given in this section (D'Cruze, Wang, Sbeit, & Ray, 2018). A clear and helpful set of specifications is offered by the Open Data Center Alliance (ODCA), and it includes the following:

Adaptability: - Depending on corporate policy, network conditions, and application requirements, the network must dynamically adjust and respond.

Automated: - Modifications to policies are automatically implemented to minimize human labor and mistakes

Maintainability: - New features and capabilities (patches, software upgrades) must be introduced smoothly with the least amount of disturbance to business operations.

Model management: - Rather than altering the design by rearranging specific network components, the network management software must provide model-level network administration. Mobility: Virtual servers and mobile user devices must be supported with control functions.

Integrated security: - Rather than being an add-on feature, network applications need to incorporate seamless security as a core function.

On-demand scaling: - It is imperative for implementations to possess the capability to adjust the network's capacity and services to accommodate requests made on-demand.

1.13. Characteristic of software-defined networking

SDN has four unique and defining characteristics according to (Chica et al., 2020):

Agility: - Administrators can modify network configurations in response to changes in business and application requirements.

Centralized management: - SDN gathers network intelligence to give an all-encompassing picture of the setup and operation of the network.

Programmability: - The capacity to swiftly and simply configure network resources and program network features directly through automated SDN services.

Open connectivity: - SDN is built on open standards and is put into practice using them. Consequently, SDN offers consistent networking in a vendor-neutral architecture and

simplifies network design.

Number	(SDN)Software Defined Networking	Traditional/classical Networking
1	*SDN is a virtual networking approach*	This network is the old conventional networking approach
2	it is centralized control	It is distributed control
3	SDN is programmable	Non-programmable
4	this is the open interface	This is a not open interface
5	dataplane and control are decoupled by software	In this kind of network data plane and control plane are not separable.

Table 1: Difference between SDN and Traditional Networking

1.14. Network Intrusion

Using a computer system or its resources without the required authorization is known as intrusion, and it can be either malicious or internal. The availability, integrity, and security of networked computer systems may be compromised as a result. The Intrusion Detection Assessment Plan of the Defense Advanced Research Projects Agency (DARPA) divides network intrusions into four categories: Attacks that use denial of service (DoS), probing, user to root (U2R), and remote to local (R2L) (Shone, Ngoc, Phai, & Shi, 2018).

1.14.1. Denials of Service (DoS)

An attack known as a denial of service (DoS) occurs when an attacker attempts to block authorized users from using a service. An attacker using a denial-of-service attack (DoS) frequently bombards the network or server with messages requesting authentication. Typical denial-of-service attacks include ping of death (PoD), buffer overflow, TCP SYN, smurf, neptune, and teardrop. (Liang, Zheng, Sheng, & Huang, 2016).

1.14.2. Probing

In a probing attack, a hacker examines a computer or network device for flaws or vulnerabilities that could be used to breach a system in the future. (Hanson & Hines, 2018). Some common probe attacks are: IPsweep that looks for a service on a certain port by scanning the network Portsweep that searches a large number of ports to find out whatever services are available on a particular host.

1.14.3. User to Root (U2R)

This is a type of exploit in which an attacker begins by gaining access to normal user accounts on a system which can be obtained by password sniffing, dictionary attacks, or social engineering and can exploit certain vulnerabilities to gain root access to the system (Hassan, 2017)

1.14.4. Remote to Local (R2L)

Occurs when a malicious party using network packet delivery capabilities finds a way to access a machine locally and becomes the user of that computer (Hassan, 2017).

1.15. Classification of Intrusion Detection Methods

Nowadays, Intrusion detection (ID) in a network system is one of the solution to monitoring if there is an intervention of any unauthorized activities. Many researchers have proposed various intrusion detection techniques to detect a malicious activities (Suthar & Patel, 2023). Intrusion detection system can categories deployment based intrusion method and detection based intrusion method

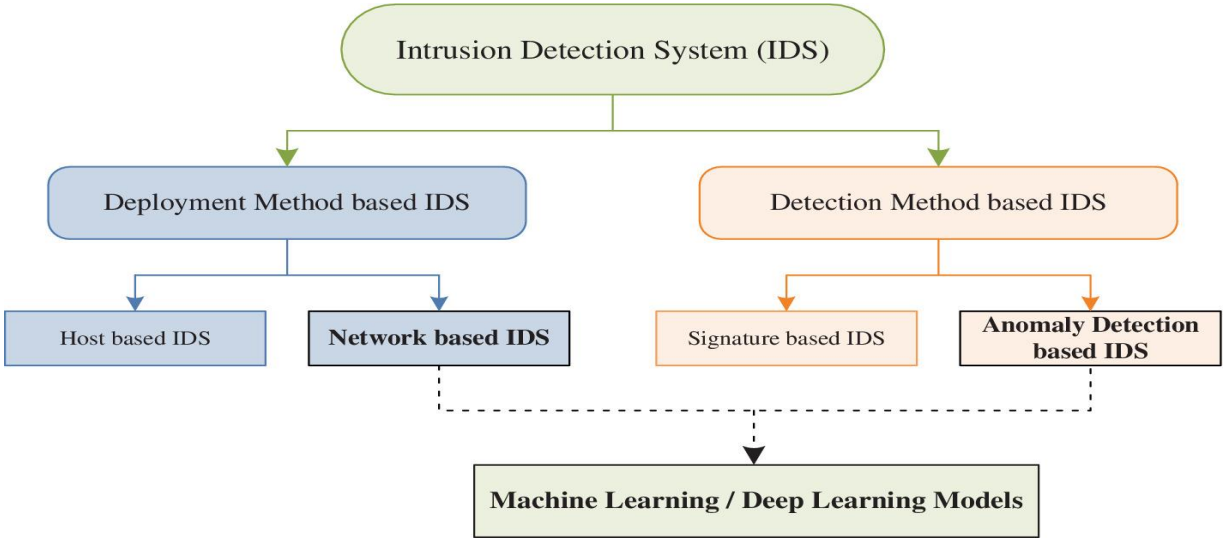

Figure 2: deployment and detection intrusion method classification taxonomy(Ahmad, 2021)

(Ahmad, 2021)mostly detection method based intrusion detection can be classified into signature and anomaly intrusion detection system: Signature-based detection this malicious attack detection methods are also recognized using signatures of well-known attacks that have been stored in the database. It has high detection accuracy for known attacks, low false positive rates and fast detection speed but it suffers from detection accuracy for variation of known attacks, unknown and zero-day attacks cannot be detected, misrepresentation of signature patterns leads to increased false negative rates. Anomaly-based detection this method establishes a baseline profile for normal traffic behavioral pattern collected over a predetermined period. It is efficient for detecting unknown and zero-day attacks with a very high detection accuracy and relatively with low detection speed, but it has more overhead and unable to detect encrypted attack patters

Supervised Learning (SL) is a common type of Machine Learning used in various studies for detecting DDoS attacks. In this approach, a predictor is trained using labeled data and then utilized to make decisions. The effectiveness of an SL algorithm can be assessed by testing it

on known samples, allowing network administrators to verify its performance before implementing it in the network. During classification tasks, an SL predictor computes the probability of each class and selects the one with the highest probability. This section reviews existing literature on SL applications, particularly in the realm of DDoS attack detection.(Eswari et al., 2023; Wang et al., 2022).

2.8 Related Work

Machine learning methods are now a crucial element in enhancing security measures against attacks. These advanced technologies can accurately differentiate between normal and malicious traffic. Numerous research projects offer effective detection and prevention techniques to combat a range of SDN threats.

(Singh et al., 2018) Proposed a DDoS defense technique using machine learning in SDN. The system comprises three modules: flow statistics collection, feature extraction, training, and network traffic categorization. Various machine learning classifiers like Support Vector Machine, Decision Tree, Logistic Regression, and Nearest Neighbors are explored to distinguish between normal and attack traffic. The classifier with the best accuracy and lowest false positive rate is selected for attack detection in this security system.

The primary objective of an intrusion detection system is to analyze extensive volumes of network traffic data. A well-structured classification system is essential to address this challenge. In the proposed system, Support Vector Machine (SVM) and Naive Bayes are utilized as machine learning algorithms to tackle categorization issues. To enhance the effectiveness of detecting both known and novel attacks, a multi-level hybrid intrusion detection model is developed in the research, incorporating support vector machines and extreme learning machines(Logeswari et al., 2023).

(Farooq et al., 2023) the authors introduced an assessment framework for evaluating the security of Software-Defined Networking (SDN) and compared it with traditional network systems. The evaluation criteria for SDN security encompass authenticity, availability, confidentiality, consistency, and integrity. While the authors acknowledge that some attacks in SDN may also occur in conventional networks, they did not delve into exploring attacks specifically targeting SDN layers and their implications on the overall SDN infrastructure.

(Alahmadi et al., 2022) the study introduced a machine learning approach that distinguished between normal and DDoS network traffic to identify the most effective classification model for detecting malicious IP addresses. The CICDDoS2019 dataset was utilized for training and testing the classification models, containing instances of DDoS attacks with 1 class attribute and 88 features, from which the top 15 features were selected. The evaluation involved four algorithms: Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), Random Forest

(RF), and Decision Tree (DT). ANN emerged as the top-performing model, achieving an accuracy rate of 94.95%. Notably, there were no false positives, although false negatives were more prevalent in the results.

In summary of related work, many of the researchers have proposed and explored various flood attack detection technique and solved some parts of the flood detection issues. But it is not enough, due to the dynamic nature of cyberattack. The detection technique also had limitations, because of lack of updated/ comprehensive dataset and feature selection, data preprocessing.ad hyper parameter tuning. Those issues lead to low accuracy detection, high false alarm rate(Suthar & Patel, 2023).

Paper title	**Used methods**	**datasets**	**short come**
(Islam et al., 2022) Detection of Distributed Denial of Service (DDoS) Attacks in IOT Based Monitoring System of Banking Sector Using Machine Learning Models	Machine learning that K-means clustering were used for feature selection and 5 fold used for cv	From kaggle bank dataset	Training phase high computational power required. On offline dataset detection was limited.
(Prasad & Chandra, 2022) VMFCVD: An Optimized Framework to Combat Volumetric DDoS Attacks using Machine Learning	Machine learning For feature selection the paper used sorting genetic algorithm	From kaggle CICIDS2017 dataset	- Imbalanced dataset class was there. - False alarm rate is high.
(Mohmand et al., 2022) A Machine Learning-Based Classification and Prediction Technique for DDoS Attacks	Machine learning, this paper used hyper parameter tuning to improve accuracy.	KDD dataset, UNWS-NB15	- Low accuracy - False alarm rate is high - Old dataset

Table 2: Related works summary

CHAPTER THREE

RESEARCH METHODOLOGY AND MATERIAL

3.1. Literature Review

The researcher conducted a comprehensive review of relevant literature, including books, articles, conference proceedings, and online sources, in order to gain an in-depth understanding of current research trends in their field. The utilization of machine learning (ML) algorithms for data analysis and application development has seen a rise in various domains such as intelligent tutoring, robotics, medical diagnosis, stress recognition, and decision support. To enhance their knowledge of key aspects ranging from data collection to data processing to defining human states, a thorough literature review was carried out.

The entire process of data analysis utilizing ML algorithms in these domains comprises three primary stages. Firstly, data collection is performed using a variety of tools. Subsequently, feature extraction is conducted on the gathered data. Lastly, the extracted features are classified to assess their effectiveness. This phase entails a review of literature to gain a comprehensive understanding of the current state of Software Defined Networking (SDN) and an exploration of machine learning solutions.

The researcher engaged in reviewing papers to understand related work, examining articles on potential threats to industrial control systems, and exploring diverse ML-based approaches for SDN design. Additionally, they identified, studied, and applied various techniques and tools relevant to their research study.

