Machine Learning to Detect and Prevent Sinkhole Attacks in Cluster-based Routing Protocol of WSNs

Aregay Nigusie Gebrekirstose

A Thesis Submitted to

The Department of Computer Science and Engineering

School of Electrical Engineering and Computer Science

Presented in Partial Fulfillment of the Requirement for the Degree of

Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February, 2023

Adama, Ethiopia

Machine Learning to Detect and Prevent Sinkhole Attacks in Cluster-based Routing Protocol of WSNs

Aregay Nigusie Gebrekirstose

Advisor

Ketema Adere (Ph.D.)

A Thesis Submitted to

The Department of Computer Science and Engineering

School of Electrical Engineering and Computer Science

Presented in Partial Fulfillment of the Requirement for the Degree of

Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February, 2023

Adama, Ethiopia

DECLARATION

I hereby declare that this Master Thesis entitled. "**Machine Learning to Detect and Prevent Sinkhole Attacks in Cluster-based Routing Protocol of WSNs"** 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

Aregay Nigusie	_____	_____
Name of the student	Signature	Date

RECOMMENDATION

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Machine Learning to Detect and Prevent Sinkhole Attacks in Cluster-based Routing Protocol of WSNs"** prepared under my guidance by **Aregay Nigusie** submitted in partial fulfillment of the requirements for the degree of Mater's of Science in Computer Science and Engineering. Therefore, I recommend the submission of revised version of the thesis to the department following the applicable procedures

Ketema Adere (Ph.D.)	_____	_____
Major Advisor	Signature	Date
_____	_____	_____
Co-advisor	Signature	Date

APPROVAL PAGE

I/we, the advisors of the thesis entitled "**Machine Learning to Detect and Prevent Sinkhole Attacks in Cluster-based Routing Protocol of WSNs"** and developed by **Aregay Nigusie** , hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis

Ketema Adere (Ph.D.)	_____	_____
Major Advisor	Signature	Date
_____	_____	_____
Co-advisor	Signature	Date

APPROVAL OF BOARD OF REVIEWERS

We, the undersigned, members of the Board of Examiners of the thesis by **Aregay Nigusie** have read and evaluated the thesis entitled "**Machine Learning to Detect and Prevent Sinkhole Attacks in Cluster-based Routing Protocol of WSNs"** and examined the candidate during 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 is 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

First of all, I would like to thank my advisor **Dr. Ketema Adere** for his guidance and advice until the end of this work. Throughout, he provided me with all the support I needed for this thesis work. This thesis work wouldn't have been possible without his endless support and creative ideas from him. I would like to offer my thanks to all my friends here who have had good and constructive suggestions for my work.

Table of Contents

List of figures

List of Tables

List of Acronyms

ABR	Associatively Based Routing
ADC	Analog to Digital Converter
AODV	Adhoc On-demand Distance Vector
APDR	Average Packet Delivery Ratio
AUC	Area under Curve
BS	Base Station
CM	Confusion matrix
CPU	Central Processing Unit
CSV	Comma Separated Value
CV	Cross-Validation
DOS	Denial of Service
DSR	dynamic source routing
DT	Decision Tree
EED	End-to-End Delay
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GEAR	Geographic and Energy-Aware Routing
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
KNN	K-nearest Neighbor
LEACH	Low-Energy Adaptive Clustering Hierarchy
LR	Logistic Regression
MAC	Medium Access Control
MANET	Mobile Ad hoc NETwork
MECN	Minimum Energy Communication Network
ML	Machine Learning
NB	Naïve Bayes

NS3	Network Simulator version 3
QoS	Quality of Service
RAM	Random Access Memory
RERR	Route-Error
RF	Random Forest
RREP	Route-Reply
RREQ	Route-request
SMECN	Small Minimum-Energy Communication Network
SSA	Stability based Adaptive
TDMA	Time Division Multiple Access
TN	True Negative
TORA	Temporally-Ordered Routing Algorithm
TP	True Positive
WSN	Wireless Sensor Network

ABSTRACT

The Wireless Sensor Network (WSN) is a collection of tiny sensor nodes that can collect and process data and transmit it to the base station. Medical, Military, and Agriculture are some of the applications of WSNs. Due to the nature of wireless sensor networks; nodes are highly vulnerable to many security threats. Among those threats is the sinkhole attack. A sinkhole is a malicious node that advertises the best path to a base station and interferes with the data to modify and drop packets, it is also used to launch other attacks on the network such as selective forwarding and wormhole attacks. For this reason, there is a need to strengthen the security of wireless sensor networks. A machine learning-based intrusion detection system is an essential security tool to protect wireless sensor network infrastructure and services from these unpredictable and invisible attacks. Previous studies of machine learning have been proposed for intrusion detection in wireless sensor networks and have achieved reasonable results. However, these works still need to be more accurate and well-organized against imbalanced data problems in network traffic. In this paper, an efficient machine learning-based Intrusion Detection System for WSNs is proposed to detect and monitor the network activities against wireless sensor network routing attacks (Sinkhole, Blackhole, Flooding, and TDMA) more efficiently. We first describe the challenges in detecting sinkhole attacks in wireless sensor networks; followed by an analysis of methods to classify and detect sinkhole attacks including the above-mentioned related network routing attacks. Naive Bayes, Logistic Regression, Decision tree Random forest, and K-nearest neighbors are used as a classifier under different conditions to increase the performance of our model. The WSN-DS dataset is used to train and test our proposed models After experimenting with various combinations of supervised classifiers, we have found that the detection rate of the proposed attack detection system using Random Forest was obtaining an accuracy of 99.52%, Precision 99.74%, Recall 99.72%, and F1-Score 99.73% respectively. The experimental outcome we got an accuracy of 99.27%, precision 99.87%, Recall 99.32%, and F1-Score 99.60% has done after using SMOTE method which solves the data imbalance problems that happened in the dataset. Based on the finding, this research concludes that the Random forest classifier is better than the other classifiers and recommends that network infrastructure managers customize and upgrade their intrusion detection systems (IDSs) by employing this classifier.

Keywords*:* ML Algorithms, *IDS, Sinkhole attack, LEACH, WSN, Random Forest*

CHAPTER ONE

1 Introduction

1.1. Background of the Study

A wireless sensor network (WSN) is a collection of sensor nodes that are capable of sensing and processing data and then sending back the collected data to the Base Station (BS). These sensor nodes can communicate among themselves using radio signals. When sensors are deployed in the network, they automatically configure themselves and thereby forward the data to the Base Station. The Base station (the gateway node) is typically a gateway to another network, which is treated as the most resource-rich node, a powerful data processing, and good storage capacity (Waoo & Tiwari, 2021), (Carlos-Mancilla et al., 2016)

Wireless sensor network technology is prominently being used in different and variety of applications for example, in military activities, fire detection mechanisms, healthcare, and detecting environmental conditions like temperature, sun rays, sound, disaster, prediction, and so on. (Hidoussi et al., 2015) . Unfortunately, it is found that most of the wireless networks are actually deployed and working in not-friendly, unsecured areas. Due to resource constraints and limitations which contain low memory storage, low computational power, limited energy capacity, low power supply, processing capacity, and low communication range (Bhushan & Sahoo, 2018), the network is exposed to different malicious attackers that create trouble by attacking the networks easily. One such attack is the sinkhole. It is one of the severe routing attacks which have the capability of attracting the whole network traffic and modifying secret information, by making changes to data packets and dropping them before they reach the determined destination node. It also occurs by broadcasting fake shortest path routing information and gradually drawing all network traffic towards itself (Tiwari & A. Waoo, 2020). A sensor node's lifetime is determined by the battery that supplies energy to the memory unit, sensing unit, and transceiver. Hence, it is important to use an energy-efficient routing protocol that will increase the overall lifetime of WSNs (Radoglou Grammatikis et al., 2019), (K. Kaur & Singh, 2017), (Khan et al., 2021).

The routing protocols can be classified into three categories block-based routing, hierarchical routing, and location-based routing. LEACH is a hierarchical-based routing protocol that uses a random rotation of the nodes required to be the cluster heads to evenly distribute energy consumption in the network (Sehrawat et al., 2018). The LEACH protocol is the most efficient clustering protocol. The enhancements made in LEACH are achieved by making changes in the technique of routing, and the target of almost all enhancements is to optimize energy utilization and maximization of the overall lifetime of the sensor network (A. Kaur & Grover, 2015). The LEACH protocol consists of two parts: namely the setup and the steady state part. Cluster heads are elected during the setup phase, while data packets are transferred from cluster nodes to cluster heads and from cluster heads to the base station during the steady-state phase (Kumar et al., 2017).

Security is a challenging issue in the wireless platform and a major issue in the resource-constrained WSN (Vishwas et al., 2016). Sensor networks are subject to several attacks at each layer. The physical layer attacks are jamming, and tampering, Data link layer attacks are Jamming and collision; the Network attacks are selective forwarding attacks, sinkhole attacks, Sybil attacks, black hole attacks, and wormhole attacks; the Transport layer attacks are flooding attacks, de-synchronization attack. One of the hazardous and very difficult-to detect attacks is a sinkhole attack because using this attack we can perform any type of attack in the WSN. (R, 2014).

1.2. Motivation of the study

Any kind of network, whether it is wired or wireless needs to achieve basic security features including the availability of network services, confidentiality, and integrity of the data. Wireless communication networks such as MANET, VANET, and WSN, are highly vulnerable to different types of security threats or attacks compared to wired networks. This resulted, because of their features like open medium, dynamic changing of topology, and lack of central monitoring and management. WSNs face several kinds of routing attacks. A sinkhole attack is one of the severe attacks that occur in a wireless ad hoc network which has the capability of attracting the whole network traffic and modifying secret information, by making changes to a data packet or dropping them before it reaches the destination node.

Based on some drawbacks associated with the existing studies to protect against a sinkhole attack in WSNs, This work is motivated based on the issues: of low detection accuracy and performance of the existing IDS. Besides, feature selection and some best parameter selection methods are provided to enhance the detection accuracy and performance of the previous works.

1.3. Statement of the Problem

Since the growth of wireless communication networks is gaining strength with an increasing number of applications, they can easily be applicable everywhere. However, the network layer is more vulnerable to a variety of attacks than all layers. The network layer attacks are the most sophisticated attack that destructs the normal operation of the network. The performance will be reduced if a sinkhole attack with additional related attacks (Blackhole, Wormhole, and Flooding attack) has occurred. They will interrupt normal network processes, increase delay, rise routing load, decrease throughput, lose packets, and consume excessive power are some of its consequences (Amrita & Ahmed, 2012).

Because of limited power sources, limited memory storage, limited computational capability, and transmission range, the WSNs become exposed to the above-stated threats. The detection of these threats has become a critical issue in wireless sensor network security. An intrusion is an activity that tries to compromise the normal functioning of a wireless network

Most research studies that have been done regarding the detection and classification of sinkhole attacks such as redundancy mechanism, hop count, and trust management techniques do not take into account the effect of power consumption which will be used by attackers as the main gate to enter into the network path. Additionally, the existing Intrusion detection systems still have their shortcomings. Some of them are low detection rates and high training time. To overcome these challenges, in recent years, there have been various attempts to propose efficient IDS. But still, there is a need for further improvement in these systems. This research is proposed to study and evaluate sinkhole attacks as a major threat with additional related routing attacks for the network which uses LEACH as a routing protocol. Using Intrusion detection system protection mechanisms by selecting the best parameters of ML algorithms can improve the performance of the models for making the wireless sensor network better secured

1.4. Research Questions

This research has proposed to answer the following research question:

1. How to analyze the impact of wireless sensor network layer attacks in addition to sinkhole attacks on cluster-based WSNs?
2. Which attributes are relevant to improve the attack detection model?
3. Which machine learning technique can effectively detect network layer attacks (Sinkhole, Blackhole, Flooding, and TDMA)?
4. How to improve the intrusion detection performance rate by removing imbalance distribution among classes using SMOTE method?

1.5. Objectives

1.5.1. General objective

The general objective of this thesis is to propose, implement and improve the wireless sensor network layer attacks (Sinkhole, Blaockhole, Flooding, and TDMA) detection system using machine learning-based algorithms to ensure the network infrastructure more secured

1.5.2. Specific objectives

The specific objectives of this research work are to:

- To review related literature studies
- To explore relevant datasets that will be used for model training and testing
- To implement a simulation setup based on the provided dataset
- To apply preprocessing tasks using one-hot-encoder on categorical data and scale up encoded data using standard scalar
- To design an effective ML classification model for better attack detection.
- To implement the proposed models for attack detection and classification
- To apply Synthetic Minority Oversampling Technique (SMOTE) to solve data balance problems that occurred in the given dataset
- To evaluate and select the efficiency of machine learning algorithms for our study.

1.6. Significance of the study

The security of wireless networks has become one of the hot research areas (issues) for the last couple of decades. Since there are numerous kinds of security attacks (threats) that needs to be addressed. Thus the significance of this thesis is to address the challenges that are faced in the detection of network layer attacks by using the proposed approach. It enhances the security performance of the wires sensor network routing path which uses LEACH-based routing protocols by protecting it from a sinkhole, black holes, and flooding attacks. As a result, we believe that this thesis paper fills the gap of the earlier studies by improving the detection accuracy of intrusion detection systems to detect a group of WSNs routing attacks

1.7. The expected outcome of the study

In WSNs, security is the primary issue to achieve and run network operations securely. Since a sinkhole attack is an active attack, it drops a large amount of data packets. To make better-quality security in Hierarchical WSNS under LEACH routing protocol, we have to minimize the impact of sinkhole attacks and other related network routing attacks using an anomaly-based IDS approach. The final result of this study is to create a risk-free routing path and improve security, performance as well as network reliability from sinkhole attacks using Machine learning models.

1.8. Delimitation Scope

1.8.1. Scope of the Study

However, there are many attacks in the wireless sensor network, this thesis was limited to detecting only sinkhole attack by inducing blackhole, flooding, and TDMA attacks. This paper of study is conducted to propose and implement five machine learning models by classifying the network traffic into normal and attack. Among different routing protocols, the LEACH protocol is chosen. Naive Bayes, Logistic Regression, Decision tree Random forest, and K Neighbors Classifier algorithms are the only algorithms that have been checked for the detection method.

1.8.2. Limitations of the Study

This study presents several limitations on a different phase and may not accommodate all security of attack vulnerabilities that happened in the WSN. It is limited only to Sinkhole attacks. Additionally, the tool used as a simulator, in the study depends on the computer we used which needs better processes, storage, RAM sizes, and good graphics to run it perfectly.
One of the limitations of this research work is that only one dataset (which contains network traffic data) is used to build models that can classify the input data into normal or attack. Also, one feature selection method is applied to the dataset to select relevant features for model building and testing

1.9. Ethical Consideration

This research respects the procedural ethics of research, thus researchers' work used as a reference in this paper does not copy the words as it is, and rather it takes the idea and then takes further analysis on the limitations to which it would fill the gap accordingly.

1.10. Organization of the Project

This project contains six chapters: Chapter 1 describes the introduction part of the project including the statement of the problem, objectives, scope of the study, methodology, and project organizations. The literature review and related works are discussed in Chapter 2. Chapter 3 discusses the research methodology dealing with all of the components required to build the project by including hardware, electrical and electronic materials. The proposed design of this study is described in Chapter 4. Design, simulation, practical demonstration results, and the overall operation of the study are included in Chapter 5. Conclusion, research contribution, and future work are all discussed

CHAPTER TWO

2. Literature Review and Related works

In this Chapter, a review of the literature on WSNs, the Architecture of WSN, sensor node components, Sinkhole attack, and an introduction is one of the vital tasks in WSNs called sensor node deployment in the clustering-based routing protocol is described. Furthermore, the most significant part is a description of previous work-related research on the recognition of sinkhole attacks on cluster-based routing protocols

2.1. Wireless Sensor Network

Generally, the WSNs can be divided into infrastructure-based and infrastructure-less wireless networks (ad hoc wireless networks). Infrastructure-based wireless networks rely on a device that provided service as an access point which is a node that acts like a bridge by interconnecting both wireless and wired networks. On the other hand, an infrastructure-less wireless network does not depend on any fixed infrastructure or access point to communicate with other wired networks. It is commonly known as an ad-hoc wireless network. The word ad-hoc can be translated as "not organized," to describe MANET (Mobile Ad Hoc Networking). Mobile Ad Hoc Networking is an interrelated system of nodes that communicate over a wireless link network. (Babaeer et al., 2020), (Raza et al., 2016). Wireless sensor networks have nodes that communicate with each other and that also send their data to a special node called the base station (or gateway), which in turn sends it to the end user. This communication is made possible through radio signals (Mathew & Terence, 2018) (Bhushan & Sahoo, 2018) (Anwar et al., 2015) (Babaeer et al., 2020).

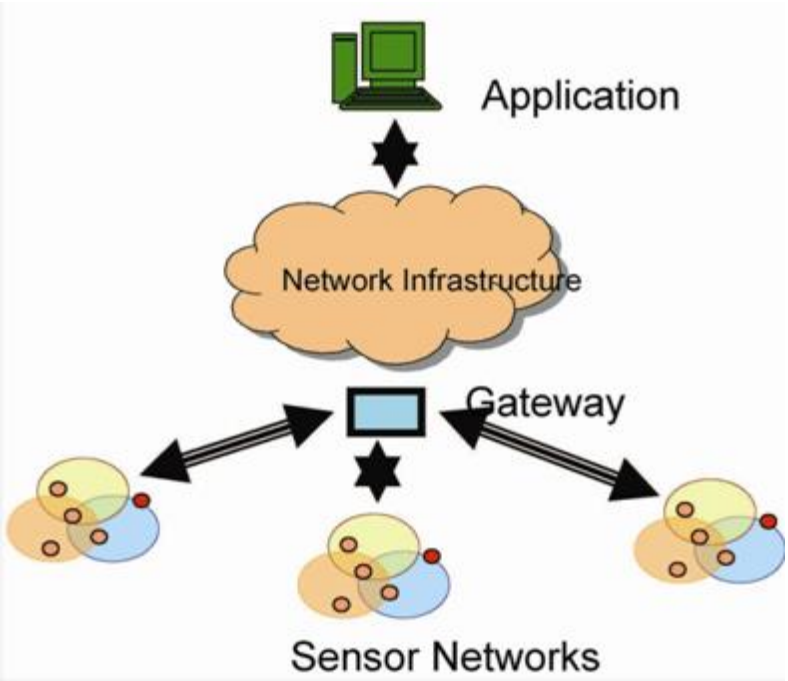


Figure 1 Wireless Sensor Network Architecture

2.1.1. Wireless sensor network Hardware components

There are four basic components in sensor nodes: sensing, processing, transceiver units, and a power unit (Babaeer et al., 2020).

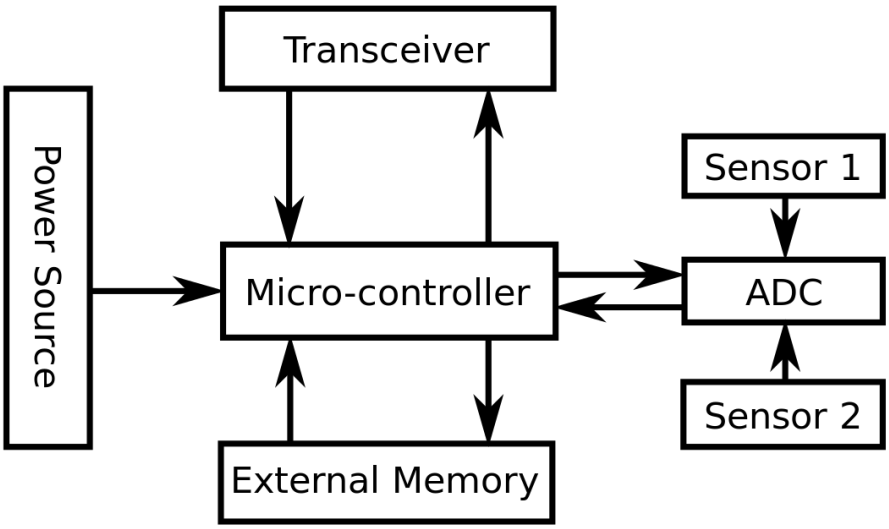


Figure 2 Sensor node Structure

Sensing units: Sensing units are basically of two types: (a) passive sensors like cameras and (b) active sensors like radar.

Processing unit: It includes a processor (microcontroller) and storage (RAM). The responsibility of the processing unit includes gathering information from numerous sources and then processing and storing it.

Communication unit: It uses a transceiver which consists of a transmitter as well as a receiver.

Power unit: The task of the power unit is to provide energy to the sensor The lifetime of the sensor nodes relay on the battery or power generator which is connected to the power unit. The power unit is required for the efficient use of the battery.

