

**SECURE COMMUNICATION
METHOD IN WIRELESS SENSOR
NETWORKS**

TEE YEW CHUN

**FACULTY OF COMPUTING AND INFORMATICS
UNIVERSITI MALAYSIA SABAH
2024**

**SECURE COMMUNICATION
METHOD IN WIRELESS SENSOR
NETWORKS**

TEE YEW CHUN

**THESIS SUBMITTED IN PARTIAL FULFILLMENT
FOR THE DEGREE OF BACHELOR OF COMPUTER
SCIENCE WITH HONOURS
(NETWORK ENGINEERING)**

**FACULTY OF COMPUTING AND INFORMATICS
UNIVERSITI MALAYSIA SABAH**

2024

NAME : TEE YEW CHUN

MATRIC NUMBER : BI20110050

TITLE : SECURE COMMUNICATION METHOD IN WIRELESS SENSOR
NETWORKS.

DEGREE : BACHELOR OF COMPUTING WITH HONOURS
(NETWORK ENGINEERING)

CERTIFIED BY;

1. SUPERVISOR

DR. SALMAH BINTI FATTAH

Signature


SALMAH BINTI FATTAH
Lecturer
Faculty of Computing and Informatics
Universiti Malaysia Sabah
(DATE: 23/1/2024)

DECLARATION

I hereby declare that the material in this thesis is entirely mine, except for quotations, equations, summaries, and references, which have been properly acknowledged.

26 APRIL 2023

YewChun

TEE YEW CHUN

BI20110050

ACKNOWLEDGEMENT

I would like to express my gratitude to my supervisor, Dr. Salmah Fattah, who gave me lots of guidance, direction, and counsel for me to complete this final year project. My final year project would not have been possible without her help and counsel.

TEE YEW CHUN

26 APR 2023

ABSTRACT

Wireless sensor networks (WSNs) are vulnerable to various attacks, including unauthorized access, distributed denial of service (DDoS), eavesdropping, node capture, wormhole attack, and Sybil attack. However, the performance of attack detection methods in WSNs is not well-defined. This project aims to address these challenges by investigating and selecting a secure communication detection method for WSN attacks, implementing it, and evaluating its performance in terms of accuracy, precision, and time elapsed. The experimental design consists of several steps, including data collection, data preparation and preprocessing, feature extraction, data training and testing, detection using machine learning algorithms, and performance evaluation. The experiment is conducted using Python programming language and relevant libraries such as SKLEARN, PANDAS, and NUMPY. Multiple machine learning algorithms, namely Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Artificial Neural Network (ANN), are utilized for WSN DDoS attack detection. The three selected classifiers (SVM, KNN, and ANN) are employed to detect WSN DDoS attacks. Based on the experimental results, KNN emerges as the most effective classifier. It achieves an accuracy of 99.63%, indicating a high level of correct predictions. Additionally, KNN demonstrates a precision of 99.64%, signifying a low rate of false positives. Furthermore, KNN exhibits the shortest time elapsed, with an average of 91.05 seconds (approximately 1.52 minutes) to process the extensive dataset. The project's outcomes contribute to enhancing the security of WSNs by providing insights into the performance of different detection methods. The findings emphasize the superiority of KNN over SVM and ANN in terms of accuracy, precision, and computational efficiency. These results can guide researchers and practitioners in choosing an appropriate detection method for securing wireless sensor networks against various attacks.

ABSTRAK

Rangkaian sensor tanpa wayar (Wireless Sensor Networks atau WSN) mudah terdedah kepada pelbagai serangan, termasuk akses tidak sah, penafian perkhidmatan (DDoS), pendengaran sulit, penangkapan nod, serangan terowong, dan serangan Sybil. Walau bagaimanapun, prestasi kaedah pengesanan serangan dalam WSN tidak ditakrifkan dengan baik. Projek ini bertujuan untuk menangani cabaran ini dengan menyiasat dan memilih kaedah pengesanan komunikasi yang selamat untuk serangan WSN, melaksanakannya, dan menilai prestasinya dari segi ketepatan, kecermatan, dan masa yang diambil. Reka bentuk eksperimen terdiri daripada beberapa langkah, termasuk pengumpulan data, penyediaan dan pra pemprosesan data, penyarian sifat latihan dan pengujian data, pengesanan menggunakan algoritma pembelajaran mesin, dan penilaian prestasi. Eksperimen dijalankan menggunakan bahasa pengaturcara Python dan pustaka berkaitan seperti SKLEARN, PANDAS, dan NUMPY. Beberapa algoritma pembelajaran mesin, iaitu Mesin Vector Sokongan (SVM), Jiran Terdekat K (K-Nearest Neighbors atau KNN), dan Rangkaian Neural Buatan (Artificial Neural Network atau ANN), digunakan untuk pengesanan serangan DDoS dalam WSN. Tiga pengelas yang dipilih (SVM, KNN, dan ANN) digunakan untuk mengesan serangan DDoS dalam WSN. Berdasarkan hasil eksperimen, KNN muncul sebagai pengelas yang paling berkesan. Ia mencapai ketepatan 99.63% yang menunjukkan tahap jangkaan yang tinggi. Selain itu, KNN menunjukkan kejulitan 99.64%, menandakan kadar positif palsu yang rendah. Tambahan pula, KNN menunjukkan masa pemprosesan eksperimen yang paling singkat, dengan purata 91.05 saat (kira-kira 1.52 minit) untuk memproses dataset yang besar. Hasil projek ini menyumbang kepada peningkatan keselamatan WSN dengan memberikan pandangan mengenai prestasi kaedah pengesanan yang berbeza. Kajian menekankan kelebihan besar KNN berbanding SVM dan ANN dari segi ketepatan, kejulitan, dan kecekapan komputasi. Hasil kajian ini boleh membimbing penyelidik dan pengamal dalam memilih kaedah pengesanan yang sesuai untuk mengamankan rangkaian sensor tanpa wayar daripada pelbagai serangan.

TABLE OF CONTENTS

DECLARATION.....	I
ACKNOWLEDGEMENT.....	II
ABSTRACT	III
ABSTRAK.....	IV
TABLE OF CONTENTS	V-VII
List of Tables.....	VIII
List of Figures.....	IX-X
CHAPTER 1 INTRODUCTION	1
1.1 Introduction.....	1-2
1.2 Project Background	3
1.3 Project Statements.....	3
1.4 Project Objectives	3
1.5 Project Scopes	4
1.6 Organization of the Report.....	5
CHAPTER 2 LITERATURE REVIEW	6
2.1 Introduction.....	6
2.2 Wireless Sensor Network Attack	6-9
2.3 Feature Extraction	10
2.4 Machine Learning Algorithms.....	10
2.4.1 Support Vector Machines (SVMs).....	11
2.4.2 K-Nearest Neighbors (KNN).....	11
2.4.3 Artificial Neural Network (ANN)	12
2.5 Wireless Sensor Networks DDoS Attack Detection in past 5 years.....	12-19
2.6 Summary of Wireless Sensor Networks DDoS Attack Detection in past 5 years	20-22
2.7 Summary.....	22
CHAPTER 3 METHODOLOGY	23
3.1 Introduction.....	23
3.2 Project Methodology	23-25
3.3 Software and Hardware Requirements	25
3.4 Flowchart of Activities.....	26
3.5 Project Milestones	27
CHAPTER 4 EXPERIMENTAL DESIGN	28
4.1 Introduction.....	28

4.2 Experiment Design Plan	28-29
4.3 Data Collection.....	30
4.4 Data Preparation and Preprocessing.....	30-31
4.5 Training and Testing Data	31
4.6 Wireless Sensor Network DDoS attack detection using Multiple ML Algorithms.....	31
4.7 Performance Evaluation	32
4.7.1 Accuracy	32
4.7.2 Precision	32
4.7.3 Time Elapsed.....	32
4.7.4 Regression Metrics	33
4.8 Interface Design	34
4.9 Summary.....	34
CHAPTER 5 IMPLEMENTAION	35
5.1 Introduction.....	35
5.2 Experiment Implementation	35-36
5.3 Feature Extraction	36-38
5.4 Testing and Training of Dataset.....	38-42
5.5 Summary.....	42-43
CHAPTER 6 RESULTS AND DISCUSSION	44
6.1 Introduction.....	44
6.2 Experiment Result of Feature Extraction.....	44-45
6.3 Experiment Results of Machin Learning Algorithms.....	45
6.3.1 Accuracy	45-46
6.3.2 Precision	46-47
6.3.3 Time Elapsed.....	47
6.3.4 Mean Squared Error (MSE)	48
6.3.5 Mean Absolute Error (MAE).....	48
6.3.6 Mean Absolute Percentage Error (MAPE).....	48
6.3.7 Confusion Matrix.....	49
6.4 Experiment Result for SVM (Support Vector Machine)	50-51
6.5 Experiment Result for KNN (K-Nearest Neighbors).....	52-53
6.6 Experiment Result for ANN (Artificial Neural Network)	54-55
6.7 Discussion	55-57
6.8 Interface Design	58-61
6.9 Summary.....	62

