Detection of Denial of Service (DOS) Attack in Wireless Sensor Networks (WSNs) using Machine Learning

A Dissertation submitted to the Jawaharlal Nehru Technological University, Hyderabad in partial fulfillment of the requirement for the award of degree of

MINOR PROGRAMME
IN
CYBER SECURITY

Submitted by

DUSSA PAVAN	**20B81A05L3**
JIRRA VIGNESHWAR	**20B81A0457**

Under the guidance of
Ms. K. Sinduja
Assistant Professor CSE(AIML)
Department of CSE (Cyber Security)

DEPARTMENT OF CSE (Cyber Security)
CVR COLLEGE OF ENGINEERING
(An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510
APRIL 2024

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

DEPARTMENT OF CSE (Cyber Security)

CERTIFICATE

This is to certify that the project report entitled **"Detection of Denial of Service Attack in Wireless Sensor Networks using Machine Learning"** is a bonafide record of work carried out by **Dussa Pavan (20B81A05L3)** and **Jirra Vigneshwar (20B81A0457)** submitted to **Ms. K. Sinduja** for the requirement of the award of **Minor Degree** in **Cyber Security** to the CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad during the year 2023-2024.

Project Guide
Ms.K. Sinduja
Assistant Professor (CSE-AIML)
Department of CSE (Cyber Security)

Head of the Department
Dr. Lakshmi H N
Professor & HOD-
Department of CSE(Cyber Security)

Project Coordinator
Dr. M. Sunitha
Professor, Coordinator CSE (Cyber Security)

External Examiner

Mr. Sravan Kumar G
Assistant Professor, Coordinator CSE (Cyber Security)

DECLARATION

We hereby declare that the project report entitled **"Detection of Denial of Service Attack in Wireless Sensor Network using Machine Learning"** is an original work done and submitted to Department, CSE (Cyber Security), CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University Hyderabad in partial fulfilment for the requirement of the award of Minor Degree Programme and it is a record of bonafide project work carried out by us under the guidance of **Ms. K. Sinduja**, Assistant Professor CSE (AIML), Department of CSE (Cyber Security)

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

Signature of the Student

Dussa Pavan

Signature of the Student

Jirra Vigneshwar

Date:

Place:

ACKNOWLEDGEMENT

We are thankful for and fortunate enough to get constant encouragement, support, and guidance from all **Teaching staff of CSE (Cyber Security) Department** which helped us in successfully completing this project work.

We thank **Dr. M. Sunitha** and **Mr. Sravan Kumar G**, Project Coordinators and Project Review Committee members: **Ms. L. Roshini**, Sr Assistant Professor, **Ms. Sunitha Rekha**, Sr. Assistant Professor for their valuable guidance and support which helped us to complete the project work successfully.

We respect and thank our internal guide, **Mr. Sravan Kumar G**, Assistant Professor, Department of CSE (Cyber Security), for giving us all the support and guidance, which made us complete the project duly.

We would like to express heartful thanks to **Dr. H. N. Lakshmi**, Professor & Head of the Department, for providing us an opportunity to do this project and extending support and guidance.

We thank our Vice-Principal **Prof**. **L. C. Siva Reddy** for providing excellent computing facilities and a disciplined atmosphere for doing our work.

We wish a deep sense of gratitude and heartfelt thanks to **Dr. Rama Mohan Reddy**, Principal, and the Management for providing excellent lab facilities and tools. Finally, we thank all those guidance helpful to us in this regard.

ABSTRACT

Wireless sensor networks (WSNs) are like a bunch of tiny smart devices that communicate wirelessly to gather and share information. Wireless sensor networks (WSNs) are widely used in many areas due to their features and effectiveness. However, they face a significant challenge – security threats, with denial-of-service (DoS) attacks being a common issue. These attacks can disrupt the normal functioning of WSNs.

A Denial-of-Service (DoS) attack is meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic or sending it information that triggers a crash. A denial-of-service (DoS) attack is a security threat that occurs when an attacker makes it impossible for legitimate users to access computer systems, network, services or other information technology (IT) resources. Attackers in these types of attacks typically flood web servers, systems or networks with traffic that overwhelms the victim's resources and makes it difficult or impossible for anyone else to access them.

This project aims to develop a system that detects DOS attacks in Wireless sensor networks (WSN) using Machine Learning. Earlier proposed approaches use Decision Trees (DT) to detect DOS attacks which provided an accuracy of about 92% but they are subjected to problems like being prone to overfitting and unstable to noise and missing data. This project aims to overcome the earlier problems using other Machine Learning Algorithms such as Naive Bayes, XGBoost, RandomForest, etc. which help to enhance the performance in the detection of DOS Attacks in WSN.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
DOS	Denial-of-Service
WSN	Wireless Sensor Networks
CSV	Comma Separated Values
ML	Machine Learning
DL	Deep Learning
IOT	Internet of Things
UI	User Interface
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
IT	Information Technology
DT	Decision Tree
CNN	Convolutional Neural Network
NLP	Natural Language Processing
DNN	Deep Neural Networks
IDS	Intruder Detection System
RNN	Recurrent Neural Network
ANN	Artificial Neural Network

1.INTRODUCTION

Wireless sensor networks (WSNs) have the potential to create a new generation of distributed systems and satisfy the requirements of many critical real-time applications. Moreover, most Internet of Things (IoT) devices are based on wireless sensor node technology because they provide an adequate communication platform. WSNs are the core and important fundamental components of the IoT [1], [2]. WSNs have a wide range of potential uses, and have recently attracted considerable attention.

WSNs are one of the most significant technologies in the twenty-first century [3]. However, these networks face many challenges compared with traditional networks. Security and energy efficiency are significant challenges [4]. In terms of security, Denial of Service (DoS),node duplication, and node damage are the main threats to WSNs [5].Using an unguided transmission medium renders WSNs more vulnerable to security attacks than other networks that use a guided transmission medium. In addition, the limits of computing power, batteries, and memory make the most common security measures inapplicable to WSNs (e.g., public key cryptography). Therefore, these types of networks require security solutions that cause a very low overhead, which is difficult to achieve.

WSNs are simple, easy, and inexpensive to implement in various critical fields, and meet the requirements of real-life applications. However, their constraints and limitations of computational capacity make WSNs highly susceptible to various types of security attacks [6], [7], particularly DoS attacks. DoS attacks are among the most frequently occurring attacks on these networks [8], [9], [10], and are very difficult to defend [11].

This project explores on WSN constraints, vulnerabilities, and attack classification. It investigates recent methods to counter DoS attacks, aiming to develop a lightweight detection method for DoS attacks in WSNs, as the author is highly motivated to introduce a solution that meets the WSN constraints. The contributions of this study can be summarized as follows.

- It provides comprehensive details on WSN constraints, vulnerabilities, and attacks classification, with a focus on DoS attacks.

• It demonstrates the insufficient and drawbacks of cryptographic methods to resolve security issues of WSNs.

• It investigates recent machine learning and deep learning methods to counter DoS attacks and evaluate their achievements and limitations.

• It develops an enhanced version of the WSN-DS dataset by applying an adequate feature selection method to improve the dataset. Evidence for this improvement has been provided through the results achieved.

• It Introduces a lightweight machine learning detection approach based on XGBoost , Naïve Bayes ,Random Forest. This approach achieved a high classification accuracy with an acceptable overhead.

• It compares the proposed detection approach with some detection approaches that have been recently presented to counter DoS attacks. The comparison results indicate that the proposed detection approach outperforms other classifiers.

• It provides a thorough discussion of the experimental results obtained.

1.1 MOTIVATION

The motivation for embarking on this project lies in the persistent threat posed by denial-of-service (DoS) attacks, which continue to challenge the security of wireless sensor networks (WSNs). Traditional methods of detecting these attacks often fall short in keeping pace with the evolving strategies of cyber attackers. As these attackers become more sophisticated, employing dynamic techniques to overwhelm WSNs, there is a pressing need for a more advanced and adaptable approach to detection.

By harnessing the power of machine learning algorithms, we aim to develop a robust system capable of effectively detecting and mitigating DoS attacks in WSNs. Building upon previous research, which has shown promise with Decision Trees but faced limitations such as overfitting and instability, our project seeks to explore alternative machine learning techniques like Naive Bayes, XGBoost, RandomForest, and others. By doing so, we hope to improve the accuracy and resilience of DoS attack detection in WSNs, ensuring the continued reliability and integrity of these networks in the face of malicious threats.

Moreover, as the importance of user privacy and data security continues to grow, it is imperative to develop models that can detect and mitigate DoS attacks based on features specific to WSNs, without compromising user privacy. By focusing on comprehensive feature selection and leveraging innovative techniques, such as image embeddings, our project aims to deliver a detection system that not only addresses the current challenges of DoS attacks but also anticipates and adapts to future threats.

1.2 OBJECTIVES

- To develop a machine learning-based system to detect denial-of-service (DoS) attacks in wireless sensor networks (WSNs).
- To Explore and implement alternative machine learning algorithms, such as Naive Bayes, XGBoost, and RandomForest, to enhance the accuracy of DoS attack detection.
- To Address the limitations of previous approaches, including overfitting and instability, to create a more robust and adaptable DoS detection system.
- To Prioritize user privacy by focusing on feature selection and innovative techniques like image embeddings for detecting and mitigating DoS attacks.
- To Improve the resilience of WSNs by ensuring the proposed system can effectively discern between legitimate network activities and malicious DoS attempts.
- To Align the project with evolving legal frameworks and user privacy concerns to create a detection system that adheres to ethical and legal standards.
- To Provide a comprehensive solution that not only tackles current DoS attack challenges but anticipates and adapts to emerging threats in the cybersecurity landscape.
- To Establish a user-friendly interface for easy monitoring, enabling non-experts to understand the status and security of wireless sensor networks in real-time.

1.3 PROBLEM STATEMENT

Wireless Sensor Networks (WSNs) face a critical security challenge due to the prevalent threat of denial-of-service (DoS) attacks. These malicious activities disrupt the normal functioning of WSNs, impacting their reliability and effectiveness. Existing detection methods, particularly those relying on Decision Trees, exhibit limitations such as overfitting and instability.

To address these issues, this project aims to develop an advanced machine learning-based system that effectively detects and mitigates DoS attacks in WSNs. By exploring alternative algorithms like Naive Bayes, XGBoost, and RandomForest, the project seeks to enhance accuracy and overcome the existing problems while prioritizing user privacy and aligning with evolving legal frameworks. The goal is to create a comprehensive solution that not only tackles current DoS attack challenges but also anticipates and adapts to emerging threats in the dynamic cybersecurity landscape.

1.4 REPORT ORGANIZATION

The report is divided into 6 chapters including the introductory page. Chapter 1 deals with introduction to this report on the Detection of Denial-of-Service Attack(DOS) in wireless sensor Networks. Chapter 2 includes the literature survey which provides insight into existing models and their limitations. Chapter 3 summarizes functional and system requirements along with software and hardware specifications. Chapter 4 shows the design and overall working of the project using various UML diagrams. It also includes the information related to all the algorithms used in this project. Chapter 5 shows the implementation of the project in neat steps which includes the results and screenshots of execution. The report ends with a conclusion at Chapter 6 along with future scope and enhancements for this project.

2.LITERATURE SURVEY

2.1 Existing Work

Existing works of Detection of Dos Attack in Wireless Sensor Network explores different approaches to detect DOS like deep learning, machine learning techniques.

2.1.1 Advancements in Wireless Sensor Network Security Using Artificial Neural Networks

Wireless Sensor Networks (WSNs) have emerged as critical components of modern information and communication systems, facilitating the collection and transmission of data in various applications. However, the proliferation of cyber threats poses significant challenges to the security and integrity of WSNs, necessitating the development of robust intrusion detection and localization techniques. This literature review explores recent advancements in WSN security, particularly focusing on the utilization of Artificial Neural Networks (ANNs) for detecting and localizing multiple attacks within WSN environments.

Recent research has demonstrated the efficacy of ANN-based intrusion detection and localization schemes in WSNs. These approaches leverage the inherent learning capabilities of ANNs to analyse sensor data, detect anomalous behaviour, and localize malicious nodes. The proposed scheme utilizes a multilayer perceptron artificial neural network (MLPANN) to achieve high detection and localization accuracy across multiple benchmark datasets, including UNSW-NB, WSN-DS, NSL-KDD, and CICIDS2018.

The proposed MLPANN scheme exhibits notable contributions to WSN security:

- High Detection Accuracy: The scheme achieves impressive detection accuracy rates across various benchmark datasets, underscoring its effectiveness in identifying malicious nodes within WSNs.
- Optimized Localization: The localization approach outperforms traditional distance vector hop techniques, demonstrating superior accuracy in pinpointing the location of malicious nodes within the network.

- Validation and Evaluation: Rigorous validation and evaluation of the proposed scheme using industry-standard tools and methodologies confirm its efficacy and applicability in real-world WSN scenarios.

Limitations:

- ANNs, especially deep learning models, often require significant computational resources for training and inference. This can pose challenges in resource-constrained WSN nodes with limited processing power, memory, and energy supply.
- ANNs typically require large amounts of labelled training data to learn complex patterns and generalize well to unseen data. dynamic and diverse nature of network traffic and security threats.
- Imbalanced datasets, where the number of normal instances significantly outweighs the number of anomaly instances, can bias the learning process and result in poor performance in detecting rare security events.
- Training and inference processes in ANNs can consume significant energy, which is a critical concern in battery-powered WSN nodes.

2.1.2 Deep Learning Techniques for DoS Attacks Detection in Wireless Sensor Networks

WSNs play a crucial role in diverse applications, including healthcare, telecommunications, environmental monitoring, and IoT. Despite their simplicity and cost-effectiveness, WSNs face significant security challenges due to resource constraints and deployment in hostile environments. DoS attacks pose a serious threat to WSNs by depleting node resources and disrupting network operations.

Recent studies have investigated the application of DL techniques, including Dense Neural Networks (DNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and hybrid architectures, for detecting DoS attacks in WSNs. These approaches typically involve training DL models on specialized datasets containing instances of DoS attacks, such as the WSN-DS dataset, which encompasses various DoS attack types, including Blackhole, Grayhole, Scheduling, and Flooding attacks.

The proposed solution demonstrates several key contributions:

- Experimentation with DL-Based IDSs: Researchers have developed and evaluated multiple DL-based IDSs for DoS detection in WSNs, aiming to achieve high detection accuracy while minimizing computational complexity.
- Comparative Evaluation: Studies have compared the performance of different DL architectures, such as DNNs, CNNs, RNNs, and hybrid models, in terms of accuracy, robustness, and efficiency.
- Future Directions: The literature suggests future research directions, including feature selection techniques to enhance model efficiency and scalability, as well as optimization of training parameters to improve detection performance.

Limitations :

- Training Time and Epoch Selection: Longer training times and increased epochs may improve model performance but require careful consideration of computational resources and convergence.
- Feature Selection and Dataset Size: Feature selection techniques can help reduce the size of training data and enhance model efficiency, but the selection process requires domain expertise and experimentation.

2.1.3 Enhancing Intrusion Detection Systems with Machine Learning Techniques

Intrusion Detection Systems serve as proactive defense mechanisms, identifying and classifying intrusions or policy violations within network and host infrastructures. NIDS and HIDS represent two primary categories of IDS, each leveraging distinct methodologies to analyse and detect malicious activities.

Recent research has increasingly focused on harnessing the power of machine learning algorithms to augment intrusion detection capabilities. While classical machine learning classifiers have shown promise, recent advancements in deep learning, particularly DNNs, offer significant potential for improved detection accuracy and scalability.

Several noteworthy contributions have been made in the realm of intrusion detection research:

Exploration of DNNs: A comprehensive analysis of the performance of DNNs in detecting and classifying cyberattacks across various publicly available benchmark datasets. This includes evaluation on datasets such as KDDCup 99, NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017.

Hybrid IDS Framework: Introduction of a scalable hybrid intrusion detection framework, known as SHIA, capable of processing large volumes of network and host-level events in real-time. This framework combines classical machine learning algorithms with DNNs to achieve enhanced detection capabilities.

Advanced Text Representation Techniques: Investigation into natural language processing (NLP) methods to capture contextual and semantic similarities in host-level events, thereby improving the accuracy of intrusion detection.

In summary, while existing research has made significant strides in enhancing IDS with machine learning techniques, our project seeks to advance this field further by focusing on the specific challenge of DoS attack detection in WSNs and evaluating a diverse range of machine learning algorithms for optimal performance. Through comprehensive experimentation and systematic evaluation, we aim to develop a flexible and effective IDS capable of detecting and mitigating unforeseen cyber threats in real-world scenarios.

Limitations:

- Machine learning models may become overly complex and fit too closely to the training data, resulting in poor generalization to unseen data.
- In real-world scenarios, the distribution of normal and anomalous data may be highly imbalanced, with a much larger proportion of normal data compared to intrusion data.
- Selecting relevant features from high-dimensional data can be challenging and may require domain expertise.
- Intruders may intentionally manipulate or evade IDS by exploiting vulnerabilities in the machine learning models themselves.

Method	**Accuracy**	**Key Models**
Advancements in Wireless Sensor Network Security Using Artificial Neural Networks	99.1%	Artificial Neural Networks
Deep Learning Techniques for DoS Attacks Detection in Wireless Sensor Networks	98.79%	CNN+RNN, DNN
Enhancing Intrusion Detection Systems with Machine Learning Techniques	99.3%	KNN, DT, LR
Proposed Method	99.76%	Naïve Bayes, XGBoost, Random Forest

Table 1: Comparison Results of Existing vs Proposed

3.SOFTWARE AND HARDWARE REQUIREMENTS

Software and Hardware requirements are the most important part of any project. Hardware requirements specify the underlying hardware required for the system to execute. Software requirements specify all the software and the packages required for the execution of the program. Therefore, these requirements must be met before we start developing the project.