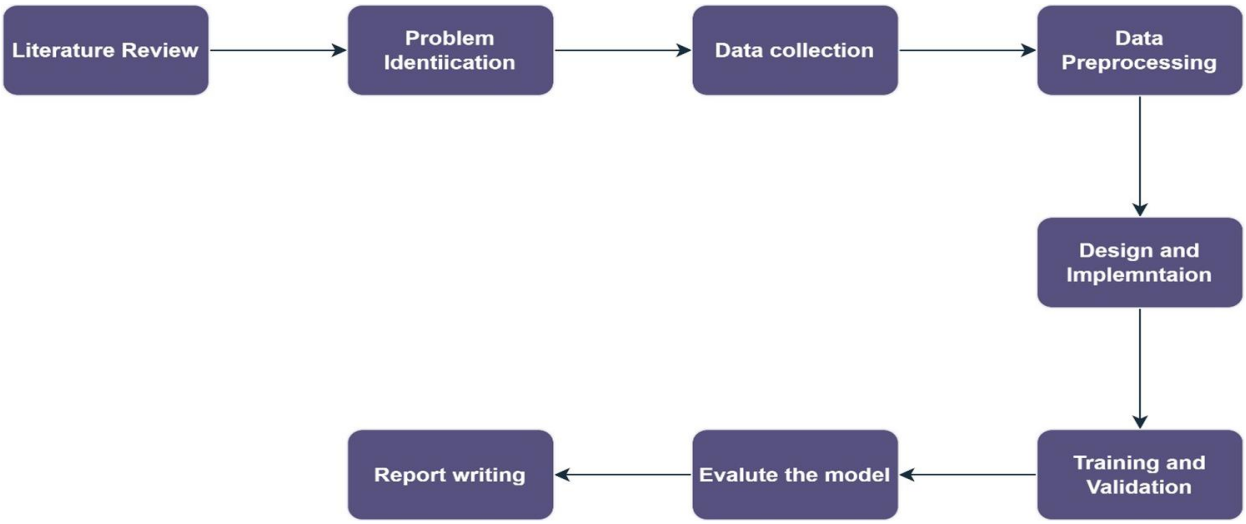

Figure 3**:** research methodology

3.2. Dataset

In this research, the new reference dataset NetSDN was utilized to assess the effectiveness

of SDN models. IoT data collection involves using sensors to monitor the status of physical objects, allowing devices and technologies connected through the Internet of Things to gather and analyze data in real time. This data can be transferred, stored, and retrieved as needed, facilitating real-time fault detection, scheduling based on time, and predictive maintenance. These capabilities contribute to reducing energy expenses, enhancing productivity, and prolonging the life span of equipment.

The collection of data plays a crucial role in supporting the objectives of SDN to achieve desired outcomes and study goals effectively. By tapping into diverse SDN data sources through various collection methods, researchers can address research challenges efficiently. IoT data encompasses information captured by sensors embedded in connected devices, wearables, and similar technologies, albeit not all sensor data types are equally complex. Evaluating the performance of a dataset involves employing different metrics such as full network configuration, complete traffic analysis, labeled diversity assessment, consideration of heterogeneity, assessment of feature sets, and availability of metadata. (Zhou, Yan, Liu, & Atiquzzaman, 2019).

3.3. Performance Evaluation Method

After doing the data selection, scaling the features and applying the feature selection techniques,choosing the ML algorithm and of course implementing the model and taking the output as a layer, the next step is to find understand how well the model is performing against certain metrics using test data sets. The metrics we choose to evaluate the ML model are important because they influence how the model's performance is measured and compared. The goal of anintrusion detection model is to minimize false alarm rates or maximize detection rates. High detection rate and low false positive rate are the main criteria of any ML-based intrusion detection model. Generating performance metrics is an important aspect of model performanceanalysis. Various metrics were generated in this survey (Amin et al., 2018). These are explainedin detail as follows:

3.3.1. Accuracy

One technique to evaluate the effectiveness of a classification model is to count the total numberof samples classified as true and false. To represent these values, a confusion matrix is often used. The confusion matrix is a tabular visual representation of the results of supervised learningalgorithms. The number of instances in an actual class is displayed in rows while the number is the frequency of correctly classified positive samples among all real positive samples. The instances in the prediction layer is shown in columns. Practically and simply, the measure ofaccuracy is calculated as follows to summarize the matrix of this study:

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

3.3.2. Precision

Often insufficient when assessing the effectiveness of a learning model. Although accuracy indicates whether the model was successfully trained, it does not provide application-specific information. As a result, other performance criteria are used, such as accuracy. Accuracy is thepercentage accuracy of a detected positive or a true positive. False-positive results can be problematic in a variety of situations. The high false positive rate in this study implies that trafficis incorrectly classified as hostile when it is not. This can lead to wasted time and money outsideof academia (Dawoud et al., 2018). Here's how the precision is calculated:

$$\textbf{Precision} = \frac{TP}{TP + FP} \quad (2)$$

3.3.3. Recall

Another performance measure is recall. Recall refers to the number of positives actually foundor recalled. It is also an important number because hidden positives or false negatives can havean important aspect in some cases. For example, if a model does not remember all DoS attacks,the descriptive network traffic may go unnoticed, increasing the risk of harm to systems and users (Azam et al., 2020).

$$\textbf{Recall} = \frac{TP}{TP + FN} \quad (3)$$

3.3.4. F-measure (F1-score)

F-metric is a measure that combines accuracy and recall to produce an overall accuracy score for a model. A high F measurement indicates that a model has minimal false positives and negatives, indicating that it correctly identifies hazards while generating few false alarms (Carpaet al., 2017).

$$\textbf{F-Measure} = \frac{2(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (4)$$

3.3.5. Detection Rate

Detection rate is the ratio of correctly identified attacks to all attack instances. It is also known as the true positive rate (TPR), sensitivity, or recall. The detection rate specifically focuses on accurately identifying abnormal traffic patterns. It evaluates a classifier's capability to recognize positive samples. The worst-case scenario has a value of 1.0, while the best-case scenario is 0.0. The formula for DR is: DR = Sensitivity = 1 - Type 2 error**.**

$$\text{Detection Rate } (\textbf{DR}) = \frac{TP}{TP + FP} \quad (5)$$

False Alarm Rate (FAR)

False-positive rate is another name for false-alarm rate (FAR). It displays the percentageof wrongly generated alerts for normal records to all normal records (Faraj, Megías, Ahmad, & Garcia-Alfaro).

$$\text{False Alarm Rate (FAR)} = \frac{FP}{FP + TN} \quad \text{---------- (6)}$$

True Negative Rate (TNR)

Its formula is: Known as specificity it is defined as the ratio of the number of correctly classified Normal samples to all the samples that are Normal. Specificity = 1 – Type 1 error or defined as follow:

$$\text{Negative Rate (}\mathbf{TNR}\text{)} = \frac{TN}{TN + FP} \quad \text{---------- (7)}$$

Computation Time: Compute time is the ultimate performance measure used in this work. This is not directly related to the classification but to the time required to train a model. This measure shows the efficiency of the model. Time is calculated using a Windows machine with 4GB RAM and cor I5 processor.

3.4. Software Tools

The primary rationale for selecting Python stems from two key advantages. Firstly, Python boasts a straightforward syntax that is easily comprehensible and learnable. Secondly, Python provides extensive support for scientific computing and data science, making it an ideal choice for an intrusion detection system for the Internet of Things (IoT). Python stands out as one of the most popular programming languages, offering simplicity, robustness, and benefiting from community-driven contributions to open-source projects.

Python's versatility and wide range of applications contribute to its popularity. From developing software and web services to conducting data analytics, visualization, and training machine learning models, Python serves varied purposes effectively and at no cost. Leveraging development tools facilitates the creation of efficient and dependable Python solutions, with offerings such as Integrated Development Environments, Python package management tools, and productive extensions streamlining the processes of testing, debugging, and deploying software solutions for production environments..

Jupyter Notebook is one of the most a web-based IDE for experimenting with code and displaying the results. It is fairly popular among data scientists and machine learning practitioners. It allows them to run and test small sets of code and view results instead of running the whole file.

.

CHAPTER FOUR

PROPOSED SOLUTION

4.1 Introduction

From a range of supervised machine learning algorithms, we have selected three common techniques representing different SL categories for our test. These include Figure 11 illustrates the Architecture, highlighting the dataset preprocessing in the initial section. The preprocessing is segmented into standardization and data cleaning. Following this, a feature selection process is implemented, leading to the subsequent classification phase. Subsequently, the training and testing phases are conducted to assess the model's performance and achieve the desired thesis output.

The rapid growth of IoT devices has made the Internet of Things a prominent technology domain. These intelligent and internet-connected devices require a more dynamic and secure network architecture to meet evolving demands. Traditional networks may not sufficient in this changing landscape. SDN, a groundbreaking technology, provides complete control over network behavior and helps prevent network congestion. The SDN controller offers essential debugging tools for enhancing security within the IoT community (Shone et al., 2018). Figure 2 illustrates how machine learning model can help with IoT security. We can detect the malicious intrusion attack into independent subsets using the machine learning algorithm model.

4.2. Data Descriptions

We have used SDN Intrusion detection dataset from kaggle, it was created in 2020 for the aim of to produce a comprehensive SDN dataset to verify the performance of intrusion detection systems and the dataset is very vast and contains 61616 samples and 86 records(Elsayed & Jurcut, 2020). The creation of the SDN Intrusion detection dataset addresses existing limitations by establishing a comprehensive and distinct reference dataset for intrusion detection. This dataset aims to construct user profiles that encapsulate abstract representations of network events and behaviors. The dataset encompasses various attack types, including DoS, Mirai, Scan, and MITM such us ARP. But, due to resources constraint we took 17,427 records. Which is currently stored in comma-separated values (CSV) files. Each instance provides details on the IP stream generated by the network device, incorporating statistical stream features like duration, packet count, byte count, packet length, and more.

No.	Group of attributes	Attributes	Description
1	Network identifiers(3 attributes)	Source Port, Destination Port, Protocol	These properties contain all information regarding the source and destination of the Internet stream, i.e., protocol and ports
2	Flow descriptors (34 attributes)	Total Fwd Packets, Total Bwd Packets, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean, Fwd Packet Length Std, Bwd Packet Length Max, Bwd Packet Length Min, Bwd Packet Length Mean, Bwd Packet Length Std, Flow Bytes, Flow Packets, Min Packet Length, Max Packet Length, Packet Length Mean, Packet Length Std, Packet Length Variance, Down Up Ratio, Avg Fwd Segment Size, Avg Bwd Segment Size, Fwd Avg Bytes Bulk; Fwd Avg Packets Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes Bulk, Bwd Avg Packets Bulk, Bwd Avg Bulk Rate, Init Win bytes forward, Init Win bytes backward, act data pkt fwd, min seg size forward, Label	These attributes contain all the information regarding the Internet flow, i.e., packet count, volume and standard deviation among others in inbound and outbound directions.

3	Inter-arrival times (9 attributes)	Flow Duration, Flow IAT Mean, Flow IAT std, Flow IAT Max, Flow IAT Min, Fwd IAT Total, Fwd IATMean, Fwd IAT Std, Fwd IAT Max	These attributes hold all the information related to the interarrival times inthe forward and backward direction
4	Sub-flow descriptors (4 attributes)	Subflow Fwd Packets, Subflow Fwd Bytes, Subflow Bwd Packets, Subflow Bwd Bytes	If there are sub flows, these properties present all information regarding the number of packets per flow and their volume in the inbound and outbound directions
5	Header descriptors (2 attributes)	Fwd Header Length, Bwd Header Length	Among these attributes, the header information isstored
6	Flow timers (4 attributes)	Idle Mean, Idle std, Idle max, Idle min	These properties store information about eachthread's active and inactive duration

Table 3: Dataset features and explanation

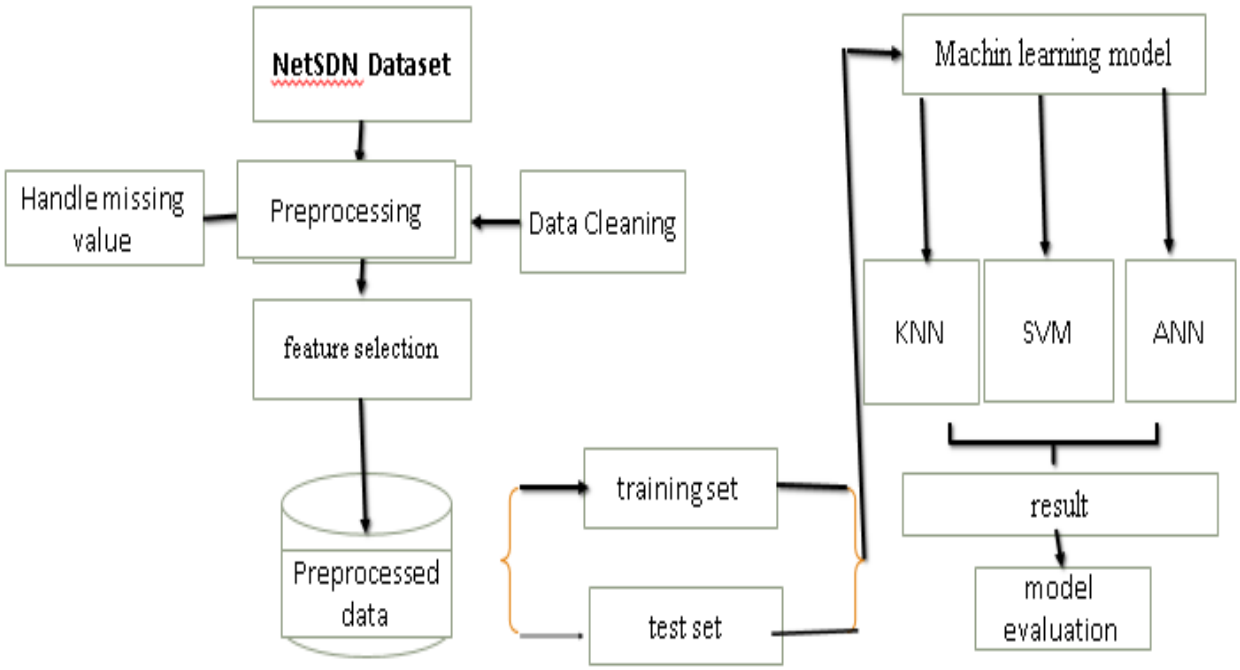

Figure 4: proposed System Architecture

4.3. Model Train Techniques

4.3.1. Data preprocessing

Data preprocessing involves preparing raw data to align it with a machine learning model. Clean, formatted data is not always readily available when constructing a machine learning project. Therefore, it is essential to clean and format the data through preprocessing. Real-world data typically includes noise, missing values, and may be in an unsuitable format for direct use in machine learning models. Data preprocessing is crucial for cleaning and formatting the data to better fit the machine learning model, ultimately enhancing accuracy and efficiency. Data preprocessing includes the following steps:

- Get necessary dataset for modeling
- Import necessary library for model training
- Import he selected dataset that is used to train the model
- Replace or remove null, missing and redundant data in the dataset
- Encryption of categorical data
- Splitting dataset into training and testing set
- Feature scaling

In order to develop a machine learning model, the initial requirement is a dataset as the model operates based on data. The dataset, which is collected in a suitable format for a specific problem, is referred to as a dataset. Typically, this dataset is stored in a CSV file format, which stands for "Comma Separated Values" and allows for the storage of tabular data like spreadsheets. CSV files are beneficial for managing large datasets and can be easily integrated into programs.