CHAPTER 7 Conclusion.....	63
7.1 Introduction.....	63
7.2 Objective Achievement	63-64
7.3 Project Constraint.....	64
7.4 Future Work	64
7.5 Summary.....	64
REFERENCES	65-67
APPENDIX	68-70

LIST OF TABLES

Table 1.1: Project Scope.....	4
Table 2.1: Summary result of the Experiment of K.Nirmala et al., 2021	18
Table 2.2: Summary of Wireless Sensor Networks DDoS Attack Detection in past 5 years ..	20
Table 3.3: Software requirements and description	25
Table 3.2: Project Milestones	27
Table 6.1: Summary of Results.....	55

LIST OF FIGURES

Figure 2.1: Attacks in WSN	7
Figure 2.2: Classification of DDoS Attack.....	8
Figure 2.3: DDoS attacks on Two Architectures	9
Figure 2.4: Support Vector Machine.....	11
Figure 2.5: K-Nearest Neighbour (KNN)	11
Figure 2.6: Artificial Neural Network (ANN)	12
Figure 2.7: Result Analysis for Normal IoT System and IoT Threat System.....	13
Figure 2.8: Accuracy and F-Score analysis of SHOQNN-AC approach with other IDS methods	14
Figure 2.9: Comparison between Different Classifier.....	15
Figure 2.10: Result of Evaluation of Quality of Classifiers.....	17
Figure 3.1: Flowchart of Project Methodology	23
Figure 3.2: Flowchart of Activities.....	26
Figure 4.1: Overview of the Overall Design.....	29
Figure 5.1: Original Wireless Sensor Network DDoS Attack Dataset from Kaggle.....	35
Figure 5.2: Experiment Setup to get the Feature Importance Score	36
Figure 5.3: Experiment Setup to get the Important Features	37
Figure 5.4: Importing Libraries	38
Figure 5.5: Importing .csv Dataset.....	38
Figure 5.6: Extracting Features	39
Figure 5.7: Applying Label Encoding	39
Figure 5.8: Splitting Dataset.....	39
Figure 5.9: Creating instance for the classifier	39
Figure 5.10: Create Variables to track training progress.....	40
Figure 5.11: Training Loop.....	40
Figure 5.12: Calculating Total Accuracy and Precision.....	41
Figure 5.13: Calculating Average Training and Validation Loss	41
Figure 5.14: Calculating Total Time Elapsed	42
Figure 6.1: Outcome Feature Importance Score.....	44
Figure 6.2: Outcome Important Features	45
Figure 6.3: One of the experiment result screenshots of SVM	50
Figure 6.4: Confusion Matrix for SVM.....	50
Figure 6.5: Experiment Result for SVM.....	51

Figure 6.6: One of the experiment result screenshots of KNN	52
Figure 6.7: Confusion Matrix for KNN	52
Figure 6.8: Experiment Result for KNN.....	53
Figure 6.9: One of the experiment result screenshots of ANN	54
Figure 6.10: Confusion Matrix for ANN	54
Figure 6.11: Experiment Result for ANN.....	55
Figure 6.12: Result of MSE, MAE and MAPE of ANN, KNN and SVM.....	56
Figure 6.13: Result of Accuracy and Precision of ANN, KNN and SVM.....	56
Figure 6.14: Total Time Elapsed of ANN, KNN and SVM.....	57
Figure 6.15: WSN DDoS Attack Prediction Dashboard.....	58
Figure 6.16: Dashboard view of WSN DDoS Attack Prediction Dashboard.....	59
Figure 6.17: Train with Feature Selection	60
Figure 6.18: Train without Feature Selection	61

CHAPTER 1

INTRODUCTION

1.1 Introduction

Wireless sensor networks (WSNs) are networks of specialized, spatially dispersed sensors that monitor and record environmental physical variables and transmit the data to a central location. WSNs can measure environmental parameters such as temperature, noise, pollution levels, humidity, and wind. (Ullo et al., 2020).

WSNs are used in wide range of fields, including monitoring of habitats, environmental or earth sensing (air quality monitoring, detecting forest fires, detecting landslides, monitoring water quality, and preventing natural disasters), industrial monitoring (monitoring of machine health, data logging, and water/wastewater monitoring), threat detection (incident monitoring), and supply chains. WSNs can be used to follow the flow of items in a warehouse, keep tabs on the health of patients in a hospital, and even regulate the temperature and lighting in a home, to give a clear perspective.

WSNs may encounter a variety of difficulties, including limited resources (memory, compute, and power) and unreliable communications (data transfer, disputes, and delay). However, new communication techniques and algorithms that address these issues have been created because of technological and research advancements, making WSNs more dependable, effective, and scalable.

Unauthorised Access, Distributed Denial of Service (DDoS) Attack, Eavesdropping, and Node Capture are some of the potential security risks of WSN. Additionally, attackers can capture nodes to obtain access to sensitive information or to initiate DDoS, Wormhole, or Sybil attacks (Sohraby et al., 2007).

A distributed-denial-of-service (DDoS) attack is a sort of cyber-attack that attempts to disrupt the availability of a network, system, or service by flooding it with unauthorised requests or exploiting system weaknesses to deplete system resources. DDoS attacks can be

extremely damaging in the context of wireless sensor networks (WSNs) since they can disrupt regular network operation, reduce network performance, and jeopardise data collection and communication integrity.

The common DDoS Attacks can be classified into few categories:

- 1) Flooding attacks: These attacks involve overloading the network with a large number of illegitimate communications, which overwhelms the network's resources and causes congestion.
- 2) Jamming attacks: These jamming attacks consist of emitting radio signals on the same frequency as the WSN which causing interference and disrupting communication between sensor nodes or between sensor nodes and the base station.
- 3) Resource depletion attacks: These attacks involve exploiting vulnerabilities in sensor nodes or the base station to deplete its resources, such as CPU, memory, or energy.
- 4) Spoofing attacks: These attacks disrupt the normal operation of the network by impersonating legitimate nodes or the base station. For instance, an adversary can impersonate a valid sensor node or base station and send fraudulent messages or commands to disrupt communication or mislead the network.
- 5) Physical attack: These attacks involve physically damaging or destroying sensor nodes or the base station, thereby disrupting the network's operation.

In a nutshell, a DDoS attack is one that attempts to bring down a computer system or network such that its intended users cannot access it. DDoS attacks do this by sending data to the target that causes it to crash or by flooding it with traffic. A DDoS attack, in its most basic form, denies legitimate users, such as employees, members, or account holders, access to resources or services.

DDoS attacks routinely target well-known organisations' web servers, including banks, e-commerce, media, and even governmental and commercial organisations. DDoS assaults can be time and money consuming to cope with, even though they rarely result in the theft or loss of significant data or other assets.

1.2 Project Background

Due to the distributed nature, wireless sensor networks (WSNs) are currently exposing to security threats such as Unauthorised Access, Distributed Denial of Service (DDoS) Attack, Eavesdropping, and Node Capture. Attackers can easily capture nodes to obtain access to sensitive information or to initiate DDoS, Wormhole, or Sybil attacks.

For instance, the problem background of secure communication methods in WSNs involves ensuring the performance of the security communication among sensor nodes while considering the unique challenges and constraints of WSNs, such as limited resources and unreliable communications of the network. Developing effective and efficient secure communication protocols for WSNs is a difficult task that requires careful consideration of these factors to ensure the security and reliability of WSNs in various applications, include environment monitor, healthcare, industrial automation, and smart agriculture.

1.3 Project Statements

- Lack of the information which classification is more suitable for wireless sensor network's attack detection method.
- Wireless sensor network's DDoS attack detection method's classification is not yet well defined in their performance and suitability in past five years research.

1.4 Project Objectives

- 1) To investigate and choose the most suitable classification for DDoS attack detection.
- 2) To implement an experiment of DDoS attack detection for wireless sensor network.
- 3) To evaluate the performance of DDoS attack detection for wireless sensor network attack detection.

1.5 Project Scopes

Table 1.1: Project Scope

Project Justification:	<ul style="list-style-type: none">• A researched based project on secure communication detection method for wireless sensor network attack.
Project Objectives:	<ul style="list-style-type: none">• Investigate and choose the most suitable classification for DDoS attack detection.• Build up an experiment of DDoS attack detection for wireless sensor network.• Evaluate the performance of DDoS attack detection for wireless sensor network attack detection.
Data Source:	<ul style="list-style-type: none">• Public dataset from Kaggle website
Project limitation:	<ul style="list-style-type: none">• Only focus on a few secure communication detection methods and several attacks for wireless sensor network.
Estimated time to complete:	<ul style="list-style-type: none">• Around one year (from February 2023 to January 2024)
Project acceptance criteria:	<ul style="list-style-type: none">• Success determines the most effective secure communication detection method for wireless sensor network with high performance.