2.1.2. Applications of WSN

The WSNS is a free, spatially distributed listener that impresses physical objects or monitors surrounding natural data and collectively delivers data to base stations. The WSN is applied in several areas like agricultural accuracy, animal tracking, natural monitoring, safety and supervision, smart cities, health care, and so on

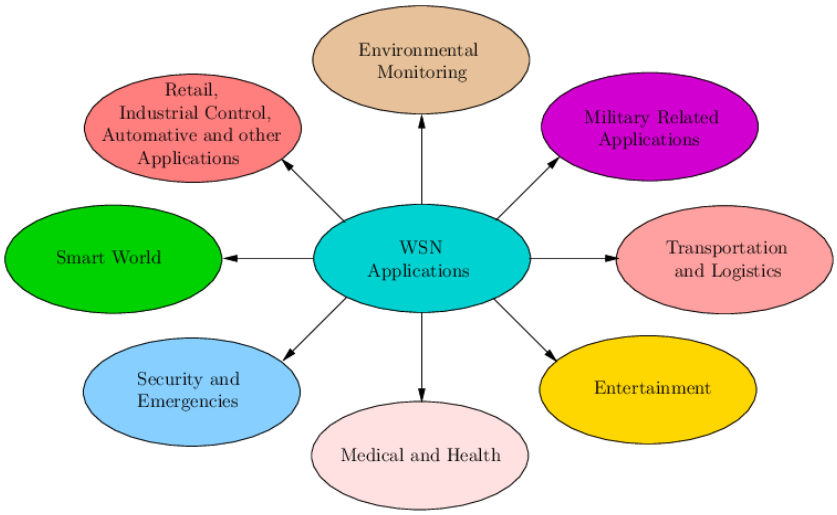


Figure 3: applications of Wireless Sensor Network

2.2. WSN Routing Protocols

Routing protocols describe how nodes communicate with each other and how information is distributed over the network. Ad-Hoc routing protocol determines the best route from the source to the destination node based on different parameters.

2.2.1. Location-based protocols

In location-based protocols, sensors are addressed by using their locations. Information location in sensor nodes is required for sensor networks by the routing protocols to calculate the distance between two particular nodes so that energy usage can be estimated. Sample of location-aware routing protocols proposed for WSNs: Geographic and Energy-Aware Routing (GEAR), Small Minimum-Energy Communication Network (SMECN), (Bhushan & Sahoo, 2019).

2.2.2. Data-Centric Protocols

In these protocols, when source nodes send their data to the sink, middle sensors can perform data aggregation from multiple source sensors and send the aggregated data to a sink. This process can result in power savings as less transmission is required to send data from the sources to the sink. Example of such protocol Energy-Aware Data-Centric Routing (EAD) (Singh et al., 2017). (Shi et al., 2015).

2.2.3. Hierarchical Protocols

The hierarchical routing protocol is the most effective method to decrease energy consumption, and it is separated into two layers. The primary layer is utilized to select an appropriate cluster head, and the second one is used for the routing procedure. Furthermore, merging and aggregating the data are performed to reduce the number of sent messages to the base station, (Ishaq et al., 2018)**.** The Low-Energy Adaptive Clustering Hierarchy (**LEACH**), the Energy Efficient Clustering Scheme (**EECS**), and the Base station Controlled Dynamic Clustering Protocol (**BCDCP**) are the most important protocols, which belong to the hierarchical routing protocol, (Dong & Liu, 2009)

2.2.3.1. LEACH (Low-Energy Adaptive Clustering Hierarchy)

LEACH developed by W. R. Heinemann et al. is the most popular hierarchical routing algorithm with energy efficiency, which has been proposed for WSNs. It uses a clustering technique that divides the network into two levels: cluster heads and member nodes, (Pawa, 2011).

LEACH protocol has a steady-state phase and a setup phase. In the steady-state phase, the cluster is formed based on the threshold value, where each node calculates its value by selecting a random number from 0 to 1 and broadcasts the value to its neighbors. For a node whose value is under the threshold limit will be nominated as the cluster head and the remaining nodes will join as member nodes, (Isaac Sajan & Jasper, 2020) (Xu et al., 2021).

2.2.3.2. Objectives of Clustering

***Scalability*:** When designing routing protocols for WSNs, scalability is very important. WSN can be scaled if we can add network nodes after design The routing protocol can be said to be efficient and flexible, with changes to the network topology from time to time ensuring good performance, (K. Kaur & Singh, 2017).

Data *aggregation*: The data aggregation collects helpful information from nodes, which helps for saving energy to extend the network lifetime of the nodes, (Rajeshkumar & Valluvan, 2016).

Collision Avoidance: This can influence the entire system's efficiency, so it should prevent collision to improve the network lifetime because excessive collisions result in losses, wasted resources, and retransmissions of data, (Samara et al., 2020)

Load Balancing: Performing clusters equal in size is important for improving network lifetime as precocious energy consumption of cluster heads is avoided and we should balance the load between CHs to achieve the required performance, where they are selected from available sensors (Kumar et al., 2017).

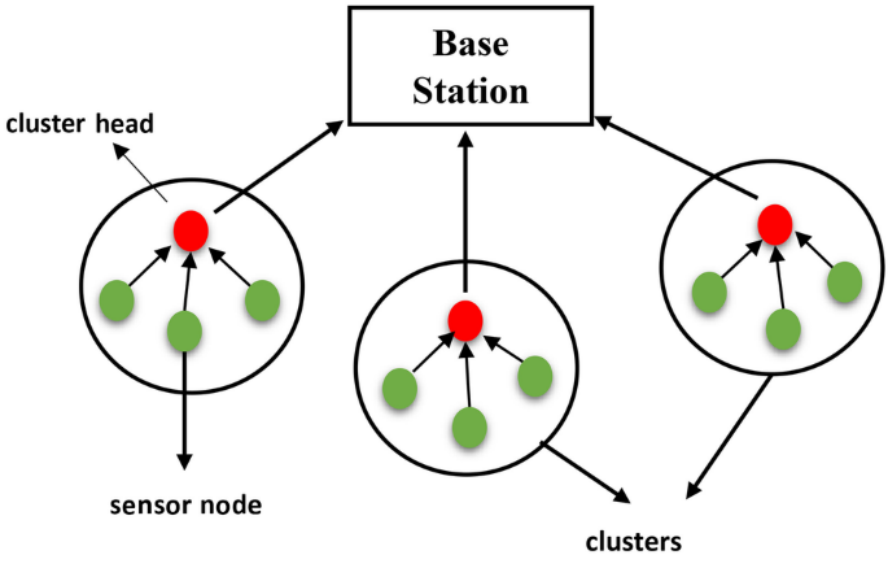


Figure 4: Wireless after clustering formation

2.2.4. Cluster formation

The decision of whether or not to move a node to the cluster header is made dynamically at each interval. The threshold function is defined as

$$T(n) = \begin{cases} \frac{P}{1-P(r \bmod \frac{1}{P})} & \text{if n} \in \text{G} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Where **n** is the given node, **P** is the a priori probability of a node being elected to be cluster head, **r** is the round number, and **G** is the number of nodes that have not been elected as cluster heads in the last 1 / P rounds. Each node generates a random number between 0 and 1 when choosing a cluster header. If the number is less than the threshold (T(n)), the node will become a cluster head. Following cluster establishment, cluster heads will create a transmission Schedule and broadcast the schedule to all nodes in their respective cluster. The schedule consists of TDMA slots for each neighboring node. This scheduling scheme minimizes power consumption since nodes can turn off the radio at any time except when scheduled.

2.3. Ad hoc On-Demand Distance Vector (AODV) Protocol

The AODV is a routing protocol that is categorized as an on-demand or reactive type of routing protocol in MANET. Exchange of AODV messages (Hello, Messages) can be used to detect and monitor nearby links. When Hello messages are used, each active node regularly sends a Hello message to all of its neighbors. Because nodes regularly send hello messages, a link failure is detected when a node stops receiving hello messages from a neighbor (Marina & Das, 2006) (Sehrawat et al., 2018). When a source has data to transmit to an unknown destination, it sends a route request (RREQ) for that destination. A path to the source is created at each intermediate node upon receipt of an RREQ. If the receiving node has not previously received this RREQ, is not the destination, and has no current path to the destination, it returns the RREQ. If the receiving node is the destination or has a current route to the destination, create Route Reply (RREP). RREP is unicast to the hop-by-hop source. During RREP propagation, each intermediate node builds a path to the destination.

2.4. Security in Wireless Sensor Networks

2.4.1. Security Principles in Wireless Sensor Networks

(Babaeer et al., 2020), wireless sensor networks are typically deployed insecure environments where security is a major concern and where these networks are vulnerable to external and internal attacks Because it is a large-scale network it is not possible to monitor and protect every node. Some of the security principles of wireless sensor networks are

Confidentiality: The transmission of data should be kept secret and not accessible to an unauthorized user

Authentication: The unwanted results can be protected by ensuring the identity of the node with which it is communicating.

Integrity*:* It ensures the correctness of the data i.e. the data is not altered by the intruder.

Availability: The service should be accessible all the time.

Freshness*:* It suggests that data should be new no old data should be replayed.

2.4.2. Wireless Sensor Newark Attacks

(Verma et al., 2021) (Yadav, 2018) Wireless sensor networks are susceptible to security attacks due to the broadcast nature of the transmission medium. These attacks are divided into two categories, namely passive and active attacks.

Passive attack: Monitoring and interception of a communication channel by unauthorized attackers are passive attacks.

Active Attacks: The unauthorized attackers monitor, listen to, and modify the data stream in the communication channel is known as an active attack. The following attacks are active. Selective Forwarding, Sinkhole Attacks, Wormholes Attacks, and HELLO flood attacks: Physical Attacks are some examples.

2.4.3. Attacks on Different Layers in Wireless Sensor Networks

(Anwar et al., 2015) (Jamil et al., 2021)**,** many different types of attacks can affect the network. The attacker's goal is to attract almost all traffic in the network through a malicious node. To achieve this, the malicious node is intentionally placed near the base station to discard or corrupt the received packets before sending them to the next node.

(Babaeer et al., 2020), The architecture of WSN is divided into five layers, which are the physical layer, data link layer, network layer, transport layer, and application layer. The malicious attacks corresponding to various layers of the WSN stack are listed in bellow table.

Table 1: Attacks on Different Layers

Layer	Attack
Transport Layer	Flooding, Desynchronization
Network Layer	Routing Attacks like a blackhole, Sinkholes, Wormholes etc.
Data link layer	Collision, Exhaustion, Unfairness
Physical layer	Jamming Attack, Tempering
Application	Cloning, Denial-of-service

2.4.4. Network layer attacks

According to (Rao et al., 2011), routing attacks are generally handled through key management and secure routing schemes although costly to implement on WSN due to the characteristics of sensors

Blackhole In a black hole attack, an attacker can physically capture and re-program a set of nodes in the network to block the packets they receive instead of forwarding them to the BS. Any information that enters the black hole region is then compromised, and the information does not reach the destination. This produces high end-to-end delay and a decrease in packet delivery ratio and network throughput. As a result, the information cannot reach the destination node within the required time (Dhanaraj et al., 2021).

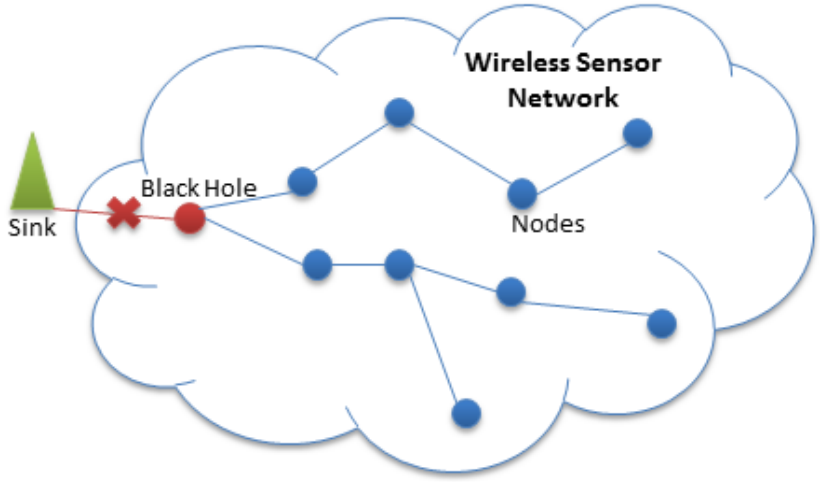


Figure 5 Blackhole Attack

Wormhole attack: This is an attack where two attackers strategically position themselves on the network. Attackers then continue keeping on listening on the network and record wireless information. The two attackers are placed in a strong strategic position in the wireless network In wormhole attacks, the attacker receives packets at one point in the network and forwards them to another part of the network, and then replays them down the network from that point onward, (Sharif & Ahmed, 2010).

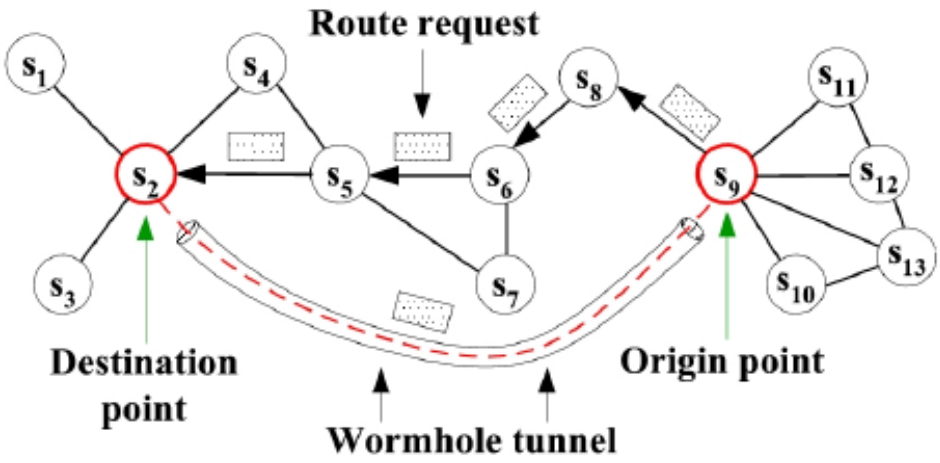


Figure 6 Wormhole Attack

Hello flood attack: In this attack, broadcasts hello messages, and declares itself as a neighbor node. When a sensor node receives this massage assuming that the sending node is in communication range, the node starts communicating with that node and makes an entry into its routing table as a neighbor. Thus, all nodes assume that the hello message path is the shortest pathway from the base station and that the attacker node is the base station in WSNs, and then start communication with that attacker node (S. Patil & Chaudhari, 2016).

Sinkhole Attack: It is an active security attack that is considered one of the severe security threats in WSNS. It is caused by the malicious node that tends to divert the whole network traffic to ward itself instead of the intended route. The Sinkhole node attracts network traffic by broadcasting fake routing information with a higher sequence number and lower hop count. As it receives an RREQ from the source node it creates a fake RREQ message, to pretend as if it is on the shortest route to specific nodes. Thereafter it alters the data packets or drops the packets silently and finally, it destroys the whole network. (Jamil et al., 2021) , (A. Patil et al., n.d.), (Kibirige, n.d.).

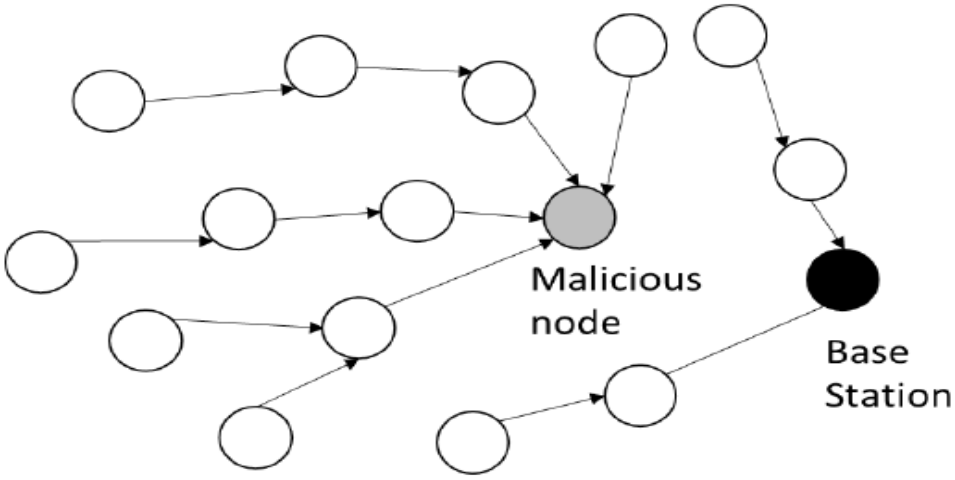


Figure 7 Sinkhole Attack

2.5. Machine Learning (ML)

Machine learning is the practice of computer programming to learn from data it is trained rather than explicitly programmed and it needs to learn from the historical data, optimize for better computations, and also generalize itself to provide proper results (Alqahtani et al., 2019). Currently, machine learning algorithms are used for classification, regression, clustering, and reduction tasks on large datasets with a large number of dimensions. As a result, the task of detecting network intrusions comes under ML as it involves the classification of data into normal and attack behavior. Algorithms used in machine learning can be classified based on the outcome of the algorithm and types of input fed during training as supervised, unsupervised, or semi-supervised learning (Sivasakthi & Pandiyan, n.d.).

2.5.1. Machine learning types

The main task of using machine learning techniques is to create a model that describes the relationships and dependencies between the characteristics of the input and the expected objective results, (Ifzarne et al., 2021). Machine learning techniques are categorized based on the output provided. Therefore four machine learning algorithm types such as supervised learning, unsupervised learning, and semi-supervised learning are examples of machine learning algorithm types (Ioannou & Vassiliou, 2019).

2.5.1.1. Supervised Learning

Supervised learning asks the machine to learn from our data when we specify a target variable. It is a process by which a function is derived with the help of labeled data samples. The dataset used in the supervised algorithm is classified into a training dataset and a testing dataset, which helps us to predict the outcomes of the data, (Ioannou & Vassiliou, 2019). The algorithm analyzes the training data and produces a function, which will be used for input-to-output mapping. A supervised learning model has two major tasks to be performed, classification and regression.

Classification: Classification is about predicting output labels (Classes) or categorical responses to the input for what the model learned during the training phase A classification problem can be binary classification (with two classes) or multiclass classification when there are three or more classes. Common algorithms for performing classification are Logistic Regression, Random Forest and gradient boosting, k-nearest neighbors, Decision Trees, Naïve Bayes, and so on (Ikonomakis, 2005).

Regression: in contrast to classification, regression is concerned with predicting the numeric value for the class label. Regression models use input data features and continuous numeric output values to learn specific relationships between the inputs and corresponding outputs, (Ioannou & Vassiliou, 2020).

2.5.1.2. Unsupervised Learning

In this method, the algorithm is provided with an unlabeled dataset and it predicts a pattern in the data. It doesn't have a target variable as we did in classification and regression. In contrast to supervised learning, there are no explicit target output labels associated with each input. It is used to group unsorted information based on patterns, similarities, and differences without any prior training data (Yau et al., 2019). Unsupervised learning contains two major types: Clustering and Association.

2.5.1.3. Semi-supervised Learning

Semi-supervised learning is another type of machine learning that combines both labeled (supervised) and unlabeled (unsupervised) data. This type is used in a small amount of labeled data with large unlabeled data

2.6. Evaluation Matrices

Evaluation metrics are used to measure the quality of the machine learning models or algorithms of any project. There are many different types of evaluation metrics available to test a model. These include classification accuracy, confession matrix precision, recall, and fl-score, and others (Qassim et al., 2016). All the classification metrics that we mentioned here are based on actual labels and predicted labels of samples. The actual labels of samples are already available in the dataset. However, predicted labels are obtained using the model.

2.6.1. Confusions Metrics

The confusion matrix is an N x N matrix used to evaluate the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values to those predicted by the machine-learning model, (Alqahtani et al., 2019).

Each cell in the confusion matrix represents one of the TP, TN, FP, and FN outcomes of the prediction results of a model and each column in it represents classified instance counts based on predictions from the model, and each row of the matrix represents instance counts based on the actual class labels.