3.1 HARDWARE REQUIREMENTS

Laptop or PC with the following specifications:

- **RAM**: 8GB is recommended or higher
- **Processor**: Intel Core i3, Recommended i5 or i7, clock speed of at least 2.4 GHz
- **ROM**: 128GB or higher
- **Graphic Card**: Not Required

3.2 SOFTWARE REQUIREMENTS

- **Operating System**: Windows/Mac
- **Execution Environment**: Anaconda - Jupyter Notebook, VS Code
- **Programming Language**: Python 3.0 and above
- **Libraries**: Scikit-learn, Pandas, Matplotlib, Seaborn, NumPy, Tkinter.

4.PROPOSED SYSTEM DESIGN

4.1 PROPOSED SOLUTION

The goal of this project is to accurately detect denial of service attack in wireless sensor networks while solving many of the limitations of existing systems. The methodology to achieve this goal is described in brief below.

Methodology Overview:

Feature Extraction: Utilize NLP preprocessing techniques to extract meaningful features from the data, focusing on textual components associated with WSN activities.

Model Development: Implement various Machine Learning algorithms including Naive Bayes, XGBoost, Random Forest, etc., to build predictive models capable of identifying patterns indicative of DoS attacks in WSNs.

Model Evaluation: Assess the performance of each model using metrics such as accuracy, precision, recall, and F1-score to determine the most effective algorithm for DoS attack detection in WSNs.

Comparison with Previous Approaches: Conduct a comparative analysis with earlier proposed methods, particularly those utilizing Decision Trees, to highlight the improvements achieved through the adoption of alternative Machine Learning algorithms.

Detailed Approach:

Feature Engineering: Extract relevant features from the WSN dataset, considering parameters such as packet loss, latency, and network congestion.

Algorithm Selection: Experiment with various Machine Learning algorithms, including Naive Bayes, XGBoost, Random Forest, etc., to identify the algorithm(s) yielding the highest detection accuracy.

Model Training and Validation: Divide the dataset into training and validation sets to train the selected models. Employ cross-validation techniques to ensure robustness and generalizability.

Performance Evaluation: Evaluate the trained models using appropriate metrics and compare the results with the baseline accuracy achieved by earlier approaches utilizing Decision Trees.

Optimization and Fine-Tuning: Fine-tune hyperparameters and explore ensemble methods to optimize the chosen algorithm(s) further.

Implementation: Develop a scalable and efficient system capable of real-time detection of DoS attacks in WSNs, integrating the most effective Machine Learning algorithm(s) identified during the experimentation phase.

Expected Outcomes:

Enhanced accuracy in the detection of DoS attacks in WSNs compared to previous approaches. Identification of the most suitable Machine Learning algorithm(s) for DoS attack detection in WSN environments. Development of a scalable and adaptable system capable of addressing evolving threats in WSNs.

This project aims to address the challenge of detecting DoS attacks in Wireless Sensor Networks by leveraging the capabilities of Machine Learning algorithms. Through systematic experimentation and evaluation, the project endeavors to enhance the accuracy and reliability of DoS attack detection, thereby contributing to the security and resilience of WSN deployments.

4.2 USE CASE DIAGRAM

A use case diagram shows the dynamic aspect of the system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. The below figure depicts the high-level functionality of a system and tells how the user handles a system.

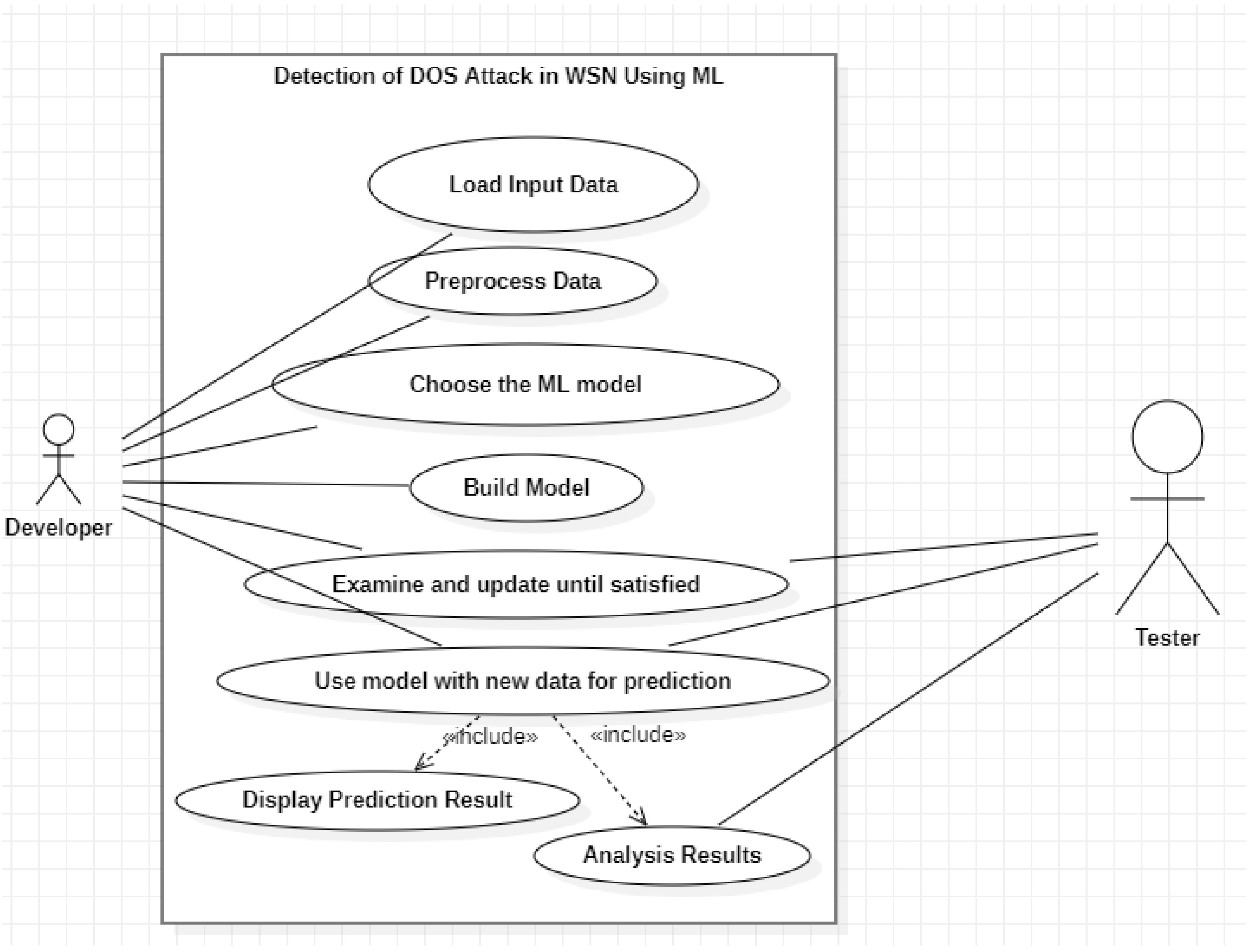

Detection of DOS Attack in WSN Using ML
Load Input Data
Preprocess Data
Choose the ML model
Build Model
Examine and update until satisfied
Use model with new data for prediction
«include»
«include»
Display Prediction Result
Analysis Results
Developer
Tester

Fig 4.2.1: Use Case Diagram

4.3 CLASS DIAGRAM

A class diagram is a graphical representation that illustrates the structure and relationships among classes in a system. It provides a static view of the system, highlighting the classes, their attributes, and the associations between them. In a class diagram, classes are depicted as rectangles, with the class name at the top, attributes in the middle, and methods at the bottom. Relationships between classes, such as associations, generalizations, or dependencies, are represented by connecting lines.

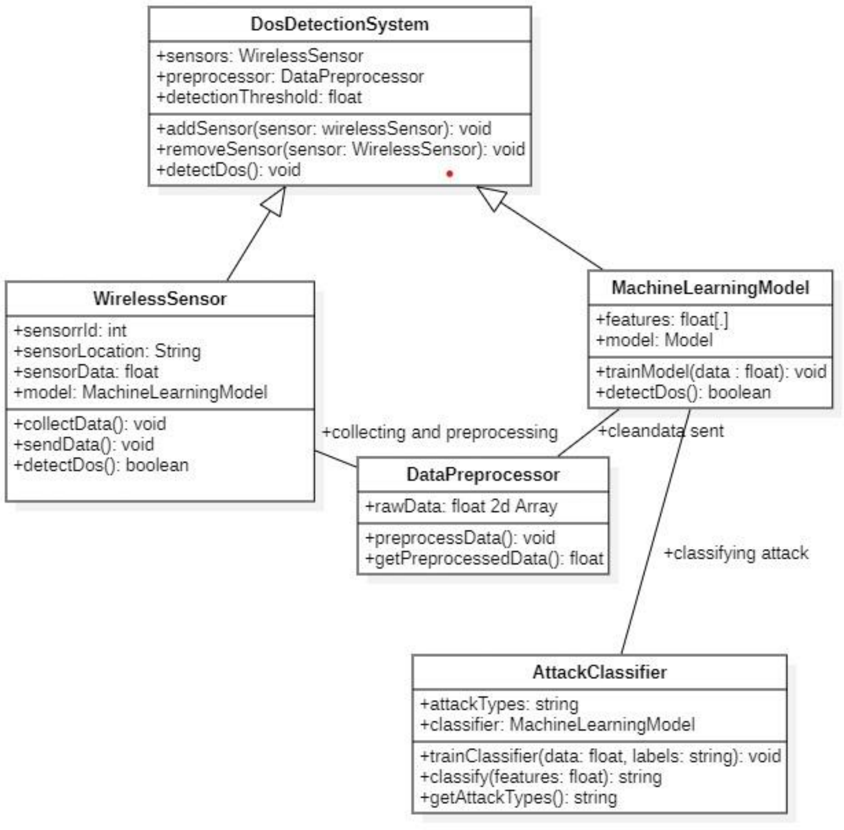

DosDetectionSystem
+sensors: WirelessSensor
+preprocessor: DataPreprocessor
+detectionThreshold: float
+addSensor(sensor: wirelessSensor): void
+removeSensor(sensor: WirelessSensor): void
+detectDos(): void
WirelessSensor
+sensorrId: int
+sensorLocation: String
+sensorData: float
+model: MachineLearningModel
+collectData(): void
+sendData(): void
+detectDos(): boolean
MachineLearningModel
+features: float[.]
+model: Model
+trainModel(data : float): void
+detectDos(): boolean
+collecting and preprocessing
+cleandata sent
DataPreprocessor
+rawData: float 2d Array
+preprocessData(): void
+getPreprocessedData(): float
+classifying attack
AttackClassifier
+attackTypes: string
+classifier: MachineLearningModel
+trainClassifier(data: float, labels: string): void
+classify(features: float): string
+getAttackTypes(): string

Fig 4.5.1 Class Diagram

4.4 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram that visualizes the interactions between objects or components in a system over time. It shows the flow of messages or method calls between these objects along a timeline, illustrating the order of execution. Typically, actors or objects are represented as lifelines, and messages are depicted as arrows between them.

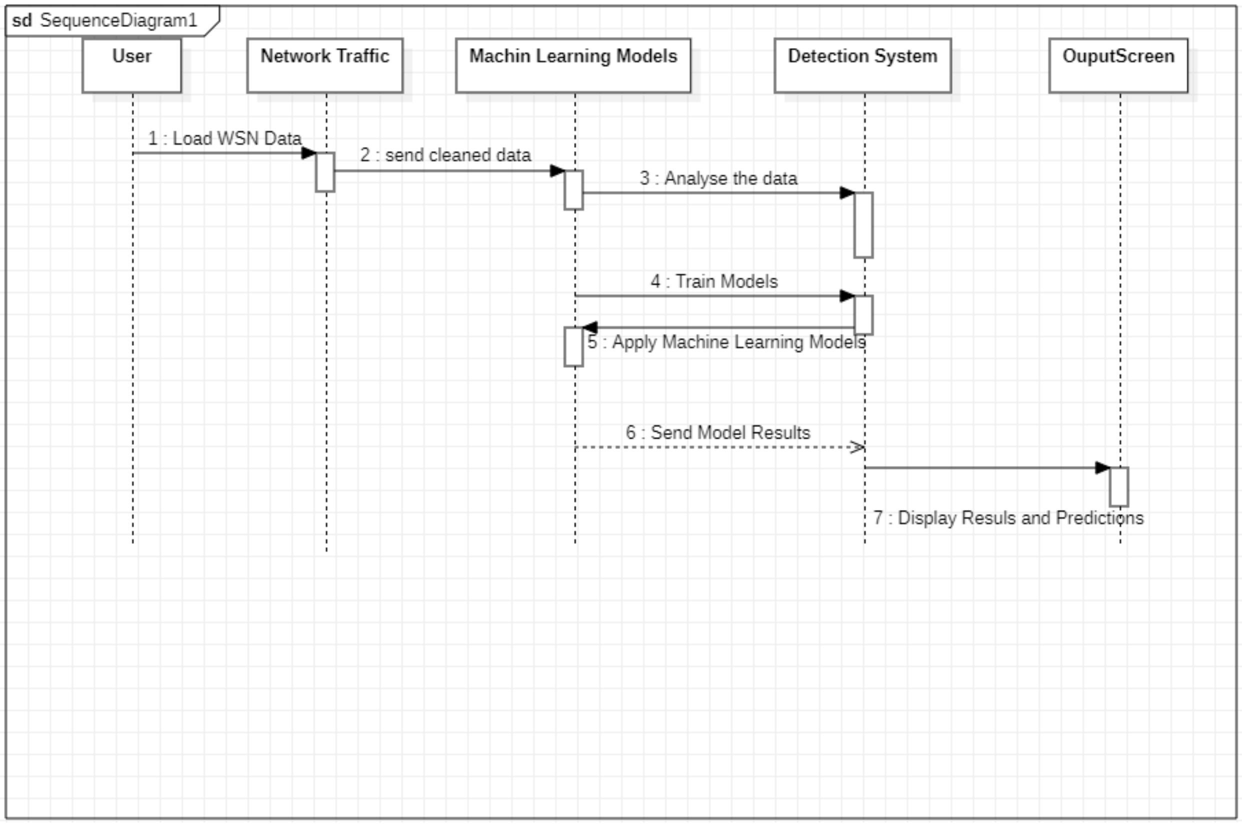


Fig 4.4.1 Sequence Diagram

4.5 ACTIVITY DIAGRAM

An activity diagram is a visual representation of a system's workflow or business process, depicting the sequence of activities and actions performed within that system. It helps illustrate the dynamic aspects of a process, showing the flow of control from one activity to another.

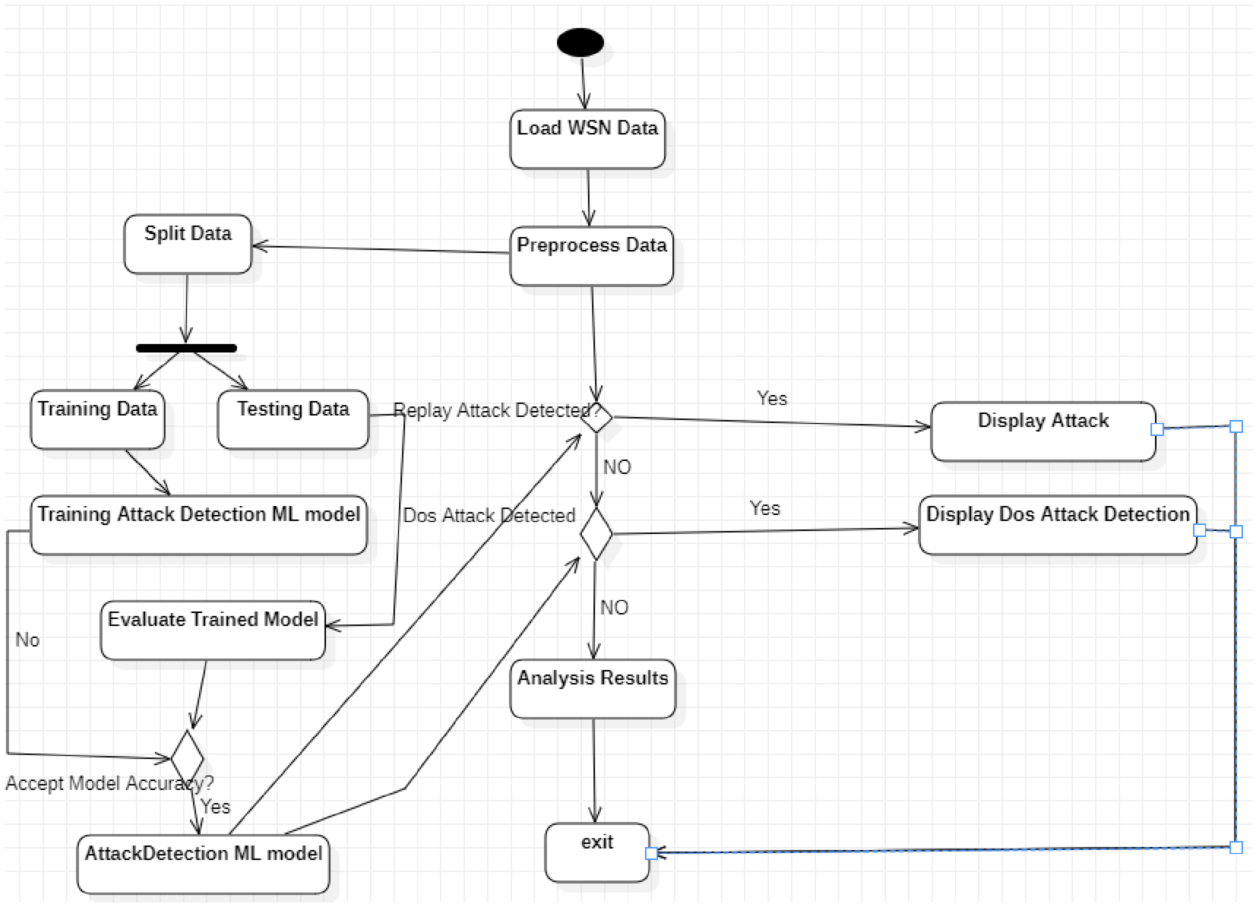


Fig 4.3.1: Activity Diagram

4.6 SYSTEM ARCHITECTURE

The diagram below shows a three-level architecture diagram for the system. At the very core, we have the datasets which are used for our project. Upon initializing the tkinter app, we load the required datasets for comparisons and computations. These datasets are used for list-based comparison, to fine tune our ML model as well as to calculate visual similarity score. At the top level is our Tkinter app that provides a user-friendly dashboard.