1.6 Organization of the Report

This final year project report is organized into several key sections, each serving a distinct purpose in presenting the research and findings related to Wireless Sensor Network (WSN) DDoS attack detection using multiple Machine Learning (ML) algorithms. The declaration, acknowledgment, and abstract sections provide introductory elements. Following these, the table of contents outlines the sequential structure of the report, offering a comprehensive overview of its contents.

Chapter 1, the introduction, initiates the report by providing a general overview of the research topic, followed by delineating the project background, statements, objectives, and scopes. The organization of the report is elucidated towards the end of this chapter, outlining the subsequent chapters' content.

Chapter 2 delves into the literature review, presenting a detailed examination of Wireless Sensor Network attacks, feature extraction techniques, and various ML algorithms such as Support Vector Machines (SVMs), K-Nearest Neighbors (KNN), and Artificial Neural Networks (ANN). The chapter further reviews DDoS attack detection in WSN over the past five years, culminating in a summary.

Chapter 3 outlines the methodology employed in the project, including project methodology, software and hardware requirements, a flowchart of activities, and project milestones. Chapter 4 elaborates on the experimental design, detailing the experiment design plan, data collection, preparation, and preprocessing, as well as the training and testing of data. The use of multiple ML algorithms for WSN DDoS attack detection is discussed, along with performance evaluation metrics and an interface design.

Chapter 5 focuses on the implementation of the experiments, covering the process from feature extraction to the testing and training of the dataset. Chapter 6 presents preliminary results, including outcomes of feature extraction, ML algorithm experiments, and specific results for SVM, KNN, and ANN. A comprehensive discussion of the findings is included. A interface design of a WSN DDoS attack prediction dashboard is generated.

Chapter 7 provides the conclusion, summarizing the key aspects of the research. The discussions on the results and findings are further expanded. The report concludes with a reference section and an appendix, providing additional details and supporting materials. This organized structure facilitates a systematic presentation of the research process, methodology, experimentation, and analysis, offering a thorough exploration of WSN DDoS attack detection using ML algorithms.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The purpose of the literature review is to study past research which is almost similar and related to the proposed research. The literature review is to know the limitations and to study how to improve the performance of past research. Studies on wireless sensor network attack and machine learning have been done to have a better understanding for future work. Most of the Wireless Sensor Network attack detection is conducted by using machine learning algorithms and there is a need to understand the consequences.

2.2 Wireless Sensor Network Attack

An against act designed to undermine or disrupt the operation of a wireless sensor network is referred to as a wireless sensor network (WSN) attack. WSNs are networks of multiple tiny, self-contained sensor nodes that exchange data wirelessly for a variety of uses, including surveillance, automation of industrial processes, and environmental monitoring.

According to Chen et al., 2015, a variety of applications for wireless sensor networks based on IPv6 are being developed as a result of the rising popularity of IPv6 technology and wireless sensor networks. Nevertheless, IPv6-based wireless sensor networks are susceptible to wormhole attack, a destructive attack. Two malicious nodes work together to create a low-latency, high-quality, out-of-band tunnel between them in the wormhole attack. These two nodes are the only ones with access to the tunnel, which is also known as a "wormhole link." The packets sent by the surrounding nodes get detected by one node (the attacker), who subsequently relays them to the other malicious node (the collaborating node) for replay. Nodes near the attacker would wrongly believe that packets transmitted via the wormhole link require fewer hops and travel less distance because the collaborating node can send packets

to the sink with fewer hops. As a result, packets would prefer to be transmitted over the wormhole link. Because nodes won't be able to find the typical paths, this may have a negative effect on the routing algorithm.

In addition, Kaur et al., 2016, notes that with the development of this technology, security is currently one of the main issues. WSNs are susceptible to numerous security risks known as Attacks. These can generally be divided into two categories: passive attacks and active attacks.

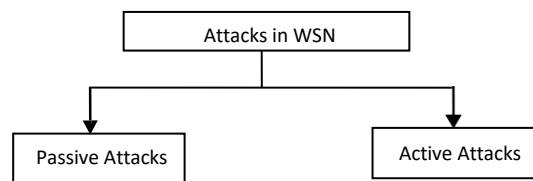


Figure 2.1: Attacks in WSN

In passive attacks, the intruder will listen to the channel or watch the traffic but won't interfere with the sender and receiver's conversation. Thus, passive attacks are challenging to identify. It can also be divided into Data Disclosure and Traffic Analysis in Data Disclosure; an intrusive party may endanger information privacy. When conducting traffic analysis, an attacker may examine the traffic and look for additional factors if the actual information is in an unreadable format.

An attacker can change the information or stop communications between the sender and the receiver during active attacks. Attacks like this pose a greater threat than passive ones. One kind of active attack is denial of service. In this study, Kaur et al., 2016, give DOS attacks even more attention. Disruption of Service Attacks that target routing schemes are carried out by attackers and happen as a result of both think and unintentional node failure. In a DoS attack, the victim node's resources are depleted by receiving needless packets that are not authorised for or intended for that node. DoS attacks target not only hostile nodes but also situations that render the network's ability to deliver services useless. DoS attacks can take many different forms in Wireless Sensor Networks (WSNs), with jamming and manipulation at the physical layer, exhaustion, collision, and unfairness at the link layer, and black holes, misdirection, etc. at the network layer. DoS attacks happen at the transport layer as a result of desynchronization and malicious flooding.

Protocol Layer	Attack	Description
Physical Layer	Jamming	Attacker generates jamming signals in communication area, genuine nodes unable to communicate.
MAC layer	Denial of Sleep Attack	In Denial of Sleep attacks, attacker stops nodes from going into sleep mode
Network Layer	Flooding Attack	In these attacks, attacker exhausts the resources of network by sending unnecessary packets, as a result network lifetime of network decreases.
	Hello Flood Attack	In this attack, high range radio transmission range is used by the attacker to send HELLO packets to large area. Hence all other nodes send information through intruder node to reach base station.
	Black Hole Attack	In this attack, for every route request message, attacker replies with shortest and fresh route to the destination. As a result target node chooses path through attacking node, then attacking node can drop the packets.
	Worm Hole Attack	In these attacks, at least two nodes present in the network, having high speed link in between. Attacker at one side relays packets to other side through that link and convinces other nodes that they are very close to each other and have shortest path to the base station. Hence all other nodes include malicious nodes in their path, so malicious nodes can disrupt the traffic.
Transport Layer	Desynchronization Attack	In this attack, attacker will modify the sequence number of packets to disrupt a session between sender and receiver.
Application Layer	Path Based DoS	Attacker injects bogus or replayed packets once a path is established, to waste the network bandwidth.

Figure 2.2: Classification of DDoS attack

Other than that, Wireless Sensor Networks (WSN) are the most well-liked technical advancements, according to Keerthika et al., 2022, and are already a leading technology used for mission carried out in remote places as well as for future applications. Since WSN are broadcast, all users can access the wireless air interface, creating several special security challenges for researchers. A network assault known as a distributed denial of service (DDoS) involves dispersing a victim's many workstations across various independent networks. By effectively forcing the target system to shut down due to the constant stream of incoming messages, authorised users are prevented from accessing the system. A DDoS assault poses a serious security risk to the communications networks of today. A DoS attack is intended to take down any service. A DDoS attack occurs when several connected devices cooperate to complete the same task. A DDOS attack has four components:

- Attacker.
- Compromised nodes (Running a specific programme (handler) with the capacity to control multiple agents.)
- Agent nodes (Run a programme responsible for sending data streams to the victim node.)

- Victim.

DDoS assaults against the Internet Relay Chat Architecture and the Agent Handler framework. Customers, handlers, and agents make up the Agent-Handler structure. By discovering vulnerable hosts and leveraging them to generate the required number of site visitors, the attacker first creates a community of computer systems. At the cooperating hosts of the attack community, the attacker installs attack gear. An IRC channel is used in the IRC-kind arrangement to connect the clients with the agents. Instructions can be sent to the agents using IRC ports. This structure's main benefit is that an attacker may hide their presence. One illustration is the Low Orbit Ion Cannon (LOIC).