Table 2 Confusion matrix for binary classification

		Predicted Class	
		Normal	Attack
Actual Class	Normal	TP	FP
	Attack	FN	TN

- True Negative (TN): number of normal instances that are actually normal and the model predicted as normal
- False Positive (FP): number of normal instances that are actually normal but the model incorrectly predicted as an attack
- False Negative (FN): number of attack instances that are actual attacks butthe model incorrectly predicted as normal
- True Positive (TP): number of attack instances that are actual attacks and the model correctly predicted as an attack

2.6.2. Accuracy

Accuracy is the percentage of the right predictions performed by the model. In other words, it measures how the classifiers make the correct prediction. Accuracy will take value in the range of [0, 1] (Bhushan & Sahoo, 2019). If accuracy is equal to 1 that means all samples in the dataset are correctly classified. In contrast, if accuracy is equal to 0 that means none of the samples in the dataset is classified correctly (S. Patil & Chaudhari, 2016).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (2)$$

2.6.3. Precision

Precision measures the amount of actual positive cases out of all the positive cases predicted by the model based on the positive class. This is also called positive predictive value. The formula for calculating precision is as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (3)$$

2.6.4. Recall

A recall is the percentage of actual positive cases that the model can predict correctly out of the total number of positive cases. In our case, how correct is the classifier to classify all available attack instances as ‘ATTACK’

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4)$$

2.6.5. F1- score

F1-score denoted is the harmonic mean of the values of precision and recall. F1- score close to 1 means a perfect model. However, the f1- score close to 0 shows a low model's predictive capability. F1 - score of a model is calculated

\

$$\text{Fmeasure} = \text{F1} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \quad (5)$$

2.6.6. FPR

FPR also called false alarms or specificity; determines the number of incorrect predictions among all negative samples in the dataset. FPR is defined by

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TP}} \quad (6)$$

2.7. An intrusion detection system (IDS)

Intrusion is a malicious activity that compromises the security of a network or computer system either externally or internally without mandatory privileges (Vijayarajeswari & Priya, 2015). Therefore, detecting network attacks is an important step in preventing such attacks The IDS monitors the malicious activity in the network by using a KB and inference engine. The KB located in the BS stores data collected by the CH over the network, which the CH accesses using an inference engine. In this case, the inference engine is responsible for generating rules for the knowledge base to do this job. CH monitors the activity of the node and delivers the collected data to BS, If a malicious node is detected, the IDS sends a warning message along with the data regarding the malicious node to the respective CH. IDS-based systems are very effective in detecting the anomalous activity of internal network nodes and protecting the entire network from various types of malicious attacks, (Patel & Gheewala, 2015) (Mehmood et al., 2017) (Isaac Sajan & Jasper, 2020).

2.8. Related Words

This section presents and briefly discusses the research work done so far related to this thesis paper along with the methods, tools, and techniques used, the results obtained, and the challenges of these methods.

(Samara et al., 2020), proposed about Energy-Efficiency Routing algorithms in WSNs: The article discussed WSNs by considering how much important is wireless nodes' power consumption. The author pointed out items that make the node consume its energy like Collision: which happens if a node receives a lot of packets simultaneously. Idle Listening: A significant reason for wasting power occurs when a node listens to an idle traffic channel.

(Karami, 2018), the author proposed a system to identify anomalies accurately and to provide visualized information to end-users using ML techniques. The author's main goal was to achieve a higher detection rate and low false alarms. The paper of the proposed model was evaluated on NSL-KDD, UNSW-NB15, AAGM, and VPN-nonVPN datasets. Finally, the experimental results showed that the proposed method provided better lattice adjustment with a few overlapped connections among neighbors. However, it failed on connections between nodes andtheir neighbors from the other two methods

(Babaeer et al., 2020), this study focused on the impact of sinkhole attacks on cluster-based wireless Sensor networks and the way how detects based on LEACH protocols to make the network secure from attack and described how LEACH protocol is different from the other cluster methods. The election of cluster head and the communication process among cluster with nodes, cluster to cluster as well as cluster to the base station was proposed by the author

(Vijayarajeswari & Priya, 2015), have developed a system based on various ML techniques to detect and analyze distributed denial of service attacks (DDoS). They designed an anomaly-based attack detection system using four ML techniques (DT, Naïve Bayes, multilayer perceptron, and SVM). Their report showed that a multilayer perceptron achieved the highest accuracy rate. Even though the authors collected and labeled the dataset, the dataset is restricted to only two layers of a network (network and application layer).

(Ali et al., 2020), proposed the impact of Sinkhole Attacks in Wireless Sensor Networks. Communication in sensor nodes is based on multiple hops, and each sensor has limited resources such as memory, power, data rate, communication range, and processing power. These limitations make the network very vulnerable to network attacks and make it difficult to implement security measures against detection and prevention mechanisms. The wireless sensor network protocols have inherent limitations due to limitations built into wireless devices

(Kushwaha et al., 2017), planned some feature selection algorithms and developed IDS using SVM. They carried out feature investigation using various feature selection algorithms. In this approach, a filter method was used for feature selection and a K-nearest neighbor algorithm for dimensionality reduction. They also compared the results obtained by these algorithms and ranked them according to the score obtained corresponding to each attribute. They used various classification algorithms and compared their accuracy and finally, they found that the SVM realized an accuracy of 99.91% for multiclass classification.

(Nithiyanandam & Latha, 2019), this paper addresses the problem of sinkhole attack detection in WSN using a colony algorithm. This algorithm finds the discredited node by associating the node IDs assigned in the rule set. This algorithm decreases the overall participation complexity, necessitating finding the hazarded node, minimizing the receptacle loss percentile, and improving the packet delivery ratio (PDR)

(Alajmi & Elleithy, 2016), proposed a rule-based classification model for intrusion detection using data mining frameworks. The main goal of the proposed research work was to test different rule-based classifiers for IDS and to select the best one. But the work was not able to classify new attacks (attacks in the test dataset but not available in the training dataset).

(Gupta & Kulariya, 2016), proposed a fast framework to detect network attacks. They used five ML techniques (LR, SVM, RF, Naïve Bayes, and gradient boosted tree) using Apache Spark to determine whether network traffic is an attack or normal. The authors used two feature selection methods, namely correlation-based feature selection (CFS) and chi-squared feature selection. The proposed framework used two real-time network traffic datasets namely KDD-CUP and NSL-KDD. Two experiments were carried out. At first, the major weakness of

this approach is that the chi-squared method decreased the accuracy of the classifiers even though processingtime is improved

(Qassim et al., 2016), this research developed an anomaly-based network intrusion classifier to automatically classify activities from the Internet. The authors used a packet header-based anomaly detection system for network traffic collection. From the collected network traffic, they extracted features but the methods used for feature extraction were not stated

(Naseer et al., 2018), presented a deep learning model to perform anomaly detection on network traffic data. One of the uses of deep learning is its ability to learn raw input features without a need to preprocess them. The proposed approach aimed to train a deep neural network (DNN) model with raw input data. They built a model called Raw Power multi-layer DNN for network analysis. The model was built on Keras along with the TensorFlow framework on the USTCTFC2016 dataset. Furthermore, the proposed model was compared with the RF algorithm by using the same input features and their result showed that RF gained better results with 70% detection and below 3% FAR. But the model performed poorly on non-processed input data.

Table 3 Summary of Related Works

Author & Year	Title	Objectives	Dataset/ Classifiers	Gap
(Ahmad et al., 2022)	An Efficient Network Intrusion Detection and Classification System	Building an Intrusion Detection and Prevention System to mitigate the effect of the Denial of Service attack	UNSW-NB 15	Its result only depends on ANN SVM and AdaBoost models for comparison to detect nine intrusion threats
(Rezvi et al., 2021)	Data mining approach to analyzing intrusion detection of wireless sensor network	detect denial of service (DoS) attacks using data mining technique	WSN-DS	It did not explicitly set the percentage of testing and training samples

(Mighan & Kahani, 2021)	A novel scalable intrusion detection system based on deep learning	Address scalable network intrusion detection in big data framework	UNB ISCX12	No Data balance techniques have been used
(Ifzarne et al., 2021)	Anomaly Detection using Machine Learning Techniques in Wireless Sensor Networks	intelligent intrusion detection model based on incremental machine learning	WSN-DS	Its correctness depends on the threshold selection model of ID-GOPA and the detection result of each attack does not exceed 96%
(Hidru, 2020)	Modeling Network IDS Based on Anomaly Approach Using ML Techniques	Design and implement a model for the classification of network traffic based on the anomaly approach using machine learning techniques	NSL-KDD	Limited to cross-checking of its result for the models used. like SMOTE or Under sampling techniques
(Al-Issa et al., 2019)	Using Machine Learning to Detect DoS Attacks in Wireless Sensor Networks	Building an Intrusion Detection and Prevention System to mitigate the effect of the Denial of Service attack	WSN-DS	- Its analysis is limited to two algorithm models - Uses WEKA for data analysis
(A & Sundarakantham, 2019)	Machine learning based intrusion Detection system	Improve the performance of ML- based IDSs	NSL-KDD	Its result depends on the two models called SVM and Naïve Bayes
(Hidru, 2020)	Modeling Network IDS Based on Anomaly Approach Using ML Techniques	Design and implement a model for the classification of network traffic based on the anomaly approach using machine learning techniques	NSL-KDD	Limited to cross checking of its result for the models used. like SMOTE or Under sampling techniques

CHAPTER THREE

3. Research Methodology

3.1. Introduction

This chapter discusses the research methodology. It begins with a methodology for how the data were collected and processed alternatively. The following section in this chapter explains and justifies the methodology used in conducting the study. To attain the objectives of the research, an in-depth literature review on malicious wireless sensor network attack classification and related works was done. Articles, thesis papers, electronic materials, and different open-source tools have been used to collect the necessary information for this study. Based on the information obtained from the reviewed literature, the input analysis, underlying representation, and output representation techniques and tools are selected. The techniques and tools are chosen by considering various factors like their effectiveness in similar previous works.

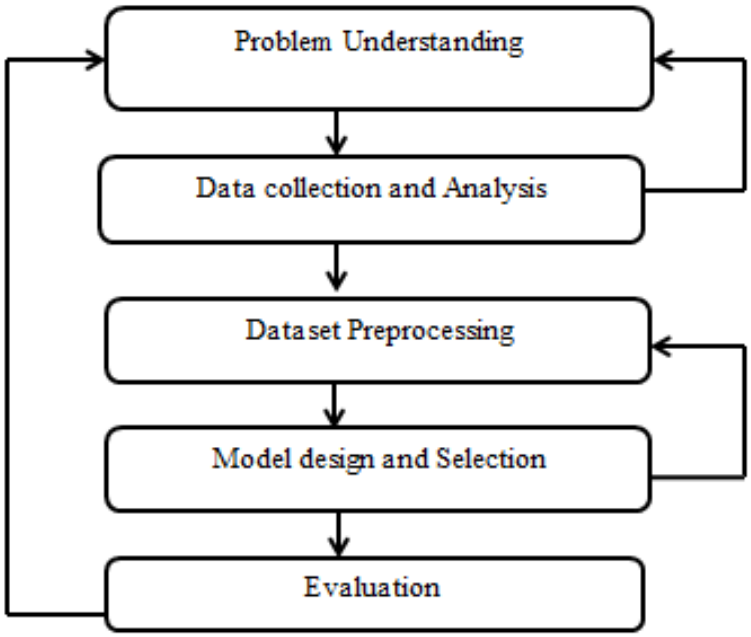


Figure 8 Research methodology

3.2. Literature Review

A detailed review will be done of journal articles, conference proceedings, books, dissertations, and the Web to have a deep understanding of the previous works in the study area. Different methods of techniques in network intrusion detection will be examined. From the review of these works, appropriate techniques and methods have been selected.

3.3. Data Collection

For implementing the proposed model, dataset samples that include malicious attacks were collected from GitHub (Shen et al., n.d.), which relevant to our research study has been done.

3.4. Design Proposed Model

Designing the proposed is the procedure of defining the architecture, flowchart; block diagram ad other diagrams that satisfy the specified research process. Proposed design prototypes help to be seen as the application of solutions in the design aspect.

3.5. Tools and Development Environments

The system designed for this research is implemented using various tools and methodologies. One of the tools we sued is: Jupyter Notebook, a laptop computer, the Scikit-learn website, and Anaconda Navigator (anaconda3)

Laptop computer

The PC used to conduct this thesis is as follows;

- Processor: Intel(R) Core(TM)i3-4510UCPU@2.00GHz 2.60GHz
- Installation Memory(RAM): 4:00GB
- System Type: 64-bit Operating System
- Hard Disk:1TB External and Internal

Anaconda Navigator

It is a graphical user interface built in Anaconda distribution which is a free Python distribution developed by Continuum Analytics that is made for scientific computing Without using command-line tools, Anaconda Navigator enables to run python programming codes and manage conda packages environments, It works across Windows, mac OS and Linux platforms (Rolon-Mérette et al., 2020). For this research work Anaconda version, 22.11 is used.

Jupyter Notebook

Jupiter Notebook is an application that allows notebook pages to be edited and run in a web browser. It contains a "Dashboard" (Notebook Dashboard), and a "control panel" and it is used to open notebook documents or shut down their kernels, in addition, it displays, edit, and run notebook documents

Spyder 5.2

Scientific Python Development Environment (Spyder) is a powerful scientific environment for scientists, engineers, and data scientists to build Python applications. It offers advanced editing, analysis, and debugging comprehensive development tools with data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of scientific packages. For this research work, the latest version Spyder 5.2 is used.

- **NumPy:** is a library supporting large multi-dimensional arrays and matrices.
- **Matplotlib**: plotting library for data visualization (for producing line charts, histograms, scatter plots, and so on)
- **Scikit-learn:** an open-source ML library in Python, which provides useful helper functions and infrastructure. It offers a set of efficient tools for Machine Learning and statistical modeling, such as classification, regression, and clustering via a Python interface.

3.6. Machine Learning Algorithms

A good starting point for this article is the basic concept of machine learning. Machine learning is a type of artificial intelligence that enables software applications to make more accurate predictions without being explicitly programmed to do so. Machine learning algorithms use historical data as inputs to predict new output values (Ray, 2019).

3.6.1. Logistic regression algorithm

Logistic regression is a model for predicting data values based on past observations of a data set. It gives the binomial outcome as it gives the probability that an event will occur or not (in terms of 0 and 1) based on the values of input variables. The characteristics of logistic regression are easy implementation, computational efficiency, learning efficiency, and easy regularization. No scaling of input functions is required. This algorithm is mainly used to solve problems on an industrial scale (Charbuty & Abdulazeez, 2021)

3.6.2. Decision tree

It is a Supervised Machine Learning approach to resolve classification and regression complications by continuously splitting data based on a certain parameter. In Classification Tree, the decision variable is categorical (outcome in the form of Yes/No) and in the Regression tree, the decision variable is continuous. The decision tree method of machine learning makes decisions at different levels using a tree data structure It is Suitable for regression and classification problems, easy to interpret, easy to handle categorical and quantitative values, possibility to fill in missing values in attributes with the most probable value, high performance due to the efficiency of the tree traversal algorithm (Ben-Haim & Tom-Tov, 2010)

3.6.3. Random Forest

The Random Forest classifier creates multiple decision trees from a randomly chosen subset of the training data set. It then aggregates the votes from the different decision trees to determine the final class of test subjects. It is more accurate than the decision trees due to the reduction in over-fitting (Iwendi et al., 2020)

3.6.4. K-Nearest Neighbor

It is simple in its implementation and robust to noisy training data. Even if the training data is large, it is quite efficient. It finds similarities between predefined classes and the classes to be classified using Euclidean distance. (Shah et al., 2020)

3.6.5. Naïve Bayes

The Naïve Bayes classifier calculates conditional probability using the Bayes theorem to divide into different classes. This theorem depends on the naïve assumption, in which input factors are independent of each other. It is extremely fast compared to other classifiers. (Shah et al., 2020)

3.7. Machine Learning Model Build

Machine learning (ML) techniques are considered to be one of the important approaches that could be used in intrusion detection systems to enhance their abilities to identify and recognize attackers. The ML classification method has been used for DoS detection in WSN

The planned system architecture consists of different modules. The first activity to be processed is to prepare the datasets for model building by loading the datasets from the local disk and the next, feature engineering carries out together static and dynamic output results to extract the feature in the form of the features vector. Thus each source code is represented as feature vectors. Finally, the classification model is trained on all feature vectors constructed by feature engineering that maps the python code with their features to the target class by using learning algorithms finally the model is ready for classification hence, this study builds a classification model mainly by using a machine learning classifier algorithms

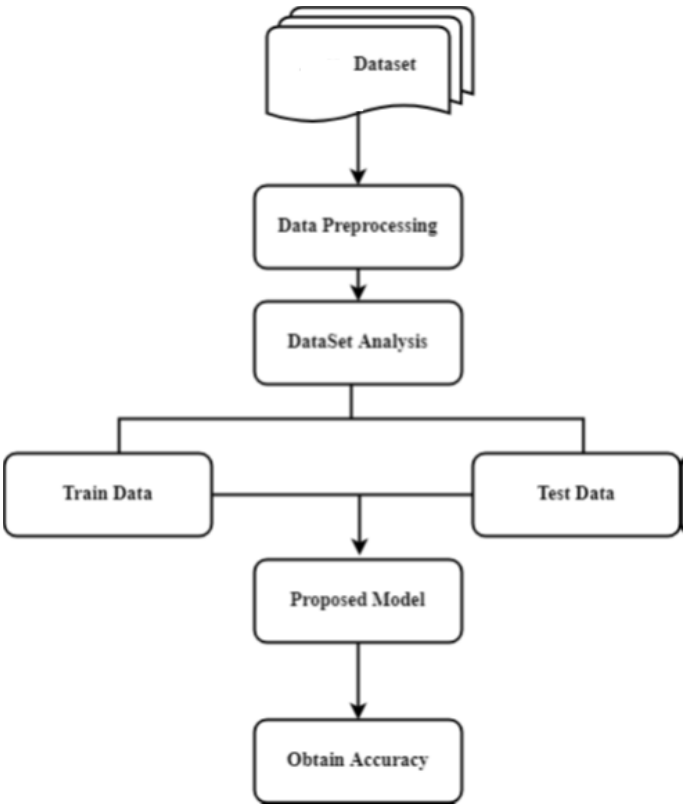


Figure 9: Model building flow diagram

3.8. Models Evaluation, Validation, and Testing

Every trained model will be tested on new data that is unseen during the training to determine how well it classifies the samples correctly. The efficiency of the models will be evaluated using classification models evaluation metrics. To test and estimate the learning ability of machine learning models. The basic concern of the machine learning model is to find an accurate estimation of the generalization error of the trained models on finite datasets. Hence we used train-test split methods and K-fold cross-validation (CV) methods. In the train-test split, the given dataset is separated into two parts: the training dataset is used to train the model and the test dataset is used to evaluate the model. In cross-validation, we used 10 folds in which the dataset is randomly divided into ten equal-sized parts. At each iteration, one single part is taken as the testing data, and the other parts are used as training data. The final result is the overall average of each iteration. For performance evaluation, we used different metrics that are appropriate for models such as confusion matrix, accuracy, precision, recall, and F-Measure.

CHAPTER FOUR

4. Proposed Solution

4.1. Introduction

This Chapter is all about the proposed system. The proposed architecture is given then the data collection and preparation are discussed. The data preprocessing, train-test split, validation split, algorithm training with default parameters, and grid search phases are presented next. Finally, model performance evaluation metrics for the five classes are described.

4.2. The System Architecture

As shown in Figure 9, the system architecture designed for this research is categorized into different phases It includes preprocessing, train-test split, validation split, and soon. The preprocessing phase consists of sub-tasks such as string indexing to convert string data to numeric, one hot encoding to convert string indices into 0s and 1s, feature selection to select relevant features, and normalization for standardizing the range of values. In the train-test split phase, the preprocessed data split into a training set (for training ML models) and a testing set (for evaluating models' performance). In the validation split phase, the training set is further split into actual training and validation set.

The trained model has then evaluated for its performance on the separated preprocessed testing data and the evaluation result is generated. The grid search phase is useful to find the best parameters for a model by constructing a parameter grid. The cross-validation searches the whole parameter grid and runs the training set for selecting the best parameters. Then, the models are evaluated on the validation set. The selected best parameters are then used for building the final trained model which is evaluated on the testing set. The result of this evaluation is evaluation result.