During the data preprocessing phase, the raw dataset undergoes processing to align it with the machine learning approach being utilized. This preprocessing involves tasks such as standardization, normalization, and data cleaning to transform the data into a more efficient format for use in data mining, machine learning, and other data science activities. By enhancing the quality of the data, preprocessing helps in identifying valuable insights and improving the performance of classifiers. Employing effective strategies early on is crucial to ensure reliable outcomes.

Data streams often generate a continuous flow of substantial amounts of data, making it impractical to utilize all samples for model training. Therefore, efficient data sampling techniques are essential to select highly representative data samples. In many instances, machine learning algorithms face challenges processing datasets due to missing values, irrelevant features, and issues with categorical columns impacting NetSDN. Consequently, various data preparation and cleaning procedures are implemented to address these issues.

(Zhou, Yan, Fu, & Yao, 2018).

Data cleaning:

Some rows or columns contain missing numeric values (NaN) or infinite values, which must be addressed before further procedures can be carried out. There are various methods to handle NaN values, including removing rows or columns with NaN values or replacing them with alternative values such as mean, median, mode, or other statistical measures from related columns, rows, or groups. The selection of the appropriate approach should be made carefully based on the dataset's characteristics. Therefor we have used mean statistical values replacement technique to fill null values.

The following important libraries are the libraries that used in the experiment

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.neural_network import MLPClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report
        import matplotlib.pyplot as plt
        from sklearn import tree
        from sklearn.svm import SVC
        from sklearn import svm
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import precision_score
        from sklearn.neighbors import KNeighborsClassifier
```

Figure 5: project libraries

To upload the experiment csv files to the python software

```
df=pd.read_csv("C:\Users\lulu\Desktop\Lamesgen_SDN\NetSDN.csv")
```

Figure 6: csv file of dataset

```
from sklearn.impute import SimpleImputer

imputer=SimpleImputer(missing_values=np.nan, strategy='mean')

imputer=imputer.fit(x[:,0:82])
```

Figure 7: null values replacement

Data Normalizes and Scales

Dependent and independent variables often vary on different scales, with one changing linearly while the other changes exponentially. For instance, salary may be in the range of multiple figures, while age is typically expressed in double digits. Normalizing and scaling

techniques are essential for adjusting data to enable computers to establish a meaningful relationship between these variables.

```
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

X_normalized_df = pd.DataFrame(X_normalized, columns=X.columns)
```

Handles Data Outliers

In the field of data analysis, it is often necessary for data practitioners to combine multiple data sources in order to create a new machine learning model. One essential technique for achieving this is principal component analysis (PCA), which plays a crucial role in reducing the number of dimensions in the training data set. By doing so, PCA enables the creation of a more streamlined and efficient representation of the data, facilitating better analysis and modeling outcomes.

Remove correlated feature

To improve the performance of machine learning models, correlated features have been eliminated from NetSDN as they do not contribute to detection tasks and may introduce performance issues and unnecessary complexity. Preprocessing steps commonly used to assess ML performance involve dividing the data into training and test sets. In cases of unbalanced datasets, a random split of data sets can result in uneven distributions that hinder accurate performance evaluation. To address this issue, data sets are divided into training and test sets using the stratification method, which ensures homogeneous distribution within the sets.

One of the key preprocessing steps in evaluating the performance of machine learning models dividing the data into training and testing sets. When dealing with an unbalanced dataset, randomly splitting the data can lead to an uneven distribution, hindering accurate performance evaluation. To address this issue, the dataset is divided into training and testing sets using a stratified method, which ensures a more balanced and representative sampling technique.

4.3.2. Features Selection

In order to efficiently and effectively develop IoT intrusion detection models that perform well, feature selection is a crucial step. Feature extraction involves selecting the most relevant features for the model, impacting its overall performance in machine learning. Feature selection is a key aspect of dimensionality reduction within the ML domain. Various techniques, including Filter-based, Wrapper, and embedded methods, are utilized for selecting optimal features. Filter-based methods, which employ statistical approaches, are

commonly used for feature selection in model optimization (Jun dong Li et al., 2017).

Wrapper methods, in contrast, aim to identify the optimal blend of attributes that enhance model accuracy. They generate a subset from the complete feature set, train the model using this subset, and evaluate its performance. This iterative process continues until the most effective subset is identified. The chosen subset is the one that maximizes the performance of a specific estimator. The selection of this subset is influenced by the algorithm and varies across different estimators.

Feature selection is imperative for the rapid and effective development of IoT intrusion detection models. It involves choosing the most relevant features for the model during feature extraction, a critical step that significantly impacts model performance. In the realm of machine learning, feature selection plays a crucial role in reducing dimensionality. Various techniques, such as filter-based, wrapper, and embedded methods, are employed for feature selection. Filter-based methods, which utilize statistical approaches, are commonly utilized for optimizing models. (Yin et al., 2018).

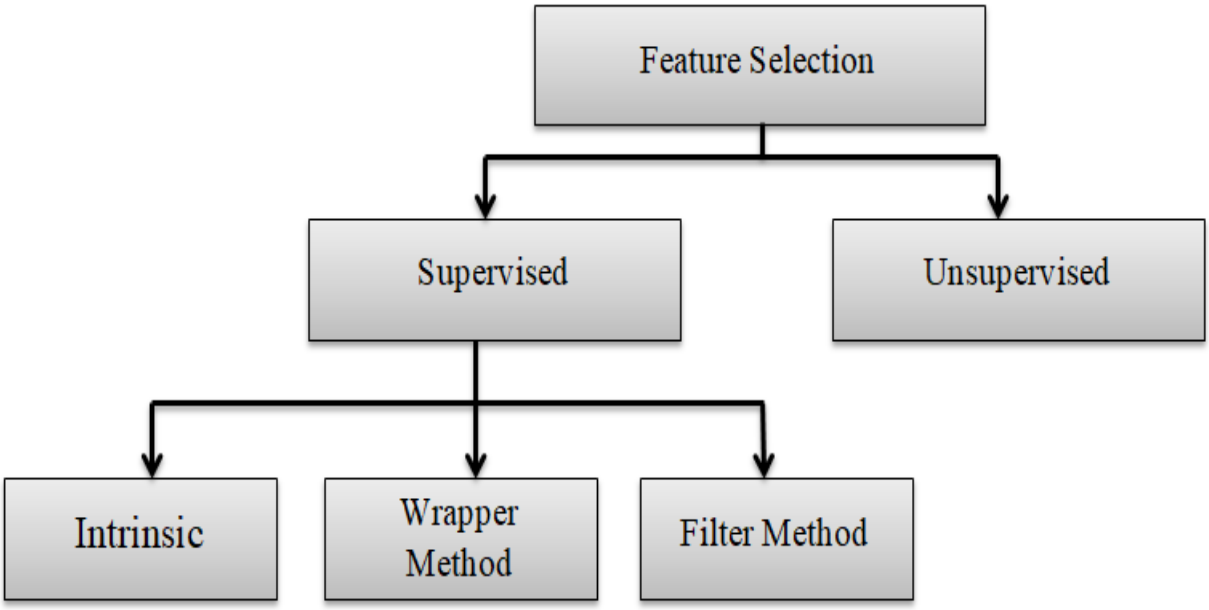

Figure 8: Feature Selection Models

Filter method: Features are eliminated in this method according to how they correlate or relate to the result. The features were discarded in accordance with the outcomes of the correlation analysis, which determined whether the features and the output label were positively or negatively associated.

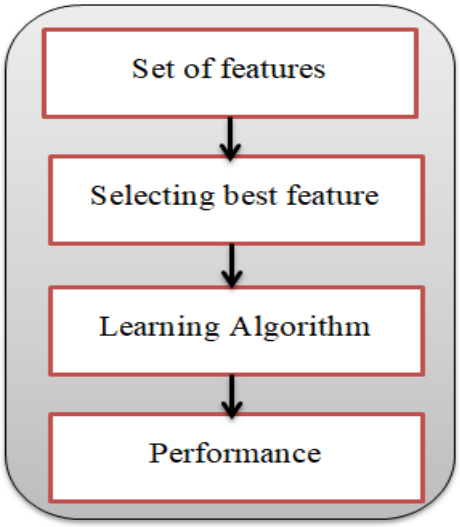

Figure 9: Filter Method flowchart

Wrapper Method: Data were separated into subsets, and those subsets were used to train a model. It is feasible to add, remove, and retrain features in the model based on its output. It assesses the correctness of every conceivable feature combination and uses a greedy strategy to build subgroups.

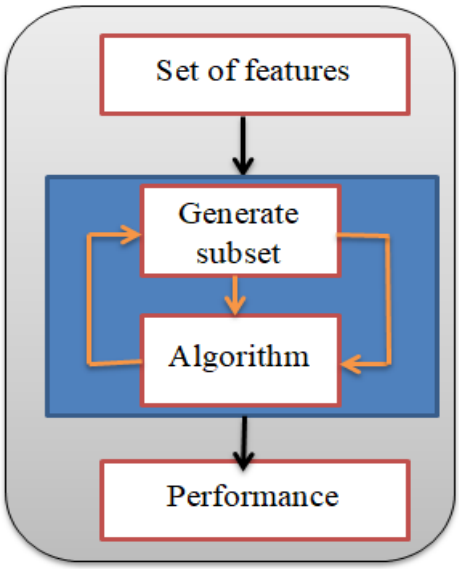

Figure 10: Wrapper Method Flowchart

Intrinsic Method: this feature selection method uses by merging the characteristics of Filter and Wrapper methods, this technique aims to generate an optimal subset. This strategy facilitates iterative machine learning processes while minimizing computational expenses.

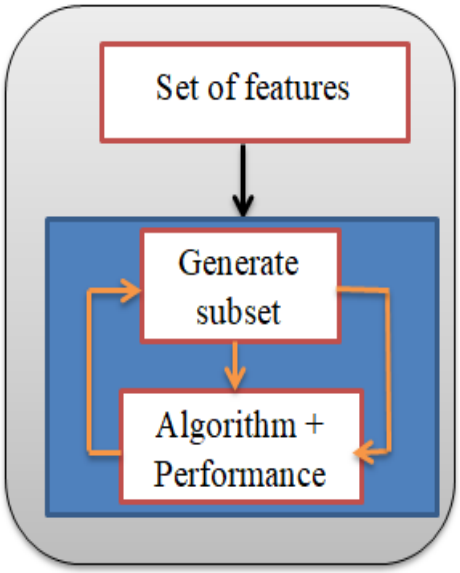

Figure 11: Intrinsic Model Flowchart

Feature selection result of wrapper recursive feature selection method:

We have explored different feature selection techniques, to select an optimal subset of features that improve our machine learning model performance. Hereby our investigation wrapper feature selection method is outperform than others.