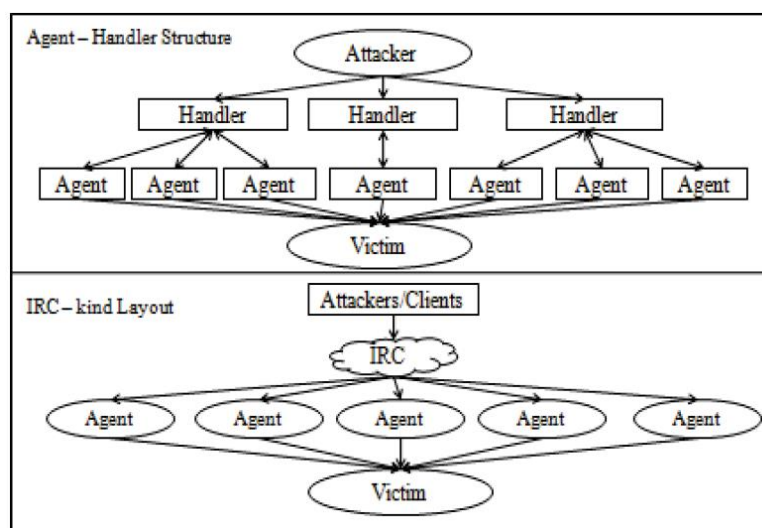


Figure 2.3: DDoS attacks on Two Architectures

In conclusion, there are a variety of threats that can damage wireless sensor networks' (WSNs') functionality and interfere with communication. DoS assaults, such as DDoS attacks, which involve numerous compromised nodes in various networks flooding a victim system with traffic and making it inaccessible to authorized users, are a serious threat to WSNs. DDoS assaults are harder to defend against because of their spread nature. To maintain the dependability and availability of wireless sensor networks for various applications, researchers and practitioners must concentrate on building strong techniques to detect, prevent, and mitigate these assaults. Security is a major concern in WSNs.

2.3 Feature Extraction

Feature extraction is a crucial step in machine learning with several key objectives. Firstly, it aims to simplify models by reducing the number of parameters, thereby enhancing their interpretability, and easing the comprehension of underlying patterns. Secondly, it contributes to decreased training times, making the learning process more efficient and practical, especially for large datasets. Additionally, feature selection plays a pivotal role in reducing overfitting and promoting better generalisation to new, unseen data. Lastly, it helps address the challenge of the curse of dimensionality by carefully choosing relevant features, mitigating the risk of unnecessary complexity and noise in the modelling process. In essence, feature selection optimises the model's performance, making it more effective, interpretable, and scalable. (Chen et al., 2020)

Random forests are one of the most popular machine-learning algorithms. They are so successful because they provide good predictive performance, low overfitting, and easy interpretability. Random Forest is an example of an embedded method for feature extraction. In the context of Random Forest, feature extraction happens naturally during the training process. As the Random Forest algorithm builds multiple decision trees using different subsets of features, it assesses the importance of each feature based on its contribution to the model's overall performance. The feature importance scores generated by Random Forest can then be used to rank and select features, effectively performing feature extraction. This embedded approach makes Random Forest particularly useful for tasks where identifying the most relevant features is essential for model interpretability and performance. (Huljanah et al., 2019)

2.4 Machine Learning Algorithm

Artificial intelligence relies on machine learning techniques, which give computers the ability to learn from data and enhance their performance. By building models from training data, these algorithms can forecast the future and make judgements without having to explicitly programme them. In many areas where conventional algorithms might not be feasible, including medical, email filtering, speech recognition, agriculture, and computer vision, machine learning has uses.

Various machine learning algorithms exist, each with its specific purpose and learning style. Some popular machine learning algorithms include:

2.4.1 Support Vector Machines (SVM)

SVM is a powerful algorithm used for classification and regression tasks. It has applications in handwriting recognition and image classification. The algorithm operates by locating the optimal hyperplane that divides the data into two classes based on the input characteristics. The SVM classifier is not susceptible to overfitting issues, making it more effective and reliable. The illustration below depicts the support vector machine algorithm that have been used in various research.

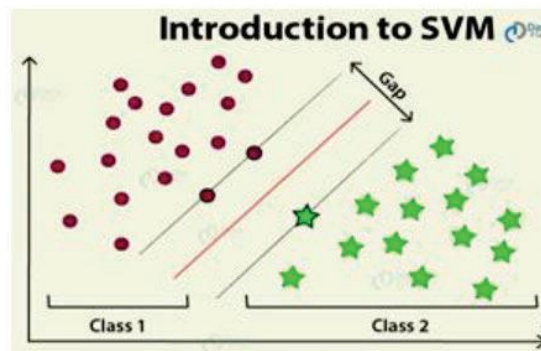


Figure 2.4: Support Vector Machine

2.4.2 k-Nearest Neighbors (k-NN)

k-NN is a simple and effective algorithm used for classification and regression tasks. It is commonly used in recommendation systems and pattern recognition. Based on the input attributes, the method discovers the K-nearest neighbours to a new data point and classifies the data point based on the majority vote of the K-nearest neighbours. Based on their proximity, KNN predicts the aggregation of individual data points. The diagram below summarises the KNN classification.

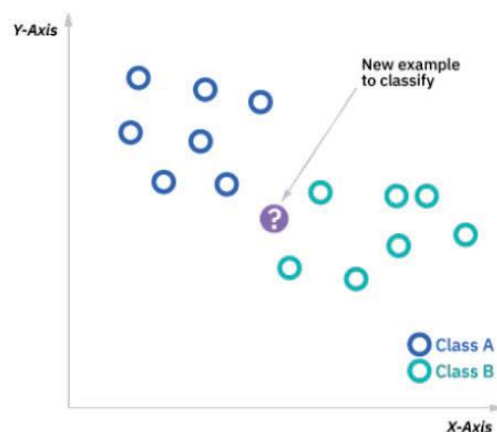


Figure 2.5: K-Nearest Neighbour (KNN)

Machine learning algorithms have become the foundation for various applications and services, enabling computers to learn from data and make accurate predictions or discoveries. Overall, machine learning continues to be a vital field within computer science and artificial intelligence, propelling advancements in various industries and reshaping the way we interact with technology and data-driven decision-making.

2.4.3 Artificial Neural Network (ANN)

A computer model known as ANN is influenced by the design and operation of the human brain. Artificial neural networks, which have the ability to learn and carry out tasks including pattern recognition, classification, regression, and decision-making, are frequently employed in the field of machine learning. Artificial neurons or perceptrons, which are interconnected nodes arranged in layers, make up an ANN. Input, hidden, and output layers are the three primary types of layers in an ANN. The input layer provides the initial data, and the output layer generates the finished product or prediction. In processing and altering the input data, the hidden layers, which are situated between the input and output layers, are extremely important.

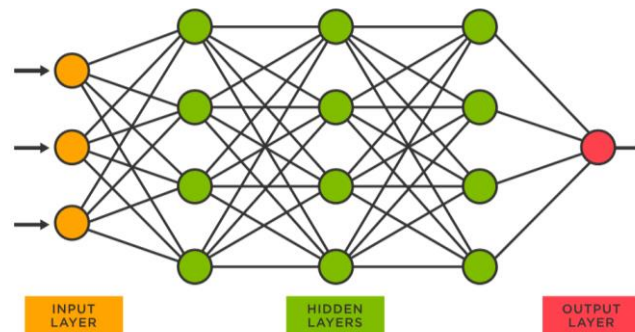


Figure 2.6: Artificial Neural Network (ANN)

2.5 Wireless Sensor Networks DDoS Attack Detection in past 5 years

Due to the resource constraints and sensor node limitations in wireless sensor networks (WSNs), detecting distributed denial-of-service (DDoS) assaults is a difficult operation. To detect WSN DDoS, there are a number of strategies that can be used.

In Wireless Sensor Networks (WSNs), it is important to recognize Distributed Denial-of-Service (DDoS) assaults. The authors of the study by Kumavat et al., 2023, compare the system's performance and working metrics before and after a Distributed Denial of Service Attack (DDoS) is discovered in the system. Using a network simulator in which they treat 10%

of nodes as attacker nodes, the authors created an IoT-enabled wireless sensor network. The authors describe a system that is secure from intrusion. Additionally, the authors use a network simulator to evaluate various performance parameters to characterize how a system performs once an attack is discovered. The final results include an evaluation of performance metrics including average PDR (packet delivery ratio), throughput, average used energy, average delay, and average overhead. According to the study's findings, a DDoS assault causes the performance of the IoT-enabled WSN system to suffer. The study demonstrates that while threat detection decreased average throughput and packet delivery ratio, the attack increased average latency, energy consumption, and overhead. The findings show that the system is unable to deliver the anticipated services or outcomes when under attack. The paper emphasizes the significance of DDoS attack detection and mitigation in IoT-enabled WSN systems to guarantee their appropriate operation and performance. a difficult task because of the limiting capabilities and resources of sensor nodes. To detect WSN DDoS, there are a number of strategies that can be used.