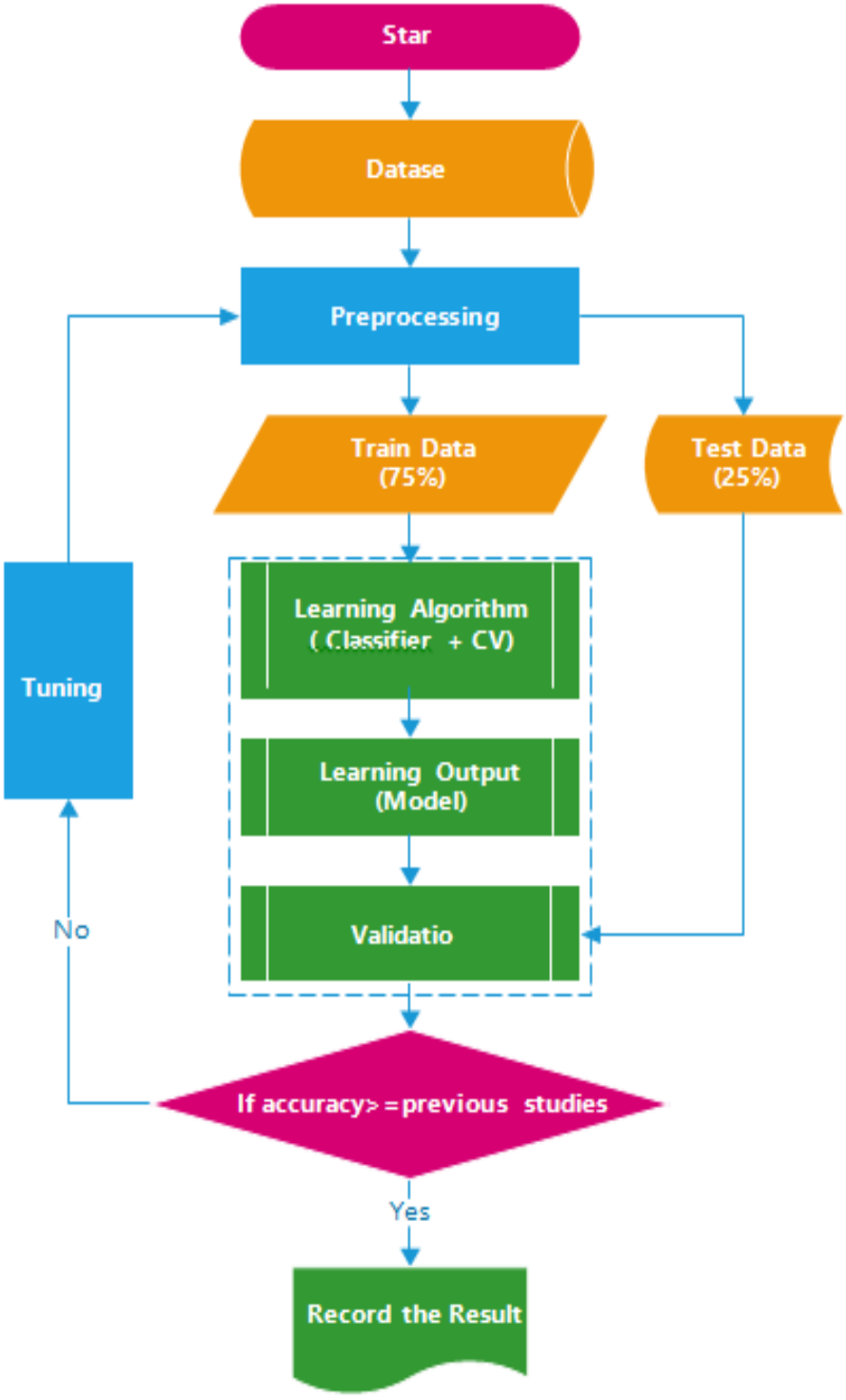


Figure 10: System Architecture

4.3. Proposed method

This paper has proposed based on the data mining techniques to detect the WSNs network routing malicious attacks, in our case the sinkhole attack as well as, other related attaches included in the provided dataset, Such as Blackhole, Flooding, and TDMA. For this research study, we applied Naive Bayes, Logistic Regression, Decision tree Random forest, and K Neighbors Classifier algorithms to see which algorithm can detect intrusion more accurately.

After the dataset has prepared, we performed preprocessing for the appropriateness of our experiment. The dataset has been split into 75% training and 25% testing respectively; also we used 10-fold cross-validation. Then we have chosen different ML classification algorithms to detect intrusions. Several standard metrics have been used to measure the result obtained from the experiment. The experimental outcome analysis can tell important information about the types of algorithms and their suitability for detecting various types of attacks. The proposed framework is illustrated in Figure 10

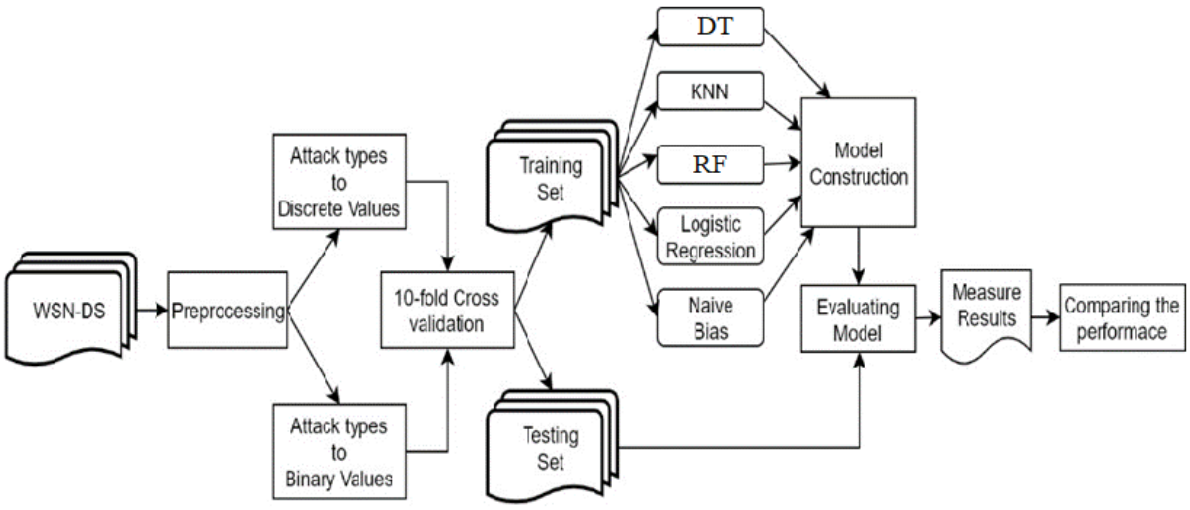


Figure 11 Proposed System

4.3.1. Data Collection and Preparation

The dataset is needed to attain the objective of this study. To develop an intrusion classification model, a network traffic dataset is needed to provide enough data to the classifiers to train them properly. To do this the first method is to collect the proper dataset by using websites. Our dataset called WSN-DS contains both malicious and non-malicious records and it comprises four different attack types, such as Flooding, Blackhole, Sinkhole, TDMA, and Normal.

4.3.2. . Data Preprocessing

The main reason why we need data preprocessing arises from the fact that redundant data and insignificant features may be unclear to the classification algorithm and this may lead to inaccurate and less generalizable models. Usually, the dataset collected from different sources may not be in a format that is suitable for a machine, may contain invalid and/or missed values, and may be too large for a specific task. The main reason for a model to be accurate and precise in predictions is that the algorithm should be able to interpret the data's features quickly

4.3.2.1. String Indexing

The dataset we used for building the proposed intrusion detection model contains three feature types, namely nominal, numeric, and binary features. Therefore, before we can run a model on these, we must first transform the data into the desired form, such as converting a string or categorical data to numeric data. To transform this type of categorical text data into model-readable numeric data, we use the StringIndexer transformer. StringIndexer encodes categorical values of a column into indices, based on the frequency it appears in the entire input column of a data frame, such that the most frequent categorical value gets an index value of 0, 1, 2, and so on.

4.3.2.2. One Hot Encoding

Since the output of string indexing is numeric, the model can be confused thinking that a column has data with some kind of order or hierarchy, but these numbers do not have any order. To avoid this, the OneHotEncode of that column is necessary. OneHotEncoder retrieves a column that has categorical data, which has been label encoded and then splits the column into multiple columns, and then indices are replaced by 1s and 0s, depending on which column has what value. This encoding allows algorithms that can only operate with numerical-valued features, to be applied to categorical features.

4.3.2.3. Feature Selection

Making use of more features to run ML algorithms tends to make models more complex and difficult to interpret. Besides this, it can often lead to the models overfitting on the training data. This led models to perform very poorly on the given data. In addition, having irrelevant features in a dataset can decrease the accuracy of models and takes much amount of time to train models. Therefore, we have to select an optimal number of features to train and create models that are highly generalizable to the data, with less time for computation. This can be done by using feature selection techniques.

4.3.2.4. Normalization

A dataset may contain many features that have values in different ranges. Some features may consist of small floating-point values and others large integer values. Therefore, to make the network learn easily, the data fed to the algorithm should take small values. Moreover, the input data fed to the machine learning model may include some missing values. These missing values can confuse the classifier during the learning and testing process. Most ML algorithms cannot handle missing input values. So, it is better to substitute all the missing values using some appropriate values 0. The number 0 means missing data and will start ignoring the value

4.3.3. Train-test Split

The main purpose of any model-building process is to develop a generalizable model on the available data which will perform well. But to estimate a model's performance, we need to have separate data. This is achieved by splitting the available dataset into training and testing parts. The training part is used by the classifier to learn what each category looks like by making predictions about the input data and then correcting itself when the predictions are wrong. The test part is used to find the accuracy (predictability) of the model and compare the final model output to the actual dataset labels. Accordingly, from the total number of datasets, 75% is used as a train and 25% is applied as test datasets, respectively. Although several studies used different data proportions for train and test datasets, this research is assuming that the more the dataset used to train the model, the more accurate the prediction can be obtained.

4.3.4. Learning Algorithms

For our study, we used five classifiers, namely DT, RF, LR, NB, and KNN, are selected for the training and testing of the model. The main reason for selecting these algorithms is the ease of understanding the model results, the ability for both binary and multiclass classification, and applicability on large datasets.

4.3.5. Grid Search

This phase provides an algorithm to validate the effectiveness of the model. The classification algorithms can have many parameters to train or learn from data but it is better to find out the best subset of parameters of the model. In this phase, both training and validation sets are passed to the grid search. Then the parameter grid (hyperparameters with default values) and algorithm are specified and passed to the cross-validation. The cross-validation runs on the training data and searches the whole parameter grid for finding the best parameters. The developed models have been validated for their effectiveness on the validation data. The selected best parameters are used to build the final model. The final model is then evaluated on the testing data.

4.3.6. Cross-validation

The cross-validation searches the whole parameter grid and runs the training section for selecting the best parameters. Then, the models are evaluated on the validation part. The selected best parameters are then used for building the final trained model which is finally evaluated on the testing set.

4.3.7. Evaluation Techniques

This technique aims to forecast and estimate how accurately a classification model will work in practice. This study uses different model evaluation metrics such as F-score, recall, and precision. Also, k-fold cross-validation techniques are applied to increase the evaluation reliability of the classifier by splitting the original dataset into *k* equally-sized folds, from which *k-1* folds are operated as a training dataset and *k* fold is used as a test dataset. Again, we use the confusion matrix as a metric to evaluate classifier performance. The matrix enables the assessment of the result of false negatives and false positives.

4.3.8. Model Performance Evaluation

An algorithm model can be evaluated in three phases: during the training phase, validation phase, and testing phase. The performance on the test section tells us how good the model is in the real world. Evaluating a model on a dataset is usually done using classification metrics. Based on the literature review, five evaluation metrics (confusion matrix, accuracy, precision, recall, and f1-score) are used for evaluating the efficiency of the models in this paper.

4.4. Summary

Intrusion detection is a process of detecting inappropriate, incorrect, or anomalous activity targeted at wireless networking devices. To detect and classify the sinkhole attack including other related routing attacks, we proposed a network IDS using machine learning techniques. The proposed system can achieve effective detection of intrusions using the collected dataset. The dataset is preprocessed using data preprocessing methods and classified into their category based on the trained models. The features selected from the dataset enable the system to speed up its processing and improve its accuracy. The grid search phase is responsible to validate the effectiveness of the models using validation data, which was not used in the training section. It helps to show that the parameters used in the training stage work better based on the evaluation metrics. The selected better hyperparameter values are then used to build the final model. Finally, the final model is used to predict (classify) new instances into their category. Based on this evaluation result a decision can be done on the efficiency and effectiveness of the system.

CHAPTER FIVE

5. Experimental Setup and Implementation

5.1. Introduction

In this Chapter, the experiments carried out to test the effectiveness of the proposed system are discussed. The development environment and software tools, the programming languages, and libraries used during the classification process, the dataset used, implementation details, and the performance evaluation results of the different models using various evaluation metrics are discussed.

In this notebook, we will use the WSN-DS Datasets downloaded from the website Github to build five models (Logistic Regression, Decision Tree, Random Forest, Naive Bayes, and KNN). After building each model, we will evaluate and compare them to determine which model suits us best. Next, we will try to optimize our model by optimizing the model hyperparameters using GridSearch.

5.2. Dataset Overview

WSN-DS uses the LEACH protocol to gather the dataset with features of wireless sensor networks and with different attacking scenarios. It contains a total of 374,661 records, and it has 19 attributes described. The data sets include four different DoS attacks, such as Blackhole (10,049), Sinkhole (14,596), Flooding (3,312), , TDMA (6,638), and normal with the remaining 340,066 records.

Table 4 Attribute Of WSN-DS Dataset

No	Feature name	Feature description
1	Id	Unique ID to distinguish the sensor node
2	Time	Current simulation time of the node.
3	Is CH	It distinguishes whether the node is Cluster head or not cluster head
4	Who CH?	The ID of the Cluster head in the current round.
5	Distance to CH	The space between the node and its CH in the current round.
6	ADV_S	The number of advertise CH's broadcast messages sent to
7	ADV_R	The number of advertising CH messages received from CHs
8	Join_S	The number of join request messages sent by the nodes to
9	Join_R	The number of join request messages received by the CH from the nodes.
10	SCH_S	The number of advertise TDMA schedule broadcast messages sent to the nodes.
11	SCH_R	the number of TDMA schedule messages received from
12	Rank	The order of this node within the TDMA schedule.
13	DATA_S	The number of data packets sent from a sensor to its CH.
14	DATA_R	The number of data packets received from CH.
15	Data_Sent_To_BS	The number of data packets sent to the BS
16	dist_CH_To_BS	The distance between the CH and the BS.
17	send_code	The cluster sending code.
18	Consumed Energy	The amount of energy consumed in the previous round.
19	Attack Type	Type of the node. It is a class of five possible values, Blackhole, Sinkhole , Flooding, and TDMA, and normal, if the node is not an attacker

5.3. Data Pre-processing

We first need to load and pre-process the dataset before starting to build our first model. This phase ensures that our model will receive good data to learn from, as they said "a model is only as good as its data". The data pre-processing will be divided into a few steps as explained below.

Loading data: To start our machine learning project in Python, we need to be able to load data properly. In this first step, we will load our dataset that has been uploaded on GitHub for an easier process.

```
In [4]: # dataframe1 = pd.read_csv("WSN-DSM.CSV")
        dataframe = pd.read_csv("WSN-DS.CSV")
        dataframe.head(10)
```

Out[4]:

	id	Time	Is_CH	who_CH	Dist_To_CH	ADV_S	ADV_R	JOIN_S	JOIN_
0	101000	50	1	101000	0.00000	1	0	0	2
1	101001	50	0	101044	75.32345	0	4	1	
2	101002	50	0	101010	46.95453	0	4	1	
3	101003	50	0	101044	64.85231	0	4	1	
4	101004	50	0	101010	4.83341	0	4	1	
5	101005	50	0	101010	31.91198	0	4	1	
6	101006	50	0	101044	24.34167	0	4	1	

Figure 12 Dataset Extraction

Features Selection: It is important to understand, which features are the most influential in determining the attack type. This will examine how well each attribute can individually differentiate attack types. This speeds up ML algorithms, improves classification accuracy, and increases a model's generalizability. Furthermore, the larger the feature length, the more computing time is required for model training (Upadhyay et al., 2021)

```
In [16]: df.drop ([' Time', ' who_CH', ' Dist_To_CH',
                  ' JOIN_S', ' JOIN_R',  'Rank', ' dist_CH_To_BS', ' send_code ', 'Expaned_Energy' ],  axis=1 , inplace = True)
```

Figure 13 Features Selection

Distribution of Class: Additional important thing to make sure of before changing our data into the model is the class distribution of the data. In our case where the expected class are divided into two outcomes, 'Normal ' and 'Attack'

```
In [9]: count = dataframe['Attack_type']
        data_distrubution = count.value_counts()
        print(data_distrubution)

Normal        340066
Sinkhole       14596
Blackhole      10049
TDMA            6638
Flooding        3312
Name: Attack_type, dtype: int64
```

Figure 14 Class distribution

Table 5 Normal and Attack data

Category		Total Records
Attacks	Sinkhole	14,596
	Black hole	10,049
	TDMA	6,638
	Flooding	3,312
Normal		340,066

```
In [40]: # Plotting the percentage of observations that fall under each class
ax = df_multi_data["Attack_type"].value_counts().sort_values().plot(kind="barh", color=["r", "g"])
totals= []
for i in ax.patches:
    totals.append(i.get_width())
total = sum(totals)
for i in ax.patches:
     ax.text(i.get_width()+.3, i.get_y()+.20,
     str(round((i.get_width()/total)*100, 2))+'%',
     fontsize=10, color='black')
plt.title("Attack_type", fontsize=20)
plt.xlabel("Count", fontsize=14)
plt.ylabel("Class", fontsize=14)
plt.show()
print(df_multi_data["Attack_type"].value_counts())
fig = ax.get_figure()
```

Figure 15 Multi-class distribution

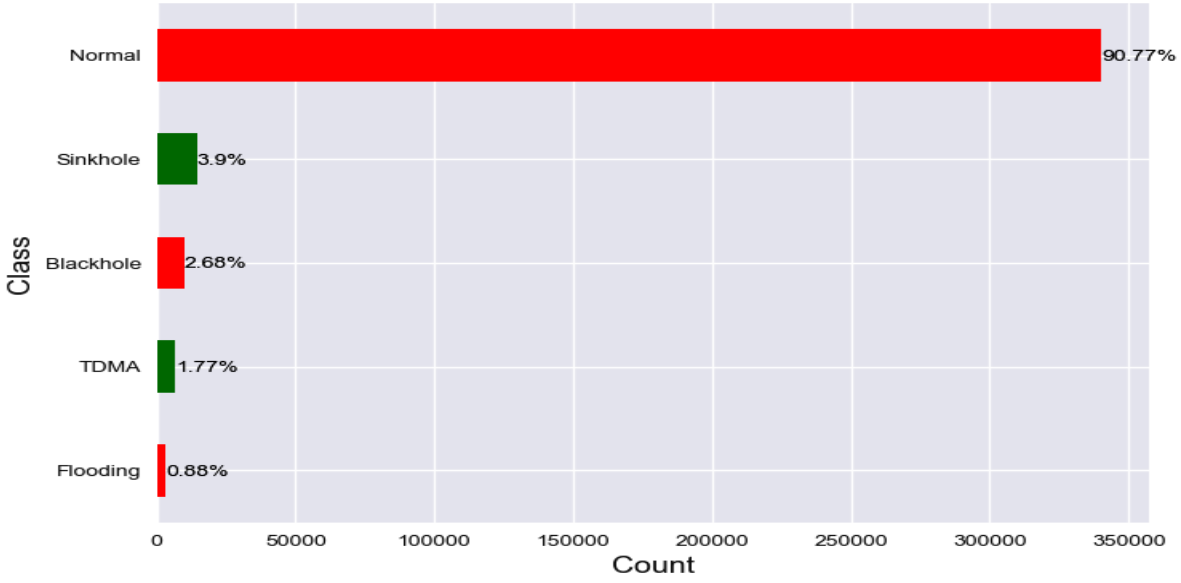


Figure 16 Multi-class distribution result

Missing Values: In some cases, our data might have missing values in some columns; this can be caused by some reasons such as human error. We can use is null () function from Pandas to check for any missing data and then use the sum () function to see the total of missing values in each column.

```
In [9]: dataframe[dataframe.isnull().any(axis=1)]
```

Out[9]:

id	Time	Is_CH	who_CH	Dist_To_CH	ADV_S	ADV_R	JOIN_S	JOIN_R	SCH_S	SCH_R	Rank	DATA_S	DATA_R	Data_Sent_To_BS	dist_CH_To_BS	send

Figure 17 Missing Value result

Numeric Data Scale: Scaling is a set of linear transformations that make all features comparable. The two most common techniques for scaling numerical data before modeling are normalization and standardization Scaling is a set of linear transformations that make all features comparable.

Encode Categorical Value: Encoding categorical data is the process of converting categorical data into an integer format so that data with converted categorical values can be fed into models to obtain and improve predictions. We will use the `OneHotEncoder () `function provided by sklearn.