```
select = RFE(RandomForestClassifier(n_estimators=100,random_state=42),n_features_to_select=10)
select.fit(X_train,y_train)
X_train_selected = select.transform(X_train)

print("X_train.shape: {}".format(X_train.shape))
print("X_train_selected.shape: {}".format(X_train_selected.shape))
```

```
X_test_selected = select.transform(X_test)

model.fit(X_train,y_train)
print("Scoree with all features: {:,.3f}".format(model.score(X_test,y_test)))

model.fit(X_train_selected,y_train)
print("Scoree with selected features: {:,.3f}".format(model.score(X_test_selected,y_test)))
```

```
Scoree with all features: 0.793
Scoree with selected features: 0.900
```

4.4. Classification Algorithms

There are various methods to classify malicious traffic according to the nature of intrusion attack behavior. Machine learning algorithm classified into three which are supervised, unsupervised and reinforcement learning. We considered to use supervised machine learning algorithm that involves the utilization of machine learning algorithms to assign labels to instances, with the class label representing the assigned value. In the training phase, a

classifier learns from the dataset values, while in the testing phase, these values are categorized into normal or abnormal labels. Various machine learning techniques are utilized for classification using training data, generating new categories of observations. An effective classification program segregates new observations into classes or groups based on insights gained from an initial dataset. IoT devices can be classified in diverse ways. Anomaly detection aims to identify patterns that deviate from regular nodes (Lakhan et al., 2022).

4.5. Supervised machine learning Algorithm

Supervised Learning is a technique that uses existing data to classify new observations. It involves categorizing fresh data into predefined categories based on a dataset or set of observations. In our study, we utilized KNN, SVM, and ANN algorithm to classify intrusion malicious traffic.

4.5.1. K-nearest neighbor (KNN)

K-Nearest Neighbors (KNN) is a straightforward machine learning algorithm that falls under supervised learning techniques. It operates on the premise of comparing new data with existing data to determine similarity and assign the new data to the most similar category. KNN retains all available data and assesses a new data point based on its resemblance to the stored data. This enables quick classification of new data into appropriate categories using the K-NN algorithm. While KNN can be applied to both regression and classification tasks, it is predominantly used for classification purposes. Notably, KNN is a non-parametric algorithm, which means it does not make assumptions about the underlying data distribution. It is also referred to as a lazy learning algorithm because it delays learning from the training set, opting to store the dataset and act upon it during classification. During the training phase, KNN only saves the dataset, and when presented with new data, it categorizes it based on its similarity to the existing data.

Steps of K-nearest neighbor algorithm to use

1: decide the number of K neighbors

2: Compute the Euclidean distance of the number of K neighbors

3: Get the K nearest neighbors based on calculated Euclidean distance.

4: Among k neighbors, count the number of data points in each category.

5: Assign new data points for the category with the maximum number of neighbors.

6: Then our model to be ready.

7: final end

4.5.2. Support Vector Machine

The learning process in a support vector machine involves two main stages. First, the inputs are mapped onto an n-dimensional space, where n represents the number of attributes or individual attribute coordinates. In the second stage, a hyperplane is used to separate the

instances. A hyperplane is a line that divides a group of data points into classes in a linear fashion. The support vector machine identifies the optimal hyperplane for effectively splitting the dataset.

Steps of Support Vector Machine algorithm to use

1: first step is Start

2: Then, a training collection of documents sample:

3: Create categorization objects using SVM

4: n-dimensional space of objects

5: Utilize a kernel trick to convert f to a linear separable unction

6: Binary classification is calculated using hyperplane

4.5.3.Artificial Neural Network

The most widely used machine learning algorithm available today is the artificial neural network. These neural networks were invented in the 1970s, but since then, computing power has increased dramatically, making them quite popular and now mainstream. Neural networks give every app you use an intelligent user interface that keeps you interested. An algorithm for machine learning based on the human neural model is called a neural network. Millions of neurons make up the human brain. It is capable of transmitting and processing chemical and electrical signals. A unique structure known as a synapse connects these neurons. Neurons can transfer messages through synapses. A neural network is constructed from a large number of simulated neurons. One method of processing information is an artificial neural network. It processes information in the same manner as the human brain. An ANN processes information by combining a large number of interconnected processing units. Additionally, they achieved outstanding outcomes (Jiang, Cao, Wu, & Yang).

ANNs Artificial Neural Networks (ANNs) are algorithms that mimic brain function and are utilized to model intricate patterns and forecast issues. ANNs, a deep learning technique inspired by the neural network concept of the human brain, aim to replicate brain functionality. While ANNs exhibit behavior similar to biological neural networks, they are not identical. These algorithms exclusively process numeric and structured data. An artificial neural network comprises artificial neurons known as units, organized in layers that constitute the network. The number of units in a layer can range from a few tens to millions, depending on the network's complexity. Typically, an artificial neural network consists of an input layer, hidden layers, and an output layer. The input layer receives external data for analysis, which is then processed through the hidden layers to generate valuable output for the final layer. The output layer presents the network's response based on the input data. A fully connected ANN is referred to as a Multi-layer Perceptron (MLP).

Steps of Artificial Neural Network Algorithm to use:

1. Give random weights to all links for starting the algorithm
2. Find the activation rate of hidden buttons by using inputs and links,
3. Find activation rates of exit nodes by using activation rates of hidden nodes and links to exits,
4. Find the error at the exit button and recalibrate all the links between the hidden button and the exit button.
5. Find at the output node, cascade the error to the hidden nodes by using the weights and errors
6. Assign again a weighting among hidden node and input node
7. Continue the procedure until the convergence condition is satisfied.
8. Note the activation of the output nodes by taking the final link weights.

4.6. Dataset Selection

The NetSDN dataset is a comprehensive collection of recent and prevalent attacks that closely resemble real-world data. It includes network traffic analysis results obtained using Flow Meter, with flows labeled based on various attributes such as timestamps, source and destination IP addresses, source and destination ports, and protocol and data types of the attack streams.

The dataset consists of 56 qualitative and quantitative characteristics, which sets it apart from previous versions of benchmark datasets. Each log in the dataset is labeled as either normal or an attack. The dataset covers five main classes for intrusion analysis: Mirai, DoS, Scan, MITM ARP, and Normal.

Mirai is a type of malware that targets smart devices running on ARC processors, transforming them into a network of remotely controlled bots.

DoS attacks, on the other hand, aim to render a computer or network inaccessible to legitimate users by flooding it with traffic or sending disruptive information. While DoS attacks do not typically result in the theft of valuable information, they can cause significant disruption and financial losses for victims. Common flood attacks fall under this category.

Overall, the NetSDN dataset provides a valuable resource for studying and analyzing different types of intrusions, enabling researchers and practitioners to develop effective intrusion detection systems and strategies.

Common flood attacks include:

- **Buffer overflow attacks** are among the most common types of Denial of Service (DoS) attacks. The idea behind this attack is to overwhelm a network address with more traffic than the system was designed to handle. This category includes various attacks, some of which exploit specific bugs in certain applications or networks**.**

One example is the ICMP Flood attack, which takes advantage of misconfigured network

devices by sending spoofed packets to ping every computer on the targeted network, rather than targeting a specific computer. This action activates the network to amplify the traffic. This attack is also referred to as a Smurf attack or death ping**.**

Another type of attack is the SYN flood, where a connection request is sent to a server but the handshake is never completed. This results in all open ports being filled with requests, preventing legitimate users from connecting.

Additionally, scanning is a technique used in ethical hacking to discover systems connected to an organization's network. It provides information about accessible systems, services, and resources on a target system. Some refer to this as an active scan, as it can potentially disrupt service on sensitive servers. Scanning is commonly employed during vulnerability assessments to identify weaknesses in existing defenses.

There are two ways to scan:

Active scanning

passive scanning

A Man-In-The-Middle (MITM) Attack is an active form of hacking where the attacker intercepts communication between two parties and can manipulate the messages exchanged or steal data packets from the victim. The victims are unaware that their communication is being controlled by the malicious attacker. Common types of MITM attacks include**:**

1. ARP Spoofing: This attack involves manipulating the Address Resolution Protocol (ARP) to redirect network traffic to the attacker's machine. By sending fake ARP packets, the attacker can intercept data and potentially overload the network switch.

2. DNS Spoofing: Similar to ARP Spoofing, DNS Spoofing involves manipulating Domain Name System (DNS) requests to redirect users to fake websites or services controlled by the attacker.

Having a comprehensive dataset like the NetSDN dataset is crucial for developing effective Intrusion Detection Systems (IDS). Key factors for a high-performance IDS dataset include complete network configuration, labeled traffic data, diverse attack scenarios, and vailability of meta data for analysis. This dataset helps in addressing the challenges of creating robust IDS systems to combat evolving cyber threats.

Number	Dataset Label	Number of Records
1	Normal	3,988
2	Mirai	40,894
3	DoS	5,875
4	Scan	7,401
5	MITM ARP Spoofing	3,458
Total Record		**61,616**

Table 4: Distribution of NetSDN dataset

(Wilson, Martinez., 2000) the number of datasets selected for the study accounted for about **28.28%** of the total NetSDNdataset records.

Number	Selected Dataset classes	# selected each records
1	Normal	1,112
2	Mirai	11,606
3	DoS	1,622
4	Scan	2,120
5	MITM ARP Spoofing	967
Total Record		**17,427**

Table 5: Distribution of selected dataset from NetSDN.

CHAPTER FIVE

IMPLEMENTATIONS OF PROPOSED SOLUTION

5.1 Introduction

This chapter explains the experiments conducted using the approach outlined and how the models were trained. In order to preserve the balance of dataset records across classes, the dataset was used. A separate input dataset is used for testing and training. The required feature selection and preprocessing methods are then used. The implementation is the path that leads into the coding procedure. Set up the necessary conditions for gaining access to the Python interface and the workspace needed for programming. This step involves the detection and classification of network traffic using a variety of machine learning (ML) approaches. The following concepts are being investigated for implementation.

- Gain insight into the functionality of interfaces and the nuances of creating interfaces in Python
- Understand the practicality of interfaces in a dynamic language like Python
- Develop an informal Python interface
- Sufficient data is available for training and testing our recommendation framework.

5.2 Experimentation Tools

An experiment is conducted to create an artificial scenario where the researcher can manipulate one or more variables while keeping all other factors constant. The objective is to measure the resulting effects. In this study, our focus is on evaluating and analyzing the performance of our proposed algorithms using Python machine learning software. Python is used to apply these algorithms to the selected dataset. The model is generated entirely using Python, utilizing frameworks such as scikit-learn and pandas.

5.2.1 Hardware Requirement

The entire project is carried out on a single computer with the following specifications: Intel(R) Core i5, processor running at 2.40GHz, 4GB main memory, L1-L3 cache memory, and Windows 10 x64 bit operating system.

5.2.2 Software Requirement

The IDS model was developed using Jupyter Notebook and other software packages mentioned below, which are essential for machine learning tools utilized in this study.

Jupyter Notebook is a cutting-edge web-based interactive development environment tailored for notebooks, code, and data. Its adaptable interface empowers users to configure and organize workflows in various fields such as data science, scientific computing, computational journalism, and machine learning. The modular design of Jupyter Notebook

facilitates extensions to enhance and broaden its functionality.

Python, an interpreted, object-oriented, high-level programming language with dynamic semantics, boasts a sophisticated integrated data structure coupled with dynamic typing and binding. This feature set makes Python highly appealing for swift application development and serves as a scripting or binding language to connect existing components seamlessly. Python's user-friendly syntax emphasizes readability, thereby reducing program maintenance costs. Moreover, Python supports modules and packages, promoting program modularization and code reuse.

Scikit-learn stands as an open-source data analysis library recognized as the benchmark for machine learning (ML) within the Python ecosystem. It encompasses algorithmic decision-making methods like classification for data categorization based on patterns, regression for predicting data values using existing and projected data averages, and clustering for automatically grouping similar data into datasets. Scikit-learn offers a range of algorithms supporting predictive analysis, from simple linear regression to neural network pattern recognition, alongside interoperability with NumPy, pandas, and matplotlib libraries.

NumPy ranks among the most widely used numerical computing libraries globally, finding applications across diverse fields such as image processing, finance, and bioinformatics. This workshop heavily relies on NumPy and its derivatives for various computations**.**

SciPy emerges as a Python library for scientific computing that builds upon NumPy's foundation. Analogous to the core of Matlab, NumPy, SciPy resembles the Matlab toolboxes in functionality**.**

Pandas, a Python package, delivers rapid, flexible, and expressive data structures tailored for seamless manipulation of "relational" or "labeled" data. It serves as a fundamental high-level building block for practical real-world data analysis in Python.

Hardware	
Item Name	**Description**
Computer	Hp
CPU	2.3GHz Intel Core i5 series 5 processor
RAM	4 GB
Storage	Disk 500 GB
Software	
OS Windows	Windows 10 64-bit
Execution platform	Jupyter Notebook
programming language	Python

Table 6: Experimental Tools

5.3 Implementation

High-ranking features are pre-processed from the NetSDN before they are exposed to three feature selection algorithms. After that, the three-classifier method was used. The first feature selection method was sequential backward processing. In terms of classification results, the decision tree and random forest feature selection techniques performed better than sequential backward processing, sequential forward processing, and recursive feature elimination. As a result, every machine learning method that was used demonstrated exceptional efficacy and efficiency. Recursive feature elimination selection combined with a decision tree classifier yields high accuracy scores. (Khraisat et al., 2019).

Training with Support Vector Machines on the NetSDN takes a lot longer than training with Random Forest and Decision Tree techniques. Support Vector Machine only requires a smaller

When the "MIN" and "AVERAGE" functions were employed, most classifiers worked well and produced impressive results compared to their results when the "MAX" operator was used.

It is interesting to notice that when the "MAX" operator was used as the aggregate function, ANN profited more than the other classifiers (RF and SVM excluded), but all classifiers performed well in the majority of tests when the "AVERAGE" operator was used. The performance of ANN somewhat decreased.

The duration of the performances was taken into consideration when evaluating them. Prediction time is another important factor to take into account when utilizing a machine learning classifier for real-time applications. Thus, in the case that machine learning classifiers are used by sensitive, real-time intrusion detection systems to stop attacks on Internet of Things devices.

5.3.1 Testing Data

After training the model with the training dataset, the next step involves testing the model using a separate test dataset. This test dataset assesses the model's performance and its ability to generalize effectively to new or unseen data. The test dataset is distinct from the training dataset and serves as an independent subset for evaluating the model. It is meticulously organized to encompass various scenarios relevant to the problem at hand, mimicking real-world conditions that the model may encounter. In most cases, the test dataset originates from the original data used in the machine learning project.

Input Features	Target Class	Dataset Split
X-training	y-training	80%
X-test	y-test	20%

Table 7: Input dataset split

At this stage, it is crucial to evaluate and compare the accuracy of the model on the test dataset against its performance on the training dataset. Discrepancies in accuracy between the two can indicate potential overfitting issues. To ensure a comprehensive evaluation, the test dataset should encompass a representative subset of the original data and be sufficiently large to enable reliable predictions.

Figure 12:- Dataset Split

To split the dataset, you can use the train_test_split function from the scikit-learn library. The following code can be used to split the dataset:

```
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=42)
```

In this code, we first import the train_test_split function from the sklearn.model_selection module. Then, we use the function to split the data into training and testing sets. The X and Y variables represent the input features and target variable, respectively.

The train_test_split function returns four variables: X_train, X_test, y_train, and y_test.

- **X_train** contains the features of the training data.
- **X_test** contains the features of the test data.
- **y_train** contains the target variable values for the training data.
- **y_test** contains the target variable values for the test data.

In the train_test_split function, we passed four parameters. The first two parameters are the input features (X) and target variable (y). The test_size parameter is used to specify the size of the test set. In this case, it is set to 0.2, which means that 20% of the data will be used for testing and 80% for training.

The random_state parameter is used to set the seed of the random number generator. This ensures that we get the same split every time we run the code.

After the model is trained with the training data, it is then tested with the test data. The training process involves three steps:

6 **Feed:**

The model is trained by feeding it the training input data.

7 **Define:**

The training data is labeled with the corresponding outputs, and the model converts the training data into text vectors or some other data representation.

8 **Test:**

The model is tested by feeding it an unseen test dataset. This step ensures that the model has been trained effectively and can generalize well to new data.

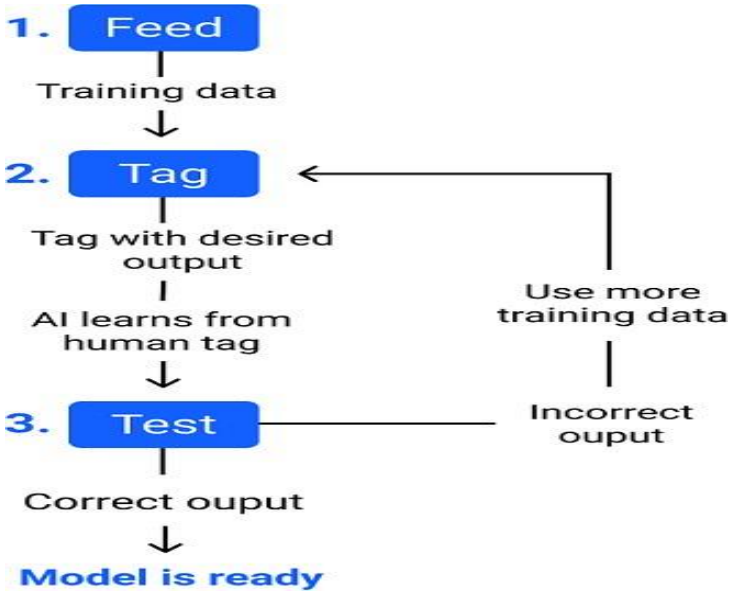

Figure 13: Training and Testing Dataset process

5.4 Training and Testing

5.2.3 Training Data

Training an ML model involves providing an ML algorithm with training data to learn from. The term "ML model" refers to the model artifact produced during the training process. The training data should contain the correct answers, known as the target or target attribute. The training algorithm identifies patterns in the training data that link the input data attributes to the target, resulting in an ML model that captures these patterns. Training data serves as the initial dataset used to train machine learning algorithms, allowing models to establish and refine their rules. It consists of sample data utilized to align the parameters of a machine learning model with its training.

During the training phase, selected algorithms are fed training data to learn and identify models within the training set that correlate input features with the target attribute. Observable patterns are leveraged to construct a model. In this thesis, the IoT dataset serves as the input data source, with the target attribute being the classification of network traffic as

either attack or norm

Our Selected Model	Modele Scikit-learn Methods and Classifiers
K-Nearest Neighbor	sklearn.neighbors.KNeighborsClassifier
Support Vector Machine	Sklearn.svm.LenearSVC
Artificial Neural Network	sklearn.neural_network.MLPClassifier

Table 8: Model building methods and classifiers

Training data can be categorized into two main types

Labeled data: This type of data consists of samples that are labeled with meaningful annotations or classifications. Labeled data is used in supervised learning, where the labels help identify specific attributes or characteristics within the data.

Unlabeled data: In contrast to labeled data, unlabeled data refers to raw data that lacks any associated labels or annotations. This type of data is utilized in unsupervised machine learning models, which aim to discover patterns or similarities within the data without predefined labels.

Quality of Training Data:

The quality of training data significantly impacts the predictive capability of an ML model. To ensure high-quality training data, the following aspects should be considered:

Relevance: The training data should directly relate to the problem being addressed, ensuring that all utilized data is pertinent to the specific issue at hand.

Uniformity: Consistency in the features of a dataset is crucial. All data related to a particular problem should originate from the same source and possess consistent attributes.

Consistency: Similar attributes in a dataset should consistently align with their respective labels, maintaining coherence within the dataset.

Comprehensiveness: Adequate training data should encompass a wide range of features to enable the model to learn effectively, including extreme cases.

Training Data Process

To effectively train an ML model, the following steps need to be specified:

- ✓ Importing the training data source
- ✓ Identifying the data attribute that contains the object to be forecasted
- ✓ Defining necessary instructions for data transformation
- ✓ Setting training parameters to regulate the learning algorithm

Training Data Parameters:

Machine learning algorithms often incorporate parameters that allow for control over various

aspects of the training process and resulting model. These parameters can be configured using tools such as the ML Console, API, or Command Line Interface (CLI). Key training parameters include:

- Maximum model size
- Maximum number of passes through training data
- Randomization type
- Normalization method
- Adjustment amount

5.5 Confusion Matrix

A confusion matrix provides a summary of the correct and incorrect predictions made by a classification model. It is commonly used to assess the performance of intrusion detection models. The key metrics derived from the confusion matrix include true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In this particular study, the dataset consists of five classes (Normal, Mirai, DoS, MITN, and Scan), resulting in a 5x5 matrix. The confusion matrix for the NetSDN dataset is defined in a similar manner, representing a 5x5 matrix with rows and columns corresponding to the different classes.

Confusion Matrix		**Actual class**	
		Normal	**Attack**
Predicted Class	Normal	True Positive (TP)	False Negative (FN)
	Attack	False Positive (FP)	True Negative (TN)

Table 9: shows the confusion matrix for flood detection.

The Confusion Matrix provides valuable insights into the performance of a classification model. Accuracy is the preferred metric when True Positives and True Negatives hold greater

Significance, making it ideal for balanced data sets.

Precision is favored when False Positives are of particular concern, while Recall is preferred when False Negatives are more critical.

The F1-Score is a comprehensive metric that considers both False Negatives and False Positives, making it well-suited for assessing imbalanced data sets.

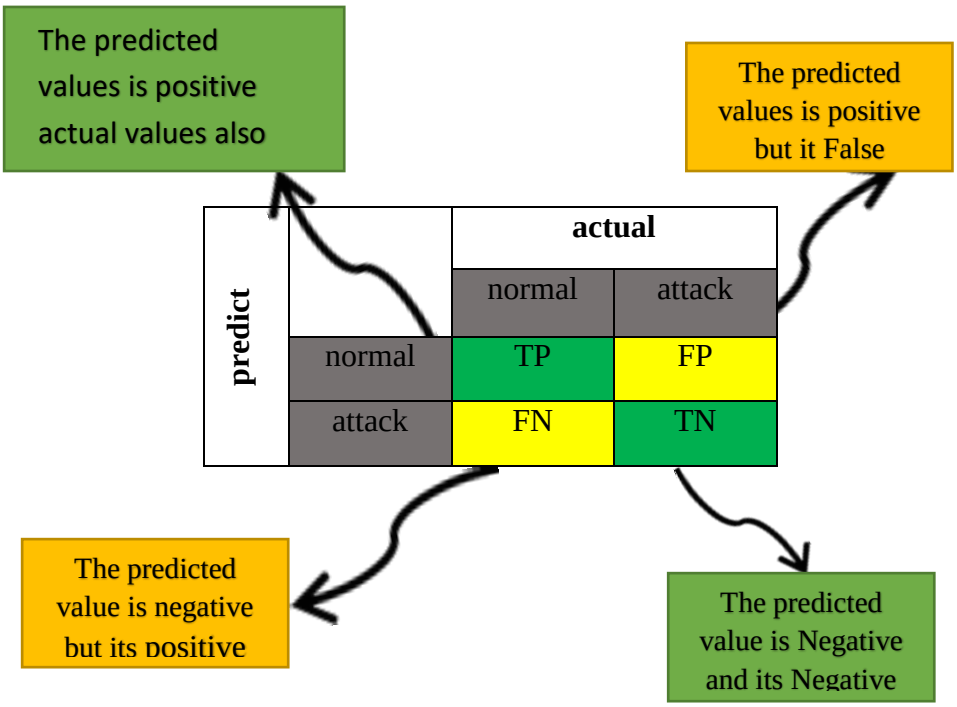

Table 10: Confusion Matrix

The principles for calculating the numbers of true positive (TP), false negative (FN), false positive (FP), and true negative (TN) for each class are as follows:

- ✓ The total number of test samples of any class is the sum of the corresponding row (i.e., the TP+FN for that class).
- ✓ The total number of FN for a class is the sum of values in the corresponding row excluding the TP.
- ✓ The total number of FP for a class is the sum of values in the corresponding column except TP.
- ✓ The total number of TN for a certain class will be the sum of all columns and rows excluding that class's column and row.

The terminologies associated with the Confusion matrix are:

True Positives (TP): Cases when the actual class of the data point was 1 (True) and the predicted is also 1 (True). In this study, it defines the number of malicious records that are correctly identified.

True Negatives (TN): Cases when the actual class of the data point was 0 (False) and the predicted is also 0 (False). In this study, it defines the number of benign records that are correctly classified.

False Positives (FP): Cases when the actual class of the data point was 0 (False) and the predicted is 1 (True). In this study, it defines the number of records that are identified as

attacks but are actually benign activities.

False Negatives (FN): Cases when the actual class of the instance was 1 (True) and the predicted is 0 (False).

5.6 The Results obtained from Dataset

13,942 records of trains and 3,485 records of tests make up the original dataset. Each record includes information about the source-port, destination-port, flow-time, protocol-types, service, flag, source bytes, destination bytes, and other features. Tables 9 and 10 display the dataset's traffic dispersion

Traffic	Training Set		Test Set	
	Size	**Distribution (%)**	**Size**	**Distribution (%)**
Normal	890	6.38	222	6.37
Attack	13,052	93.62	3,263	93.63
Total	13,942	100	3,485	100

Table 11: Traffic distribution of NetSDN in 2-class

Traffic	Training		Test	
	Size	**Distribution**	**Size**	**Distribution**
Normal	890	7.03	222	6.37
Mirai	9,284	66.59	2,322	66.63
DoS	1,298	9.31	324	9.30
Scan	1,696	12.16	424	12.17
MITM ARP Spoofing	774	4.91	193	5.53
Total	**13,942**	**100**	**3,485**	**100**

Table 12: Traffic distribution of NetSDN in multi-class.

5.7 Confusion matrix Result from model

The high number of anomaly occurrences in the confusion matrix have been classified more accurately than the limited number of normal ones. ANN classifiers with wrapper feature selection approach have been used to classify the cases correctly and wrongly.