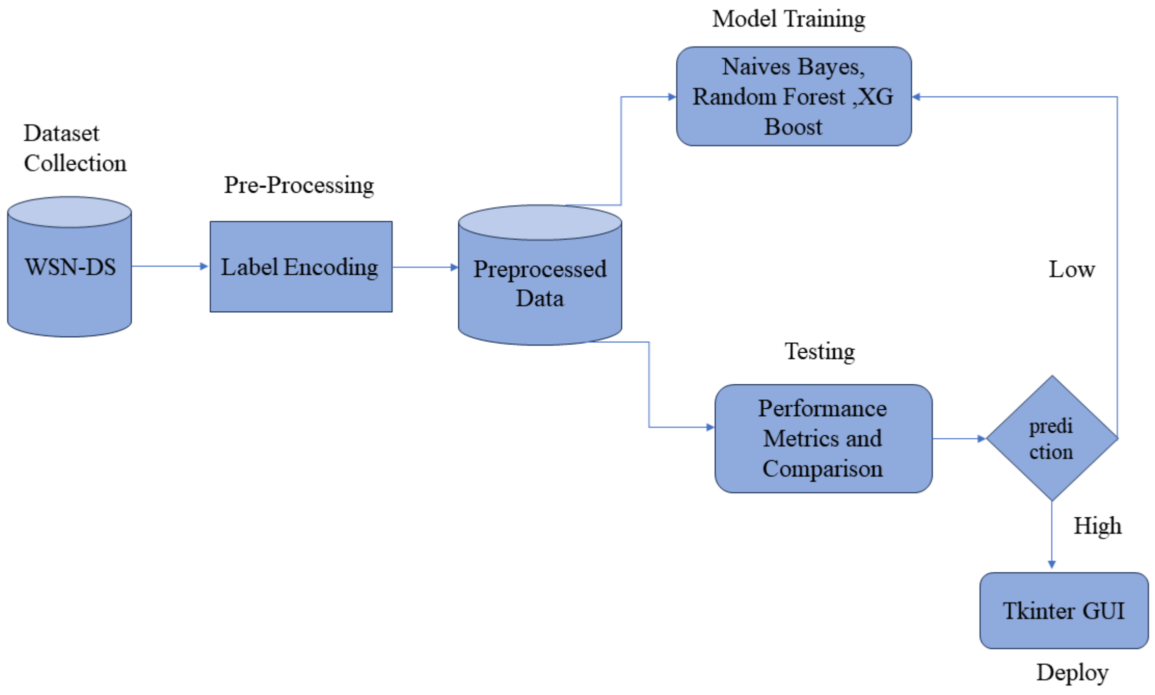


Fig 4.6.1 System Architecture

4.7 TECHNOLOGY DESCRIPTION

Scikit-Learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. Scikit-learn is extensively used for high-performance linear algebra and array operations. Furthermore, some core algorithms help in improving performance.

Pandas

Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. Pandas is fast and it has high performance & productivity for users. Pandas is built on top of two core Python libraries—matplotlib for data visualization and NumPy for mathematical operations. Pandas introduced two new types of objects for storing data that make analytical tasks easier and eliminate the need to switch tools: Series, which have a list-like structure, and Data Frames, which have a tabular structure.

Seaborn

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and closely integrated to the data structures from pandas. Seaborn aims to make visualization the central part of exploring and understanding data.

Matplotlib

`Matplotlib` is a robust and widely-used Python library designed for creating static, animated, and interactive visualizations. Established in 2003 by John D. Hunter, Matplotlib has become a cornerstone in the field of data visualization. Known for its versatility, it offers an extensive array of plotting functions and a simplified interface through the `pyplot` module, making it accessible for both beginners and experienced users. Matplotlib provides users with precise control over the customization of plots, allowing for the creation of diverse chart types and styles. With a strong community and a wealth of resources, Matplotlib stands as a go-to solution for generating high-quality visualizations in Python.

Tkinter

`Tkinter` is a standard Python library for creating graphical user interfaces (GUIs) with a simple and intuitive structure. It serves as the default GUI toolkit for Python and is included with most Python installations. Tkinter is based on the Tk GUI toolkit and provides a convenient way to build windows, dialogs, buttons, and other GUI elements. Its ease of use makes it a popular choice for beginners learning GUI programming in Python. Tkinter allows developers to design visually appealing applications with a straightforward approach, utilizing features like event-driven programming to respond to user interactions.

NumPy

`NumPy` is a powerful and fundamental Python library for numerical and scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays. NumPy forms the foundation for many other scientific computing libraries in Python and is a key component in the Python data science ecosystem. Introduced in 2005, NumPy enhances computational efficiency by executing operations in compiled code, making it significantly faster than equivalent operations in pure Python. NumPy's versatile array operations, broadcasting capabilities, and efficient memory management make it essential for tasks ranging from mathematical calculations to data manipulation in fields such as machine learning, statistics, and engineering. Its widespread adoption and continuous development underscore its importance in the Python programming landscape.

5.IMPLEMENTATION AND TESTING

The project is briefly described as mentioned below:

- Dataset Exploration.
- Algorithm Implementation
- Visualization
- Front-End and Real Time System Implementation
- Evaluation

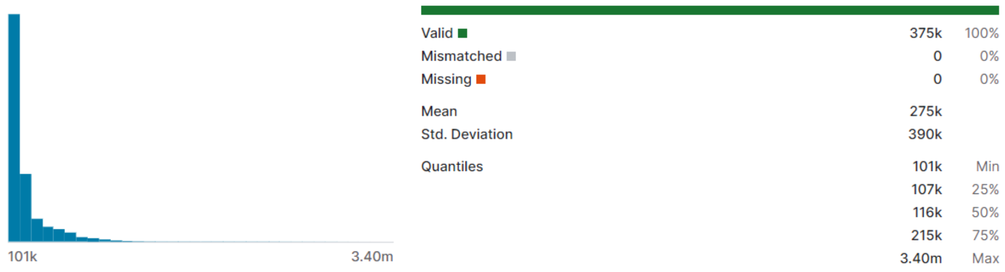
5.1 Dataset Exploration

The WSN-DS dataset is a specialized collection designed for intrusion detection systems (IDS) in Wireless Sensor Networks (WSNs). It comprises **19** features and **374,661** records, representing various types of **Denial of Service** (DoS) attacks. These attacks include the likes of **Blackhole**, **Grayhole**, **Flooding**, and **Scheduling** attacks. Researchers and practitioners can utilize this dataset to enhance the effectiveness of IDS in detecting and classifying different DoS attacks within WSNs.

Name of the File: WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks

The 19 features of the data set are described below:

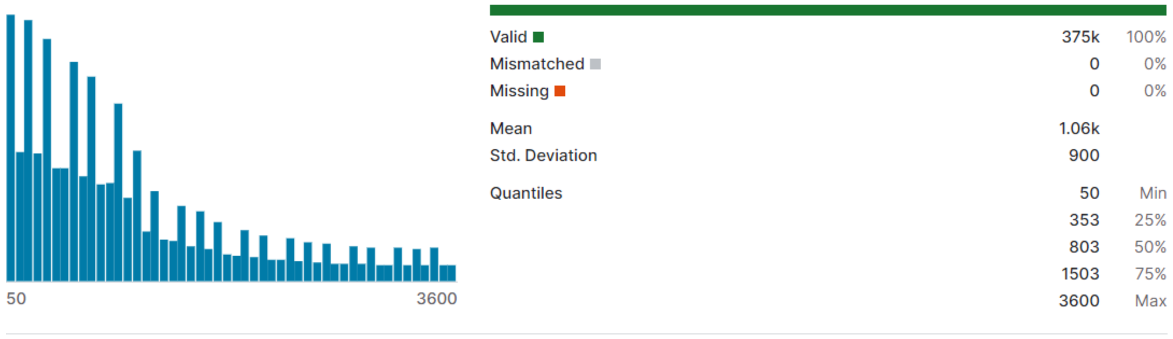
- **Id**.
  Node Id: The identification number or code assigned to each sensor node can help in tracking and identifying anomalous behaviour or nodes involved in the attack. A unique ID to distinguish the sensor node in any round and at any stage. For example, node number 25 in the third round and in the first stage is to be symbolized as 001 003 025.Each sensor node has a unique identifier. In the case of a DoS attack, monitoring the behaviour of individual nodes can help identify if any specific nodes are behaving anomalously or are being compromised to participate in the attack.



Valid 🟩	375k	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	275k	
Std. Deviation	390k	
Quantiles	101k	Min
	107k	25%
	116k	50%
	215k	75%
	3.40m	Max

Fig 5.1.1 Node id and value count

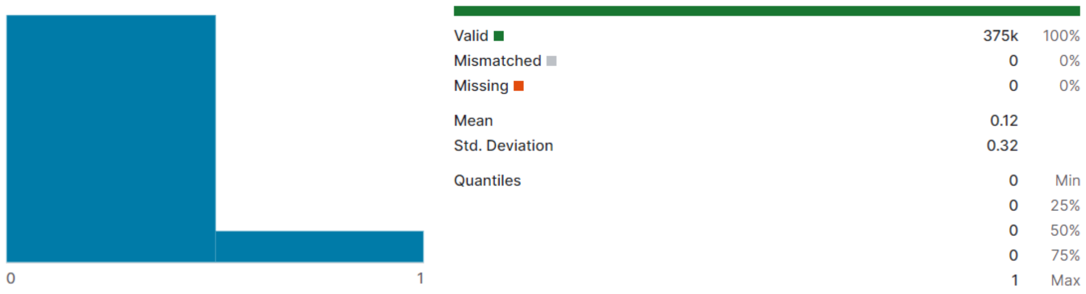
- **Time**: The current simulation time of the node. Timestamp indicating the time at which the data was collected or an event occurred. Time stamps can be crucial for identifying sudden spikes or abnormal patterns in network traffic. Unusually high traffic at specific times could be indicative of a DoS attack.



Valid 🟩	375k	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	1.06k	
Std. Deviation	900	
Quantiles	50	Min
	353	25%
	803	50%
	1503	75%
	3600	Max

Fig 5.1.2 Time stamp

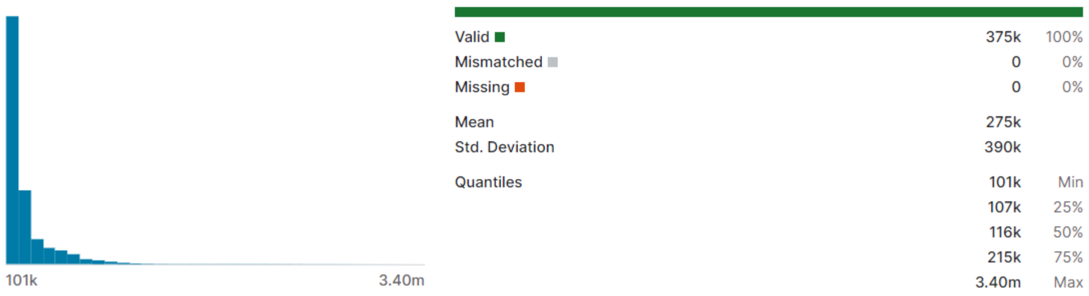
- **IS_CH** (Is Cluster Head): A flag to distinguish whether the node is CH with value 1 or normal node with value 0. A binary indicator (typically 0 or 1) that signifies whether the node is acting as a Cluster Head or not. Cluster Heads play a significant role in managing communication within their respective clusters. An attack could target these nodes to disrupt cluster operations, leading to a denial of service for the entire cluster.



Valid 🟩	375k	100%
Mismatched ⬜	0	0%
Missing 🟥	0	0%
Mean	0.12	
Std. Deviation	0.32	
Quantiles	0	Min
	0	25%
	0	50%
	0	75%
	1	Max

Fig 5.1.3 Is cluster Head count

- **who CH (who Cluster Head):** The ID of the CH in the current round. Indicates the identity or unique identifier of the Cluster Head if the node is one.

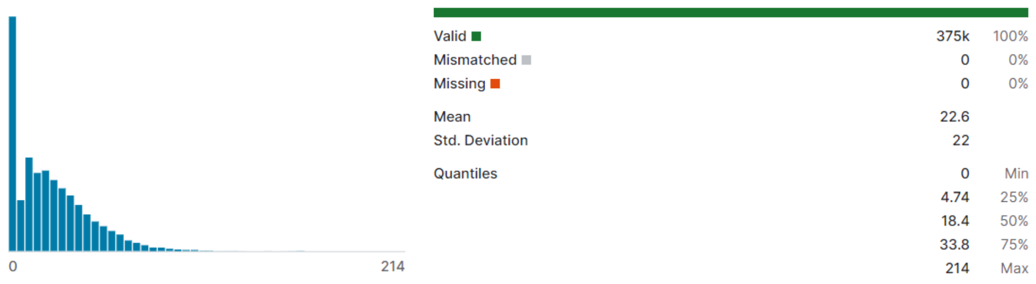


Valid 🟩	375k	100%
Mismatched ⬜	0	0%
Missing 🟥	0	0%
Mean	275k	
Std. Deviation	390k	
Quantiles	101k	Min
	107k	25%
	116k	50%
	215k	75%
	3.40m	Max

Fig 5.1.4 who cluster Head value count

- **Dist_To_CH (Distance to Cluster Head):** The distance between the node and its CH in the current round. The distance between a sensor node and its associated Cluster Head. This metric is crucial for routing and data aggregation purposes. Monitoring the distance of nodes to Cluster Heads can help detect if certain nodes are being overwhelmed with traffic, potentially indicating a DoS attack.

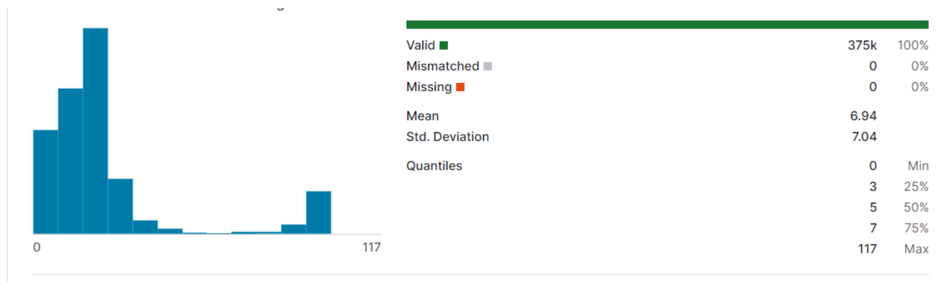
- **ADV_S (Advertisement from Sensor):** The number of advertise CH's broadcast messages sent to the nodes. Information sent by sensor nodes to advertise their presence or status. Anomalies in advertisement messages exchanged between nodes can be indicative of malicious activity, such as flooding the network with fake advertisement messages to disrupt communication.
- **ADV_R (Advertisement from Receiver):** The number of advertise CH messages received from CHs. Information sent by potential receivers to announce their availability or requirements. Anomalies in signalling messages exchanged between nodes can be indicative of malicious activity, such as flooding the network with fake advertisement messages to disrupt communication.



Valid 🟩	375k	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	22.6	
Std. Deviation	22	
Quantiles	0	Min
	4.74	25%
	18.4	50%
	33.8	75%
	214	Max

Fig 5.1.5 Adverstiment from the receiver

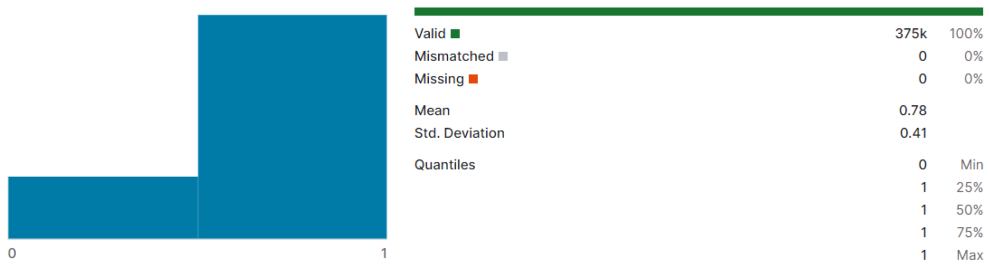
- **JOIN_S (Joining Signal from Sensor):** The number of join request messages sent by the nodes to the CH. Signal sent by a sensor node to request joining a cluster or network. The "JOIN_S" field indicates the count of join request messages transmitted by sensor nodes seeking to join a cluster or network, particularly addressed to the Cluster Head (CH).



Valid 🟩	375k	100%
Mismatched ▨	0	0%
Missing 🟥	0	0%
Mean	6.94	
Std. Deviation	7.04	
Quantiles	0	Min
	3	25%
	5	50%
	7	75%
	117	Max

Fig 5.1.6 Joining request messages count

- **JOIN_R (Joining Signal from Receiver):** The number of join request messages received by the CH from the nodes. Signal sent by a receiver to indicate its willingness to accept new nodes or join a network.
- **SCH_S (Scheduling Signal from Sensor):** The number of advertise TDMA schedule broadcast messages sent to the nodes. Signal sent by a sensor node to synchronize its activities or schedule with other nodes or the network.