Parameters	Normal-IoT	Threats-IoT
Avg. Throughput (Kbps)	61.075	49.52
PDR (%)	88.16	70.09
Avg. Delay (Seconds)	0.9277	1.1294
Avg. Energy Consumption (Nj)	1571	1796
Overhead	8.21	15.18

Figure 2.7: Result Analysis for Normal IoT System and IoT Threat System

Another work by Santhosh et al., 2022, suggests a mechanism for identifying and thwarting DDoS attacks known as UDP reflection amplification attacks. The article analyzes potential dangers to wireless sensor networks, such as DDoS assaults, and suggests using the Cumulative Sum Algorithm as a novel technique for identifying and thwarting such assaults. In the study's introduction, various dangers to WSNs are covered, including DDoS attacks and the shortcomings of existing methods for detecting them. The proposed method for identifying and avoiding UDP reflection amplification assaults is then thoroughly explained in the paper, along with the procedures for putting the Cumulative Sum Algorithm into practice. A section on experimental data, which summarizes the results of simulations used to gauge the viability of the suggested approach, is also included in the paper. The discussion of the results, which

show that the proposed method achieves a high detection rate while retaining a low false positive rate and false negative rate, and the potential implications of the suggested method for enhancing the security of wireless sensor networks round up the paper.

In addition, according to Chaitanya et al., 2023, one of the major issues that WSNs are currently facing is security. However, using sensor nodes (SNs) from an unattended platform makes networks susceptible to a variety of potential attacks, and the inherent power and memory constraints of SNs make conventional security solutions impractical. The Spotted Hyena Optimizer with Quantum Neural Network for DDoS Attack Classification (SHOQNN-AC) approach for WSN is developed in this paper. The SHOQNN-AC technique's primary goal is to correctly identify DDoS attacks in the WSN. The SHOQNN-AC method uses a min-max scaler to perform data scaling in order to achieve this. The SHOQNN-AC technique makes use of a QNN classification model for DDoS attack detection, which is highly effective in identifying DDoS attacks in the network. The SHO algorithm is used for the parameter selection process in the SHOQNN-AC technique to increase the attack detection effectiveness. The benchmark WSN-DS dataset is used to test the SHOQNN-AC technique's performance validation.

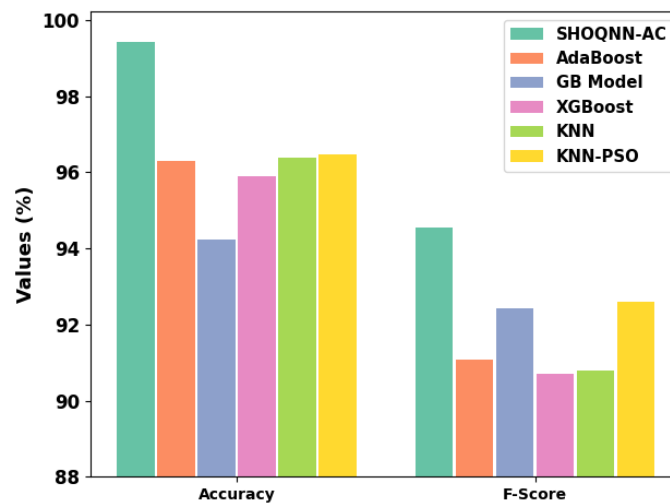


Figure 2.8: Accuracy and F-Score analysis of SHOQNN-AC approach with other IDS methods

According to the accuracy graph above, the SHOQNN-AC technique has an increase of 99.43% whereas the Adaboost, GB, XGB, KNN, and KNN-PSO models have decreases of 96.30%, 94.23%, 95.91%, 96.40%, and 96.47%, respectively. While the Adaboost, GB, XGB, KNN, and KNN-PSO strategies achieve decreasing of 91.09%, 92.43%, 90.70%, 90.79%, and 92.59% respectively, the SHOQNN-AC technique sees an increase of 94.54% based on F-Score. The results of the experiments show how important the SHOQNN-AC algorithm is in comparison to other models.

Utilize the sources of pandemic modeling to IoT networks made up of WSNs, according to the research of Aysa et al., 2020. We create a suggested framework to identify unusual defense activity. IoT-specific characteristics, such as inadequate processing capacity, power restrictions, and node density, have a substantial impact on the development of a botnet. For the two active well-known attacks, Mirai and Bashlite, we use standard datasets. To identify unusual activity like DDOS features, we also used a variety of machine learning and data mining methods, including SVM, Neural Network, Decision Tree, and Random Forest. In the experimental results, we discovered that decision-tree (J-48), Neural BP network, and Random Forest produced a higher True Positive Rate (TPR) when compared to LSVM. These classifiers were applied to three different datasets using four different classifiers. Additionally, we discovered that, when using the same iterations and threshold as Neural BP, Decision-tree (J-48) and Random Forest took the least amount of time. We also discovered that the combination of decision tree and random forest produced a high degree of accuracy in identifying attacks.

Device Id	Classifier	Time seconds	Threshold	Iteration	TPR %	FPR %	Precision %	Recall %	F-measure %	Error (MAE) %
1	LSVM	280	0.003	300	89.7	15	91.2	89.7	88.5	6.8
	NN	393	0.003	300	98.4	0.7	99.4	99.4	99.4	0.83
	J-48	96	0.003	300	99.7	0.3	99.7	99.7	99.7	0.31
	RF	113	0.003	300	99.7	0.3	99.7	99.7	99.7	0.37
2	SVM	112	0.005	300	89.2	11.6	90.2	89.2	87.9	7.9
	NN	224	0.005	300	99.3	0.6	98.3	98.3	98.3	0.47
	J-48	80	0.005	300	99.5	0.4	99.5	99.5	99.5	0.34
	RF	106	0.005	300	99.7	0.3	99.7	99.7	99.7	0.42
3	SVM	98	0.0008	300	99.4	0.4	99.5	99.4	99.4	0.37
	NN	186	0.0008	300	99.6	0.3	99.6	99.6	99.6	0.42
	J-48	30	0.0008	300	99.5	0.4	99.6	99.5	99.5	0.37
	RF	86	0.0008	300	99.7	0.3	99.7	99.7	99.7	0.38

Figure 2.9: Comparison between Different Classifier

According to Kumavat et al., 2023, the whole corporate and academic world has gone digital during the course of the Covid-19 pandemic. Digitalization has directly led to an increase in the frequency of attacks and breaches on Internet-based systems. A service or application that depends on the infrastructure of the Internet can be easily brought down by the new and deadly distributed denial of service (DDoS) and distributed reflective denial of service (DRDoS) cyberattacks. Cybercriminals constantly hone their assault strategies while avoiding discovery

by employing antiquated approaches. Since the amount of data being created and stored has increased significantly over the past several years, traditional detection techniques are not equipped to detect innovative DDoS attacks. The project focuses on creating frameworks and models for intrusion detection to recognize and stop these kinds of attacks. The paper also goes over a number of potential algorithms and methods for identifying and stopping DDoS and DRDoS assaults in wireless sensor networks. The Matching Pursuit algorithm, a signal processing method that can be used to find anomalies in network data, is one such approach. Cooperative IoT packet sampling is a different method that uses several IoT devices to sample network traffic and find anomalies. The study's suggested approach is a dynamic DDoS detection system that makes use of categorization algorithms under the supervision of a fuzzy logic system. This system seeks to dynamically choose from a collection of categorization algorithms that can recognize various DDoS assault types. Based on the characteristics of the incoming traffic and the existing network conditions, the fuzzy logic system aids in the formulation of wise decisions. This method enables the system to modify and adapt its detection techniques in order to efficiently recognize and counteract DDoS attacks in real-time.

DDoS attacks are one of the most dangerous risks in the current environment, according to Chandan et al., 2022. Since these attacks are growing more sophisticated and commonplace every day, it is difficult to recognize them and mount an effective defense. Therefore, it's critical to recognize and stop this kind of attack before it has an impact. This study examines a fresh dataset of DDoS attacks, including HTTP Flood and UDP Flood attacks. In this work, nodes are categorized as malicious or non-malicious nodes using machine learning methods such as SVM, KNN, and Random Forest. Additionally, fuzzy inference criteria are used to accurately categorize malicious nodes as very harmful, moderately malicious, or not malicious. These nodes are categorized as malicious by machine learning algorithms (SVM, KNN, and Random Forest). Finally, a choice is made to remove all the highly dangerous nodes utilizing the information that has been evaluated. Additionally, the researchers have produced a dataset called AVV-DDos2286 that will be made available to the public for additional research on DDoS attacks. In comparison to traditional DDoS attack detection systems constructed using stand-alone machine learning algorithms, the research's suggested method offers a better hybrid approach that combines machine learning algorithms with Fuzzy logic systems. The results have a classification accuracy of almost 94%.