```
      ANN_Confusion Matrix:
[[ 200   10    0    3    1]
 [   1 2247    0   70   18]
 [   0    1  303    0    0]
 [   0   36    0  391    3]
 [   0  105    0   40   57]]
```

Table 13: ANN classifier Confusion Matrix

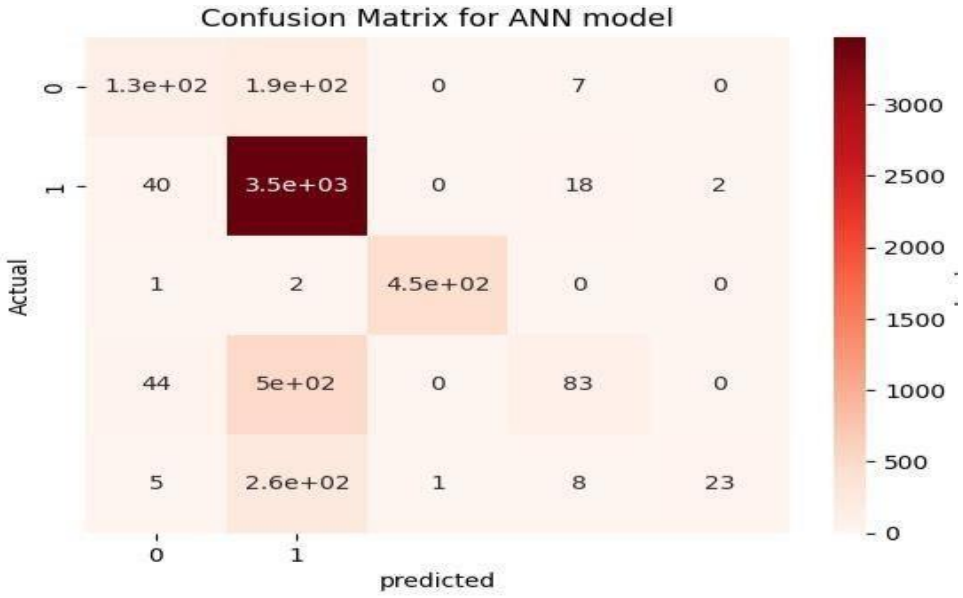

Figure 14: ANN model Confusion Matrix

y used wrapper feature Selection approach, the KNN classifiers confusion matrix has known the instances properly classified and erroneously classed with the total number of instances for datasets

KNN confusion matrix