Valid 🟩	375k	100%
Mismatched ⬜	0	0%
Missing 🟥	0	0%
Mean	0.78	
Std. Deviation	0.41	
Quantiles	0	Min
	1	25%
	1	50%
	1	75%
	1	Max

Fig 5.1.7 Number of TDMA broadcast message sent

- **SCH_R (Scheduling Signal from Receiver):** The number of TDMA schedule messages received from CHs. Signal sent by a receiver to synchronize or coordinate activities with the network or other nodes
- **Rank:** The order of this node within the TDMA schedule. A measure or value indicating the importance, priority, or capability of a node or Cluster Head within the network.
- **DATA_S (Data from Sensor):** The number of data packets sent from a sensor to its CH. Actual data generated or collected by sensor nodes. Abnormal patterns in data transmission, such as sudden spikes or a significant increase in data volume, can signal a DoS attack aimed at overwhelming the network bandwidth or consuming resources.
- **DATA_R (Data from Receiver)**: Data received or collected by receiver nodes. Abnormal patterns in data transmission, such as sudden spikes or a significant increase in data volume, can signal a DoS attack aimed at overwhelming the network bandwidth or consuming resources.
- **Data_Sent_To_BS (Data Sent to Base Station):** The number of data packets sent to the BS. Data transmitted to the base station for processing or analysis. Disruption in data transmission to the base station or an unusual increase in distance between

Cluster Heads and the base station could indicate an attack targeting the communication infrastructure of the WSN.

- **dist_CH_To_BS (Distance from Cluster Head to Base Station):** The distance between the CH and the BS. Distance between the Cluster Head and the Base Station
- **send_code:** The cluster sending code. A code or identifier associated with the method or protocol used for sending data or signals.
- **Expaned Energy:** The amount of energy consumed in the previous round. Energy consumed or expended by a node for various operations or communications within the network.
- **Attack type:** It is a class of five possible values, namely, Blackhole, Grayhole, Flooding, and TDMA (Scheduling), in addition to normal, if the node is not an attacker.

  - **Blackhole**: DDoS blackhole routing/filtering (sometimes called blackholing), is a countermeasure to mitigate a DDoS attack in which network traffic is routed into a "black hole," and is lost. (10049)
  - **Grayhole**: Gray Hole Attack is an advanced transformation of black hole attack. Both of them are a common type of attack in Wireless Sensor Network (WSN). Malicious nodes may constantly or randomly drop packets and therefore reduce the efficiency of the networking system. (14596)
  - **Flooding**: A flooding attack is a malicious activity where an attacker overwhelms a network or system with a massive volume of unnecessary and often fabricated data packets. The goal is to consume network resources, causing congestion, disrupting normal communication, and potentially leading to a denial of service (DoS) situation. Flooding attacks exploit vulnerabilities in the network's capacity to handle excessive traffic, impacting its functionality and performance. (3312)
  - **TDMA Flooding**: TDMA (Time Division Multiple Access) flooding refers to a type of network attack where an adversary inundates the communication channels in a TDMA-based system with an excessive number of transmissions. (6638)

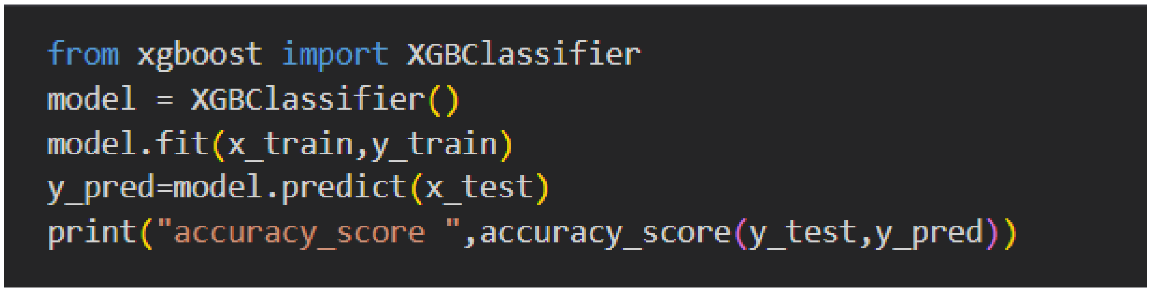
5.2 Algorithm Implementation

An algorithm is a step-by-step set of instructions or rules designed to solve a specific problem or perform a particular task. It's like a recipe that guides a computer to accomplish a task efficiently. Algorithms are essential in computer science for solving problems, making decisions, and automating processes, forming the backbone of various technologies we use daily.

In our project, we used three different methods (algorithms) to train the data how to recognize and deal with denial-of-service attacks in wireless sensor networks. By testing these algorithms with a dataset, we checked if they work well and can effectively identify and handle these attacks, ensuring the security of the sensor networks.

The algorithms implemented are XGBoost, Gaussian Naive Bayes, Random Forest.

XGBoost: XGBoost, short for extreme Gradient Boosting, is a powerful and efficient machine learning algorithm used for supervised learning tasks, especially in predictive modeling and classification problems. It belongs to the gradient boosting family and combines the strengths of decision trees with a boosting technique, enhancing predictive accuracy and computational efficiency. XGBoost is widely employed in various domains due to its ability to handle complex relationships in data, mitigate overfitting, and provide robust predictions. Its popularity in data science competitions and real-world applications is attributed to its versatility, speed, and superior performance in diverse machine learning tasks.

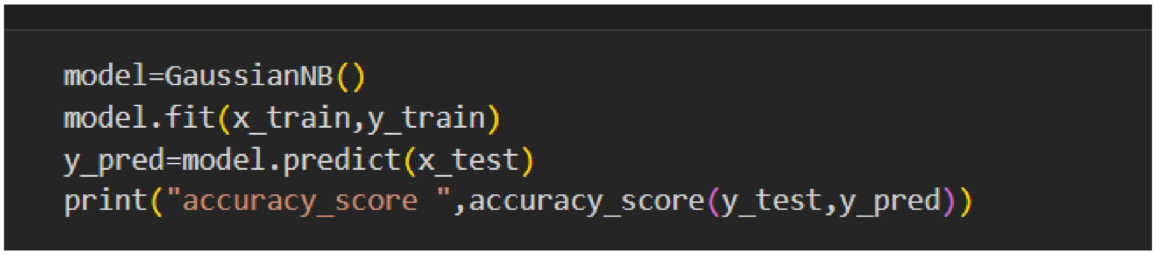


```
from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print("accuracy_score ",accuracy_score(y_test,y_pred))
```

Fig 5.2.1 XGBoost Implementation

XGBClassifier is a classification model implemented in the XGBoost library, specifically designed for tasks where the goal is to categorize data into different classes or groups.

NaiveBayes: Naive Bayes is a simple yet effective probabilistic classification algorithm used in machine learning. It's based on Bayes' theorem and assumes that features are independent, which simplifies calculations. Naive Bayes is often used for text classification, spam filtering, and sentiment analysis due to its efficiency, ease of implementation, and ability to handle high-dimensional data.

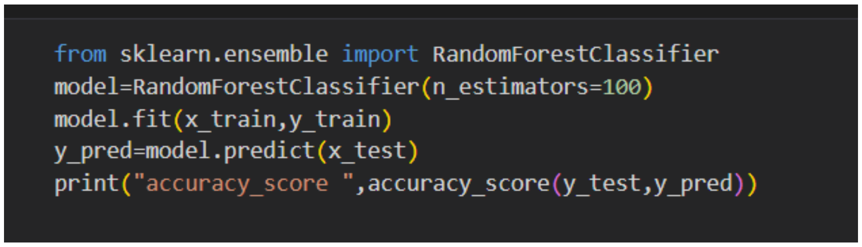


```
model=GaussianNB()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print("accuracy_score ",accuracy_score(y_test,y_pred))
```

Fig.5.2.2 Naïve Bayes Implementation

GaussianNB refers to the Gaussian Naive Bayes classifier, a specific implementation of the Naive Bayes algorithm designed for datasets where the features follow a Gaussian (normal) distribution. This classifier is commonly used for classification tasks, particularly when dealing with continuous or real-valued data.

Random Forest: Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their predictions for more accurate and robust results. It operates by introducing randomness during the tree-building process, using different subsets of data and features. This technique enhances the model's generalization ability, making Random Forest a popular and powerful tool for classification and regression tasks in machine learning.



```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=100)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print("accuracy_score ",accuracy_score(y_test,y_pred))
```

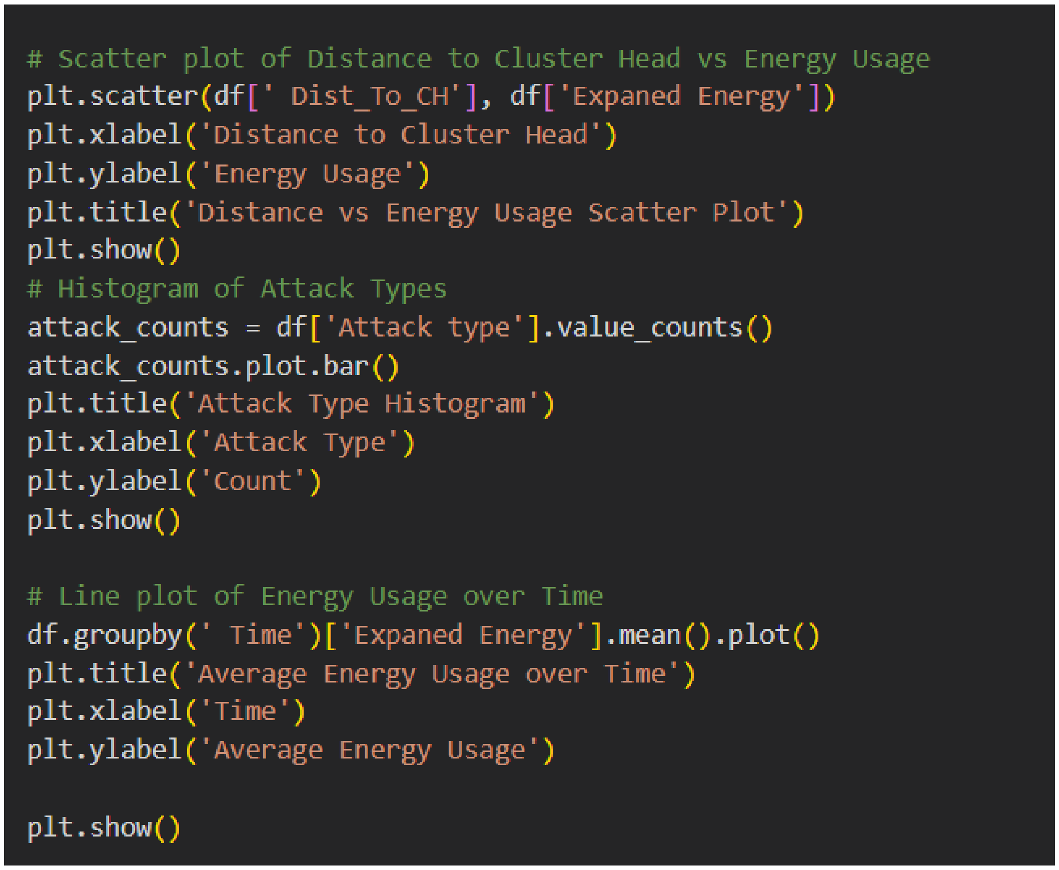
Fig.5.2.3 Random Forest Implementation

RandomForestClassifier is an implementation of the Random Forest algorithm for classification tasks in machine learning.

5.3 Visualization

Visualization is integral to machine learning as it provides a visual understanding of data patterns, aids in model evaluation, and facilitates effective communication of results. Graphical representations help practitioners explore data relationships, interpret complex models, and choose appropriate algorithms. Visualizations such as confusion matrices and ROC curves enhance model evaluation, while clear visuals also enable non-technical stakeholders to comprehend and make informed decisions based on machine learning outcomes. Ultimately, visualization plays a pivotal role in improving algorithm implementation, ensuring robust models and successful real-world applications.

In the project the visualization as represented most of the attributes showing their usage at different levels and different time stamps.



```
# Scatter plot of Distance to Cluster Head vs Energy Usage
plt.scatter(df[' Dist_To_CH'], df['Expaned Energy'])
plt.xlabel('Distance to Cluster Head')
plt.ylabel('Energy Usage')
plt.title('Distance vs Energy Usage Scatter Plot')
plt.show()
# Histogram of Attack Types
attack_counts = df['Attack type'].value_counts()
attack_counts.plot.bar()
plt.title('Attack Type Histogram')
plt.xlabel('Attack Type')
plt.ylabel('Count')
plt.show()

# Line plot of Energy Usage over Time
df.groupby(' Time')['Expaned Energy'].mean().plot()
plt.title('Average Energy Usage over Time')
plt.xlabel('Time')
plt.ylabel('Average Energy Usage')

plt.show()
```

Fig 5.3.1 Code Snippet for visualization

Scatter Plot:

- The first part of the code creates a scatter plot using `plt.scatter()`, where `df['Dist_To_CH']` represents the x-axis values (Distance to Cluster Head) and `df['Expaned Energy']` represents the y-axis values (Energy Usage).
- The `plt.xlabel()`, `plt.ylabel()`, and `plt.title()` functions set labels and title for the x-axis, y-axis, and the plot respectively.
- Finally, `plt.show()` displays the scatter plot.

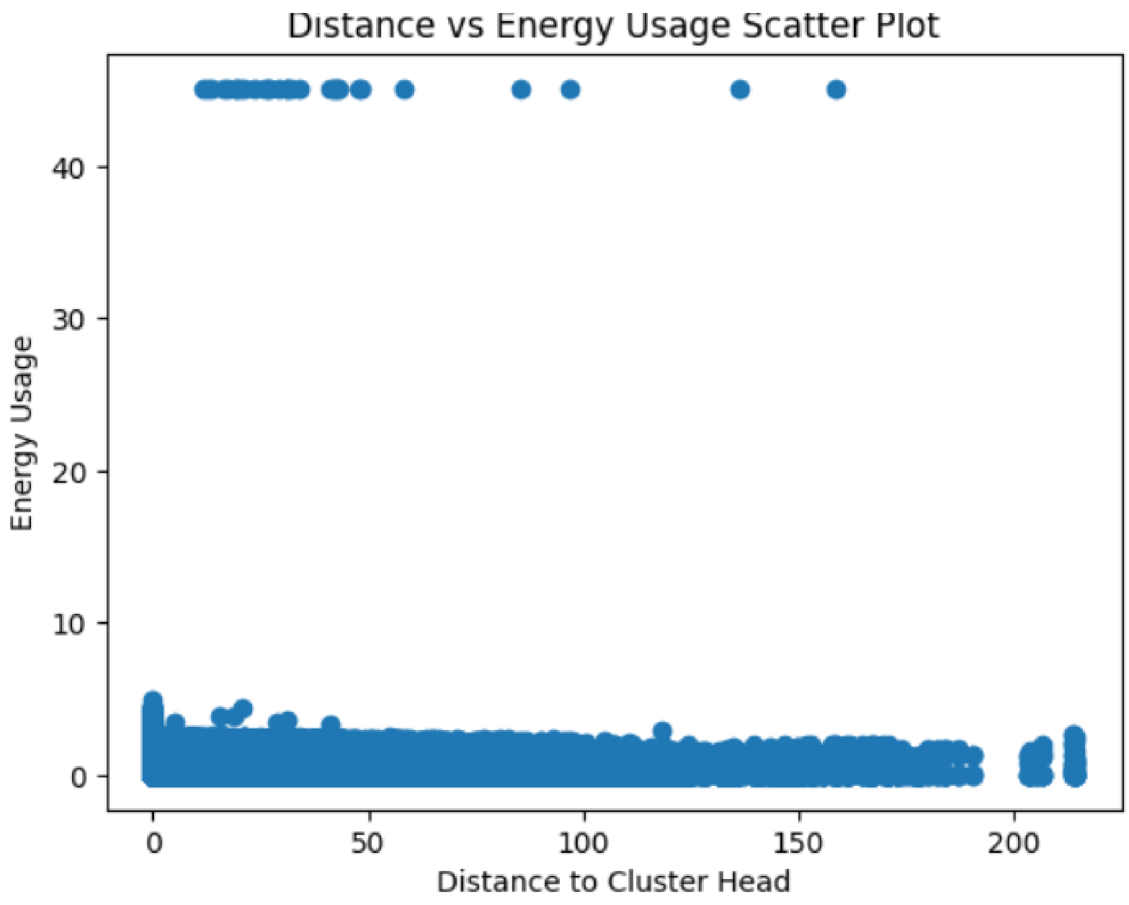


Fig 5.3.2 Visualization of distance to cluster head vs Energy Usage

The scatter plot depicting the relationship between the distance to the cluster head and energy usage is instrumental in detecting Denial of Service (DoS) attacks in Wireless Sensor Networks (WSNs). By visually assessing this plot, analysts can identify anomalies in energy consumption patterns that may indicate malicious activity. For instance, abrupt increases in energy usage at certain distances from the cluster head could signal the presence of DoS attacks, such as Blackhole or Grayhole attacks, where compromised nodes consume excessive energy resources, disrupting network functionality and communication. Understanding the correlation between distance and energy usage is crucial for pinpointing potential attack points and implementing targeted defense mechanisms to mitigate their impact effectively.

Histogram of Attack Types:

- The second part of the code generates a histogram of attack types using `df['Attack type'].value_counts()`, which counts the occurrences of each attack type in the dataframe.
- The `plot.bar()` function creates a bar plot based on the attack type counts.
- `plt.title()`, `plt.xlabel()`, and `plt.ylabel()` set the title, x-axis label, and y-axis label respectively.
- `plt.show()` displays the histogram.

The histogram illustrating the distribution of attack types provides valuable insights into the prevalence and nature of attacks observed in the Wireless Sensor Network (WSN). By analyzing this histogram, security analysts can identify the most common types of attacks occurring within the network, enabling them to prioritize the development of detection and mitigation strategies tailored to these specific threats. For example, if the histogram reveals a high frequency of Flooding attacks compared to other attack types, it signals the need for robust detection mechanisms capable of identifying and mitigating such attacks promptly. Understanding the distribution of attack types empowers security teams to deploy proactive measures to safeguard the network against prevalent threats, enhancing overall security posture.