Dataset Splitting: To complete the data pre-processing stages, we will split our data into two parts such as training and testing. In this case, because we have enough data we will split the data with a ratio of 75:25, meaning 75% of the dataset has been assigned for training and 25% for testing purposes.

```
In [48]: # splitting the dataset 75% for training and 25% testing
         from sklearn.model_selection import train_test_split   # 20 % for testing
         X_train, X_test, y_train, y_test = train_test_split(X , y, shuffle = True, test_size= 0.25, random_state=1) # 25 % for testing
```

Figure 18 Dataset Split code

Table 6: Training and Testing division of dataset

Type of attack	Number	Training set (75%)	Test set (25%)
Normal	340066	255,050	85,016.5
Sinkhole	14596	10,947	3,649
Black hole	10049	7,536.75	2,512.25
TDMA	6638	4,978.5	1,659.5
Flooding	3312	2484	828
Total	374661	280995.8	93665.25

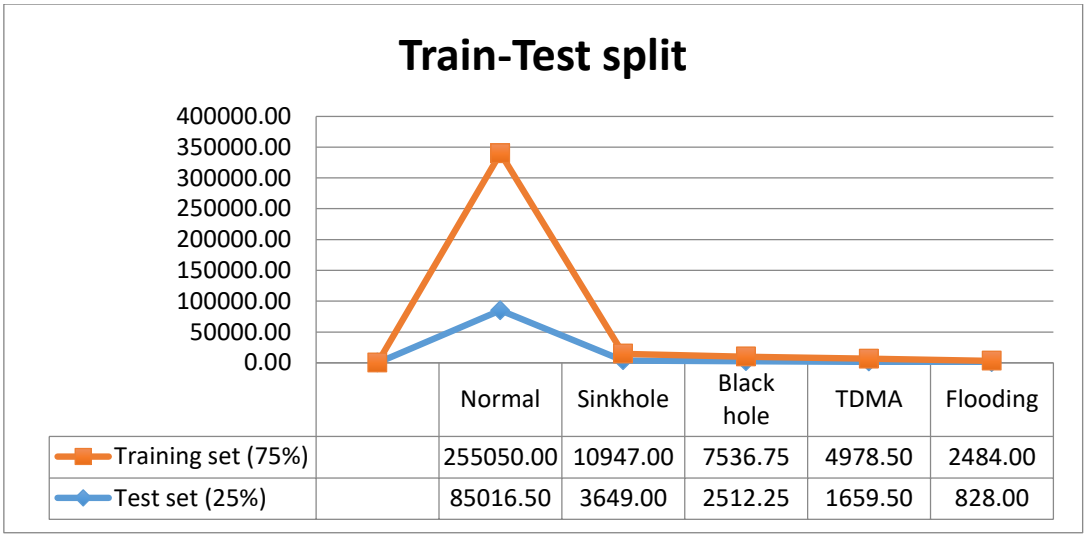


Figure 19 Train Test result

5.4. Data Modeling and Evaluation matrix

5.4.1. Data Modeling

After making sure our data is worthy and ready, we can continue to build our model. In this study, we used four ML model algorithms. After completing building the entire model, we can compare these models to how well each performs. To evaluate our models, we applied a confusion matrix and as well as AUC as the base for the score. The Area under Curve (AUC) indicates how well the probabilities from the positive classes are separated from the negative classes

5.4.2. Evaluation Metrics

A set of evaluation metrics including accuracy (ACC), precision (PR), recall (RE), and f1-score are utilized to evaluate and compare the results of the proposed intrusion detection model. They were used because they produced comparable results and were frequently used in the machine learning field for evaluating and comparing its models. In our experiments, we have used the most common supervised learning classifiers to detect the attack types listed in this dataset. We applied the train-test split techniques with a 75/25 ratio meaning 75% of the data is used to train the model, and the remaining part, 25%, is used to test the model in the train-test model

5.3. Working Environment

In this study, multiple tools and development packages were used to address a proposed machine learning analysis solution to detect and prevent WSN routing malicious attacks. We use the python programming language for implementing and experimenting with each proposed solution from the data pre-processing to the model-building phase Because of this, Python is the programming language of choice for programmers, researchers, and data scientists who need to work on machine learning models.

5.4. Experimental Result and Discussion

The below presents a detailed comparative analysis of the five classification algorithms in terms of detection accuracy, recall, precision, F1-measure, and time complexity. We used 10-fold cross-validation to conduct the experiment and to train the classifiers. The results of our experiment are obtained using both 10-fold cross-validation and holdout methods on the simulated WSN-DS dataset.

5.4.1. Classification

We performed Principal Component Analysis, a dimension reduction technique, and subsequently classified using five machine learning algorithms namely Naïve Bayes, KNN, Logistic Regression, Decision Tree, and Random Forest Tree. We also applied GridSearchCV, K-fold cross-validation, and techniques of boosting –Gradient Boosting and Voting Classifier.

5.4.2. Model Comparison

In this Section, the experimental findings for the selected classification algorithms are analyzed for both situations. A 2x2-confusion matrix is used to evaluate all classification models. This confusion matrix displays the number of correct and incorrect predictions generated by the classification model compared to the data's target values by calculating the number of test samples divided into four groups (TP, TN, FP, and FN), as summarized in the respective discussion.

As shown in Table 7 and Figure 20, the DT, NB, KNN, LR, and RF achieved test accuracy of 99.17%, 88.64%, 97.71%, 90.75%, and 99.51% respectively. In this regard, we found that the

best model in terms of training and the testing accuracy is Random Forest with an accuracy rate of 99.52% over other models with the DT coming in second place

```
In [69]: gnb_train_accuracy = accuracy_score(y_train, gnb.predict(X_train))
         gnb_test_accuracy = accuracy_score(y_test, gnb.predict(X_test))

         result = result.append(pd.Series({'Model':'Gaussian Naive Bayes ','Train Accuracy':gnb_train_accuracy,
                                           'Test Accuracy':gnb_test_accuracy}),ignore_index=True)
         result
```

Figure 20 Accuracy comparison between models

Table 7 Training and testing results of each model

	Model	Train Accuracy	Test Accuracy
0	Logestic Regression	0.907717	0.907501
1	DecisionTree Classifier	0.991292	0.991737
2	RandomForest Classifier	0.995167	0.995100
3	Gaussian Naive Bayes	0.886952	0.886426
4	K-Nearest Neighbors	0.984213	0.977057

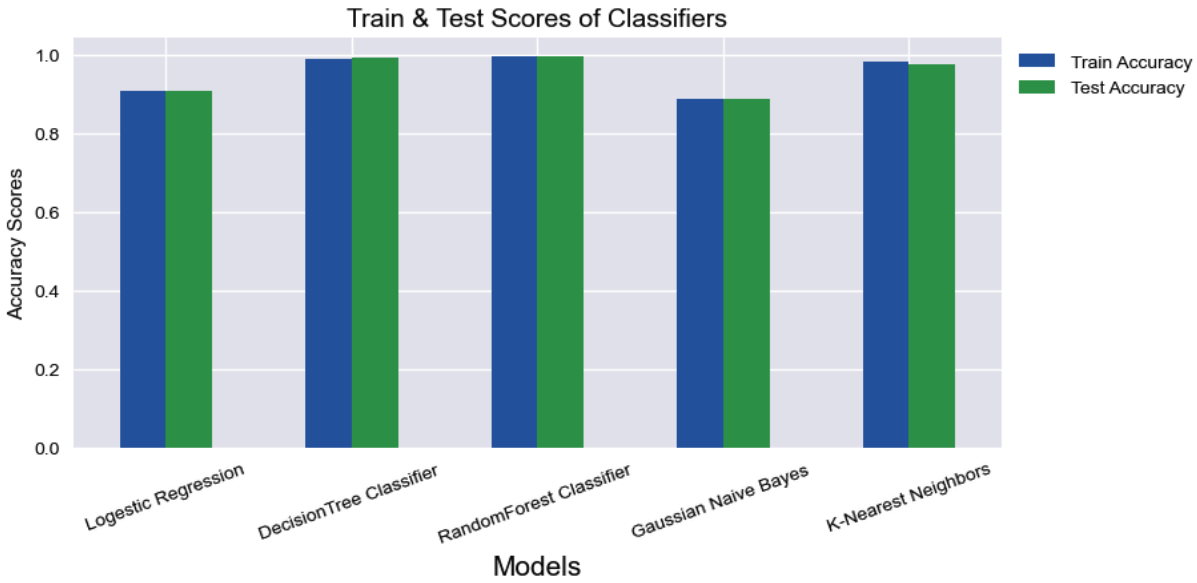


Figure 21 Training and testing accuracy

After applying all the stated algorithms for the attack types, the complete performance is evaluated by using accuracy, precision, recall, and F1-Score for the dataset. as illustrated in Table 8 and Table 9 of the training and testing results, RF and DT show better performance than the others. Figure 21 shows the comparison of the accuracy of applied algorithms.

```
In [72]: # defining a function to compute different metrics to check performance of a classification model built using sklear
def model_performance_classification_sklearn(model, X_test, y_test):


    # predicting using the independent variables

    y_pred = model.predict(X_test)

    # Calculate accuracy, precision, recall, f1-score, and kappa score
    acc = accuracy_score(y_test, y_pred)
    recall =  recall_score(y_test, y_pred )
    precision = precision_score(y_test, y_pred )
    f1 =   f1_score(y_test, y_pred )


    # creating a dataframe of metrics
    df_perf = pd.DataFrame(
        {
            "Accuracy": acc,
            "Recall": recall,
            "Precision": precision,
            "F1": f1,

        },
        index=[0],
    )

    return df_perf
```

Figure 22 Python code running for measuring model performance

Table 8: Model Performance on Training Data

	Accuracy	Recall	Precision	F1
Logistic regression	0.907717	1.000000	0.907717	0.951627
Decision Trees Clasifier	0.991292	0.997538	0.992902	0.995214
Random Forest Clasifier	0.995167	0.997350	0.997326	0.997338
Gaussian naive bayes	0.886952	0.973164	0.908761	0.939860
KNeighborsClassifier	0.984213	0.992719	0.989917	0.991316

Table 9: Model Performance on Validation Data (Testing results)

	Accuracy	Recall	Precision	F1
Logistic regression	0.907501	1.000000	0.907501	0.951508
Decision Trees Clasifier	0.991737	0.997588	0.993335	0.995457
Random Forest Clasifier	0.995100	0.997212	0.997388	0.997300
Gaussian naive bayes	0.886426	0.972777	0.908540	0.939562
KNeighborsClassifier	0.977057	0.990153	0.984651	0.987394

Testing accuracy Visualization

Figure 23: Accuracy detection

Based on the above results Logistic Regression, Decision trees Classifier, Random Forest; K Neighbors Classifier had the highest average recall (99.62%) whereas NB had the lowest recall (97.32%) Regarding the precision results of each classifier, the classifier with the lowest average precision were Logistic Regression (90.77%) and Gaussian Naive Bayes (90.88%). The Decision trees Classifier (99.29%)), Random Forest Classifier (99.73%), and K Neighbors Classifier (98.99%) had the highest average precision (0.993 and 0.997) respectively. Regarding the results for F1 measure Random Forest Classifier, Decision trees Classifier, and K Neighbors Classifier obtained the highest average F1-measure value (99.73%, 99.52%, and 99.13%), whereas the lowest value was 95.16% and 93.99% using Logistic Regression and Gaussian Naive Baye

5.4.3. Receiver Operation Characteristics (ROC)

The ROC curve is a graphical representation that indicates the efficiency of the trained classifier models. This curve represents two parameters namely the true positive rate (synonymous with recall) and the false positive rate. The larger the area, the better the model performs. As it is displayed in Figure 23, we can see that Random Forest Classifier and Decision trees Classifier, then next KNN Classifier models accuracy result get tops the other models in the metrics we evaluated, so we can say that those Classifier are the correct choice to solve our problem. But the model for our project is Random Forest

Table 10: Area under Receiver Operating Characteristic Curve (ROC)

AUC Value	Inference
AUC = 0	The classifier predicts all negatives as positives and all positives as negatives
AUC = 0.5	The classifier cannot distinguish between positive and negative class points and thus predicts a random or fixed class for all data points
Between 0.5 and 1	It is likely that the classifier will be able to differentiate between the two classes
AUC = 1	The classifier is able to perfectly distinguish between all the positive and the negative class points correctly

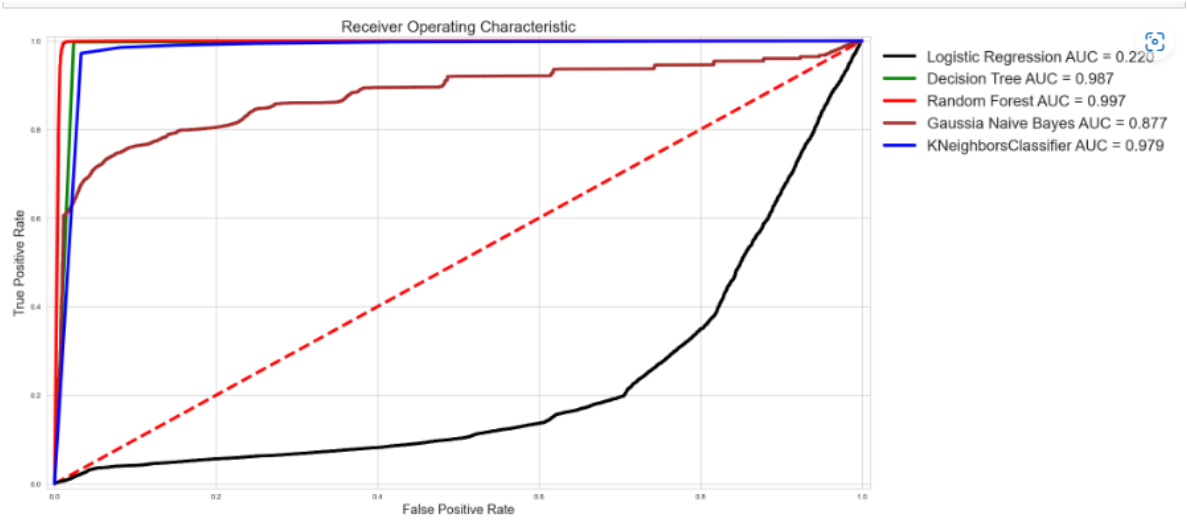


Figure 24: ROC curve

From the result of figure 24, t the detection accuracy rate of a blackhole, sinkhole, and normal is all above 95% which is the best detection status but flooding, and scheduling scores are above 85% which is also a good status result. It shows that the proposed paper in this study has a good multiclass detection effect for these types of data traffic in WSNs.

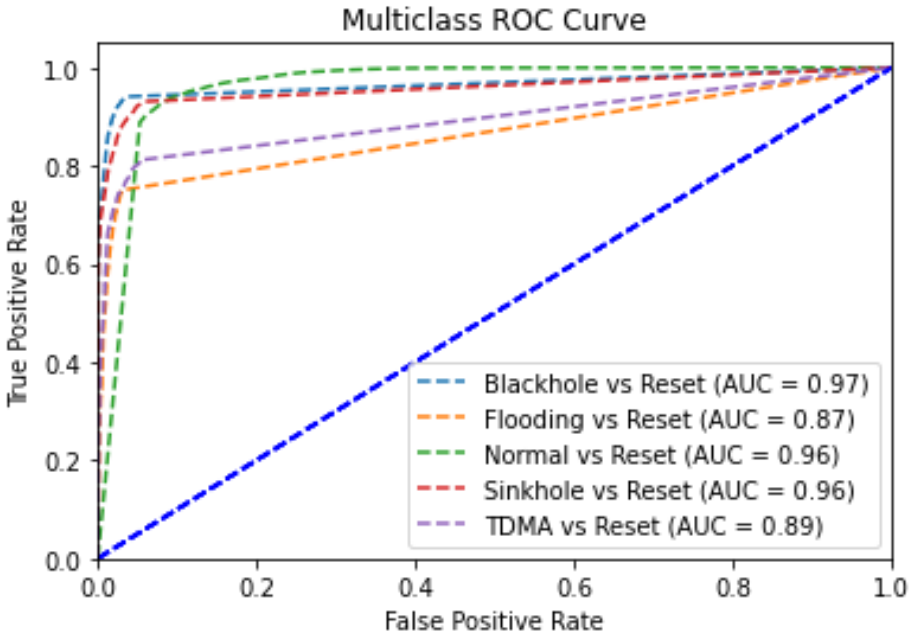


Figure 25: Multi classification results

Execution time of Classifiers

This research recorded the execution time[8] consumed by each model. As shown in Figure 15. Gaussian Naive Bayes has the fastest classifier to build at only 0.88 seconds. The KNN Classifier took more time 21.90 seconds. The reason is, that it contains a large number of connectivities that takes more time. Based on the previous results and considering all metrics, we conclude that the most correct model classifier is Random Forest Classifier since it demonstrated better results than other techniques and takes a reasonable time of amount to build. Random Forest

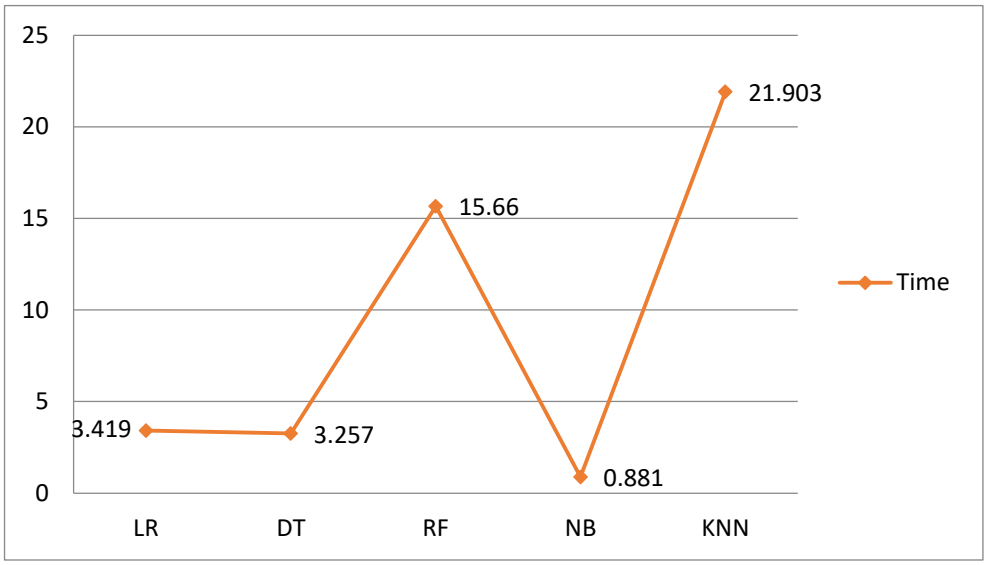


Figure 26: Model building time

5.4.4. Mean Absolute Error (MAE)

Mean absolute error is a measure of the absolute errors of our predicted value with respect to the true value. As we observed from table 11 and figure 13 results, the Random Forest has the lowest error records(0.0048) and also the decision tree and KNN also have a good result record next to the random forest by scouring 0.0081 and 0.0141 respectively

Table 11: Model comparison of MAE

Model	Mean Absolute Error(MAE)
Logistic Regression	0.0925
Decision Tree Classifier	0.0081
Random forest Classifier	0.0048
Gaussian Naive Bayes	0.1136
KNN	0.0141

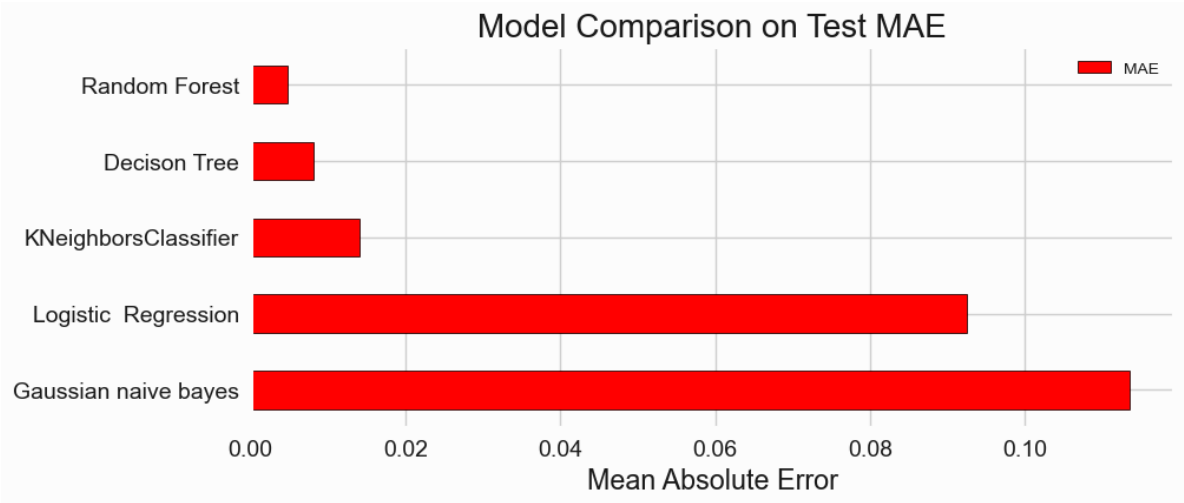


Figure 27: Mean absolute error comparison

5.4.5. Model Optimization

GridSearchCV is a technique for finding the optimal parameter values from a given set of parameters in a grid. It's cross-validation that iterates through each parameter combination to find the best-scoring parameters. The process will take longer the more combinations that are submitted. As an alternative, we used RandomizedSearchCV, which will just randomly choose

the provided amount of parameters and will likely run more quickly. Figure 27 shows a sample code of Grid search and K-fold cross-validation applied for Random Forest

K-fold Cross-validation:

Cross-validation is a resampling technique used to assess machine learning models on a limited data sample. The process has a parameter called K, which refers to the number of groups into which a given data sample should be split. Therefore, this procedure is often referred to as K-fold cross-validation. When a specific value for K is selected, it may be used in place of K in the reference to the model, such as K=10 becoming 10-fold cross-validation. Based on the result findings listed in table 12 and shown in figure 29. The Random Forest and Decision Trees classifiers scored best results than all others also NB scores the lowest values

Random Forest

GridSearch CV

```
In [110]: tuned_parameters_rfc = {'n_estimators' : [i for i in range(100,160,10)], 'criterion' : ['gini', 'entropy'],
                                  'max_depth' : list(range(15,40,5)), 'min_samples_split' : list(range(2,5)),
                                  'min_samples_leaf' : list(range(1,5)), 'max_leaf_nodes' : list(range(10,210,50)),'max_features' : ['auto'

In [111]: '''random_forest = RandomForestClassifier(random_state = 0)
          rfc = GridSearchCV(random_forest, tuned_parameters_rfc, cv=10, scoring='accuracy')
          rfc.fit(X_train, Y_train)
          rfc.best_score_'''

Out[111]: "random_forest = RandomForestClassifier(random_state = 0)\nrfc = GridSearchCV(random_forest, tuned_parameters_rfc, cv=10, scori
          ng='accuracy')\nrfc.fit(X_train, Y_train)\nrfc.best_score_"

In [112]: #rfc.best_params_
```

K-fold Cross Validation

```
In [113]: cv = KFold(n_splits=10, random_state=0, shuffle=True)
          # evaluate model
          rf_auc = cross_val_score(rf, X_train, y_train, scoring='roc_auc_ovo', cv=cv, n_jobs=-1)
          rf_accuracy = cross_val_score(rf, X_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1)

In [114]: pd.DataFrame([[accuracy_score(y_test, rf_ypredict_test), '-'],[np.mean(rf_accuracy), np.mean(rf_auc)]],
                       columns = ['Accuracy', 'AUC'], index = ['Prediction', 'K-fold'])
```

Figure 28 GridSearchCV and KFOLD for RF

Prediction Accuracy K-Fold Accuracy AUC score

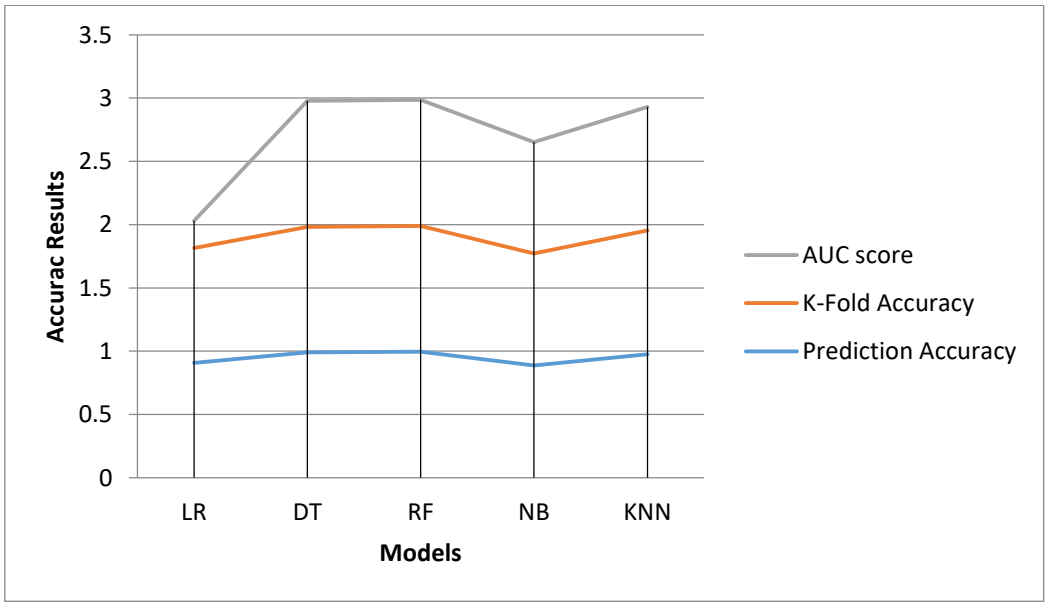
```
In [125]: accuracy = [[accuracy_score(y_test, lr_ypredict_test), np.mean(lr_accuracy), np.mean(lr_auc)],
                      [accuracy_score(y_test, dt_ypredict_test), np.mean(dt_accuracy), np.mean(dt_auc)],
                      [accuracy_score(y_test, rf_ypredict_test), np.mean(rf_accuracy), np.mean(rf_auc)],
                      [accuracy_score(y_test, gnb_ypredict_test), np.mean(gnb_accuracy), np.mean(gnb_auc)],
                      [accuracy_score(y_test, knn_ypredict_test), np.mean(knn_accuracy), np.mean(knn_auc)]]


          pd.DataFrame(accuracy, columns = ['Prediction Accuracy', 'K-Fold Accuracy', 'AUC score'],
                       index = ['Logistic Regression',  'Decision Tree Classifier', 'Random Forest Classifier' , 'Gaussian naive bayes','KNeighb
```

Figure 29 K-fold prediction accuracy

Table 12: Comparison of accuracy, 10-fold cross-validation, and AUC score

	Prediction Accuracy	K-Fold Accuracy	AUC score
Logistic Regression	0.907501	0.907717	0.215678
Decision Tree Classifier	0.991737	0.991092	0.995558
Random Forest Classifier	0.995100	0.995256	0.996053
Gaussian naive bayes	0.886426	0.886952	0.879473
KNeighborsClassifier	0.977057	0.976391	0.977476



: Figure 30 Accuracy vs AUC and k-fold:

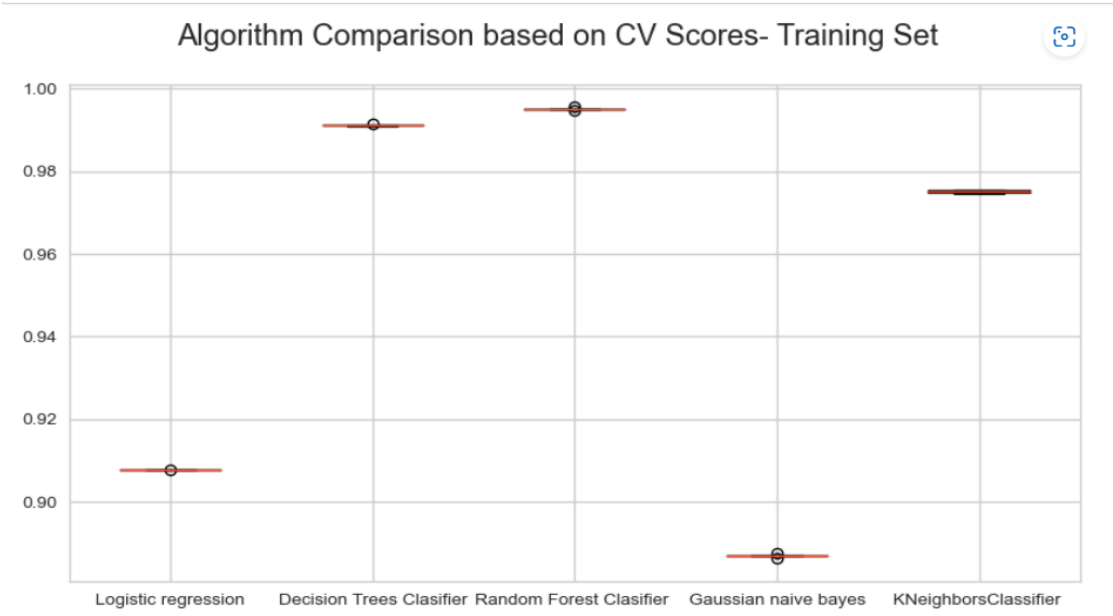


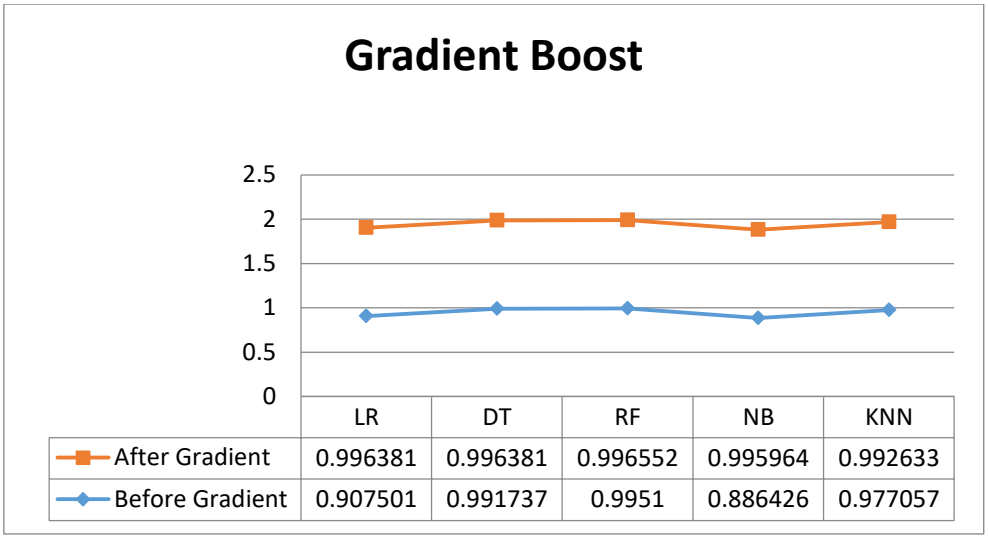
Figure 31 Comparison models based on Cross-validation

Gradient Boost

Gradient boosting is a kind of machine learning boosting technique that creates multiple weak models and then combines them to get better performance as a whole, and it minimizes the overall prediction error. Accrediting the below results shown in Table 13, the model converted the weak learners like Logic regression and NB into a strong model. But still, now the Random Forest leads by its best performance of result

Table 13: Gradient Boost model

	Gradient Boosting
Logistic Regression	0.996391
Decision Tree Classifier	0.996381
Random Forest Classifier	0.996552
Naive Bayes	0.995964
K-Nearest Neighbor	0.992633



	LR	DT	RF	NB	KNN
After Gradient	0.996381	0.996381	0.996552	0.995964	0.992633
Before Gradient	0.907501	0.991737	0.9951	0.886426	0.977057

Figure 32 Model comparison using gradient boosting

Voting Classifier

A Voting Classifier is a machine learning model used to evaluate various trained models and predict the output (classes) based on the highest probability of the class selected as the output. It simply aggregates the results of each classifier passed to the voting classifier and predicts the output class based on the highest majority of votes. As shown in Figure 32, RF and DT have been voted as the best model for our project than the rest of other models.

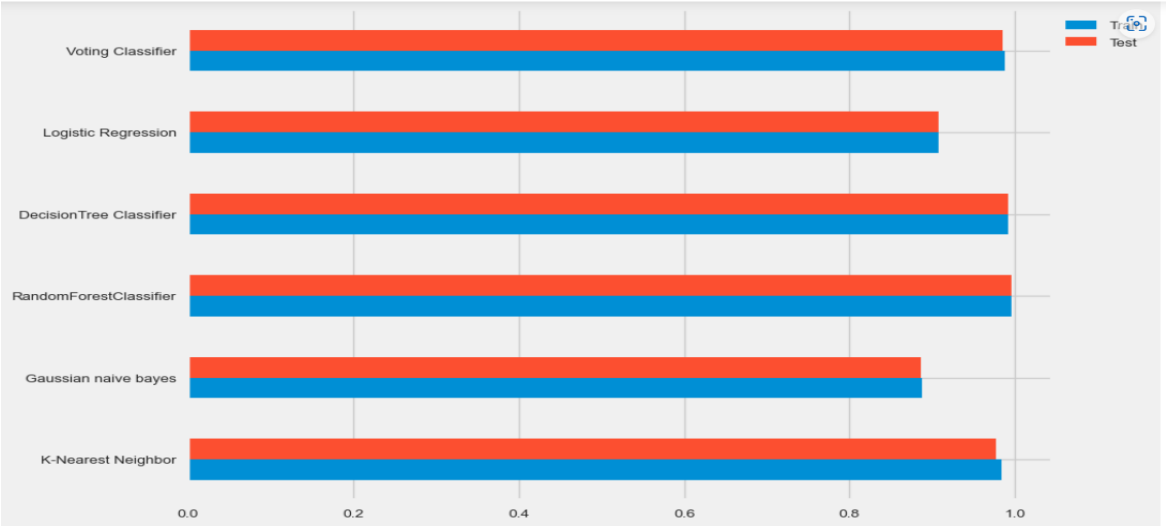


Figure 33: Voting Classifier comparison

5.4.6. Class imbalance and Sampling methods

Class imbalance (Picek et al., 2019), refers to the imbalance distribution among classes in the dataset being unequal. The dataset that has been used in this experiment exposes it to such problems. As we can observe from Figure 20, the total number of attacks in the dataset is less than that of non-attacks (Normal). Due to the imbalanced distribution of classes in a dataset, the accuracy result of an algorithm might be very high while the classifier may completely fail to predict the minority class. Hence, it is important to solve the class imbalance tricky to properly predict those attacks.

Resampling techniques have been sued to solve data imbalance that occurred in the dataset, Resampling is a method used to balance data and improve minority class scores. Accordingly SMOTE (Synthetic Minority Over-sampling Technique) and under-sampling are the main types of resampling techniques

```
In [40]: # Plotting the percentage of observations that fall under each class
ax = df_multi_data["Attack_type"].value_counts().sort_values().plot(kind="barh", color=["r", "g"])
totals= []
for i in ax.patches:
    totals.append(i.get_width())
total = sum(totals)
for i in ax.patches:
     ax.text(i.get_width()+.3, i.get_y()+.20,
     str(round((i.get_width()/total)*100, 2))+'%',
     fontsize=10, color='black')
plt.title("Attack_type", fontsize=20)
plt.xlabel("Count", fontsize=14)
plt.ylabel("Class", fontsize=14)
plt.show()
print(df_multi_data["Attack_type"].value_counts())
fig = ax.get_figure()
```

Figure 34 Class Imbalance

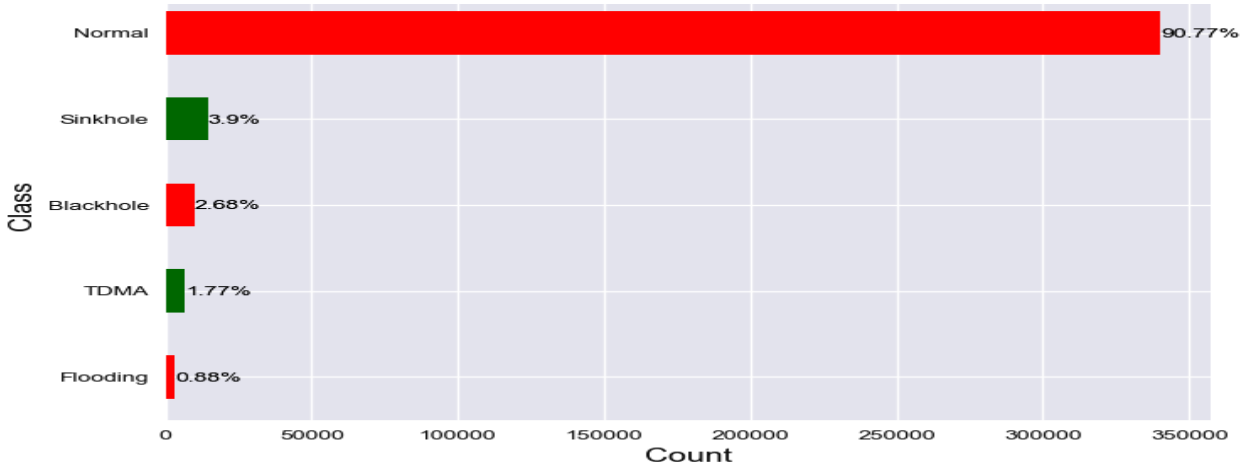


Figure 35: Imbalanced distribution between Normal and Attacks

Model Building with Oversampled data

A Sampling Method: SMOTE (Synthetic Minority Oversampling Technique)

SMOTE (Synthetic Minority Oversampling Technique)(Wang et al., 2019) is an oversampling technique that generates synthetic samples from the minority class. Rather than duplicating minority observations, SMOTE creates synthetic observations based on existing minority observations. Most of the time, we would maximize the number of minority classes to be the same as the majority class, reaching the 'balance'.

As we can be observed from Table 14 and 15, after applying SMOTE, the prediction rate of each model have different values of Accuracy, Recall, Precession, and F1 respectively compared to the earlier experiments described in table 7. Therefore, even if, the accuracy values of RF and DT slightly decrease before resampling, they still now RF recorded good performance and DT follows it by scoring good results. The worst outcome that has resulted after resampling is Logistic regression. That means logistic regression does not fit without our projects.

Table 14: Model Performance on Training Data (SMOTE)

	Accuracy	Recall	Precision	F1
Logistic regression over_sample	0.500000	0.000000	0.000000	0.000000
Decision Trees Clasifier over_sample	0.988838	0.983040	0.994574	0.988773
Random Forest Clasifier over_sample	0.993690	0.993323	0.994052	0.993688
Gaussian naive bayes over_sample	0.714140	0.919385	0.651820	0.762820
KNeighborsClassifier over_sample	0.990638	0.985576	0.995655	0.990590

Table 15 : Model Performance on Testing results after SMOTE

	Accuracy	Recall	Precision	F1
Logistic regression over_sample	0.092499	0.000000	0.000000	0.000000
Decision Trees Clasifier over_sample	0.983591	0.983212	0.998686	0.990888
Random Forest Clasifier over_sample	0.992719	0.993283	0.998687	0.995977
Gaussian naive bayes over_sample	0.880928	0.918649	0.948521	0.933346
KNeighborsClassifier over_sample	0.979929	0.981706	0.996120	0.988861

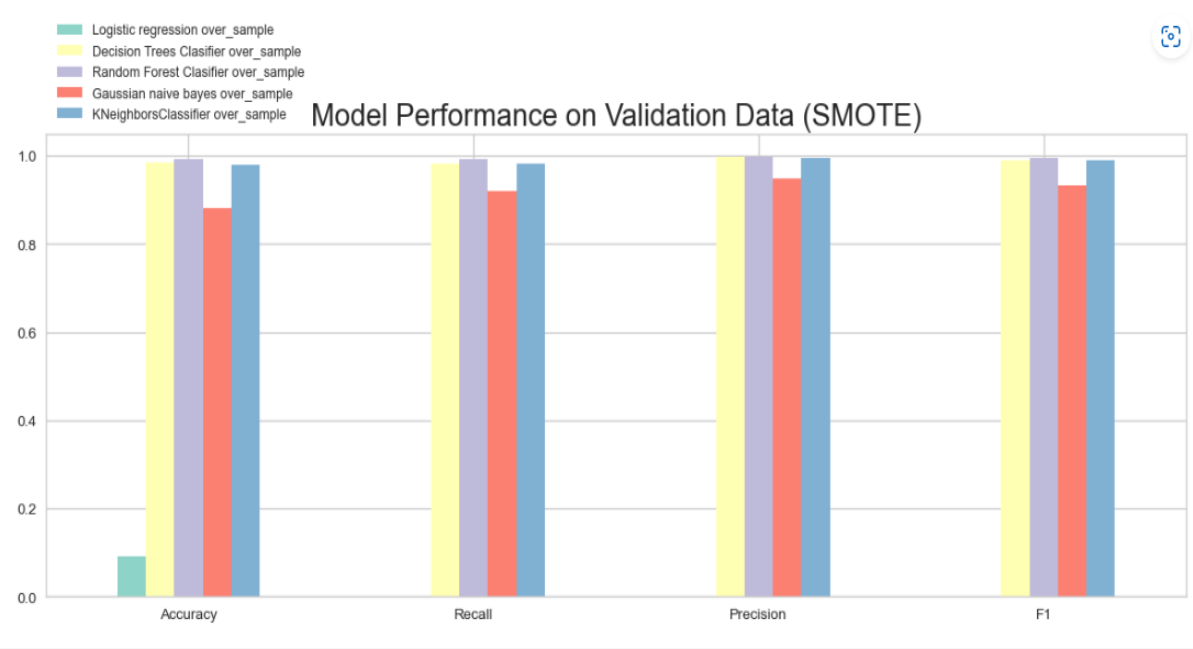


Figure 36: Model performance after SMOTE

Table 16: Summary Statistics of CV on oversampled Training Set

Out[191]:

	Logistic regression over_sample	Decision Trees Clasifier over_sample	Random Forest Clasifier over_sample	Gaussian naive bayes over_sample	KNeighborsClassifier over_sample
Min	0.499995	0.988366	0.993286	0.712296	0.985856
Average	0.500000	0.988830	0.993551	0.714036	0.986166
Max	0.500005	0.989473	0.994070	0.716099	0.986474
STD	0.000003	0.000365	0.000293	0.001229	0.000246