```
[[ 162   60    0   12    1]
[  27 2105    0  111   57]
[   0    2  324    1    0]
[  10   84    0  329    4]
[   1   80    1   25   90]]
```

Table 14: KNN Classifier Confusion Matrix Table

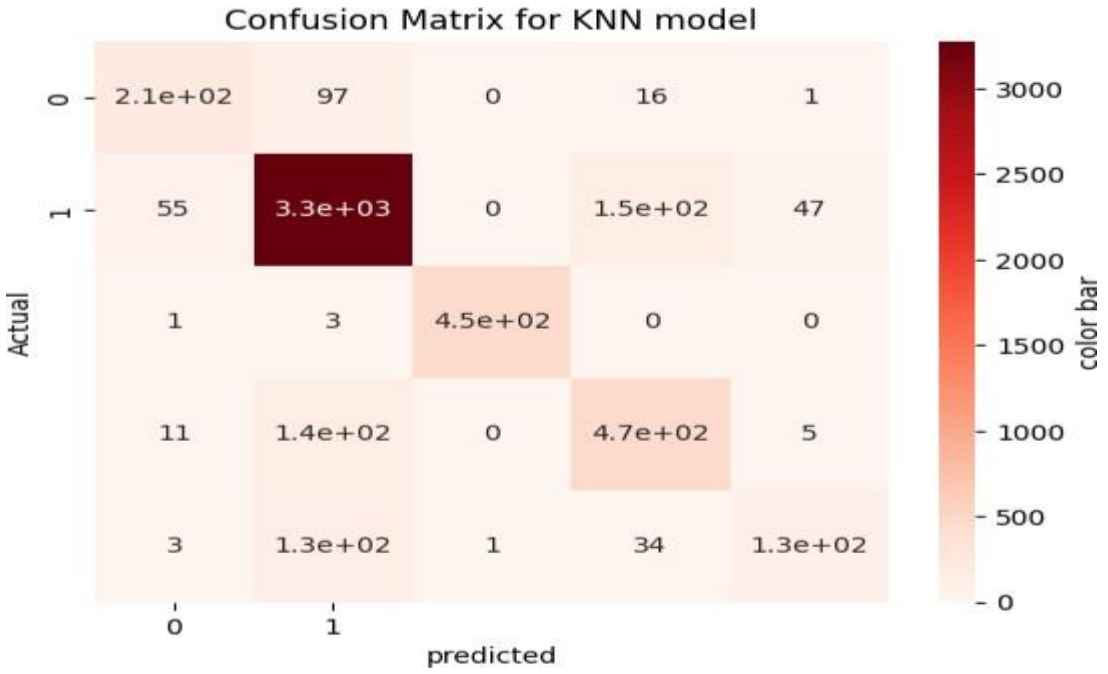

Figure 15: KNN model Confusion Matrix

5.8 Model Performance

Based on the previous confusion matrix, the performance measures are generated for each class in order to investigate the performance of the model in more detail: Accuracy, Precision, Recall, and F-Measure. The NetSDN dataset suggests KNN, and both other approaches shine out when these performance metrics are evaluated. The KNN Classifier method performs best among the recursive Feature Elimination techniques for identifying anomalies in the NetSDN dataset. This technique uses the least amount of time to construct the model, whereas the random forest classifier approach takes longer to fit processes and has a worse accuracy in predicting anomalies.

In the assessment process, we used classification accuracy and other metrics to showcase the efficiency of model compared to surface distributed IoT-level models.

Model	**2-class**			**5-class**		
	Accuracy	**Detection Rate**	**False alarm rate**	**Accuracy**	**Detection Rate**	**False Alarm Rate**
KNN	96.56	77.99	1.16	86.67	85.22	5.67
ANN	93.63	49.30	9.91	80.30	65.19	8.98
SMV	93.78	0	0	67.47	0	0

Table 15: Accuracy of KNN, ANN and SVM model

Model	Class	Evaluation Metrics			
		Accuracy	**Precision**	**Recall**	**F1-Measure**
KNN	Train	96.56	97.53	98.84	98.18
	Test	97.20	97.83	99.20	98.51
ANN	Train	96.93	98.44	98.29	98.37
	Test	93.63	99.03	94.13	96.52
SVM	Train	93.55	93.55	100	96.37
	Test	93.78	93.78	100	96.79

Table 16: Performance Evaluation

Reference	Model	Accuracy	Dataset
(Mohmand et al., 2022)	MLP/SVM	88.4/79.0	KDD dataset, UNWS-NB15
(Prasad & Chandra, 2022)	ML /Not specified the model	95.0	CICIDS2017 dataset
(Islam et al., 2022)	SVM/KNN	96.7	bank dataset

Table 17: The comparison study of SVM, KNN, and ANN.

A pie chart is a visual representation method used to display data in a circular graph format. It is particularly effective when showing data with few variables. Pie charts are commonly utilized to illustrate sample data, showing the distribution of data points across different categories. Each slice of the pie chart is proportional to the number of data points within a specific category. In this experiment, the output of the pie chart includes labels for normal data, Mirai data, DoS data, Scan data, and MITM ARP Spoofing data.

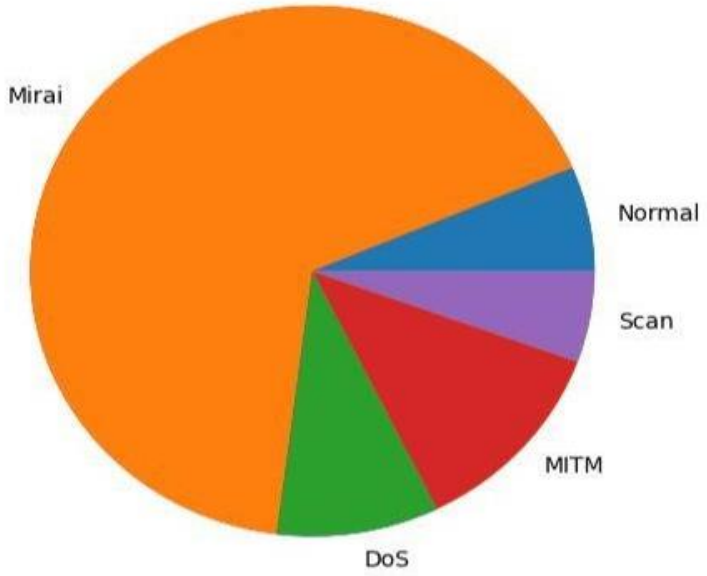

Figure 22: Pie Graph

Responses of research questions

Q1. To select more significant features that have more values in the process of machine model training, we have explored different feature section techniques from various literature. Based on our study and lab experiment we selected 55 features out of 86 features by using wrapper feature selection methods.

Q2. In intrusion detection, there is no common solution for all detection accuracy performance. Due to this reason, we had to investigate different techniques from in various sources that helps to identify what technique is used. Thus, based on our study we have selected three supervised machine learning algorithm those are K-nearest neighbor (KNN), Support Vector Machine, and Artificial Neural Network. When we selected the algorithm, we have considered we were performing malicious traffic detection and classification by using labeled historical dataset.

Q3. Model performance accuracy is the goal of this research. Therefore, we had to study evaluation methods that helps us to improve model performance during training phase and to test final model performance. So, we used different performance metrics those are k-fold cross validation for training phase and accuracy, precision, recall and F-score for final model performance test.

CHAPTER SIX

CONCLUSION AND FUTURE WORKS

The SDN security model developed is able to malicious intrusion attacks such as TCP flooding and ICMP flooding. Which has significantly changed businesses in a number of ways. It has also increased reliance on technology to obtain a competitive edge. Nonetheless, given that attackers have also changed rapidly, security concerns have emerged as one of the biggest challenges facing businesses. A solution to guard against such security concerns should exist due to the IT industry's constant evolution and the sophistication of cyberattacks on computer networks. ML methods are among the technologies used in intrusion detection systems. The models employ the DT and NN algorithms and the most recent research trends in the field as reported in the literature review section. The outcomes validate the efficacy of our suggested approaches, which may help to enhance computer network security.

6.1 Conclusion

The current network model is significantly improved by the SDN paradigm. One area where SDN could be dominant is IoT. It has been suggested that SDN will increase IoT security, scalability, and resilience. By implementing several application-level security apps, IoT security can be enhanced by deploying it on SDN-based infrastructure. SDN does, however, have significant security flaws, such as network controller threats. The traffic is received by an IoT network gateway. The gateway forwards new messages to the controller, which establishes flow rules according to the kinds of traffic it receives. This study shows that machine learning is an effective method for quickly addressing different types of attacks in Internet of Things networks. The general framework of the proposed study will be examined in this thesis research, including data preparation, feature selection for the wrapper, training and testing, model implementation, and model evaluation. To distinguish between normal and abnormal attack events on the Internet of Things, machine learning models for binary classification have been trained and validated using the NetSDN.

We evaluated ML classifiers by computing accuracy, precision, recall, and F1score. In addition to these metrics, each classifier's training, prediction, and execution times were also computed. In order to improve the detection of IoT botnet attacks, this work has developed an aggregated mutual information-based feature selection method that incorporates machine learning techniques. Four main steps comprise this approach: feature selection, classification, data preparation, and collection. Each type of assault was fed into the feature selection algorithms to produce a set of reduced features. The smaller feature set was then used to train the ML classifiers. This study has developed an aggregated mutual information-based feature

selection to improve IoT threat detection using machine learning techniques. The four primary steps of this process are data collecting data preparation, feature selection, categorization, and. Every attack type was fed into the feature selection algorithms to obtain a set of reduced feature. The set of features with lower complexity was then used to train the machine learning classifiers. The proposed approach was used to binary and multi-class classification problems. The model SVM, KNN, and ANN received performance evaluations of 93.78, 96.56, and 93.63 from the class datasets, respectively. KNN classifier algorithms achieve the objective since the KNN model yields the best accuracy results. Ultimately, an assessment of the machine learning model was conducted, and the results were reported

6.2 Future work

This work aims to create a detection model that uses the "NetSDN" to classify traffic from attacks that are either normal or abnormal. To multi-classify the attack kinds, we recommend to improve this model in our upcoming efforts.

We will also recommend trying additional algorithms and hybrid approaches in order to improve our algorithm's efficacy and efficiency.

We may also use newly gathered data to test this model in another area of future research.

Subsequent investigations can employ diverse IoT botnet datasets to implement the proposed methodology.

REFERENCES

Eswari, D. S., Lakshmi, P. V, & Kishore, P. K. (2023). *A Comprehensive Survey on the Efficacy of Machine Learning Algorithms for Detecting DDoS Attack*. 409–416.

Ahmed, A. A., & Ahmed, W. A. (2019). An effective multifactor authentication mechanism based on combiners of hash function over internet of things. *Sensors, 19*(17), 3663.

Alhaj, A. N., & Dutta, N. (2022). Analysis of security attacks in SDN network: A comprehensivesurvey. *Contemporary Issues in Communication, Cloud and Big Data Analytics: Proceedings of CCB 2020*, 27-37.

Alqahtani, A. S. (2022). FSO-LSTM IDS: hybrid optimized and ensembled deep-learning network-based intrusion detection system for smart networks. *The Journal of Supercomputing, 78*(7), 9438-9455.

Alshra'a, A. S., & Seitz, J. (2019). Using inspector device to stop packet injection attack in SDN. *IEEE Communications Letters*, *23*(7), 1174-1177.

Amin, R., Reisslein, M., & Shah, N. (2018). Hybrid SDN networks: A survey of existing approaches. *IEEE Communications Surveys & Tutorials, 20*(4), 3259-3306.

Ashley, R., Gersonius, B., & Horton, B. (2020). Managing flooding: from a problem to an opportunity. *Philosophical Transactions of the Royal Society A, 378*(2168), 20190214.

Azam, M. A., Sukarti, S., & Zaimi, M. (2020). Corrosion behavior of API-5L-X42 petroleum/natural gas pipeline steel in South China Sea and Strait of Melaka seawaters.*Engineering Failure Analysis, 115*, 104654.

Bahashwan, A. A., Anbar, M., & Hanshi, S. M. (2020). *Overview of IPv6 based DDoS and DoS attacks detection mechanisms.* Paper presented at the Advances in Cyber Security: FirstInternational Conference, ACeS 2019, Penang, Malaysia, July 30–August 1, 2019, Revised Selected Papers 1.

Carley, K. M. (2020). Social cybersecurity: an emerging science. *Computational and mathematical organization theory, 26*(4), 365-381.

Carpa, R., de AssunçÃo, M. D., Glück, O., Lefevre, L., & Mignot, J.-C. (2017). *Evaluating theimpact of SDN-induced frequent route changes on TCP flows.* Paper presented at the 2017 13th International Conference on Network and Service Management (CNSM).

Castillo, E. F., Rendon, O. M. C., Ordonez, A., & Granville, L. Z. (2020). IPro: An approach

for intelligent SDN monitoring. *Computer Networks, 170*, 107108.

Chen, C., Liao, Z., Ju, Y., He, C., Yu, K., & Wan, S. (2022). Hierarchical domain-based multicontroller deployment strategy in SDN-enabled space–air–ground integrated network. *IEEE Transactions on Aerospace and Electronic Systems, 58*(6), 4864-4879.

Comaneci, D., & Dobre, C. (2018). *Securing networks using SDN and machine learning.* Paper presented at the 2018 IEEE International Conference on Computational Science and Engineering (CSE).

D'Cruze, H., Wang, P., Sbeit, R. O., & Ray, A. (2018). *A software-defined networking (SDN) approach to mitigating DDoS attacks.* Paper presented at the Information Technology- New Generations: 14th International Conference on Information Technology.

Dawoud, A., Shahristani, S., & Raun, C. (2018). Deep learning and software-defined networks:Towards secure IoT architecture. *Internet of Things, 3*, 82-89.

Eswari, D. S., Lakshmi, P. V, & Kishore, P. K. (2023). *A Comprehensive Survey on the Efficacy of Machine Learning Algorithms for Detecting DDoS Attack*. 409–416.

Faraj, O., Megías, D., Ahmad, A.-M., & Garcia-Alfaro, J. (2020). *Taxonomy and challenges inmachine learning-based approaches to detect attacks in the internet of things.* Paper presented at the Proceedings of the 15th International Conference on Availability, Reliability and Security.

Fawcett, L., Scott-Hayward, S., Broadbent, M., Wright, A., & Race, N. (2018). Tennison: A distributed SDN framework for scalable network security. *IEEE Journal on Selected Areas in Communications, 36*(12), 2805-2818.

Hanson, M. R., & Hines, K. M. (2018). Stromules: probing formation and function. *Plant physiology, 176*(1), 128-137.

Hassan, D. (2017). Cost-sensitive access control for detecting remote to local (R2L) and user toroot (U2R) attacks. *International Journal of Computer Trends and Technology (IJCTT),43*(2), 124-129

Islam, U., Muhammad, A., Mansoor, R., Hossain, S., Ahmad, I., Eldin, E. T., Khan, J. A., Rehman, A. U., & Shafiq, M. (2022). *Detection of Distributed Denial of Service (DDoS) Attacks in IOT Based Monitoring System of Banking Sector Using Machine Learning Models*.

Jiang, S., Cao, J., Wu, H., & Yang, Y. (2020). Fairness-based packing of industrial IoT data in permissioned blockchains. *IEEE Transactions on Industrial Informatics, 17*(11), 7639- 7649.

Kareem, M. I., & Jasim, M. N. (2021). *The Current Trends of DDoS Detection in SDN*

Environment. Paper presented at the 2021 2nd Information Technology To Enhance e- learning and Other Application (IT-ELA).

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detectionsystems: techniques, datasets and challenges. *Cybersecurity, 2*(1), 1-22.

Lakhan, A., Mohammed, M. A., Obaid, O. I., Chakraborty, C., Abdulkareem, K. H., & Kadry,

S. (2022). Efficient deep-reinforcement learning aware resource allocation in SDN-enabled fog paradigm. *Automated Software Engineering, 29*, 1-25.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys (CSUR), 50*(6), 1-45.

Li, J., Zhao, Z., Li, R., & Zhang, H. (2018). Ai-based two-stage intrusion detection for softwaredefined iot networks. *IEEE Internet of Things Journal, 6*(2), 2093-2102.

Li, T., Chen, J., & Fu, H. (2019). *Application scenarios based on SDN: an overview.* Paper presented at the Journal of Physics: Conference Series.

Liang, L., Zheng, K., Sheng, Q., & Huang, X. (2016). *A denial of service attack method for aniot system.* Paper presented at the 2016 8th international conference on Information Technology in Medicine and Education (ITME).

Lynch, M., Sage, T., Hitchcock, L. I., & Sage, M. (2021). A heutagogical approach for the assessment of Internet Communication Technology (ICT) assignments in higher education. *International Journal of Educational Technology in Higher Education, 18*(1),55.

Mahjabin, T., Xiao, Y., Sun, G., & Jiang, W. (2017). A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks, 13*(12), 1550147717741463.

Nakazawa, T. (2017). Expanding the scope of studentification studies. *Geography Compass, 11*(1), e12300.

Patil, S., & Chaudhari, S. (2016). DoS attack prevention technique in wireless sensor networks.

Procedia Computer Science, 79, 715-721.

Ahmad, Z. (2021). *Network intrusion detection system : A systematic study of machine learning and deep learning approaches*. *May 2020*, 1–29. https://doi.org/10.1002/ett.4150

Alahmadi, A. A., Aljabri, M., Alhaidari, F., Alharthi, D. J., & Rayani, G. E. (2025). *DDoS Attack Detection in IoT-Based Networks Using Machine Learning Models : A Survey and Research Directions*. 1–24.

Anusuya, R. (2023). *Detection of TCP , UDP and ICMP DDOS attacks in SDN Using*

Machine Learning approach. *10*, 964–971.

Bundy, T. (2013). Dynamic, software-defined service provider network infrastructure and cloud drivers for SDN adoption. *2013 IEEE International Conference on Communications Workshops (ICC), March 2019*, 235–239. https://doi.org/10.1109/ICCW.2013.6649235

Corsetti, S., Purkayastha, A., Hasandka, A., Samon, M., Corsetti, S., Purkayastha, A., Hasandka, A., & Samon, M. (2021). *Automatic DDoS Attack Detection on SDNs Preprint Automatic DDoS Attack Detection on SDNs Preprint*. *September 2022*.

Dehkordi, A. B., Soltanaghaie, M., & Boroujeni, F. Z. (n.d.). *A New DDoS Detection Method in Software Defined Network*.

Elsayed, M. S., & Jurcut, A. D. (2020). *InSDN : A Novel SDN Intrusion Dataset*. *October*. https://doi.org/10.1109/ACCESS.2020.3022633

Engineering, C. (2004). *Review On Architecture & Security Issues of*. 6519–6524.

Eswari, D. S., Lakshmi, P. V, & Kishore, P. K. (2023). *A Comprehensive Survey on the Efficacy of Machine Learning Algorithms for Detecting DDoS Attack*. 409–416.

Farooq, M. S., Riaz, S., & Alvi, A. (2023). *Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review*.

Islam, U., Muhammad, A., Mansoor, R., Hossain, S., Ahmad, I., Eldin, E. T., Khan, J. A., Rehman, A. U., & Shafiq, M. (2022). *Detection of Distributed Denial of Service (DDoS) Attacks in IOT Based Monitoring System of Banking Sector Using Machine Learning Models*.

Jawad, F. H. M. (2018). Software defined networking: An overview. *Journal of Engineering and Applied Sciences*, *13*(SpecialIssue13), 10790–10796. https://doi.org/10.36478/jeasci.2018.10790.10796

Logeswari, G., Bose, S., & Anitha, T. (2023). *An Intrusion Detection System for SDN Using Machine Learning*. https://doi.org/10.32604/iasc.2023.026769

Mohmand, M. I., Hussain, H., Khan, A. A. L. I., Ullah, U., Zakarya, M., Member, S., Ahmed, A., Raza, M., Rahman, I. U. R., & Haleem, M. (2022). A Machine Learning-Based Classification and Prediction Technique for DDoS Attacks. *IEEE Access*, *10*, 21443–21454. https://doi.org/10.1109/ACCESS.2022.3152577

Prasad, A., & Chandra, S. (2022). VMFCVD : An Optimized Framework to Combat Volumetric DDoS Attacks using Machine Learning. *Arabian Journal for Science and Engineering*, *47*(8), 9965–9983. https://doi.org/10.1007/s13369-021-06484-9

Singh, P. K., Jha, S. K., Nandi, S. K., & Nandi, S. (2018). ML-Based Approach to Detect DDoS Attack in V2I Communication Under SDN Architecture. *TENCON 2018 - 2018*

IEEE Region 10 Conference, 144–149. https://doi.org/10.1109/TENCON.2018.8650452

Suthar, F., & Patel, N. (2023). A Survey on DDoS Detection and Prevention Mechanism. *Journal of Advances in Information Technology*, *14*(3), 444–453. https://doi.org/10.12720/jait.14.3.444-453

Wang, S., Balarezo, J. F., Chavez, K. G., Al-Hourani, A., Kandeepan, S., Asghar, M. R., & Russello, G. (2022). Detecting flooding DDoS attacks in software defined networks using supervised learning techniques. *Engineering Science and Technology, an International Journal*, *35*, 101176. https://doi.org/10.1016/j.jestch.2022.101176

Wilson, D.R., Martinez, T.R. Reduction Techniques for Instance-Based Learning Algorithms. Machine Learning 38 , 257 – 286 (2000). (2000). *286*, 2000.

Rana, D. S., Dhondiyal, S. A., & Chamoli, S. K. (2019). Software defined networking (SDN) challenges, issues and solution. *International journal of computer sciences and engineering*, *7*(1), 884-889.

Suthar, F., & Patel, N. (2023). A Survey on DDoS Detection and Prevention Mechanism. Journal of Advances in Information Technology, 14(3), 444–453. https://doi.org/10.12720/jait.14.3.444-453

Sen, S., Gupta, K. D., & Manjurul Ahsan, M. (2020). *Leveraging machine learning approach to setup software-defined network (SDN) controller rules during DDoS attack.* Paper presented at the Proceedings of International Joint Conference on Computational Intelligence: IJCCI 2018.

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*,*2*(1), 41-50.

Wu, M., Huang, W., Sun, K., & Zhang, H. (2020). *A DQN-based handover management for SDN-enabled ultra-dense networks.* Paper presented at the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall).

Yin, D., Zhang, L., & Yang, K. (2018). A DDoS attack detection and mitigation with software-defined Internet of Things framework. *IEEE Access*, *6*, 24694-24705.

Zhou, D., Yan, Z., Fu, Y., & Yao, Z. (2018). A survey on network data collection. *Journal of Network and Computer Applications*, *116*, 9-23.

Zhou, D., Yan, Z., Liu, G., & Atiquzzaman, M. (2019). An adaptive network data collection system in sdn. *IEEE Transactions on Cognitive Communications and Networking*, *6*(2),562-574.

Zhu, L., Karim, M. M., Sharif, K., Xu, C., Li, F., Du, X., & Guizani, M. (2020). SDN

controllers: A comprehensive analysis and performance evaluation study. *ACM Computing Surveys(CSUR), 53*(6), 1-40)

APPENDIXES

Appendix A: Description of Dataset Features

#	Feature Name	Description
1	Source Port	To identifies the sent data
2	Destination Port	To identifies the received data
3	Protocol	To identifies the values of layer to the data that should be passed
4	Flow Duration	Flow duration
5	Total Fwd Packets	Total packets in the forward direction
6	Total Length of Fwd	Packets Total size of packet in forward direction
7	Total Backward Packets	Total packets in the backward direction
8	Fwd Packet Length Max	Maximum size of packet in forward direction
9	Total Length of Bwd	Packets Total size of packet in backward direction
10	Fwd Packet Length Min	Minimum size of packet in forward direction
11	Fwd Packet Length Mean	Average size of packet in forward direction
12	Fwd Packet Length Std	Standard deviation size of packet in forward direction
13	Bwd Packet Length Max	Maximum size of packet in backward direction
14	Bwd Packet Length Min	Minimum size of packet in backward direction
15	Bwd Packet Length Mean	Mean size of packet in backward direction
16	Bwd Packet Length Std	Standard deviation size of packet in backward direction
17	Flow Bytes/s	Flow byte rate that is number of packets transferred per second
18	Flow Packets/s	Flow packets rate that is number of packets transferred per second
19	Flow IAT Mean	Average time between two flows
20	Flow IAT Std	Standard deviation time two flows
21	Flow IAT Max	Maximum time between two flows
22	Flow IAT Min	Minimum time between two flows
23	Fwd IAT Total	Total time between two packets sent in the forward direction
24	Fwd IAT Mean	Mean time between two packets sent in the forward direction
25	Fwd IAT Std	Standard deviation time between two packets sent in the forward direction

#	Feature Name	Description
26	Fwd IAT Max	Maximum time between two packets sent in the forward direction
27	Fwd IAT Min	Minimum time between two packets sent in the forward direction
28	Bwd IAT Total	Total time between two packets sent in the backward direction
29	Bwd IAT Mean	Mean time between two packets sent in the backward direction
30	Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
31	Bwd IAT Max	Maximum time between two packets sent in the backward direction
32	Bwd IAT Min	Minimum time between two packets sent in the backward direction
33	Fwd Header Length	Total bytes used for headers in the forward direction
34	Bwd Header Length	Total bytes used for headers in the forward direction
35	Fwd Packets/s	Number of forward packets per second
36	Bwd Packets/s	Number of backward packets per second
37	Min Packet Length	Minimum length of a flow
38	Max Packet Length	Maximum length of a flow
39	Packet Length Mean	Mean length of a flow
40	Packet Length Std	Standard deviation length of a flow
41	Packet Length Variance	Minimum inter-arrival time of packet
42	Down/Up Ratio	Download and upload ratio
43	Average Packet Size	Average size of packet
44	Avg Fwd Segment Size	Average size observed in the forward direction
45	Avg Bwd Segment Size	Average size observed in the backward direction
46	Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
47	Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
48	Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
49	act_data_pkt_fwd	Number of packets with at least 1 byte of TCP data payload in the forward direction
50	Init_Win_bytes_forward	Number of bytes sent in initial window in the forward direction
#	**Feature Name**	**Description**
51	Init_Win_bytes_backward	Number of bytes sent in initial window in the backward direction

52	act_data_pkt_fwd	Number of packets with at least 1 byte of TCP data payload in the forward direction
53	Idle Mean	Mean time a flow was idle before becoming active
54	Idle Std	Standard deviation time a flow was idle before becoming active
55	Idle Max	Maximum time a flow was idle before becoming active
56	Idle Min	Minimum time a flow was idle before becoming active

Appendix B: Sample Source Code