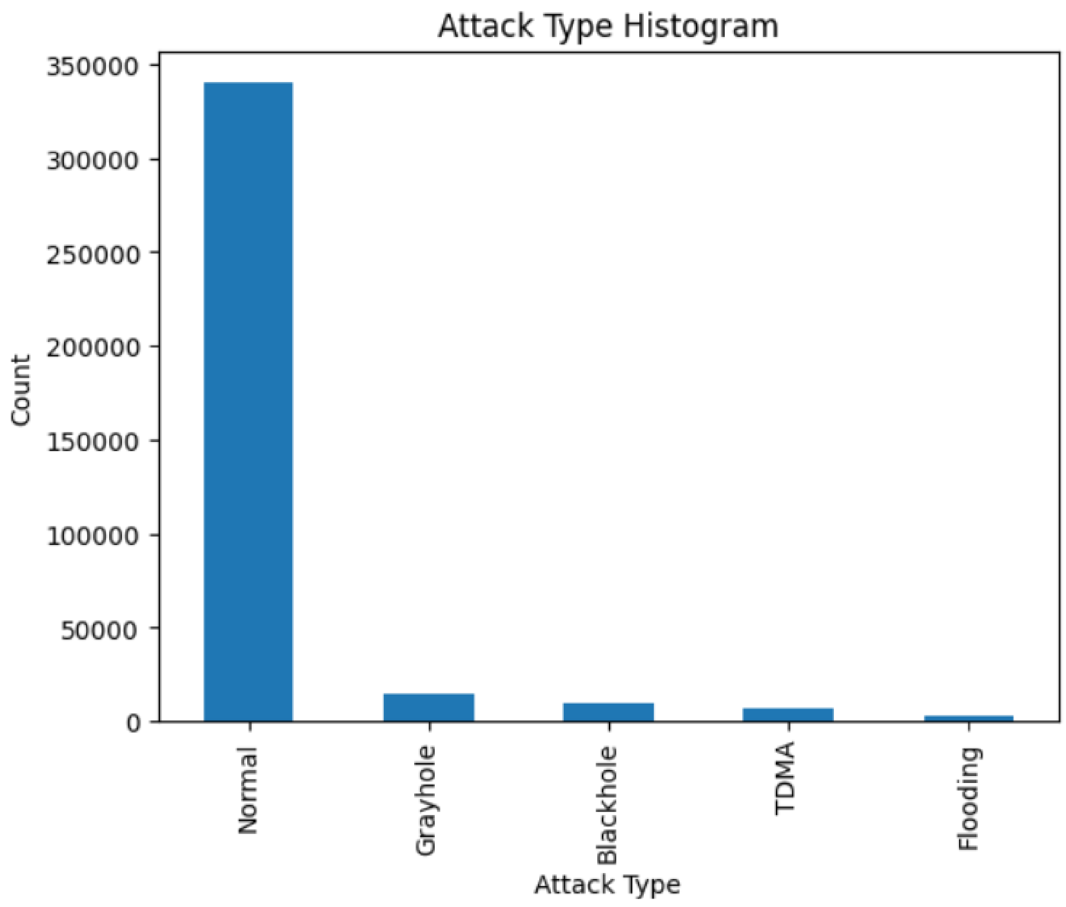


Fig 5.3.3 Graph on Attack Type vs count

Line Plot of Energy Usage over Time:

- The last part of the code plots a line chart showing the average energy usage over time. `df.groupby(' Time')['Expaned Energy'].mean()` groups the data by time and calculates the mean of 'Expaned Energy' for each time period.
- The `plot()` function creates the line plot.
- `plt.title()`, `plt.xlabel()`, and `plt.ylabel()` set the title, x-axis label, and y-axis label respectively.
- Finally, `plt.show()` displays the line plot.

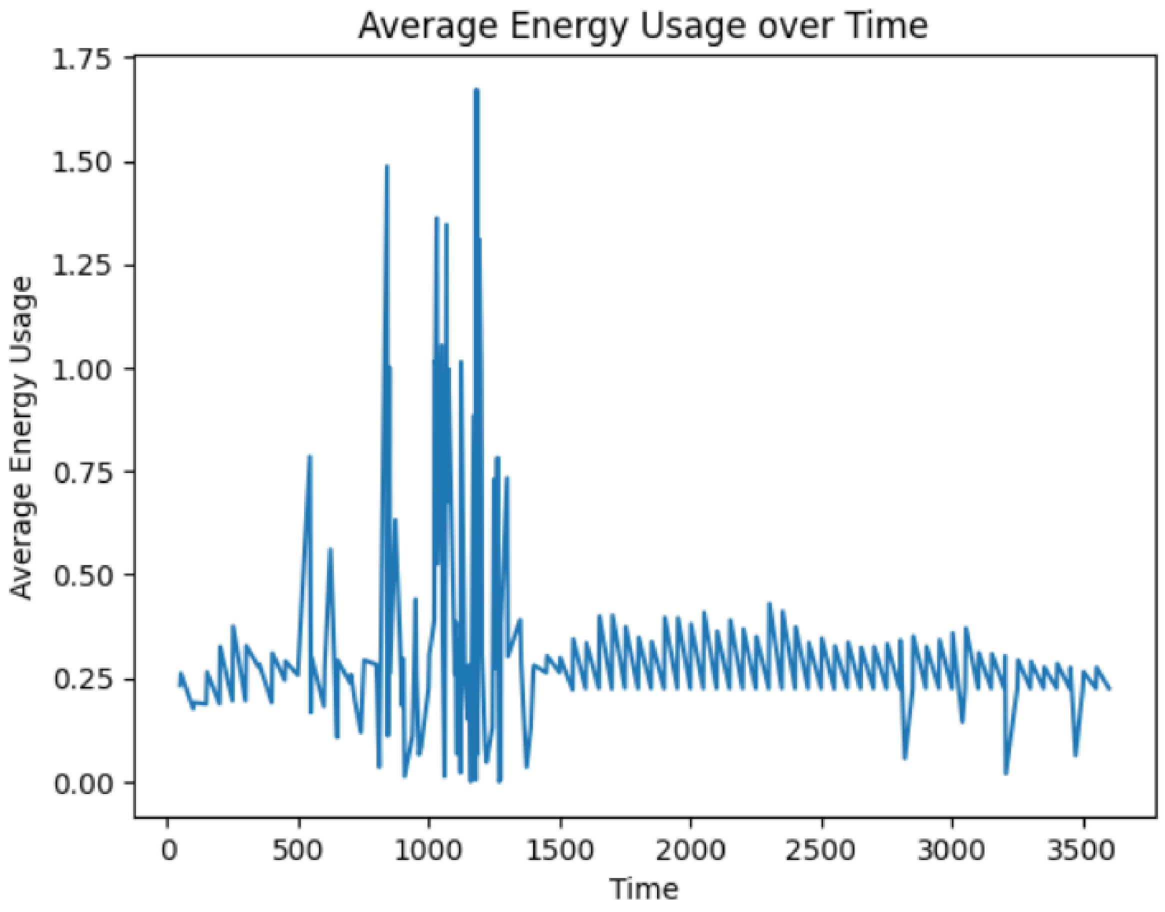


Fig 5.3.4 Time vs Average Energy Usage

The line plot showcasing the average energy usage over time serves as a critical tool for detecting anomalies indicative of ongoing Denial of Service (DoS) attacks in Wireless Sensor Networks (WSNs). By monitoring energy consumption trends over time, analysts can identify sudden spikes or drops in energy usage that may signify malicious activity within the network. For instance, if the line plot exhibits a sharp increase in energy usage followed by a sudden decline, it could indicate the occurrence of a Flooding attack, where malicious nodes flood the network with excessive traffic, causing a surge in energy consumption before disrupting legitimate communication. Detecting such anomalies in energy usage patterns enables security teams to promptly investigate and mitigate potential DoS attacks, thereby safeguarding the integrity and functionality of the WSN.

5.4 Front End and Real Time Implementation

Front-end development focuses on creating user interfaces for applications, enhancing user experience and interaction. Real-time implementation in machine learning involves updating models or predictions continuously as new data arrives. Combining front-end design with real-time capabilities provides users with dynamic interfaces, enabling seamless interaction and visualization of live machine learning outcomes, fostering better user engagement and understanding.

To display and illustrate our project's workings and system design, we've employed a handy Python tool called Tkinter. This library supports creating user interfaces (UI) with ease, making our project user-friendly and visually appealing. Tkinter helps us showcase our system in a way that's easy for anyone to understand, thanks to its GUI capabilities.

TKINTER

Tkinter is a standard GUI (Graphical User Interface) library in Python, allowing developers to create desktop applications with interactive graphical interfaces. It is included with most Python installations, making it easily accessible. Tkinter is based on the Tk GUI toolkit and provides a set of tools and functions to design windows, buttons, menus, and other GUI elements. It is beginner-friendly and widely used for creating applications that involve user interaction, such as data visualization tools, calculators, or simple games. Tkinter's simplicity and versatility make it a popular choice for developers working on projects that require a graphical interface in Python.

Tkinter offers a range of functionalities and attributes for building graphical user interfaces (GUIs) in Python. Here are some key features:

- **Widgets**: Tkinter provides a variety of widgets (GUI elements) such as buttons, labels, entry fields, textboxes, and more, allowing developers to create interactive interfaces.
- **Geometry Management**: Tkinter includes pack(), grid(), and place() methods for efficiently arranging and managing the placement of widgets within the GUI.
- **Event Handling**: It supports event-driven programming, allowing developers to define responses to user actions like button clicks, keypresses, or mouse movements.

- **Styling and Theming**: Tkinter supports basic styling options to customize the appearance of widgets, and it can be extended with themed widgets for more complex theming.
- **Dialog Boxes**: Tkinter includes built-in dialog boxes for common tasks such as file selection, color picking, and message display, simplifying user interactions.
- **Menu System**: It provides functionality for creating menus and menu items, including dropdown menus, context menus, and more.
- **Canvas Widget**: Tkinter includes a canvas widget for drawing shapes, images, and other graphical elements, allowing for custom graphics and visualizations.
- **Binding Variables**: Tkinter supports variable classes like StringVar, IntVar, and DoubleVar, which allow the linkage of widget values with variables, enabling dynamic updates.
- **Accessibility**: Tkinter is accessible, providing features for creating applications that are usable by people with disabilities, such as support for keyboard navigation.
- **Cross-Platform**: Tkinter is cross-platform, meaning that applications built with Tkinter can run on different operating systems without modification.

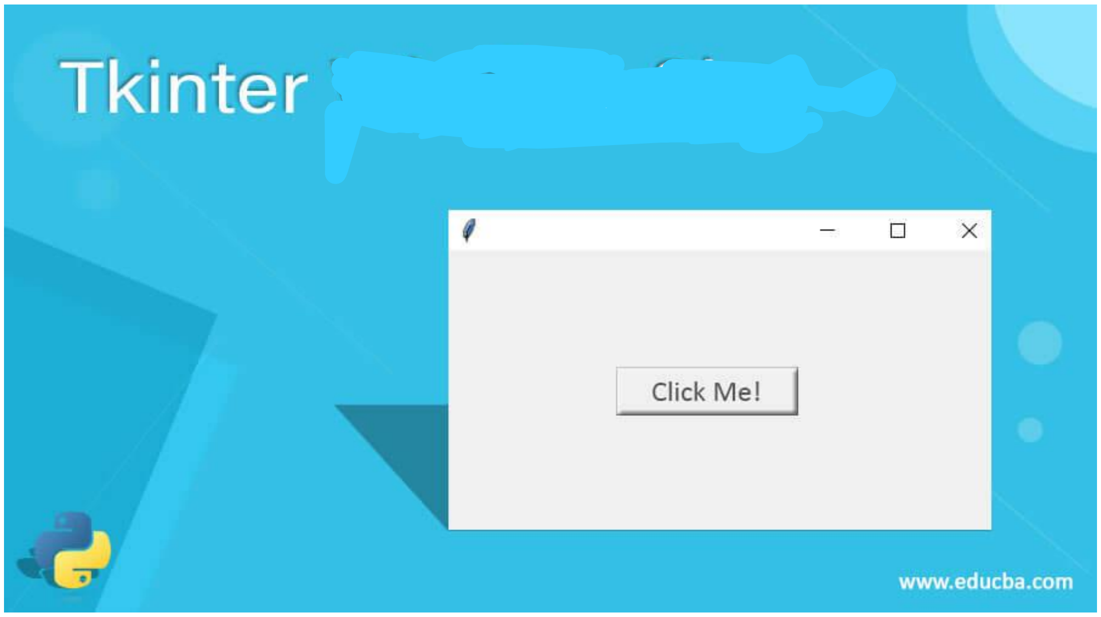
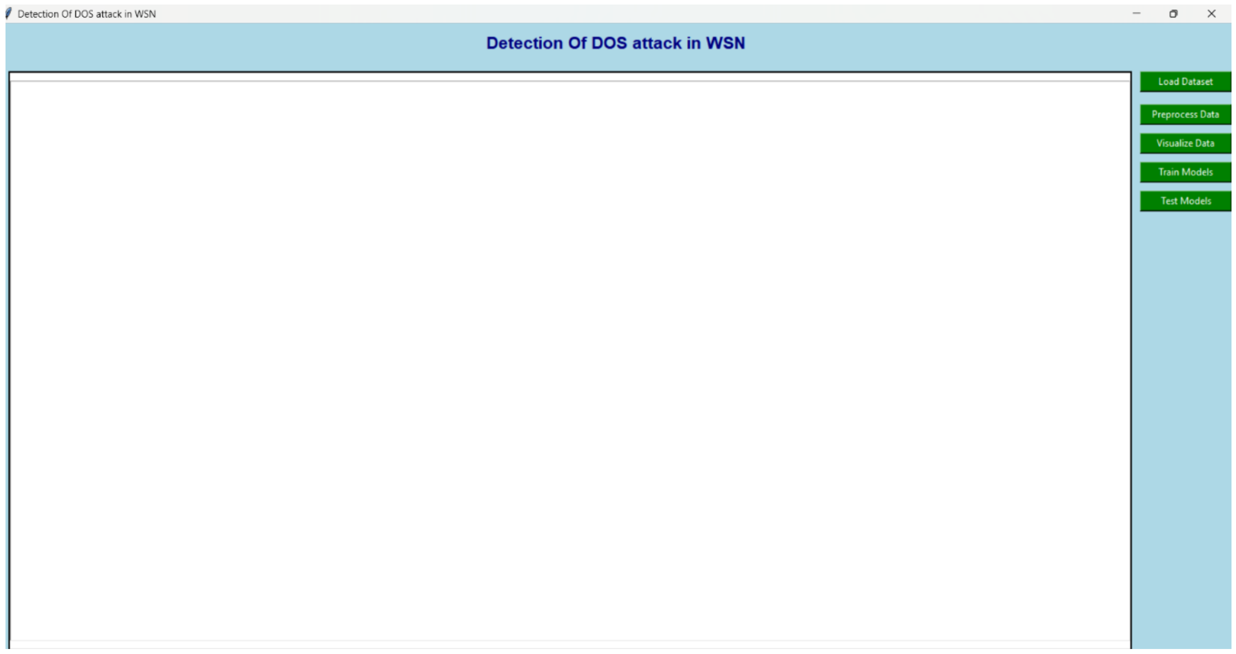


Fig 5.4.1 Tkinter GUI

UI of the proposed System:

A user interface (UI) serves as the interactive bridge between users and computer systems, encompassing the visual and functional design elements that enable seamless interaction. It includes the graphical elements, controls, and navigational tools within an application, ensuring users can easily comprehend and operate the software. An effective UI is characterized by intuitive design, logical organization of information, and responsiveness to user input. Whether it's a website, mobile app, or desktop application, a well-crafted user interface enhances the user experience by providing clarity, efficiency, and an aesthetically pleasing environment. A thoughtfully designed UI considers user needs, preferences, and the context of use, ultimately contributing to the overall success and adoption of a software product.



5.4.2 UI of DOS Attack Detection System.

The front-end user interface (UI) for our project revolves around the title "Detection of DOS in WSN" and outlines the step-by-step process of our proposed system. First, we deal with the dataset, where we gather and organize the data needed for our analysis. Next, we preprocess the data, which means cleaning it up and preparing it for training. Then comes the interesting part – visualizing the data. We create graphs and charts to better understand the patterns and characteristics of our dataset. After that, it's time to train our models, teaching the computer how to recognize denial-of-service attacks. Finally, we put our models to the test, checking how well they can identify and handle these attacks. This UI design ensures a clear and straightforward journey through our project's key steps.

The system architecture involving mainly five steps to describe entire procedure of the development. They are:

- **Load Dataset**
- **Preprocess Data**
- **Visualize Data**
- **Train Models**
- **Test Models**

Load Dataset: A dataset is a collection of organized and structured data that serves as the foundation for training and testing machine learning models. The quality and relevance of the dataset significantly impact the accuracy and effectiveness of the models, making it a crucial aspect in the success of any data-driven project.

- After clicking the load dataset button, a popup is displayed to insert the file which is our dataset. Select the file and click open.