According to Ageyev et al., 2021, monitoring network traffic is a crucial issue for the security and dependability of the network. Many techniques for finding traffic anomalies are based on the statistical model of traffic. As a result of missed attack moments, which allow an

attacker to introduce errors into the system's operation and render it unusable (for example, to carry out a DDOS attack), current modern methods of detecting attacks frequently prove to be insufficiently reliable. The study's major goal was to lessen the effects of IoT devices' limited computational capabilities on the deployment of methods for anomaly detection. In the paper, k nearest neighbors (KNN), support vector machine (SVM), decision tree (CART), random forest (RF), an adaptive boosting model over a decision tree (AdaBoost), logistic regression (LR), and Bayesian classifier (NB) were chosen for comparison.

Model	Accuracy	Precision	Recall	F1
KNN	0.973	0.941	0.593	0.969
SVM	0.716	0.696	0.037	0.612
CART	0.973	0.974	0.951	0.971
RF	0.969	0.976	0.945	0.968
AdaBoost	0.981	0.959	0.963	0.971
LR	0.953	0.938	0.923	0.964
NB	0.732	0.532	0.965	0.762

Figure 2.10: Result of Evaluation of Quality of Classifiers

The best results are demonstrated by the KNN, CART, RF, AdaBoost, LR models. Given the minimum execution time, applying a RF model to the task at hand is a reasonable choice.

G. Amudha et al., 2022, proposed a study concentrated on two kinds of MAC layer attack detection techniques. Neutral Network (NN) and Support Vector Machine (SVM) are two of the most sophisticated systems in machine learning. MLP is practised using the BP algorithm, and the probabilities of the associated Denial of Service (DoS) attack are calculated using the critical and normalised parameters of the test trials. The calculated values are utilised as training inputs for SVM and NN. In simulation results, the accuracy of the SVM result is more precise than that of the NN method. Both Support Vector Machine and Neural Network techniques are extremely useful for determining the likelihood of DoS attacks in Wireless Sensor Network.

The study of Vimal Kumar Stephen et al., 2021, uses neural network (NN) and support vector machine (SVM) machine learning techniques to detect denial-of-service (DoS) attacks on the MAC layer. After that, the efficacy of the two approaches is evaluated. It is crucial to secure the MAC layer because it grants sensor nodes access to wireless channels. The outcomes demonstrated that these algorithms are effective at securing and optimising sensor networks. NN outperforms SVM in terms of both accuracy and output time when detecting DoS attacks. Consequently, SVM has an accuracy rate of 97% while NN has an accuracy rate

of 91%. NN requires 0.75 seconds to calculate the likelihood of an attack, whereas SVM requires only 0.25 seconds. Instead of using neural networks to detect DoS attacks, SVM-based techniques are preferred.

Liu et al., 2022, proposed An Enhanced Intrusion Detection Model Based on Improved KNN in WSNs. Through the introduction of the k-Nearest Neighbour algorithm (kNN) in machine learning and the introduction of the arithmetic optimisation algorithm (AOA) in evolutionary calculation, the authors propose a WSN intelligent intrusion detection model that forms an edge intelligence framework that specifically performs intrusion detection when the WSN encounters a DoS attack. To improve the accuracy of the model, the researchers employ a parallel strategy to improve communication between populations and the Lévy flight strategy to fine-tune the optimisation. The proposed PL-AOA algorithm performs well in the benchmark function evaluation and effectively ensures the enhancement of the kNN classifier. Matlab2018b is also used to conduct simulation experiments using the WSN-DS data set, and our model obtains 99% ACC with a nearly 10% enhancement over the original kNN when detecting DoS intrusions. The experimental findings demonstrate that the proposed intrusion detection model has positive effects and practical significance.

K.Nirmala et al., 2021, proposed a paper focuses on a survey of machine learning-based attack detection, which is a crucial endeavour for network and data security. In this survey paper, the genetic k-means algorithm (GKA), optimum support vector machine (OSVM), K-nearest neighbour (KNN), and Intrusion detection model based on information gain ratio and online passive aggressive algorithm (ID- GOPA) are utilised. The researchers provide a comprehensive analysis of extant wireless sensor network (WSN) security protocols based on various machine learning methods. In the results section, performance measures are described in order to evaluate their benefits and drawbacks.

Table 2.1: Summary result of the Experiment of K.Nirmala et al., 2021.

S No.	Using Classification	Platform	Performance measures	Result
1	Genetic K-means	NS2.34	Detection rate, False positive rate	Highest detection rate, low false positive rate
2	KNN	Used three scenarios	Detection rate, Accuracy	Detection rate around 87%

3	ID-GOPA	NS-2	Accuracy, Precision, Recall, F1-Score	Whole accuracy around 96%
4	Optimal SVM (OSVM)	MATHLAB2014b	Accuracy, true positive rate, false negative rate	Accuracy 94.09%, TPR 95.53% and FNR 4.47

2.6 Summary of Wireless Sensor Networks DDoS Attack Detection in past 5 years

Table 2.2: Summary of Wireless Sensor Networks DDoS Attack Detection in past 5 years

Paper Reference	Dataset/ Techniques	Preprocessing Tasks	Machine Learning Algorithm Used /Performance Measures	Performance
Kumavat et al., 2023	Network Simulation for IoT-enabled WSN systems	Prepared a parameter for simulation (100, 200, 300, 400, 500 and 600 sensor nodes)	average PDR (packet delivery ratio), Throughput, average Consumed Energy, average Delay, and average Overhead	Average throughput and packet delivery ratio decreased due to threat detection. Average delay, energy consumption, and overhead increase due to the attack.
Santhosh Kumar et al., 2022	Contiki (memory-constrained networked computer operating system) and Cooja Simulator	Data generated by simulations and converted into small blocks of data	Cumulative Sum Algorithm and User Datagram Protocol (UDP)	High detection rate with a low false positive rate and false negative rate.
Chaitanya et al., 2023	A dataset for Intrusion Detection Systems in Wireless Sensor Networks	Comprising 374661 samples with five classes (Normal, Blackhole, Gray hole, Flooding and Scheduling Attacks)	SHOQNN-AC technique	SHOQNN-AC technique has the highest accuracy of 99.43% and highest F-Score of 94.54%.

Aysa et al., 2020	Three standard sets of normal data and abnormal or malicious data collected from three IoT devices	Data cleaning, data normalization, feature selection, feature extraction and data splitting into training and testing datasets.	LSVM, Neural Network, Decision tree and Random Forest	Decision-tree (J-48), Neural BP network and Random Forest achieved a highest True Positive Rate (TPR). Decision-tree (J-48), and Random Forest consumed minimum time.
Chandan et al., 2022	Dataset with simulation information on DDoS attacks like UDP Flood and HTTP Flood and named as AVVDDos2286.	Analyzing Network Parameters.	KNN, SVM, Random Forest and Fuzzy Interference Rule System to identify the nodes	94% of classification accuracy.
Ageyev et al., 2021	A network traffic flow creator that creates a network traffic features collection.	Eliminating unsuitable or damaged data which will have an effect on the accuracy of the dataset.	KNN, SVM, CART, RF, AdaBoost, LR and NB.	The best results are demonstrated by the KNN, CART, RF, AdaBoost, LR models.
G. Amudha et al., 2022	Dataset that included different	MLP is practiced utilizing BP algorithms.	Neutral Network (NN), Support	NN have accuracy of 92.33 and 0.74s of detection time while

	probabilities of attack.		Vector Machine (SVM)	SVM have 97.13 accuracy and 0.26s detection time.
Vimal Kumar Stephen et al., 2021	Wireless Network Simulator (Prowler)	BP method is used to train MLP.	Neutral Network (NN), Support Vector Machine (SVM)	NN have accuracy of 91.43 and 0.75s of detection time while SVM have 97.14 accuracy and 0.25s detection time.
Liu et al., 2022	WSN-DS dataset stimulated from network simulator 2 (NS-2)	Processed into a data set of 23 features.	KNN classifier and Arithmetic Optimization Algorithm (AOA)	KNN has an accuracy of 89 and AOA with a 99 accuracy.
K.Nirmala et al., 2021	KDDCup99 dataset	Pre-processing, Classification and Kernel Selection.	KNN, ID-GOPA and Optimal SVM (OSVM)	Accuracy of KNN is 87, ID-GOPA 96 and Optimal SVM (OSVM) is 94.09.

2.7 Summary

This chapter discussed the literature review of the project. There are many kinds of machine learning algorithms that have been used in wireless sensor network DDoS attack detection in the past 5 years. The pros, cons and performance of the machine learning algorithms has been investigated thoroughly. The literature review above helps in providing improvements for machine learning algorithms used in detection of the attacks for secure communication in wireless sensor networks. According to the findings in literature review, mostly of the study just compared on two kind of machine learning algorithms only and in this project wish to use SVM (Support Vector Machine), KNN (K-Nearest Neighbors) and ANN (Artificial Neural Network) three famous machine learning algorithms to be compared in terms of accuracy, precision, and time elapsed.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter will introduce the methodology used for the experiment of detection of the wireless sensor network attacks. The first part introduces the project methodology, and the flow of the project will be explained. Each part will be brief on what process happens in each phase in completing the project. The flowchart of the project is also included.