```
In [23]: X_train_prediction = modelSVM.predict(X_train)
         training_data_accuracy = accuracy_score(y_train, X_train_prediction)
         X_test_prediction = modelSVM.predict(X_test)
         test_data_accuracy = accuracy_score(y_test, X_test_prediction)
         print('The Precision on training data by using SVM Model : ', round(test_data_accuracy*100, 2), '%')
         print('The Precision on Test data by using SVM Model : ', round(test_data_accuracy*100, 2), '%')
```

```
The Precision on training data by using SVM Model :  93.78 %
The Precision on Test data by using SVM Model :  93.78 %
```

```
In [17]: print('The Mean squared error of SVM model prediction is: ', round(mean_squared_error(SVMpred, y_test)*100, 2), '%')
         print('The r2 Score of SVM model prediction is: ', round(r2_score(SVMpred, y_test)*100, 2), '%')
         print('The Mean Absolute error of SVM model prediction is: ', round(mean_absolute_error(SVMpred, y_test)*100, 2), '%')
```

```
The Mean squared error of SVM model prediction is:  6.22 %
The r2 Score of SVM model prediction is:  0.0 %
The Mean Absolute error of SVM model prediction is:  6.22 %
```

In [17]:
```
from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import cohen_kappa_score

def get_data():
    X, y = make_circles(n_samples = 1000, noise = 0.1, random_state = 1)
accuracy = accuracy_score(y_test, ANNpred)
print(f"accuracy of the Test set: {accuracy}")

recall = recall_score(y_test, ANNpred, average= 'weighted')
print ('Recall of the Test set: %f' % recall)
precision = precision_score(y_test, ANNpred, average= 'weighted')
print ('Precision of the Test set: %f' % precision)
f1 = f1_score(y_test, ANNpred, average= 'weighted')
print ('F1-Measure of the Test set: %f' %f1)

#auc = roc_auc_score(y_test, ANNpred)
#print ('ROC AUC of the Test set: %f' %auc)

kappa = cohen_kappa_score(y_test, ANNpred)
print ('Cohens kappa of the Test set: %f' %kappa)

matrix = confusion_matrix (y_test, ANNpred)
print (matrix)
```

In [12]:
```
from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import cohen_kappa_score

KNNpred = modelKNN.predict(X_train)

def get_data():
    X, y = make_circles(n_samples = 1000, noise = 0.1, random_state = 1)
accuracy = accuracy_score(y_train, KNNpred )
print(f"Accuracy of the training set: {accuracy}")

recall = recall_score(y_train, KNNpred, average= 'weighted')
print ('Recall of the training set: %f' % recall)
precision = precision_score(y_train, KNNpred, average= 'weighted')
print ('Precision of the training set: %f' % precision)
f1 = f1_score(y_train, KNNpred, average= 'weighted')
print ('F1-Measure of the training set: %f' %f1)

#auc = roc_auc_score(y_train, SVMpred)
#print ('ROC AUC of the training set: %f' %auc)

kappa = cohen_kappa_score(y_train, KNNpred)
print ('Cohens kappa of the training set: %f' %kappa)

matrix = confusion_matrix (y_train, KNNpred)
print (matrix)
```

In [62]:
```
import numpy as np
print ("Normalized Confusion Matrix")
#cm_n = np.round(cm / np.sum(cm, axis = 1).reshape (-1,1), 2)
#print(cm_n)
sns.heatmap(cm_n, cmap = "Reds", annot = True, cbar_kws = {'orientation': 'vertical', 'label':'color bar'},
          xticklabels = [0, 5], yticklabels = [5, 0])
plt.xlabel("predicted")
plt.ylabel("Actual")
plt.title("Normalized Confusion Matrix")
plt.show()
```

Appendix C: Selected dataset Columns and rows

Out[2]:

	Source Port	Destination Port	Protocol	Flow_Duration	Tot_Fwd_Pkts	Tot_Bwd_Pkts	TotLen_Fwd_Pkts	TotLen_Bwd_Pkts	Fwd_Pkt_Len_Max	Fwd_Pkt_Len_Min
0	10000	10101	17	75	1	1	982	1430	982	982
1	2179	554	6	5310	1	2	0	0	0	0
2	52727	9020	6	141	0	3	0	2806	0	0
3	52964	9020	6	151	0	2	0	2776	0	0
4	36763	1900	17	153	2	1	886	420	452	434
...	...	...	...	...	...	...	...	...	...	...
17422	39310	7760	6	74	0	2	0	0	0	0
17423	52930	9020	6	73	0	2	0	2776	0	0
17424	56361	10101	17	293	0	5	0	168	0	0
17425	53190	9020	6	73	0	2	0	2776	0	0
17426	56361	10101	17	286	1	1	202	1430	202	202

17427 rows × 56 columns

Appendix D: X_train and X_test Model Output

```
Source Port  Destination Port  Protocol  Flow_Duration  Tot_Fwd_P
kts  \
6930          1632              554         6           2675
0
5072          2974              554         6           6096
0
13277          443              51875       6            146
1
3320          2225              554         6           9229
0
6744          56361             10101       17           199
1
...             ...               ...       ...          ...
...
14565         56361             10101       17           224
3
15649         10000             10101       17            76
0
```

```
10123        64775              9988        17              7
3
5600         52727              9020         6             74
1
14000         5223             53207         6             74
1
```

```
       Tot_Bwd_Pkts  TotLen_Fwd_Pkts  TotLen_Bwd_Pkts  Fwd_Pkt_Len_Max
\
6930              2                0                0               0
5072              2                0                0               0
13277             1             1448             1448            1448
3320              2                0                0               0
6744              1              914             1430             914
...             ...              ...              ...             ...
14565             1             2880             1430            1430
15649             2                0               72               0
10123             1               96               32              32
5600              1                0                0               0
14000             1               53                0              53
```

```
       Fwd_Pkt_Len_Min  ...  Subflow_Bwd_Pkts  Subflow_Bwd_Byts  \
6930                 0  ...                 2                 0
5072                 0  ...                 2                 0
13277             1448  ...                 1              1448
3320                 0  ...                 2                 0
6744               914  ...                 1              1430
...                ...  ...               ...               ...
14565               20  ...                 1              1430
15649                0  ...                 2                72
10123               32  ...                 1                32
5600                 0  ...                 1                 0
14000               53  ...                 1                 0
```

```
Init_Fwd_Win_Byts  Init_Bwd_Win_Byts  Fwd_Act_Data_Pkts  Idle_Mea
n
\
6930                 -1              14600                  0
                     2675.0
0
5072                 -1              14600                  0
                     6096.0
0
13277                -1                252                  1
                     146.0
0
3320                 -1              14600                  0
                     9229.0
0
6744                 -1                 -1                  1
                     199.0
0
...                 ...                ...                ...
                    ..
.
14565                -1                 -1                  3
                     74.6
7
15649                -1                 -1                  0
                     76.0
0
10123                -1                 -1                  3
                     2.3
3
5600                 -1              32422                  0
                     74.0
0
14000                -1                387                  1
                     74.0
0

       Idle_Std Idle_Max
Idle_Min Label6930   0.00      2675   2675 1
5072       0.00      6096      6096      1
13277      0.00       146       146      1
3320       0.00      9229      9229      1
6744       0.00       199       199      1
...         ...       ...       ...    ...
14565      8.08        84        70      1
15649      0.00        76        76      1
10123      0.58         3         2      1
5600       0.00        74        74      1
14000      0.00        74        74      1

[12198 rows x 56 columns]        Source Port Destination
Port ProtocolFlow_Duration  Tot_Fwd_Pkts \
8594         56204              9020         6            150
0
```

```
14273        10000             10101        17            291
0
14678        60134              8899        17              4
1
10012         9020             49784         6            479
0
12756        52738              9020         6            121

11896        60099              8899         17           25
3
8809          8305               554         6           2573
0
870           9020             49784         6            343
2
1240         64775              9988        17              9
8
16800        51992              8899        17             24
2

       Tot_Bwd_Pkts  TotLen_Fwd_Pkts  TotLen_Bwd_Pkts
       Fwd_Pkt_Len_Max
\
8594              3                0             4164
                  0
14273             5                0              160
                  0
14678             1               32               32
                  32
10012             3                0             1388
                  0
12756             2                0             1418
                  0
...             ...              ...              ...
                ...
11896             1               96               32
                  32
8809              2                0                0
                  0
870               1             1418             1388
                  1388
1240              1              256               32
                  32
16800             1               64               32
                  32

       Fwd_Pkt_Len_Min ... Subflow_Bwd_Pkts
Subflow_Bwd_Byts  \8594                    0  ...            3  4164
14273                0  ...                 5              160
14678               32  ...                 1               32
10012                0  ...                 3             1388
12756                0  ...                 2             1418
...                ...  ...               ...              ...
11896               32  ...                 1               32
8809                 0  ...                 2                0
```

```
870                30  ...                 1             1388
1240               32  ...                 1               32
16800              32  ...                 1               32

Init_Fwd_Win_Byts  Init_Bwd_Win_Byts  Fwd_Act_Data_Pkts  Idle_Mea
n  \
8594                 -1               1869                  0
                     75.0
0
14273                -1                 -1                  0
                     72.7
5
14678                -1                 -1                  1
                     4.0
0
10012                -1              32505                  0
                     239.5
0
12756                -1               1869                  0
                     121.0
0
...                 ...                ...                ...
                    ..
.
11896                -1                 -1                  3
                     8.3
3
```

```
8809                  -1              14600                    0    2573.0
0
870                   -1               1869                    2     171.5
0
1240                  -1                 -1                    8       2.2
5
16800                 -1                 -1                    2      12.0
0

          Idle_Std  Idle_Max  Idle_Min  Label
8594      2.830000        77        73      1
14273     0.500000        73        72      1
14678     0.000000         4         4      1
10012   231.223917       403        76      0
12756     0.000000       121       121      1
...            ...       ...       ...    ...
11896     4.930000        14         5      1
8809      0.000000      2573      2573      1
870     132.230000       265        78      0
1240      0.500000         3         2      1
16800     9.900000        19         5      1

[5229 rows x 56 columns]
```