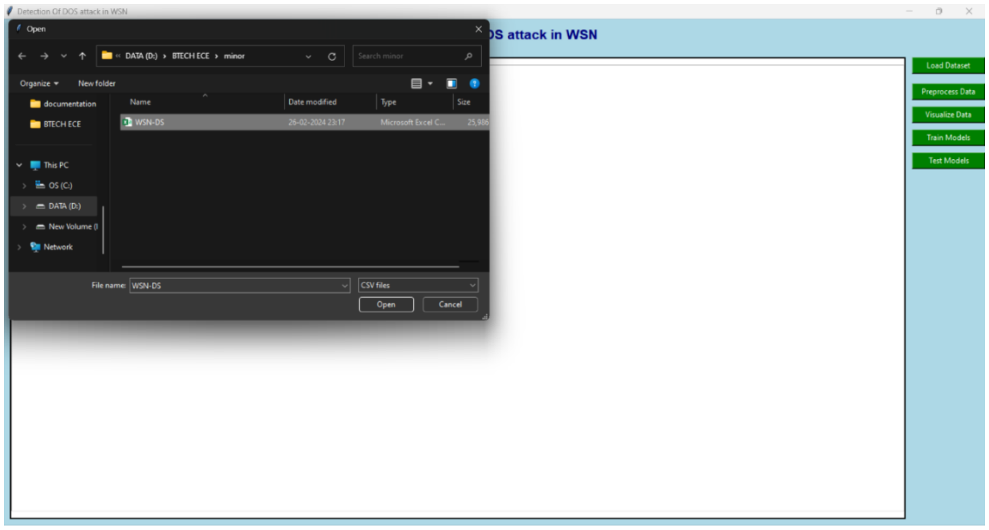
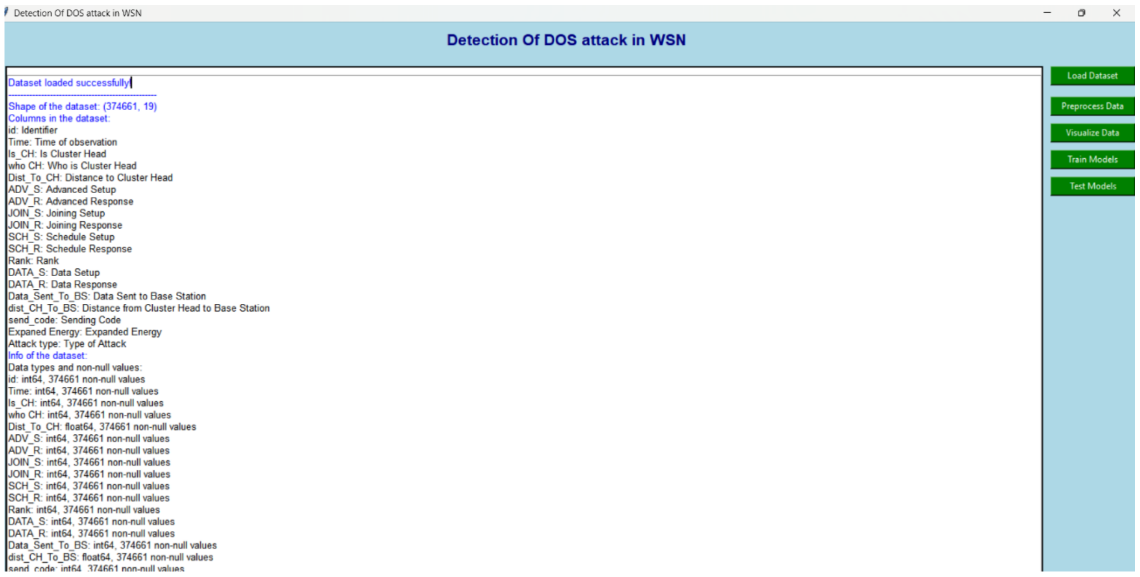


Fig 5.4.3 Loading the Dataset

- After opening the dataset, the you will get a message **“Dataset loaded Successfully”.** The attributes of the dataset are displayed as shown below. It represents the attributes that are helping in detecting the dos attack. The info of the dataset is also displayed.

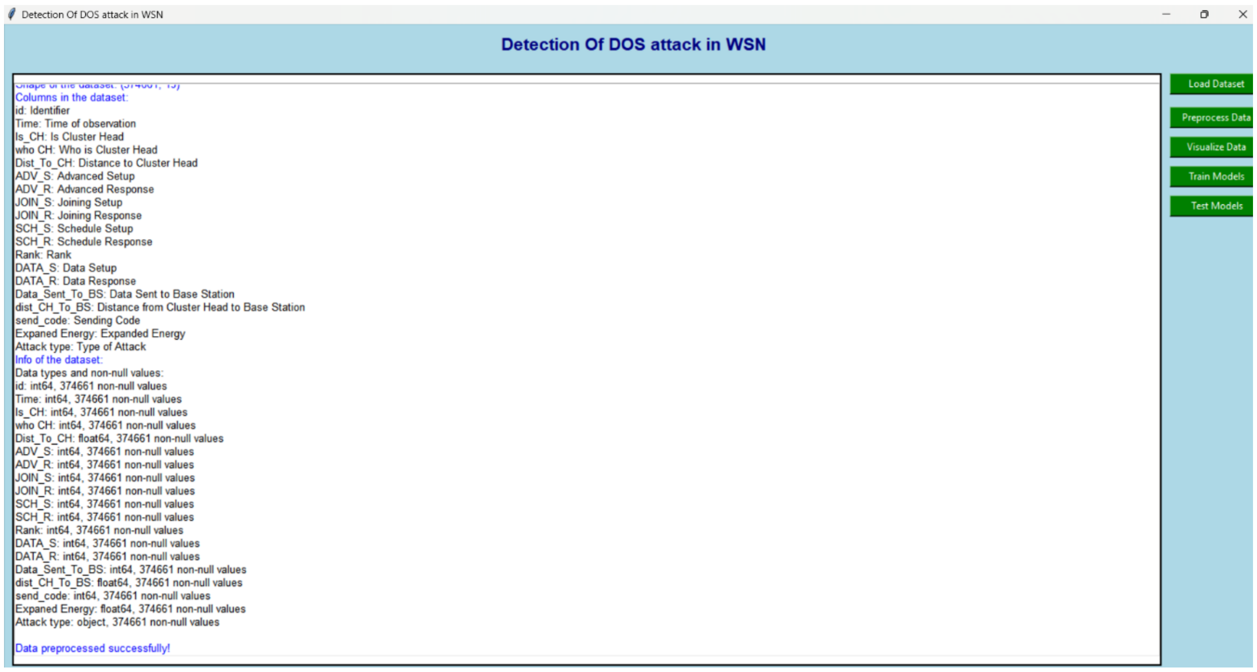


```
Dataset loaded successfully
------------------------------------------------
Shape of the dataset: (374661, 19)
Columns in the dataset:
id: Identifier
Time: Time of observation
Is_CH: Is Cluster Head
who CH: Who is Cluster Head
Dist_To_CH: Distance to Cluster Head
ADV_S: Advanced Setup
ADV_R: Advanced Response
JOIN_S: Joining Setup
JOIN_R: Joining Response
SCH_S: Schedule Setup
SCH_R: Schedule Response
Rank: Rank
DATA_S: Data Setup
DATA_R: Data Response
Data_Sent_To_BS: Data Sent to Base Station
dist_CH_To_BS: Distance from Cluster Head to Base Station
send_code: Sending Code
Expaned Energy: Expanded Energy
Attack type: Type of Attack
Info of the dataset:
Data types and non-null values:
id: int64, 374661 non-null values
Time: int64, 374661 non-null values
Is_CH: int64, 374661 non-null values
who CH: int64, 374661 non-null values
Dist_To_CH: float64, 374661 non-null values
ADV_S: int64, 374661 non-null values
ADV_R: int64, 374661 non-null values
JOIN_S: int64, 374661 non-null values
JOIN_R: int64, 374661 non-null values
SCH_S: int64, 374661 non-null values
SCH_R: int64, 374661 non-null values
Rank: int64, 374661 non-null values
DATA_S: int64, 374661 non-null values
DATA_R: int64, 374661 non-null values
Data_Sent_To_BS: int64, 374661 non-null values
dist_CH_To_BS: float64, 374661 non-null values
send_code: int64, 374661 non-null values
```

Fig 5.4.4 Displaying Dataset Attributes

Preprocess data: Data preprocessing involves cleaning and transforming raw data into a format suitable for machine learning. It includes handling missing values, scaling features, encoding categorical variables, and other steps that enhance data quality, ensuring optimal performance of machine learning models.

- After clicking on the preprocess data button, a message will be displayed as **Data Pre-processed successfully.**

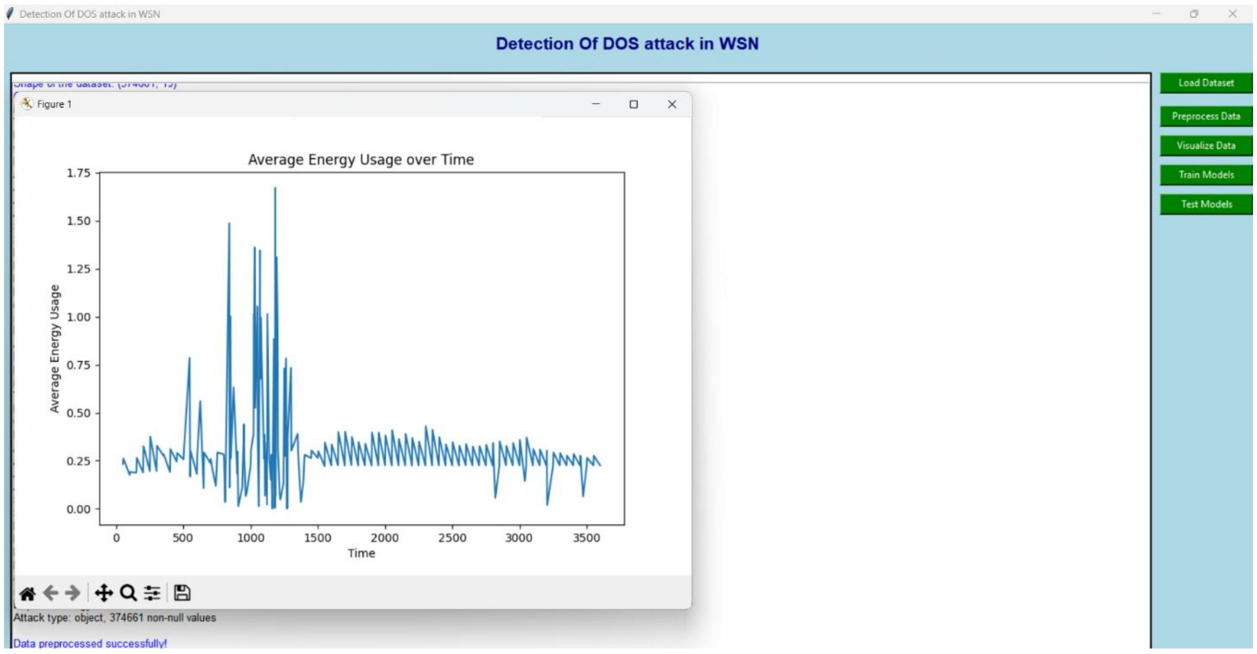


```
Shape of the dataset: (374661, 19)
Columns in the dataset:
id: Identifier
Time: Time of observation
Is_CH: Is Cluster Head
who CH: Who is Cluster Head
Dist_To_CH: Distance to Cluster Head
ADV_S: Advanced Setup
ADV_R: Advanced Response
JOIN_S: Joining Setup
JOIN_R: Joining Response
SCH_S: Schedule Setup
SCH_R: Schedule Response
Rank: Rank
DATA_S: Data Setup
DATA_R: Data Response
Data_Sent_To_BS: Data Sent to Base Station
dist_CH_To_BS: Distance from Cluster Head to Base Station
send_code: Sending Code
Expaned Energy: Expanded Energy
Attack type: Type of Attack
Info of the dataset:
Data types and non-null values:
id: int64, 374661 non-null values
Time: int64, 374661 non-null values
Is_CH: int64, 374661 non-null values
who CH: int64, 374661 non-null values
Dist_To_CH: float64, 374661 non-null values
ADV_S: int64, 374661 non-null values
ADV_R: int64, 374661 non-null values
JOIN_S: int64, 374661 non-null values
JOIN_R: int64, 374661 non-null values
SCH_S: int64, 374661 non-null values
SCH_R: int64, 374661 non-null values
Rank: int64, 374661 non-null values
DATA_S: int64, 374661 non-null values
DATA_R: int64, 374661 non-null values
Data_Sent_To_BS: int64, 374661 non-null values
dist_CH_To_BS: float64, 374661 non-null values
send_code: int64, 374661 non-null values
Expaned Energy: float64, 374661 non-null values
Attack type: object, 374661 non-null values

Data preprocessed successfully!
```

Fig 5.4.5 Data Pre-processed Successfully

Visualize Data: Visualizing the data entails creating graphical representations like charts and graphs to gain insights into patterns, trends, and relationships within the dataset. This step enhances understanding, aids in feature selection, and guides subsequent decisions in the machine learning process.

- After clicking on the visualize data button a popup will be displayed will showing all the attributes and their relation to the parameters.
- Each attribute has its specification in determining the dos attack.
- Energy consumption at a specific time assists in determining data reception and transmission, providing insights into the amount of energy expended. This parameter allows us to ascertain traffic information based on the observed energy consumption patterns.



Attack type: object, 374661 non-null values

Data preprocessed successfully!

Fig 5.4.6 Visualizing the data

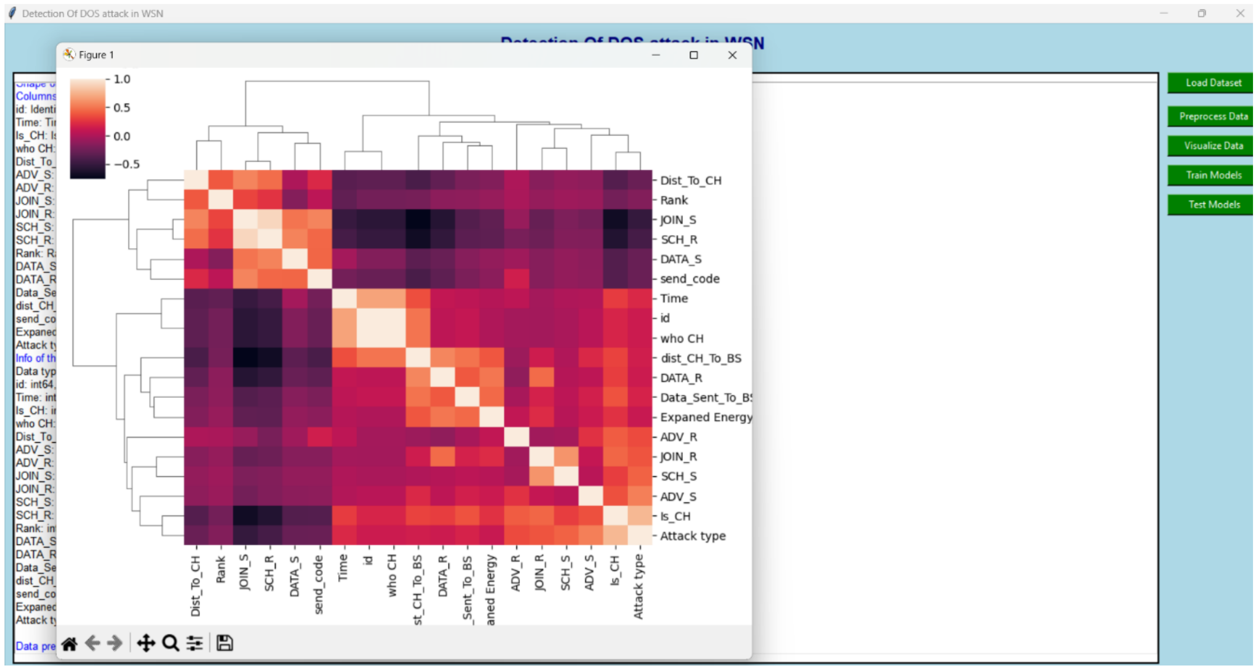
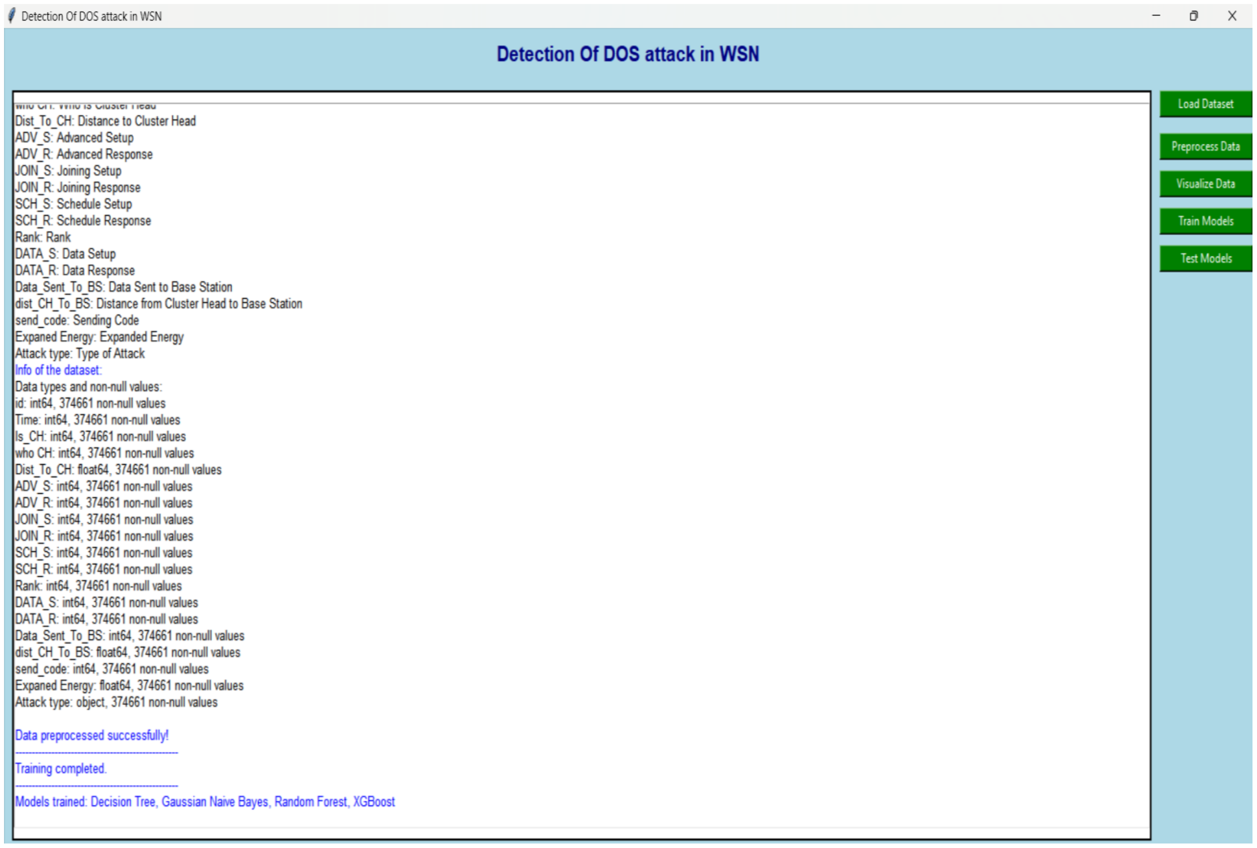


Fig 5.4.7 Confusion Matrix

Train Models:

Training models involves utilizing a machine learning algorithm on a labelled dataset to enable the model to learn patterns and relationships within the data. This process fine-tunes the model's parameters, allowing it to make accurate predictions or classifications when presented with new, unseen data.