3.2 Project Methodology

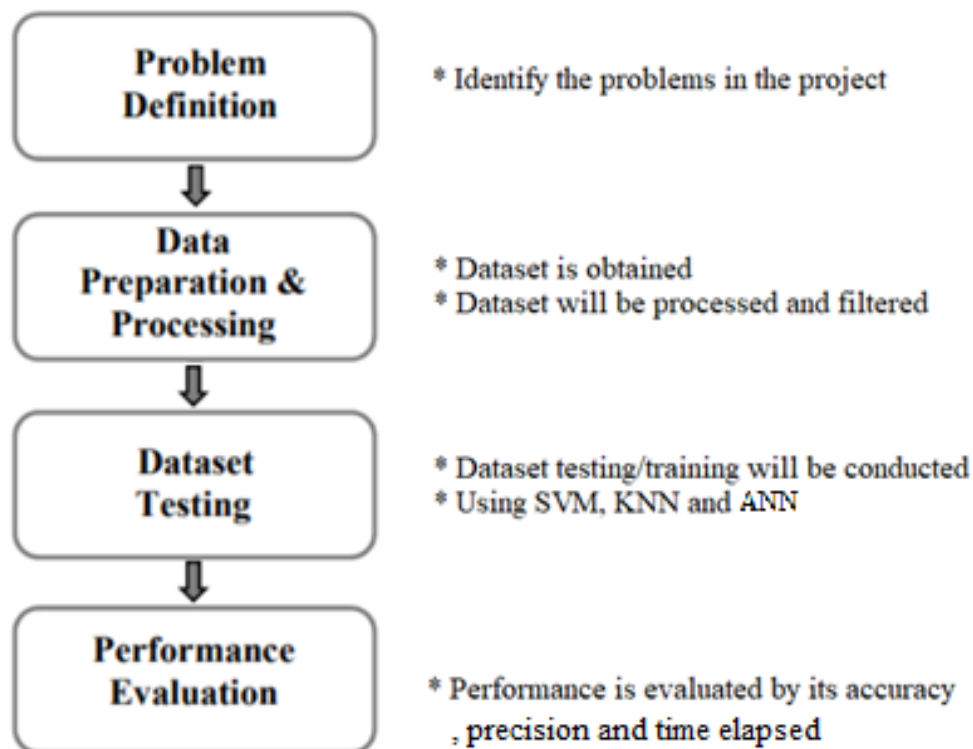


Figure 3.1: Flowchart of Project Methodology

i. Problem Definition

As shown in the figure 3.1, the first step is problem definition. The problem of this project is identified. Based on the previous research, mostly of researchers only do comparison between two machine learning algorithms only. For now, there still don't have comparison between those three popular algorithms in machine learning. In this project, comparison between the three algorithms will be done and the performance of the algorithms will be evaluated in terms of accuracy, precision and time elapsed.

ii. Data Preparation and Preprocessing

Next, for data preparation and preprocessing, data to be used in this research is text data. The research will use dataset obtained from dataset website "Kaggle" which named as "WSNBFSFDataset" (Wireless Sensor Network Blackhole Flooding Selective Forwarding). The dataset has around 16 features and 312 106 rows of raw text data set was obtained and prepared after the necessary pre-processing was performed. The dataset consists of 4 different traffics that is Blackhole, Flooding, Selective Forwarding attack traffic and Normal traffic which know as DDOS attack, and it is recorded as .csv format. The text data will be processed to filter out all the unstructured text and standardize all the text data. Before the training process, dataset will be an ongoing feature extraction method by using random forest classifier and cross validation to improve the quality and compatibility of the data. This involves applying the Random Forest Classifier to extract important features from the dataset while ensuring robustness through cross-validation, which helps evaluate the model's generalization capability. The resulting set of features is then used for subsequent model training and analysis.

iii. Dataset Testing

Following that, the dataset testing/training will take place, with the dataset being placed in a desirable algorithm in order to train and test the model. The identification of assaults in wireless sensor networks will be done using machine learning algorithms such as Support Vector Machine (SVM), K-Nearest Neighbour (KNN), and Artificial Neural Network (ANN). One of the best algorithms will be picked for future development.

iv. Performance Evaluation

Finally, evaluating a machine learning model's performance is a key step in determining the model's usefulness and capacity to detect and categories assaults effectively. The accuracy, precision, time elapsed and some of the measurement metrics of the approaches will be used to assess their performance. While accuracy is defined as the proportion of correctly classified instances in relation to the total number of instances in the dataset. It assesses the model's overall performance as well as its ability to appropriately recognize attacks in wireless sensor network.

3.3 Software and Hardware Requirements

Table 3.1: Software requirements and description

Software	Description
PYCHARM (PYTHON)	Coding Software
Window 11	Operating System
Microsoft 365 Word	Documentation Tools
Microsoft 365 Excel	Dataset Overview and Tools

3.4 Flowchart of Activities

The figure below shows the flowchart or flow work in this project. It shows the overall process form the start until the end of the project.