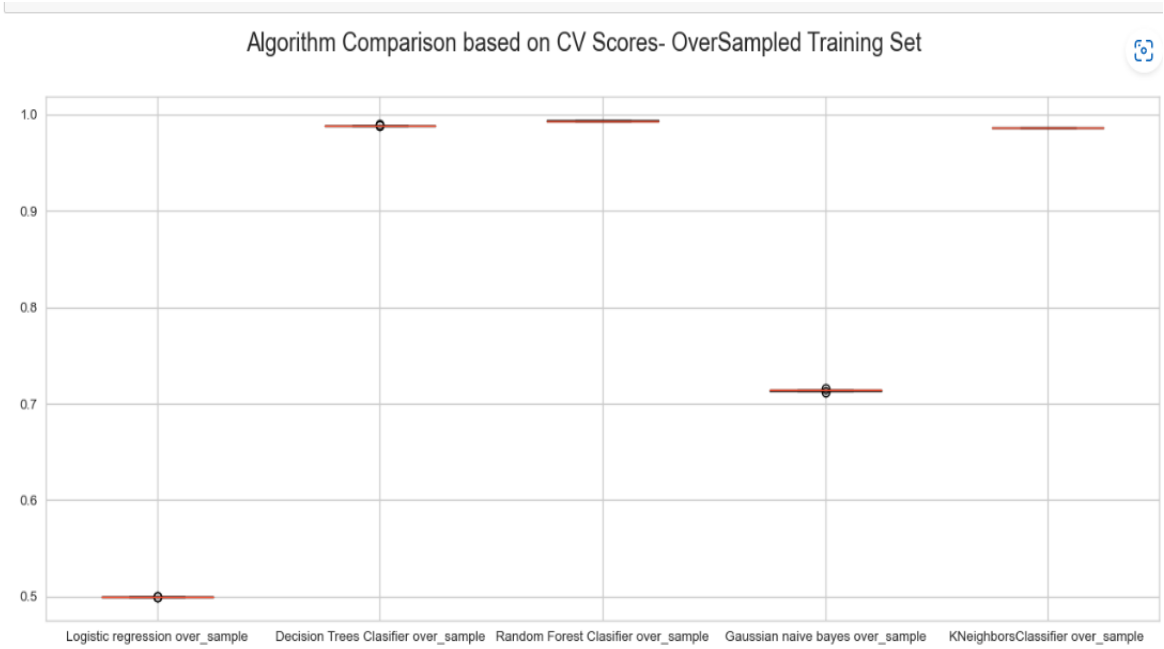


Figure 37: Model compassion based on Cross-validation after SMOTE

Model Building with Under sampled data

A Sampling Method: RUS (Random Under Sampling)

Under-sampling removing data points from the majority class to reduce its population is known as under-sampling. Random Under-sampling is a technique wherein random samples of the majority class are deleted until the distribution of data is equal. as we can see from table 20, table 21, and figure 21 like the result found in SMOTE, we have found different values scored by those several models. Accordingly, the results are decreased than those of the original and SMOTE. But in this process, Random Forest has a higher value compared to others Table 17: Model Performance on Training Data after under-sampled**.** Table 18, figure 19, and table 20 summarized all the methods we have used.

Table 17: Model Performance on Training Data after under-sampled

	Accuracy	Recall	Precision	F1
Logistic regression under_sample	0.500000	0.000000	0.000000	0.000000
Decision Trees Clasifier under_sample	0.985635	0.983803	0.987421	0.985609
Random Forest Clasifier under_sample	0.988026	0.988624	0.987443	0.988033
Gaussian naive bayes under_sample	0.715186	0.919170	0.652835	0.763441
KNeighborsClassifier under_sample	0.962323	0.953492	0.970636	0.961987

Table 18: Model Performance on Validation Data (Testing results) after under-sampled

	Accuracy	Recall	Precision	F1
Logistic regression under_sample	0.500000	0.000000	0.000000	0.000000
Decision Trees Clasifier under_sample	0.985635	0.983803	0.987421	0.985609
Random Forest Clasifier under_sample	0.988026	0.988624	0.987443	0.988033
Gaussian naive bayes under_sample	0.715186	0.919170	0.652835	0.763441
KNeighborsClassifier under_sample	0.962323	0.953492	0.970636	0.961987

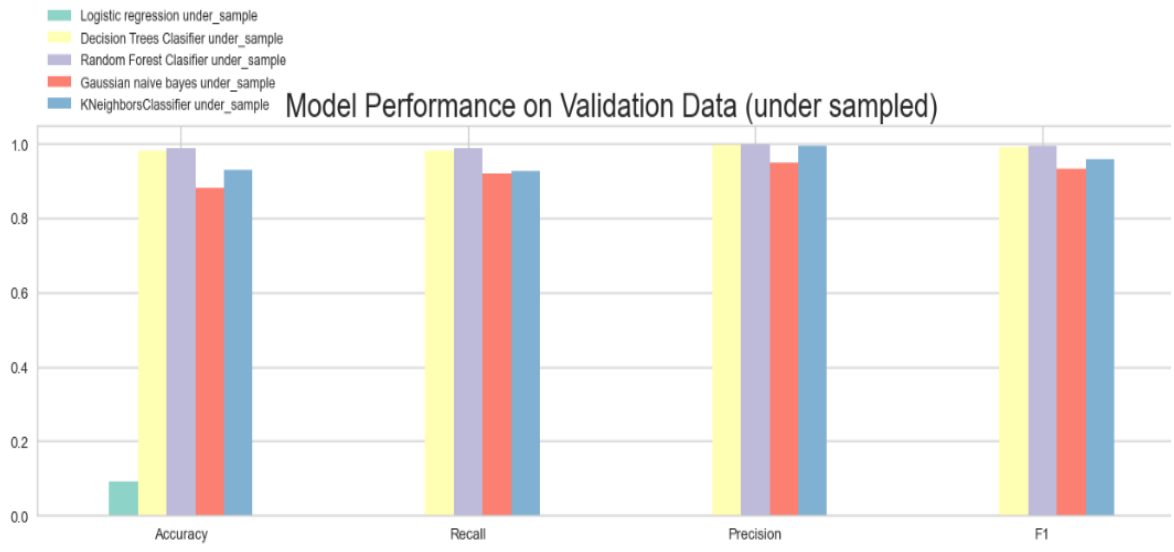


Figure 38: Model performance after RUS

Table 19: Performance Summary of all Models on Training Set

:

	Accuracy	Recall	Precision	F1
Logistic regression	0.907717	1.000000	0.907717	0.951627
Decision Trees Clasifier	0.991292	0.997538	0.992902	0.995214
Random Forest Clasifier	0.995167	0.997350	0.997326	0.997338
Gaussian naive bayes	0.886952	0.973164	0.908761	0.939860
KNeighborsClassifier	0.984213	0.992719	0.989917	0.991316
Logistic regression over_sample	0.500000	0.000000	0.000000	0.000000
Decision Trees Clasifier over_sample	0.988838	0.983040	0.994574	0.988773
Random Forest Clasifier over_sample	0.993690	0.993323	0.994052	0.993688
Gaussian naive bayes over_sample	0.714140	0.919385	0.651820	0.762820
KNeighborsClassifier over_sample	0.990638	0.985576	0.995655	0.990590
Logistic regression under_sample	0.500000	0.000000	0.000000	0.000000
Decision Trees Clasifier under_sample	0.985635	0.983803	0.987421	0.985609
Random Forest Clasifier under_sample	0.988026	0.988624	0.987443	0.988033
Gaussian naive bayes under_sample	0.715186	0.919170	0.652835	0.763441
KNeighborsClassifier under_sample	0.962323	0.953492	0.970636	0.961987

Table 20: Performance Summary of all Models on Testing Set

	Accuracy	Recall	Precision	F1
Logistic regression	0.907501	1.000000	0.907501	0.951508
Decision Trees Clasifier	0.991737	0.997588	0.993335	0.995457
Random Forest Clasifier	0.995100	0.997212	0.997388	0.997300
Gaussian naive bayes	0.886426	0.972777	0.908540	0.939562
KNeighborsClassifier	0.977057	0.990153	0.984651	0.987394
Logistic regression over_sample	0.092499	0.000000	0.000000	0.000000
Decision Trees Clasifier over_sample	0.983591	0.983212	0.998686	0.990888
Random Forest Clasifier over_sample	0.992719	0.993283	0.998687	0.995977
Gaussian naive bayes over_sample	0.880928	0.918649	0.948521	0.933346
KNeighborsClassifier over_sample	0.979929	0.981706	0.996120	0.988861
Logistic regression under_sample	0.092499	0.000000	0.000000	0.000000
Decision Trees Clasifier under_sample	0.983174	0.982730	0.998709	0.990655
Random Forest Clasifier under_sample	0.987359	0.987330	0.998727	0.992995
Gaussian naive bayes under_sample	0.881280	0.919049	0.948531	0.933557
KNeighborsClassifier under_sample	0.930370	0.927896	0.995042	0.960297

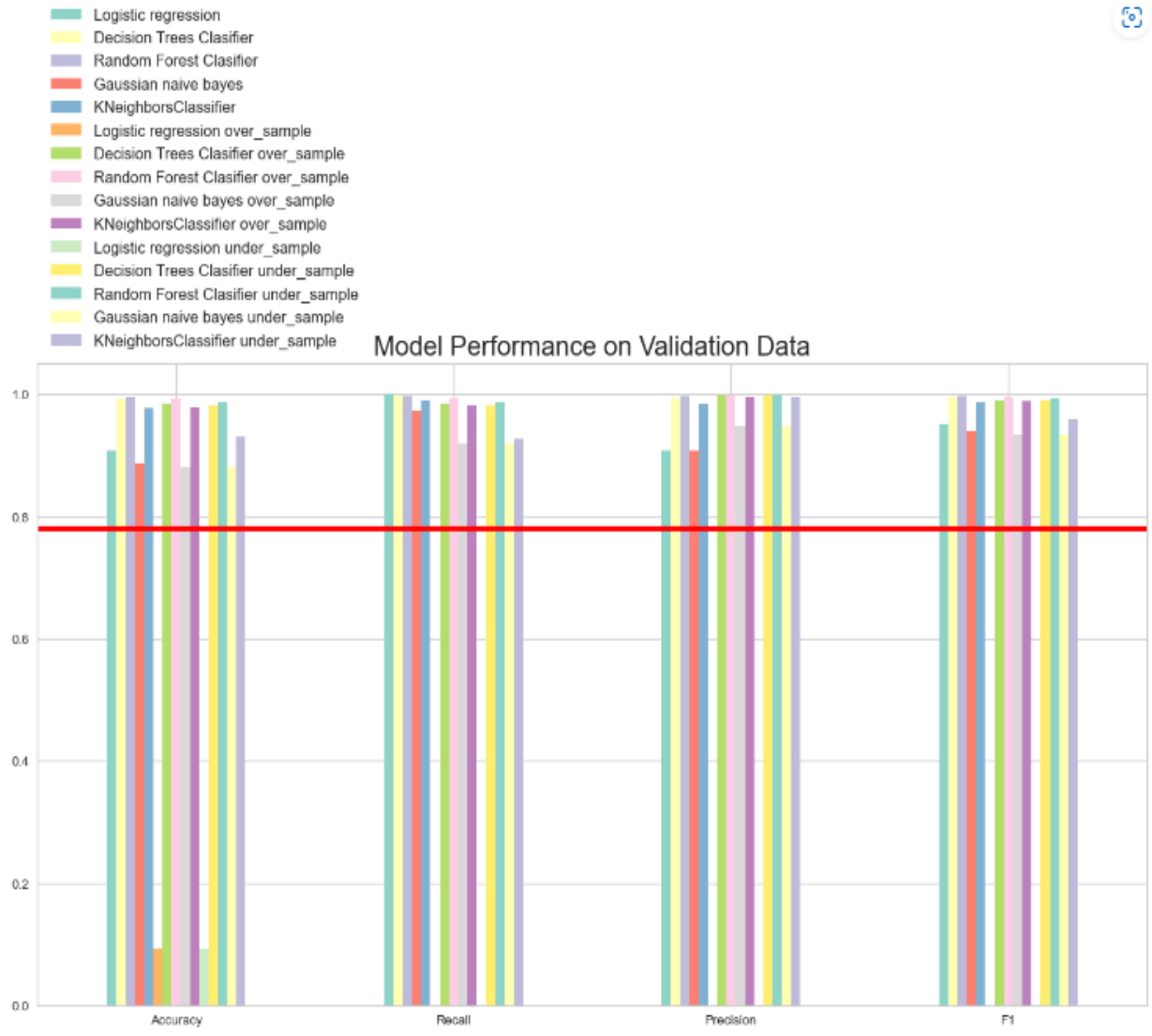


Figure 39: Model performance comparison of before resampling and aster resampling

Experiment Result comparison with other related work models

Table 21 Accuracy comparisons with related works

Title	Selected Model	Accuracy (%)	Difference
An Efficient Network Intrusion Detection and Classification System	DT	99	0.52
Data mining approach to analyzing intrusion detection of wireless sensor network	KNN	98.4	1.12
A novel scalable intrusion detection system based on deep learning	SVM	90.2	9.32
Anomaly Detection using Machine Learning Techniques in Wireless Sensor Networks	ID-GOPA.	86	13.52
Using Machine Learning to Detect DoS Attacks in Wireless Sensor Networks	DT	98.86	0.66
Machine learning-based intrusion Detection system	SVM	97.29	2.23
Modeling Network IDS Based on Anomaly Approach Using ML Techniques	NN	98	1.52
Our Proposed model	RF	99.52	

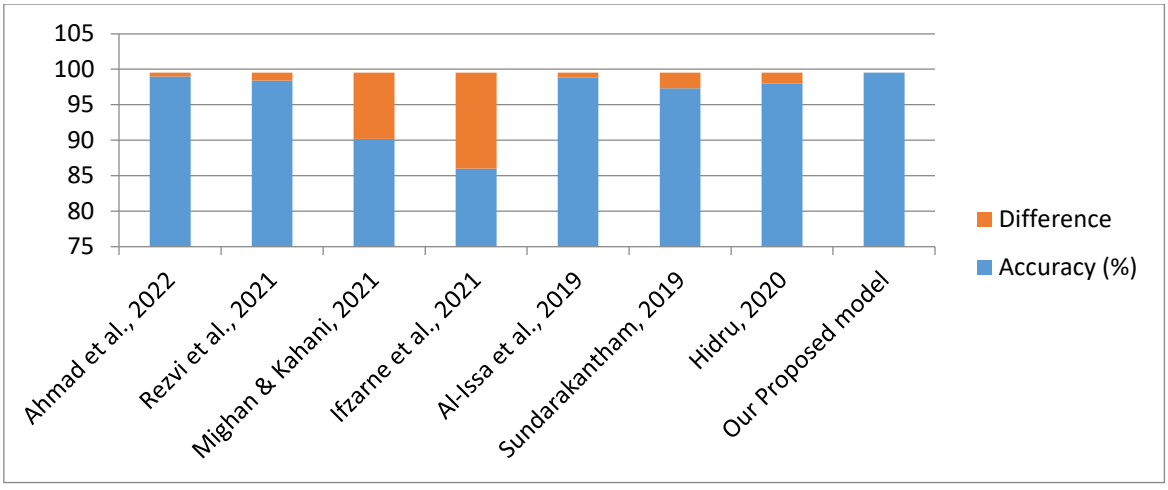


Figure 40 Model Comparison with Related models

CHAPTER SIX

6. Conclusion and future

6.1. Conclusion

In this thesis, the network layer routing attack detection model using a machine learning algorithm in WSNs under the LEACH routing protocol has been developed. Blackhole attack, Sinhole attack flooding attack, and TDMA along with the normal traffic were identified as classes of a dataset. The main goal of this work was to investigate intelligent intrusion detection and classification mechanisms that could work effectively to mitigate sinkholes and other network routing attacks on wireless sensor networks.

To achieve this, a special dataset called WSN-DS was analyzed to classify the four types of DoS attacks as Black Hole, Sinkhole, Flood, and TDMA, grouped into normal and malicious (abnormal) network traffic, in addition to the experimental results, mathematical validation of the created dataset has been provided to ensure its correctness

In this paper, We evaluated machine learning classification techniques for detecting wireless sensor network routing attacks. Five different classification techniques were then used, namely logistic regression, decision regression, random forest tree, naive Bayes, and ANN. GridSearchCV, K-fold cross-validation, gradient boosting, and voting classifier. The Voting Classifier model with Gradient Boosting achieves the highest accuracy but the Decision Tree Classifier, the K Neighbors Classifier, and the Random Forest model achieve the best accuracy result.

Based on the results of our experiment, we found that the random forest classifier and the decision tree classifier perform better than the other techniques in classification with a detection accuracy of over 99%. But compared to the best-mentioned classifications, the random forest classifier was chosen because it had the best overall performance. Also, among the regressors, the random forest regressor was chosen because it scored the lowest error result. Based on the results of the findings, we can conclude that accurate and timely detection of attacks will limit and stop the damage they cause on wireless sensor networks.

After using GridSearchCV to optimize the model, the result is not significantly different from the results that might indicate that these models have reached their limits. As a result, the random forest classifier works best, followed by a decision tree.

The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on, the minority class, although typically it is the performance of the minority class that is most important. Therefore one of the approaches for addressing the imbalanced datasets, SMOTE was chosen as the sampling method to oversample the minority class. We also used Random under Sampling. It uses the deletion of random data samples in the majority which may balance the ratio between classes, but reduces the achievement of the model.

To compare the performance of the classifiers, we also compared the time taken to build each model in seconds since it contains a large number of connectivities that takes time. Based on the previous results and considering all metrics, we conclude that the best-scored model is Random Forest Classifier since it demonstrated better results than other techniques and takes a reasonable time of amount to build. Random Forest Classifier has the best accuracy in detecting attacks due to the selection of multiple trees and can more effectively handle a large volume of data. The worst technique to detect the WSN routing DoS attacks was Gaussian Naive Bayes; although it was the fastest technique, it also had the worst results compared to other techniques.

From the result depicted in Table 19, the Random Forest classifier achieved the best result while NB was poorly performed. However, the performance metrics of KNN and DT were approximately similar; the former performs better than the latter. Therefore, the proposed work was an efficient method to detect the network layer attack in WSNs.

6.2. Future Work

The final result of this work shows the proposed approach performs well for a certain number of network routing attacks. In the future, this research will be extended to include other types of classifiers and Machine Learning techniques.

For better performance of the proposed work, network layer attacks like the Sybil attack and rushing attack should be incorporated into the dataset. Other machine learning algorithms should also be considered for the detection process. The work should be extended not only in WSNs layers but also for various wireless networks like MANET and VANET

REFERENCES

A, A. H., & Sundarakantham, K. (2019). Machine Learning Based Intrusion. *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Icoei*, 916–920.

Ahmad, I., Haq, Q. E. U., Imran, M., Alassafi, M. O., & Alghamdi, R. A. (2022). An Efficient Network Intrusion Detection and Classification System. *Mathematics*, *10*(3), 1–15. https://doi.org/10.3390/math10030530

Al-Issa, A. I., Al-Akhras, M., Alsahli, M. S., & Alawairdhi, M. (2019). Using machine learning to detect dos attacks in wireless sensor networks. *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2019 - Proceedings*, 107–112. https://doi.org/10.1109/JEEIT.2019.8717400

Alajmi, N. M., & Elleithy, K. M. (2016). A multi-layer framework for detection selective forwarding attacks in WSNs. *Asian Journal of Scientific Research*, *9*(5), 242–247. https://doi.org/10.3923/ajsr.2016.242.247

Ali, M., Nadeem, M., Siddique, A., Ahmad, S., & Ijaz, A. (2020). Addressing Sinkhole Attacks In Wireless Sensor Networks - A Review. *International Journal of Scientific and Technology Research (IJSTR)*, *9*(08), 406–411.

Alqahtani, M., Gumaei, A., Mathkour, H., & Ismail, M. M. Ben. (2019). A genetic-based extreme gradient boosting model for detecting intrusions in wireless sensor networks. *Sensors (Switzerland)*, *19*(20). https://doi.org/10.3390/s19204383

Amrita, & Ahmed, P. (2012). a Study of Feature Selection Methods in Intrusion Detection System: a Survey. *International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)*, *2*(3), 1–25.