- After clicking Train models button, you will be the output displayed as “**Training Completed**”.
- The models trained are: Decision Tree, Gaussian Naïve Bayes, Random Forest, XGBoost.



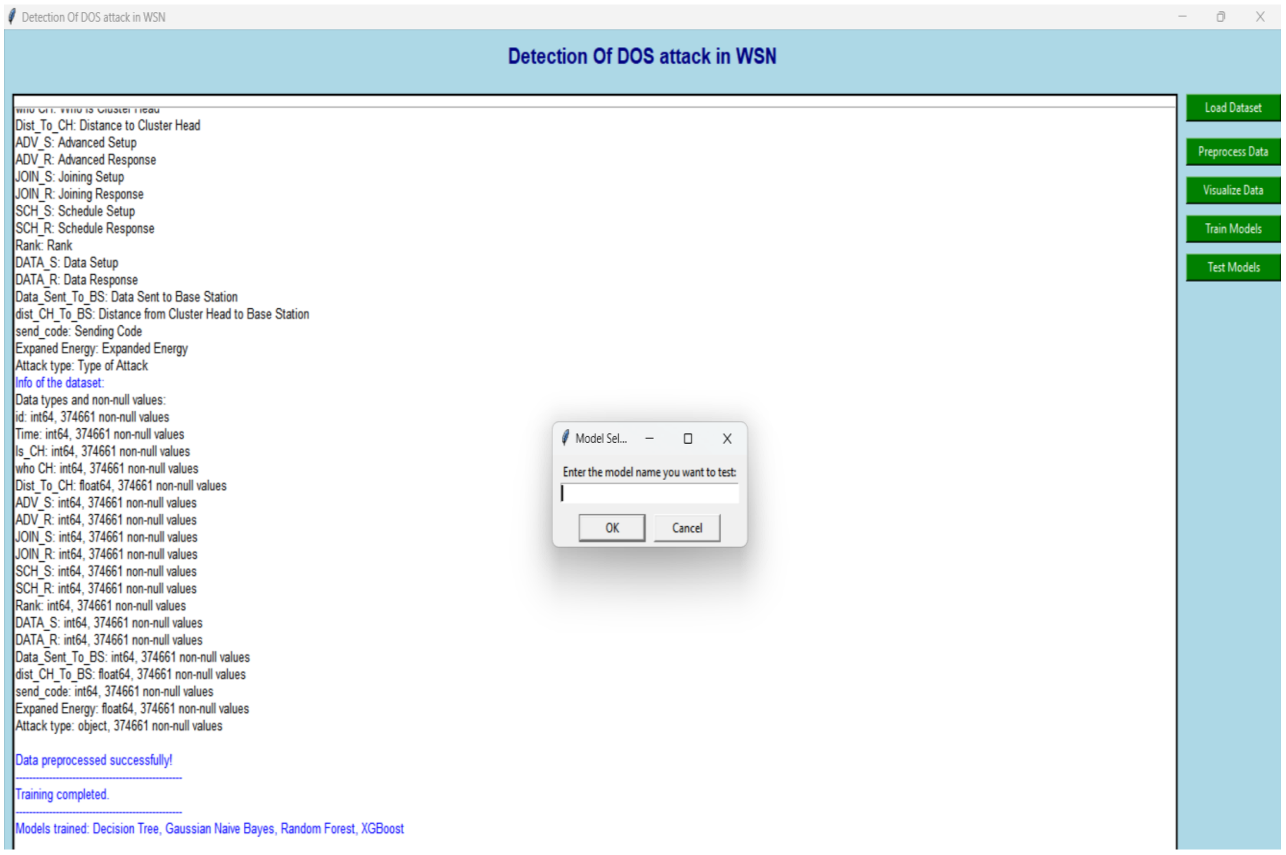
```
who CH: Who is Cluster Head
Dist_To_CH: Distance to Cluster Head
ADV_S: Advanced Setup
ADV_R: Advanced Response
JOIN_S: Joining Setup
JOIN_R: Joining Response
SCH_S: Schedule Setup
SCH_R: Schedule Response
Rank: Rank
DATA_S: Data Setup
DATA_R: Data Response
Data_Sent_To_BS: Data Sent to Base Station
dist_CH_To_BS: Distance from Cluster Head to Base Station
send_code: Sending Code
Expaned Energy: Expanded Energy
Attack type: Type of Attack
Info of the dataset:
Data types and non-null values:
id: int64, 374661 non-null values
Time: int64, 374661 non-null values
Is_CH: int64, 374661 non-null values
who CH: int64, 374661 non-null values
Dist_To_CH: float64, 374661 non-null values
ADV_S: int64, 374661 non-null values
ADV_R: int64, 374661 non-null values
JOIN_S: int64, 374661 non-null values
JOIN_R: int64, 374661 non-null values
SCH_S: int64, 374661 non-null values
SCH_R: int64, 374661 non-null values
Rank: int64, 374661 non-null values
DATA_S: int64, 374661 non-null values
DATA_R: int64, 374661 non-null values
Data_Sent_To_BS: int64, 374661 non-null values
dist_CH_To_BS: float64, 374661 non-null values
send_code: int64, 374661 non-null values
Expaned Energy: float64, 374661 non-null values
Attack type: object, 374661 non-null values

Data preprocessed successfully!
--------------------------------------------
Training completed.
--------------------------------------------
Models trained: Decision Tree, Gaussian Naive Bayes, Random Forest, XGBoost
```

Fig 5.4.8 Training the models

Test Models:

Testing models refers to evaluating the trained machine learning models on new or unseen data to assess their performance, accuracy, and generalization ability. This step helps ensure that the models can make reliable predictions or classifications beyond the data used for training, providing insights into their real-world effectiveness.

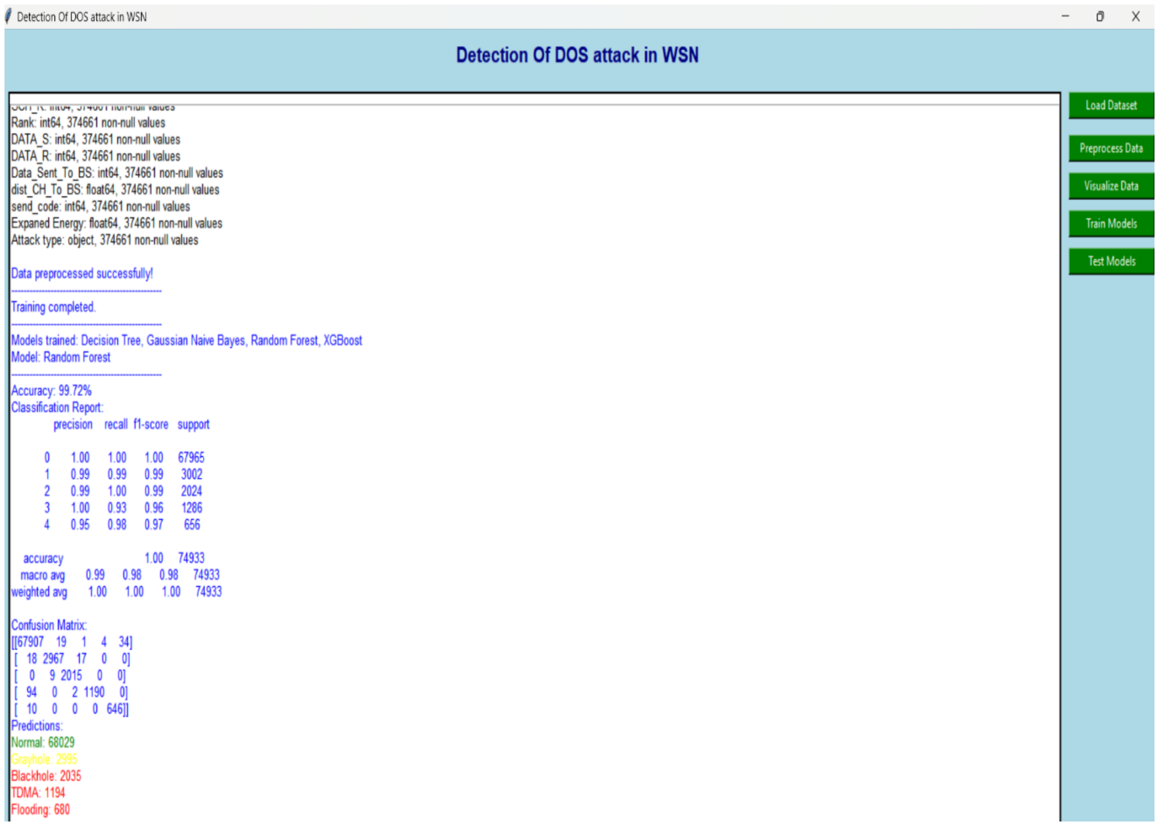
- After clicking on the test models button a pop-up is displayed showing "**enter the model's name you want to test**".


Detection Of DOS attack in WSN
Detection Of DOS attack in WSN
Load Dataset
Preprocess Data
Visualize Data
Train Models
Test Models
Model Sel...
Enter the model name you want to test:
OK
Cancel

```
Dist_To_CH: Distance to Cluster Head
ADV_S: Advanced Setup
ADV_R: Advanced Response
JOIN_S: Joining Setup
JOIN_R: Joining Response
SCH_S: Schedule Setup
SCH_R: Schedule Response
Rank: Rank
DATA_S: Data Setup
DATA_R: Data Response
Data_Sent_To_BS: Data Sent to Base Station
dist_CH_To_BS: Distance from Cluster Head to Base Station
send_code: Sending Code
Expaned Energy: Expanded Energy
Attack type: Type of Attack
Info of the dataset:
Data types and non-null values:
id: int64, 374661 non-null values
Time: int64, 374661 non-null values
Is_CH: int64, 374661 non-null values
who CH: int64, 374661 non-null values
Dist_To_CH: float64, 374661 non-null values
ADV_S: int64, 374661 non-null values
ADV_R: int64, 374661 non-null values
JOIN_S: int64, 374661 non-null values
JOIN_R: int64, 374661 non-null values
SCH_S: int64, 374661 non-null values
SCH_R: int64, 374661 non-null values
Rank: int64, 374661 non-null values
DATA_S: int64, 374661 non-null values
DATA_R: int64, 374661 non-null values
Data_Sent_To_BS: int64, 374661 non-null values
dist_CH_To_BS: float64, 374661 non-null values
send_code: int64, 374661 non-null values
Expaned Energy: float64, 374661 non-null values
Attack type: object, 374661 non-null values

Data preprocessed successfully!
----------------------------------------------------
Training completed.
----------------------------------------------------
Models trained: Decision Tree, Gaussian Naive Bayes, Random Forest, XGBoost
```

Fig 5.4.9 Entry of Model Name for testing

- After giving the name and clicking the enter you will get the results displayed as accuracy, classification report, confusion matrix, predictions such as attacks.



```
Rank: int64, 374661 non-null values
DATA_S: int64, 374661 non-null values
DATA_R: int64, 374661 non-null values
Data_Sent_To_BS: int64, 374661 non-null values
dist_CH_To_BS: float64, 374661 non-null values
send_code: int64, 374661 non-null values
Expaned Energy: float64, 374661 non-null values
Attack type: object, 374661 non-null values

Data preprocessed successfully!
--------------------------------------------------
Training completed.
--------------------------------------------------
Models trained: Decision Tree, Gaussian Naive Bayes, Random Forest, XGBoost
Model: Random Forest
--------------------------------------------------
Accuracy: 99.72%
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     67965
           1       0.99      0.99      0.99      3002
           2       0.99      1.00      0.99      2024
           3       1.00      0.93      0.96      1286
           4       0.95      0.98      0.97       656

    accuracy                           1.00     74933
   macro avg       0.99      0.98      0.98     74933
weighted avg       1.00      1.00      1.00     74933

Confusion Matrix:
[[67907    19     1     4    34]
 [   18  2967    17     0     0]
 [    0     9  2015     0     0]
 [   94     0     2  1190     0]
 [   10     0     0     0   646]]
Predictions:
Normal: 68029
Grayhole: 2995
Blackhole: 2035
TDMA: 1194
Flooding: 680
```

Fig 5.4.10 Output of the tested Model

5.5 Evaluation:

Testing and evaluation in the context of machine learning involve assessing the performance, accuracy, and effectiveness of trained models on new or unseen data. This crucial phase aims to validate the model's ability to generalize well and make reliable predictions or classifications in real-world scenarios. Various metrics and techniques, such as precision, recall, and accuracy, are employed to thoroughly evaluate the model's overall effectiveness and identify areas for improvement.

The Evaluation is performed on the dataset named WSN-DS dataset. It contains 19 attributes 374,661 records, representing various types of Denial of Service (DoS) attacks. These attacks include the likes of Blackhole, Grayhole, Flooding, and Scheduling attacks. Researchers and practitioners can utilize this dataset to enhance the effectiveness of IDS in detecting and classifying different DoS attacks within WSNs.

We calculate four metrics to check the model's performance. They are:

1. Accuracy
2. Precision
3. Recall
4. F1 score.

Accuracy

Accuracy is one of the most used metrics for evaluating the performance of a machine learning model during testing. It is calculated as the number of correctly classified instances divided by the total number of instances in the test dataset.

Precision

Precision is a measure of the accuracy of a classifier model, specifically its ability to correctly identify the positive instances (true positives) among all the instances it classified as positive (true positives plus false positives). Precision is calculated as the ratio of true positives to the total number of positive predictions made by the model (true positives plus false positives).

Recall

Recall is a measure of how well a model can correctly identify all positive instances in a dataset. Specifically, recall is calculated as the ratio of true positives to the total number of actual positive instances in the dataset (true positives plus false negatives).

F1 Score

F1 score is a commonly used metric for evaluating the overall performance of a classification model. It is calculated as the harmonic mean of precision and recall, two commonly used metrics for evaluating classification models.

In this project we have tested mainly three algorithms they are XG Boost, Gaussian Naïve Bayes, Random Forest and for comparison we have implemented decision tree.

Evaluation result of XG Boost:

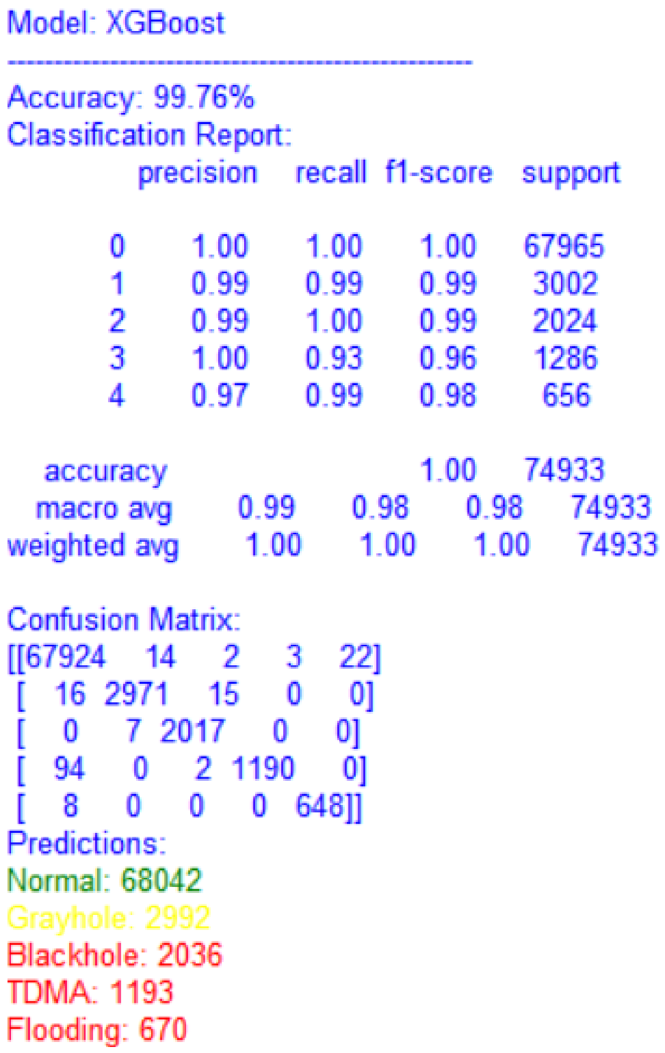
Accuracy Score: The accuracy score of 0.997 indicates that the XGBoost classifier achieved a high level of accuracy in classifying instances correctly. This score implies that 99.75% of the instances were classified accurately.

Precision: Precision refers to the ratio of true positive predictions to the total number of positive predictions made by the model. In this report, high precision scores across all classes (ranging from 0.97 to 1.00) indicate that the classifier performed well in minimizing false positives. Specifically, for class 0, precision is 1.00, indicating that all instances classified as class 0 were indeed true positives.