Anwar, R. W., Bakhtiari, M., Zainal, A., & Qureshi, K. N. (2015). A survey of wireless sensor network security and routing techniques. *Research Journal of Applied Sciences, Engineering and Technology*, *9*(11), 1016–1026. https://doi.org/10.19026/rjaset.9.2595

Babaeer, H. A., Al-Ahmadi, S. A., Sen, J., Abdulmalik Danmallam Bello, Dr. O. S. Lamba,

Ramírez Gómez, J., Vargas Montoya, H. F., León Henao, Á., Samara, G., Albesani, G., Alauthman, M., Al Khaldy, M., Hitesh Yadav, Ms. Sana Tak, Singh, P. P., Gupta, O., Saini, S., Kumar, S. R., Centre, S. R., … Dr. O. S. Lamba. (2020). How to Detect and Mitigate Sinkhole Attack in Wireless Sensor Network (WSN). *International Journal of Engineering Research And*, *7*(05), 98–105. https://doi.org/10.17577/ijertv9is050099

Ben-Haim, Y., & Tom-Tov, E. (2010). A streaming parallel decision tree algorithm. *Journal of Machine Learning Research*, *11*, 849–872.

Bhushan, B., & Sahoo, G. (2018). Recent advances in attacks, technical challenges, vulnerabilities and their countermeasures in wireless sensor networks. *Wireless Personal Communications*, *98*(2), 2037–2077. https://doi.org/10.1007/s11277-017-4962-0

Bhushan, B., & Sahoo, G. (2019). Routing protocols in wireless sensor networks. *Studies in Computational Intelligence*, *776*(2), 215–248. https://doi.org/10.1007/978-3-662-57277-1_10

Carlos-Mancilla, M., López-Mellado, E., & Siller, M. (2016). Wireless sensor networks formation: Approaches and techniques. *Journal of Sensors*, *2016*. https://doi.org/10.1155/2016/2081902

Charbuty, B., & Abdulazeez, A. (2021). Classification Based on Decision Tree Algorithm for Machine Learning. *Journal of Applied Science and Technology Trends*, *2*(01), 20–28. https://doi.org/10.38094/jastt20165

Dhanaraj, R. K., Krishnasamy, L., Geman, O., & Izdrui, D. R. (2021). Black hole and sink hole attack detection in wireless body area networks. *Computers, Materials and Continua*, *68*(2), 1949–1965. https://doi.org/10.32604/cmc.2021.015363

Dong, Q., & Liu, D. (2009). Resilient cluster leader election for wireless sensor networks. *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2009*. https://doi.org/10.1109/SAHCN.2009.5168966

Gupta, G. P., & Kulariya, M. (2016). *A Framework for Fast and Efficient Cyber Security*

Network Intrusion Detection using Apache Spark. *93*(September), 824–831. https://doi.org/10.1016/j.procs.2016.07.238

Hidoussi, F., Toral-Cruz, H., Boubiche, D. E., Lakhtaria, K., Mihovska, A., & Voznak, M. (2015). Centralized IDS Based on Misuse Detection for Cluster-Based Wireless Sensors Networks. *Wireless Personal Communications*, *85*(1), 207–224. https://doi.org/10.1007/s11277-015-2734-2

Hidru, T. (2020). *ADDIS ABABA UNIVERSITY COLLEGE OF NATURAL SCIENCES Modeling Network Intrusion Detection System Based on Anomaly Approach Using Machine Learning Techniques*.

Ifzarne, S., Tabbaa, H., Hafidi, I., & Lamghari, N. (2021). Anomaly Detection using Machine Learning Techniques in Wireless Sensor Networks. *Journal of Physics: Conference Series*, *1743*(1). https://doi.org/10.1088/1742-6596/1743/1/012021

Ikonomakis, M. (2005). *Text Classification Using Machine Learning Techniques*. *4*(8), 966–974.

Ioannou, C., & Vassiliou, V. (2019). Classifying security attacks in IoT networks using supervised learning. *Proceedings - 15th Annual International Conference on Distributed Computing in Sensor Systems, DCOSS 2019*, 652–658. https://doi.org/10.1109/DCOSS.2019.00118

Ioannou, C., & Vassiliou, V. (2020). Accurate Detection of Sinkhole Attacks in IoT Networks Using Local Agents. *2020 Mediterranean Communication and Computer Networking Conference, MedComNet 2020*. https://doi.org/10.1109/MedComNet49392.2020.9191503

Isaac Sajan, R., & Jasper, J. (2020). Trust-based secure routing and the prevention of vampire attack in wireless ad hoc sensor network. *International Journal of Communication Systems*, *33*(8), 1–22. https://doi.org/10.1002/dac.4341

Ishaq, Z., Park, S., & Yoo, Y. (2018). A Security Framework for Cluster-Based Wireless Sensor Networks against the Selfishness Problem. *Wireless Communications and Mobile*

Computing, *2018*. https://doi.org/10.1155/2018/8961239

Iwendi, C., Bashir, A. K., Peshkar, A., Sujatha, R., Chatterjee, J. M., Pasupuleti, S., Mishra, R., Pillai, S., & Jo, O. (2020). COVID-19 patient health prediction using boosted random forest algorithm. *Frontiers in Public Health*, *8*(July), 1–9. https://doi.org/10.3389/fpubh.2020.00357

Jamil, A., Ali, M. Q., & Abd Alkhalec, M. E. (2021). Sinkhole attack detection and avoidance mechanism for RPL in wireless sensor networks. *Annals of Emerging Technologies in Computing*, *5*(Special issue 5), 94–101. https://doi.org/10.33166/AETiC.2021.05.011

Karami, A. (2018). PT US CR. *Expert Systems With Applications*. https://doi.org/10.1016/j.eswa.2018.04.038

Kaur, A., & Grover, A. (2015). LEACH and Extended LEACH Protocols in Wireless Sensor Network-A Survey. *International Journal of Computer Applications*, *116*(10), 1–5. https://doi.org/10.5120/20369-2576

Kaur, K., & Singh, P. (2017). Comparative Analysis of AODV & LEACH Protocol in WSN. *Imperial Journal of Interdisciplinary Research (IJIR*, *3*(4), 1181–1185.

Khan, M. K., Shiraz, M., Shaheen, Q., Butt, S. A., Akhtar, R., Khan, M. A., & Changda, W. (2021). Hierarchical Routing Protocols for Wireless Sensor Networks: Functional and Performance Analysis. *Journal of Sensors*, *2021*. https://doi.org/10.1155/2021/7459368

Kibirige, G. W. (n.d.). *A Survey on Detection of Sinkhole Attack in Wireless Sensor Network*.

Kumar, S. R., Centre, S. R., Nadu, T., Engineering, C., Centre, S. R., Nadu, T., & Nadu, T. (2017). *Analysis of Sinkhole Attack in Leach Based Wireless Sensor Network*. *116*(24), 185–197.

Kushwaha, P., Buckchash, H., & Raman, B. (2017). *Anomaly Based Intrusion Detection Using Filter Based Feature Selection on KDD-CUP 99*. 839–844.

Marina, M. K., & Das, S. R. (2006). Ad hoc on-demand multipath distance vector routing. *Wireless Communications and Mobile Computing*, *6*(7), 969–988.

https://doi.org/10.1002/wcm.432

Mathew, A., & Terence, J. S. (2018). A survey on various detection techniques of sinkhole attacks in WSN. *Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017*, *2018-Janua*, 1115–1119. https://doi.org/10.1109/ICCSP.2017.8286550

Mehmood, A., Khanan, A., Umar, M. M., Abdullah, S., Ariffin, K. A. Z., & Song, H. (2017). Secure Knowledge and Cluster-Based Intrusion Detection Mechanism for Smart Wireless Sensor Networks. *IEEE Access*, *6*(c), 5688–5694. https://doi.org/10.1109/ACCESS.2017.2770020

Mighan, S. N., & Kahani, M. (2021). A novel scalable intrusion detection system based on deep learning. *International Journal of Information Security*, *20*(3), 387–403. https://doi.org/10.1007/s10207-020-00508-5

Naseer, S., Saleem, Y., Khalid, S., Bashir, M. K., Han, J., Iqbal, M. M., & Han, K. (2018). Deep Neural Networks. *IEEE Access*, *PP*(8), 1. https://doi.org/10.1109/ACCESS.2018.2863036

Nithiyanandam, N., & Latha, P. (2019). Artificial bee colony based sinkhole detection in wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing*, *0123456789*. https://doi.org/10.1007/s12652-019-01404-0

Patel, V., & Gheewala, J. (2015). An Efficient Session Key Management Scheme for Cluster Based Wireless Sensor Networks. *Souvenir of the 2015 IEEE International Advance Computing Conference, IACC 2015*, 963–967. https://doi.org/10.1109/IADCC.2015.7154847

Patil, A., Mumbai, N., & Mumbai, N. (n.d.). *A Survey Paper on Sink-hole Attack Resilience and Energy Efficiency using Internet of Things*.

Patil, S., & Chaudhari, S. (2016). DoS Attack Prevention Technique in Wireless Sensor Networks. *Procedia Computer Science*, *79*, 715–721. https://doi.org/10.1016/j.procs.2016.03.094

Pawa, T. (2011). *Analysis of Low Energy Adaptive Clustering Hierarchy (LEACH) protocol*. *107*, 1–37.

Picek, S., Heuser, A., Jovic, A., Bhasin, S., & Regazzoni, F. (2019). The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, *2019*(1), 209–237. https://doi.org/10.13154/tches.v2019.i1.209-237

Qassim, Q. S., Zin, A. M., Juzaiddin, M., & Aziz, A. (2016). *Anomalies Classification Approach for Network-based Intrusion Detection System*. *January*.

R, R. L. T. S. (2014). *Detection Techniques of Sinkhole Attack in WSNs : A Survey*. *3*(6), 12–14.

Radoglou Grammatikis, P. I., Sarigiannidis, P. G., & Moscholios, I. D. (2019). Securing the Internet of Things: Challenges, threats and solutions. *Internet of Things (Netherlands)*, *5*, 41–70. https://doi.org/10.1016/j.iot.2018.11.003

Rajeshkumar, G., & Valluvan, K. R. (2016). An Energy Aware Trust Based Intrusion Detection System with Adaptive Acknowledgement for Wireless. *Wireless Personal Communications*. https://doi.org/10.1007/s11277-016-3349-y

Rao, R., Kumar, M., & Palanivel, K. (2011). Secure Routing Based on ECC in Cluster Wireless Sensor Networks. *International Journal*, *0976*. http://www.researchgate.net/publication/233916458_Secure_Routing_Based_on_ECC_in_Cluster_Wireless_Sensor_Networks/file/d912f50cde3f788f36.pdf

Ray, S. (2019). A Quick Review of Machine Learning Algorithms. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 35–39.

Raza, N., Umar Aftab, M., Qasim Akbar, M., Ashraf, O., & Irfan, M. (2016). Mobile Ad-Hoc Networks Applications and Its Challenges. *Communications and Network*, *08*(03), 131–136. https://doi.org/10.4236/cn.2016.83013

Rezvi, M. A., Moontaha, S., Trisha, K. A., Cynthia, S. T., & Ripon, S. (2021). Data mining

approach to analyzing intrusion detection of wireless sensor network. *Indonesian Journal of Electrical Engineering and Computer Science*, *21*(1), 516–523. https://doi.org/10.11591/ijeecs.v21.i1.pp516-523

Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2020). Introduction to Anaconda and Python: Installation and setup. *The Quantitative Methods for Psychology*, *16*(5), S3–S11. https://doi.org/10.20982/tqmp.16.5.s003

Samara, G., Albesani, G., Alauthman, M., & Al Khaldy, M. (2020). Energy-efficiency routing algorithms in wireless sensor networks: A survey. *International Journal of Scientific and Technology Research*, *9*(1), 4415–4418.

Sehrawat, H., Singh, Y., & Siwach, V. (2018). Analysis of AODV protocol under sinkhole attack in wireless sensor network. *International Journal of Engineering & Technology*, *7*(2.4), 153. https://doi.org/10.14419/ijet.v7i2.4.13028

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification. *Augmented Human Research*, *5*(1). https://doi.org/10.1007/s41133-020-00032-0

Sharif, L., & Ahmed, M. (2010). The Wormhole Routing Attack in Wireless Sensor Networks (WSN). *Journal of Information Processing Systems*, *6*(2), 177–184. https://doi.org/10.3745/jips.2010.6.2.177

Shen, V. R. L., Wei, C., & Juang, T. T. (n.d.). JAVASCRIPT MALWARE DETECTION USING A HIGH-LEVEL FUZZY PETRI NET. *2018 International Conference on Machine Learning and Cybernetics (ICMLC)*, *2*, 511–514.

Shi, D., Zhang, S., Wang, J., & Gong, Y. (2015). Detection and Association Based Multi-target Tracking in Surveillance Video. *Proceedings - 2015 IEEE International Conference on Multimedia Big Data, BigMM 2015*, 377–382. https://doi.org/10.1109/BigMM.2015.19

Singh, P., Gupta, O., & Saini, S. (2017). *A Brief Research Study of Wireless Sensor Network*. *10*(5), 733–739. http://www.ripublication.com

Sivasakthi, M., & Pandiyan, M. (n.d.). *Machine Learning Algorithms to Predict Students ' Programming Performance : A comparative Study Keywords : Machine learning algorithms ; Classification ; Prediction ; Computer rogramming II . RELATED WORKS*. *24*(6), 1–8.

Tiwari, V., & A. Waoo, A. (2020). Comprehensive Study on Metaheuristics FADE Based Artificial Bee Colony Optimization Algorithm to Improve Performance of Wireless Networks. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 236–243. https://doi.org/10.32628/cseit206549

Upadhyay, D., Manero, J., Zaman, M., & Sampalli, S. (2021). Gradient Boosting Feature Selection with Machine Learning Classifiers for Intrusion Detection on Power Grids. *IEEE Transactions on Network and Service Management*, *18*(1), 1104–1116. https://doi.org/10.1109/TNSM.2020.3032618

Verma, A., Khare, P., & Tech, M. (2021). *A SURVEY : Detection and correction of sinkhole attack in wireless sensors networks by leach protocol*. 887–890.

Vijayarajeswari, R., & Priya, S. (2015). *AN EFFICIENT DETECTION AND PREVENTION OF DDoS ATTACK IN WSN*. *18*(2), 123–127.

Vishwas, D. B., N, I. I. C. C., & Sreenivas, I. I. I. T. H. (2016). *Discover and Prevent the Sinkhole Attacks in Wireless Sensor Network using Clustering Protocol*. *4*(2), 2–4.

Wang, Y., Wu, X., Chen, Z., Ren, F., Feng, L., & Du, Q. (2019). Optimizing the predictive ability of machine learning methods for landslide susceptibility mapping using smote for Lishui city in Zhejiang province, China. *International Journal of Environmental Research and Public Health*, *16*(3). https://doi.org/10.3390/ijerph16030368

Waoo, A. A., & Tiwari, V. (2021). *Challenges in Sinkhole Attack Detection in Wireless Sensor Network*. *August*. https://doi.org/10.35940/ijdcn.C5016.081421

Xu, Y., Jiao, W., & Tian, M. (2021). An Energy-Efficient Routing Protocol for 3D Wireless Sensor Networks. *IEEE Sensors Journal*, *21*(17), 19550–19559. https://doi.org/10.1109/JSEN.2021.3086806

Yadav, H. (2018). *Detection of Sinkhole Attack in Wireless Sensor Network Using Ad-hoc on-demand Distance Vector*. *7*(05), 188–192.

Yau, K. A., Elkhatib, Y., Hussain, A., & Al-fuqaha, A. L. A. (2019). Unsupervised Machine Learning for Networking : Techniques , Applications and Research Challenges. *IEEE Access*, *7*, 65579–65615. https://doi.org/10.1109/ACCESS.2019.2916648

APPENDIXES

Appendix A: Sample WSN-DS dataset

```
In [4]: # dataframe1 = pd.read_csv("WSN-DSM.CSV")
        dataframe = pd.read_csv("WSN-DS.CSV")
        dataframe.head(10)
```

Out[4]:

	id	Time	Is_CH	who_CH	Dist_To_CH	ADV_S	ADV_R	JOIN_S	JOIN_R	SCH_S	SCH_R	Rank	DATA_S	DATA_R	Data_Sent_To_BS	dist_CH_To_BS
0	101000	50	1	101000	0.00000	1	0	0	25	1	0	0	0	1200	48	130.08535
1	101001	50	0	101044	75.32345	0	4	1	0	0	1	2	38	0	0	0.00000
2	101002	50	0	101010	46.95453	0	4	1	0	0	1	19	41	0	0	0.00000
3	101003	50	0	101044	64.85231	0	4	1	0	0	1	16	38	0	0	0.00000
4	101004	50	0	101010	4.83341	0	4	1	0	0	1	25	41	0	0	0.00000
5	101005	50	0	101010	31.91198	0	4	1	0	0	1	18	41	0	0	0.00000
6	101006	50	0	101044	24.34167	0	4	1	0	0	1	5	38	0	0	0.00000
7	101007	50	0	101010	26.75033	0	4	1	0	0	1	21	41	0	0	0.00000
8	101008	50	0	101044	63.66485	0	4	1	0	0	1	17	38	0	0	0.00000
9	101009	50	0	101000	32.90217	0	4	1	0	0	1	12	48	0	0	0.00000

Appendix B: Data Normalization

```
In [21]: from sklearn import preprocessing
         from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()

         Scaled_Cols = [' Is_CH', ' ADV_S', ' ADV_R', ' SCH_S', ' SCH_R', ' DATA_S', ' DATA_R', ' Data_Sent_To_BS' ]

         df[Scaled_Cols] = scaler.fit_transform (df[Scaled_Cols])
```

```
In [22]: df.head()
```

Out[22]:

	id	Is_CH	ADV_S	ADV_R	SCH_S	SCH_R	DATA_S	DATA_R	Data_Sent_To_BS	Attack_type
0	101000	2.763714	0.355289	-0.985272	0.258106	-1.720360	-1.053636	4.890899	2.206935	Normal
1	101001	-0.361832	-0.129878	-0.417438	-0.104904	0.581274	-0.161081	-0.320918	-0.232198	Normal
2	101002	-0.361832	-0.129878	-0.417438	-0.104904	0.581274	-0.090616	-0.320918	-0.232198	Normal
3	101003	-0.361832	-0.129878	-0.417438	-0.104904	0.581274	-0.161081	-0.320918	-0.232198	Normal
4	101004	-0.361832	-0.129878	-0.417438	-0.104904	0.581274	-0.090616	-0.320918	-0.232198	Normal

Appendix C: Sample code for the Models

In [11]:
```
# Function to calculate missing values by column
def missing_values_table(df):
        # Total missing values
        mis_val = df.isnull().sum()

        # Percentage of missing values
        mis_val_percent = 100 * df.isnull().sum() / len(df)

        # Make a table with the results
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)

        # Rename the columns
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})

        # Sort the table by percentage of missing descending
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)

        # Print some summary information
        print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")

        # Return the dataframe with missing information
        return mis_val_table_ren_columns
```

Appendix D: Sample code for the Models

Classification Models

In [49]:
```
# train models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import VotingClassifier
from sklearn import svm


# 2. Testing Accuracy, Precision , F1 and Recall scores
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

# importing evaluation metrices.
from sklearn.metrics import r2_score,mean_squared_error,accuracy_score,recall_score,precision_score,confusion_matrix
```

Appendix E: Accuracy comparison

Testing accruacy Visulzzation

In [79]:
```
## visulization of Vlaidation data
#setting color
color=sns. set_palette("Set1");

#plotting
df_metrics_test.T.plot(kind="bar", figsize=(10, 4));
plt.legend(loc=2 , bbox_to_anchor=(1.0, 1), fontsize = 10)
plt.xticks(rotation=20);
plt.title('Model Performance on Validation Data');
```

Appendix F: Resampling

Model Performance Evaluation

Cross-Validation on OverSampled Training Data

In [190]:
```
# loop through all models to get the mean cross validated score
for name, model in models_oversample:

    kfold = StratifiedKFold(
        n_splits=5, shuffle=True, random_state=1
    )  # Setting number of splits equal to 5
    cv_result = cross_val_score(
        estimator=model, X=X_train_over, y=y_train_over, cv=kfold
    )
    results_oversample.append(cv_result)
    names_oversample.append(name)
```

In [191]:
```
print('\nSummary Statistics of CV Scores on OverSampled Training Set:')
df_CV_score_over = pd.DataFrame()
df_CV_score_over[" "] = names_oversample
df_CV_score_over["Min"] = np.array(results_oversample).min(axis=1)
df_CV_score_over["Average"] = np.array(results_oversample).mean(axis=1)
df_CV_score_over["Max"] = np.array(results_oversample).max(axis=1)
df_CV_score_over["STD"] = np.array(results_oversample).std(axis=1)
df_CV_score_over.set_index(' ',inplace=True)
df_CV_score_over.T
```

Summary Statistics of CV Scores on OverSampled Training Set:

Out[191]:

	Logistic regression over_sample	Decision Trees Clasifier over_sample	Random Forest Clasifier over_sample	Gaussian naive bayes over_sample	KNeighborsClassifier over_sample
Min	0.499995	0.988366	0.993286	0.712296	0.985856
Average	0.500000	0.988830	0.993551	0.714036	0.986166
Max	0.500005	0.989473	0.994070	0.716099	0.986474
STD	0.000003	0.000365	0.000293	0.001229	0.000246