Recall: Recall, also known as sensitivity, measures the ratio of true positive predictions to the total number of actual positive instances in the dataset. The recall scores for all classes are high, indicating that the classifier effectively captured a significant portion of the true positive instances in each class.

F1-Score: The F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics. High F1-scores across all classes (ranging from 0.96 to 1.00) indicate that the classifier achieved a good balance between precision and recall, resulting in robust performance.

Support: Support refers to the number of actual occurrences of each class in the dataset. The support values for each class indicate the distribution of instances across the classes. It determines the number of attacks that are identified and detected by the algorithm.



```
Model: XGBoost
------------------------------------------------
Accuracy: 99.76%
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     67965
           1       0.99      0.99      0.99      3002
           2       0.99      1.00      0.99      2024
           3       1.00      0.93      0.96      1286
           4       0.97      0.99      0.98       656

    accuracy                           1.00     74933
   macro avg       0.99      0.98      0.98     74933
weighted avg       1.00      1.00      1.00     74933

Confusion Matrix:
[[67924    14     2     3    22]
 [   16  2971    15     0     0]
 [    0     7  2017     0     0]
 [   94     0     2  1190     0]
 [    8     0     0     0   648]]
Predictions:
Normal: 68042
Grayhole: 2992
Blackhole: 2036
TDMA: 1193
Flooding: 670
```

Fig 5.5.1 Evaluation Result of XGBoost

Evaluation report of Random Forest:

The provided classification report for the Random Forest classifier in DoS attack detection in wireless sensor networks (WSNs) demonstrates high performance and effectiveness in identifying different types of attacks.

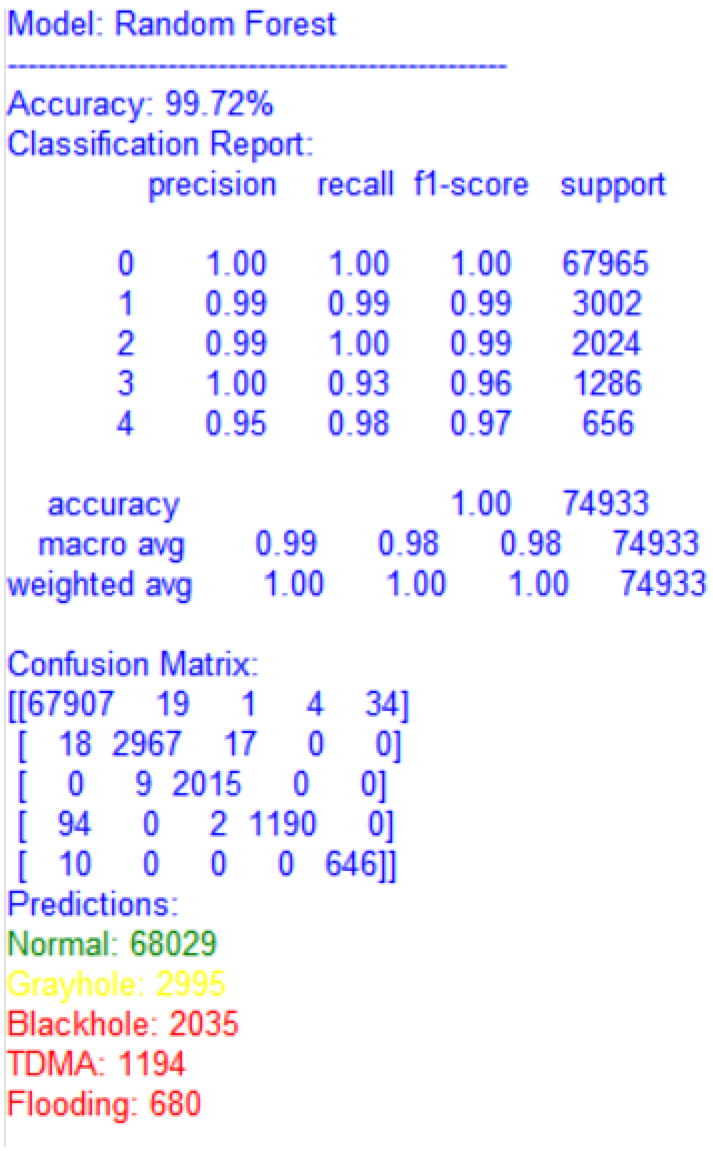
High Accuracy: The reported accuracy score of 99.72% indicates that the Random Forest classifier is adept at correctly classifying instances of both attack and non-attack traffic in the WSN dataset. This high accuracy is crucial for ensuring reliable detection of DoS attacks while minimizing false positives and negatives.

Precision and Recall: The precision and recall scores for each attack class provide insights into the classifier's ability to correctly identify instances of specific types of attacks. In this case, the classifier demonstrates high precision and recall across most attack classes, indicating its effectiveness in both minimizing false alarms and detecting actual attacks.

F1-Score: The F1-score, which is the harmonic mean of precision and recall, provides a balanced measure of the classifier's performance in terms of both false positives and false negatives. The high F1-scores reported for each attack class indicate a robust and reliable detection capability of the Random Forest classifier.

Support: The "support" column indicates the number of instances of each attack class in the dataset. This information is helpful for understanding the distribution of different types of attacks and assessing the significance of the classifier's performance across these classes.

Macro and Weighted Averages: The macro and weighted average metrics provide aggregated measures of the classifier's performance across all attack classes. In this case, both averages demonstrate consistently high scores, indicating overall effectiveness in detecting DoS attacks in the WSN dataset.



```
Model: Random Forest
--------------------------------------------------
Accuracy: 99.72%
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     67965
           1       0.99      0.99      0.99      3002
           2       0.99      1.00      0.99      2024
           3       1.00      0.93      0.96      1286
           4       0.95      0.98      0.97       656

    accuracy                           1.00     74933
   macro avg       0.99      0.98      0.98     74933
weighted avg       1.00      1.00      1.00     74933

Confusion Matrix:
[[67907    19     1     4    34]
 [   18  2967    17     0     0]
 [    0     9  2015     0     0]
 [   94     0     2  1190     0]
 [   10     0     0     0   646]]
Predictions:
Normal: 68029
Grayhole: 2995
Blackhole: 2035
TDMA: 1194
Flooding: 680
```

Fig 5.5.2 Evaluation Result of Random Forest

Evaluation result of Naïve Bayes:

The provided classification report is for the Naive Bayes classifier applied to detect Denial-of-Service (DoS) attacks in a wireless sensor network (WSN) dataset. Here's an explanation of the key metrics:

Accuracy Score (0.8486): The accuracy score measures the proportion of correctly classified instances out of the total instances. In this case, the Naive Bayes classifier achieves an accuracy of approximately 84.86%.

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positives. Naive Bayes achieves varied precision scores for each class, ranging from 0.18 to 0.99. For instance, for class 0 (non-attack), precision is high at 0.99, indicating that most instances classified as non-attacks are indeed non-attacks. However, for class 1 (attack), precision is lower at 0.18, indicating that only a small proportion of instances classified as attacks are true attacks.

Recall: Recall (also called sensitivity) is the ratio of correctly predicted positive observations to all actual positives. Naive Bayes achieves varied recall scores for each class, ranging from 0.37 to 0.93. Higher recall values indicate that the classifier can effectively identify true positive instances. For example, class 2 (attack) has a recall of 0.93, suggesting that the classifier can correctly identify a large proportion of true attacks.

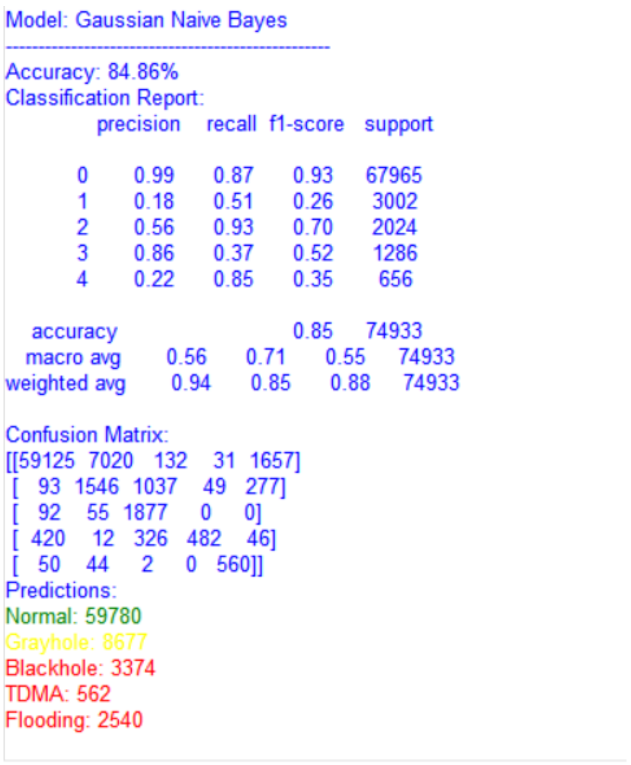
F1-Score: The F1-score is the harmonic mean of precision and recall and provides a balance between the two metrics. Naive Bayes achieves F1-scores ranging from 0.26 to 0.93 for different classes, reflecting the classifier's overall performance in terms of both precision and recall.

Support: Support refers to the number of actual occurrences of each class in the dataset. It provides context for the precision, recall, and F1-score metrics.

Macro and Weighted Averages:

Macro Average: The macro average calculates the metric independently for each class and then takes the unweighted mean of those scores. In this case, the macro average precision, recall, and F1-score are 0.56, 0.71, and 0.55, respectively.

Weighted Average: The weighted average calculates the metric for each class and then takes the weighted average based on the number of instances for each class. Naive Bayes achieves higher weighted average scores compared to the macro average, indicating that the classifier performs better when accounting for class imbalances in the dataset.



```
Model: Gaussian Naive Bayes
--------------------------------------------------
Accuracy: 84.86%
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.87      0.93     67965
           1       0.18      0.51      0.26      3002
           2       0.56      0.93      0.70      2024
           3       0.86      0.37      0.52      1286
           4       0.22      0.85      0.35       656

    accuracy                           0.85     74933
   macro avg       0.56      0.71      0.55     74933
weighted avg       0.94      0.85      0.88     74933

Confusion Matrix:
[[59125  7020   132    31  1657]
 [   93  1546  1037    49   277]
 [   92    55  1877     0     0]
 [  420    12   326   482    46]
 [   50    44     2     0   560]]
Predictions:
Normal: 59780
Grayhole: 8677
Blackhole: 3374
TDMA: 562
Flooding: 2540
```

Fig 5.5.3. Evaluation Result of Gaussian Naïve Bayes

Model	Accuracy	F1-score	Precision	Recall
XGBoost	99.76	100	99.0	98.2
Random Forest	99.72	100	98.6	98.0
Naïve bayes	84.86	85	56.2	70.6

Table 2: Evaluation of Models

6.CONCLUSION

In this project, we have tackled the persistent challenge of detecting Denial-of-Service (DoS) attacks in Wireless Sensor Networks (WSNs) using machine learning (ML) techniques. With the proliferation of IoT devices, the vulnerability to DoS attacks in WSNs has become a pressing concern. Our approach integrates three popular ML algorithms: Naive Bayes, XGBoost, and Random Forest, to enhance the existing detection methods. Through our efforts, we achieved notable improvements in accuracy, reaching 84.86% for Naive Bayes, 99.76% for XGBoost, and 99.72% for Random Forest models. Among the three models which performed well is XGBoost with an accuracy of 99.72%.

To execute this project, we commenced by implementing and evaluating these ML models individually to ascertain their efficacy in detecting DoS attacks in WSNs. Subsequently, we curated a dataset comprising network traffic features conducive to training the models. Additionally, we developed an intuitive front-end interface using Tkinter, facilitating seamless interaction with the models. The interface provides functionalities such as data loading, preprocessing, model training, visualization of results, and model testing.

To evaluate the performance of our models, we employed standard metrics including Accuracy, Precision, Recall, and F1-Score. Comparing our results with existing approaches, we observed competitive or superior performance across these metrics, as illustrated in the evaluation table (refer to Figure 6.1). Our project signifies a significant stride towards fortifying WSNs against DoS attacks, offering promising results and a user-friendly interface for practitioners and researchers in the field.

Future Scope

Future research can focus on developing techniques to reduce training time and optimize epoch selection without compromising model performance. This could involve exploring parallel processing, distributed computing, or novel optimization algorithms tailored for WSN environments.

Further refinement of optimization strategies for training parameters and model architectures can significantly improve the efficiency and accuracy of machine learning algorithms for DoS detection in WSNs. This includes exploring novel optimization algorithms, hyperparameter tuning techniques, and model compression methods to optimize resource utilization and enhance performance.

Conducting extensive validation studies in real-world WSN deployments is crucial to assess the scalability, robustness, and practicality of machine learning-based IDSs. Collaborating with industry partners or deploying prototypes in operational WSNs can provide valuable insights into real-world performance and potential challenges.

REFERENCES

[1] G. G. Gebremariam, J. Panda, and S. Indu, ‘‘secure localization against malicious nodes in IoT-based wireless sensor networks using federated learning,’’ Wireless Communication Mobile Computer., vol. 2023, pp. 1–27, Jan. 2023, doi: 10.1155/2023/8068038.

[2] M. Faris, M. N. Mahmud, M. F. M. Salleh, and A. Alnoor, ‘‘Wireless sensor network security: A recent review based on state-of-the-art works,’’ Int. J.Eng. Bus. Manage., vol. 15, Jan. 2023, Art. no. 184797902311572, doi:10.1177/18479790231157220.

[3] S. Ashraf, O. Alfandi, A. Ahmad, A. M. Khattak, B. Hayat, K. H. Kim, and A. Ullah, ‘‘Bodacious-instance coverage mechanism for wireless sensor network,’’ Wireless Commun. Mobile Comput., vol. 2020, pp. 1–11, Nov. 2020, doi: 10.1155/2020/8833767.

[4] X. Feng, X. Ding, and S. Sun, ‘‘A security detection scheme based on evidence nodes in wireless sensor networks,’’ in Proc. 6th Int. Conf. Biomed. Eng. Informat., Dec. 2013, pp. 689–693, doi:10.1109/BMEI.2013.6747027.

[5] H. Yang, L.-X. Wei, and X.-Y. Yang, ‘‘Sybil attack detection scheme in wireless sensor network,’’ Jisuanji Gongcheng/Comput. Eng., vol. 37, no. 12, pp. 122–124, 2011.

[6] S. Salmi and L. Oughdir, ‘‘Performance evaluation of deep learning techniques for DoS attacks detection in wireless sensor network,’’ J. Big Data, vol. 10, no. 1, p. 17, Feb. 2023, doi: 10.1186/s40537-023-00692-w.

[7] S. Ismail, D. W. Dawoud, and H. Reza, ‘‘Securing wireless sensor networks using machine learning and blockchain: A review,’’ Future Internet, vol. 15, no. 6, p. 200, May 2023, doi: 10.3390/fi15060200.

[8] M. N. U. Islam, A. Fahmin, M. S. Hossain, and M. Atiquzzaman, ‘‘Denialof-service attacks on wireless sensor network and defense techniques,’’Wireless Personal Commun., vol. 116, pp. 1993–2021, Feb. 2020, doi:10.1007/s11277-020-07776-3.

[9] H. S. Sharma, M. M. Singh, and A. Sarkar, ‘‘Machine learning-based DoS attack detection techniques in wireless sensor network: A review,’’ in Proc. Int. Conf. Cogn. Intell. Comput., A. Kumar, G. Ghinea, S. Merugu, and T. Hashimoto, Eds. Singapore: Springer, 2023, pp. 583–591, doi:10.1007/978-981-19-2358-6_53.

[10] V. Dani, ‘‘Detection of denial-of-service attack using weight based trust aware routing approach,’’ J. Inf. Assurance Secure, vol. 18, pp. 089–097, Jun. 2023.

[11] B. J. Santhosh Kumar and S. Sinha, ‘‘An intrusion detection and prevention system against DOS attacks for internet-integrated WSN,’’ in Proc. 7th Int. Conf. Commun. Electron. Syst. (ICCES), Jun. 2022, pp. 793–797, doi:10.1109/ICCES54183.2022.9835838.

[12] G. G. Gebremariam, J. Panda, and S. Indu, ‘‘Localization and detection of multiple attacks in wireless sensor networks using artificial neural network,’’ Wireless Commun. Mobile Comput., vol. 2023, pp. 1–29, Jan. 2023, doi: 10.1155/2023/2744706.

10.1109/ICCES54183.2022.9835838.

[13] B. K. Mishra, M. C. Nikam, and P. Lakkadwala, ‘‘Security against black hole attack in wireless sensor network—A review,’’ in Proc. 4th Int. Conf. Commun. Syst. Netw. Technol., Apr. 2014, pp. 615–620, doi:10.1109/CSNT.2014.129.

[14] R. Wazirali and R. Ahmad, ‘‘Machine learning approaches to detect DoS and their effect on WSNs lifetime,’’ Comput., Mater. Continua, vol. 70,no. 3, pp. 4921–4946, 2022, doi: 10.32604/cmc.2022.020044.

[15] S. B. Atitallah, M. Driss, W. Boulila, and I. Almomani, ‘‘An effective detection and classification approach for DoS attacks in wireless sensor networks using deep transfer learning models and majority voting,’’ in Advances in Computational Collective Intelligence, vol. 1653, C. Badicva, J. Treur, D. Benslimane, B. Hnatkowska, and M. Krótkiewicz, Eds.Cham, Switzerland: Springer, 2022, pp. 180–192, doi: 10.1007/978-3-031-16210-7_14.

[16] S. Ismail, D. Dawoud, and H. Reza, ‘‘A lightweight multilayer machine learning detection system for cyber-attacks in WSN,’’ in Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC), Jan. 2022, pp. 0481–0486, doi: 10.1109/CCWC54503.2022.9720891.

[17] H. Tabbaa, S. Ifzarne, and I. Hafidi, ‘‘An online ensemble learning model for detecting attacks in wireless sensor networks,’’ 2022, arXiv:2204.13814.