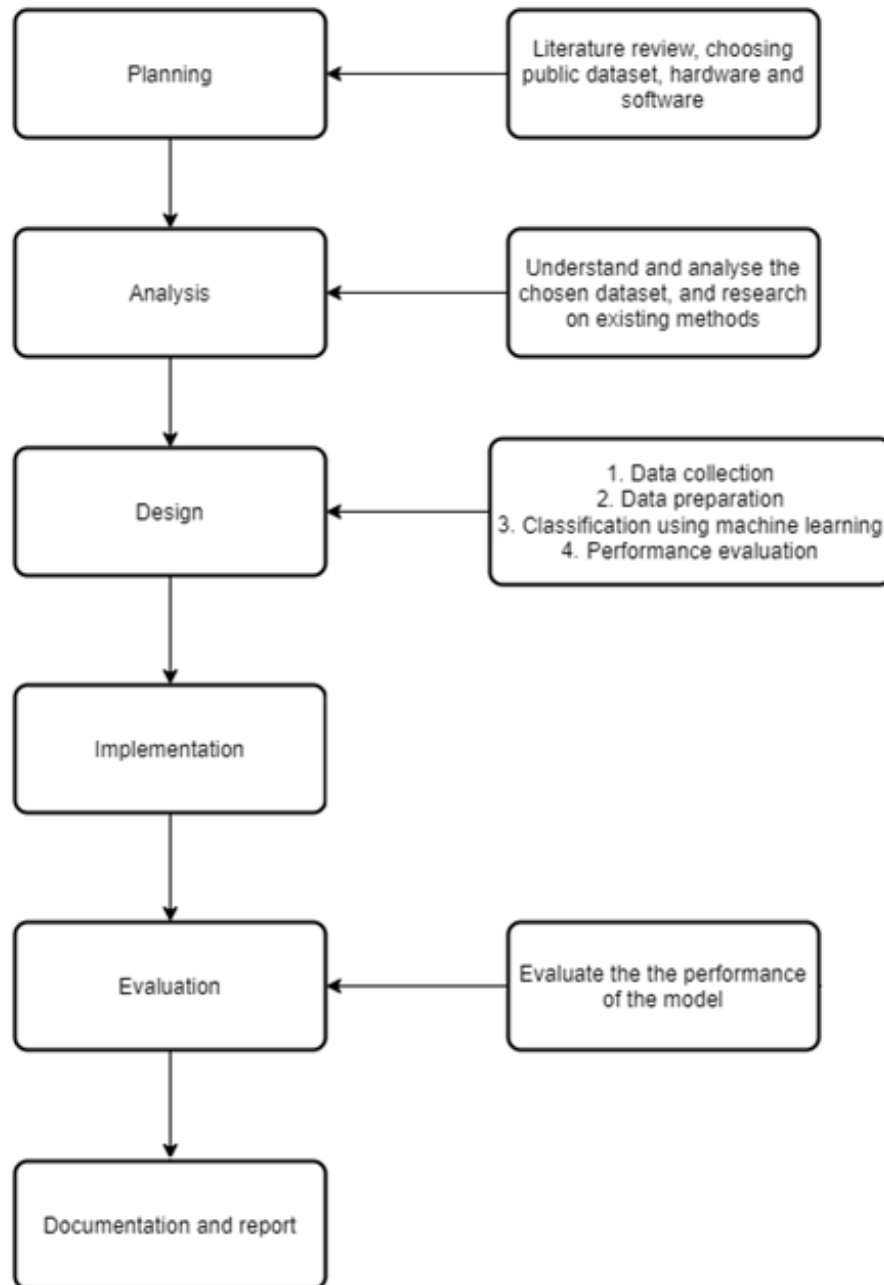


Figure 3.2: Flowchart of Activities

3.5 Project Milestones

Table 3.2: Project Milestones

Project Milestone	Description	Dates
Milestone 1	<ul style="list-style-type: none"> • Proposal Writing • FYP Title • Introduction • Literature Review • Methodology Idea • Project Objective, • Project Scope • Milestone Planning 	1/1 – 12/2
Milestone 2 (Research on recent work)	<ul style="list-style-type: none"> • Literature review on recent paperwork • Gain more idea on doing the FYP research 	14 /2 – 21 /2
Milestone 3 (Research on the methodology)	<ul style="list-style-type: none"> • Plan well the methodology of the project 	22/2 – 25-3
Milestone 4 (Dataset collection)	<ul style="list-style-type: none"> • Get all dataset solution 	26/3 – 20 /4
Milestone 5	<ul style="list-style-type: none"> • Presentation 1 	21/4 – 26/4
Milestone 6	<ul style="list-style-type: none"> • Pattern Discovery 1 	2/5 - 10/6
Milestone 7	<ul style="list-style-type: none"> • Presentation 2 	13/6 - 24/6
Milestone 8	<ul style="list-style-type: none"> • Pattern Discovery 2 	11/7 - 29/7
Milestone 9	<ul style="list-style-type: none"> • Model Evaluation 1 	1/8 - 12/8
Milestone 10	<ul style="list-style-type: none"> • Pattern Discovery 3 	15/8 - 2/9
Milestone 11	<ul style="list-style-type: none"> • Model Evaluation 2 	5/9 - 16/9
Milestone 12	<ul style="list-style-type: none"> • Pattern Discovery 4 	19/9 - 7/10
Milestone 13	<ul style="list-style-type: none"> • Model Evaluation 3 	10/10 - 21/10
Milestone 14	<ul style="list-style-type: none"> • Result Visualization 	24/10 - 4/11
Milestone 15	<ul style="list-style-type: none"> • Report Writing 	7/11 - 18/11
Milestone 16	<ul style="list-style-type: none"> • Report Conclusion Writing 	21/11 - 25/11

CHAPTER 4

EXPERIMENTAL DESIGN

4.1 Introduction

This chapter will introduce the experiment's design that will be conducted in the overall experiment process. The experiment is carried out to achieve the three objectives of this project as mentioned in Chapter 1.

4.2 Experiment Design Plan

There are few steps that will be conducted so that the experiment will achieve all three objectives of this project which is data collection, data preparation and preprocessing, data training and testing, detection using machine learning algorithms and performance evaluation. The design will be implemented by using Python programming language in the PyCharm Community Edition 2023. There are several libraries that will be used in this project such as sklearn, pandas and numpy.

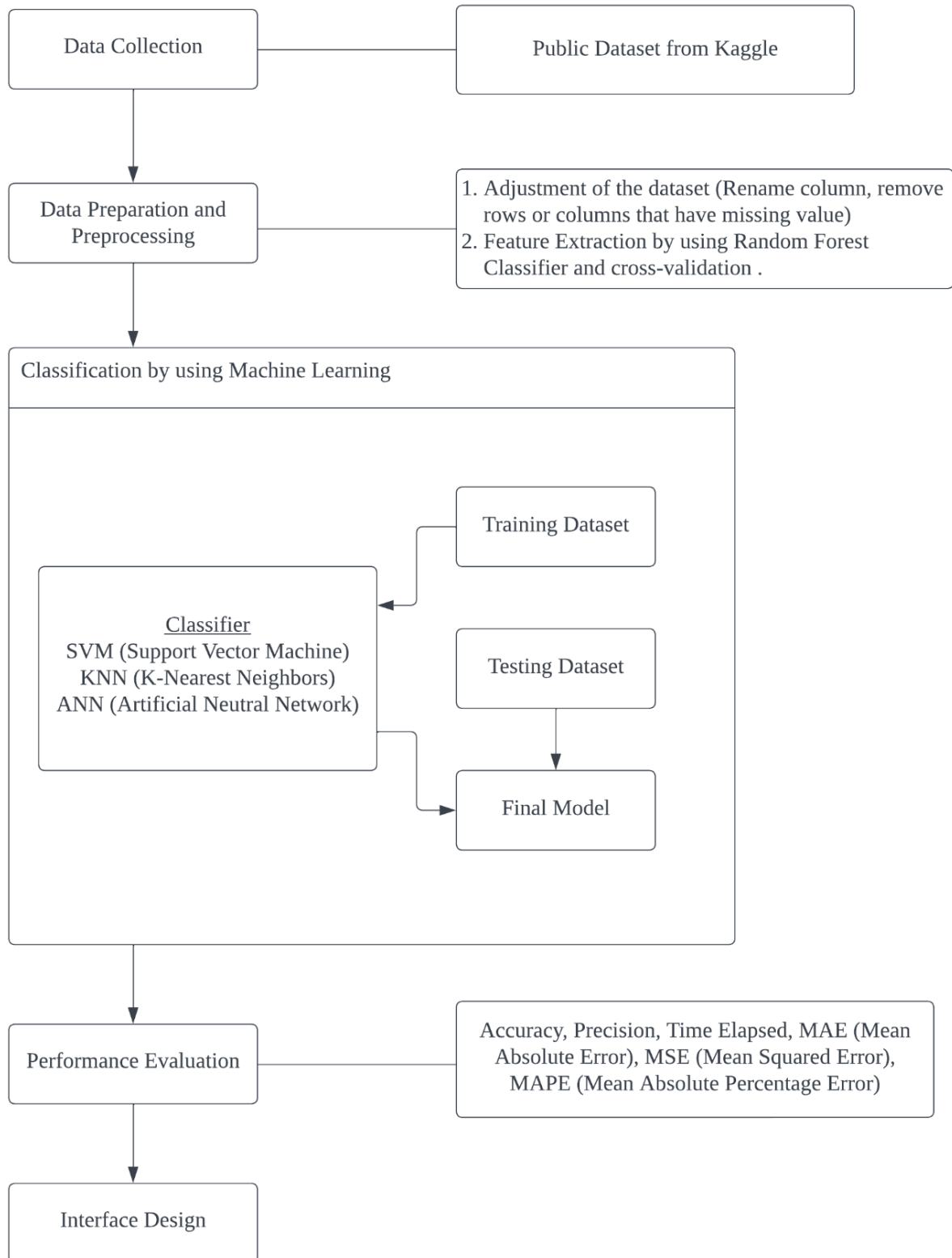


Figure 4.1 Overview of the Overall Design

4.3 Data Collection

In this project, to conduct a wireless sensor network DDoS attack, a public dataset will be used, which is obtained from Kaggle. As the wireless sensor network DDoS attack detection will be using machine learning algorithms, the dataset consists of 312107 rows and 18 features, *Event*, *Time*, *S_Node*, *Node_id*, *Rest_Energy*, *Trace_Level*, *Mac_Type_Pckt*, *Source_IP_Port*, *Des_IP_Port*, *Packet_Size*, *TTL*, *Hop_Count*, *Broadcast_ID*, *Dest_Node_Num*, *Dest_Seq_Num*, *Src_Node_ID*, *Src_Seq_Num* and *Class*. The dataset is downloaded after checking the content and suitability to be used for the project.

4.4 Data Preparation and Preprocessing

After that, the dataset will be conducted for data preparation and data preprocessing. First, data cleaning, where all the unnecessary columns are dropped, columns are renamed to more suitable names to avoid confusion and removing rows or columns that have missing values to avoid error during the experiments. All the dataset has been seen through to remove any rows that have missing values.

Before the training process, dataset will be an ongoing feature extraction method by using Random Forest Classifier and cross-validation to improve the quality and compatibility of the data. A Random Forest Classifier is instantiated with 100 trees (`n_estimators=100`) and fitted to the training data using the fit method. The `feature importance` for each feature of the dataset have been calculated. Feature importance analysis can be valuable for understanding the contribution of different variables to the model's predictions and potentially for guiding feature selection decisions. After that, feature selection will be done using a range of threshold values for feature importance scores obtained from a Random Forest Classifier. It iterates through each threshold, selects features with importance scores above it, and trains Random Forest models for evaluation with cross-validation. After iterating through all thresholds, the script identifies the best-performing threshold based on the highest accuracy score and selects features accordingly. This process helps find an optimal balance between the number of selected features and model accuracy for further analysis and training.

Then, the preprocessing steps in the experiment are loading the dataset, the dataset is loaded from a CSV file using the `read_csv()` function. Besides extracting features and the target variable, the features and target variable is assigned by selecting the column from the dataset. Moreover, applying label encoding to the target variable. An instance of the

`LabelEncoder` class is created to convert the categorical target variable into numerical labels. Lastly, applying label encoding to categorical features. If there are additional categorical features in the dataset, the code loops through each feature specified in `categorical_features` and applies label encoding to the corresponding column in the feature matrix.

These preprocessing steps aim to prepare the data for subsequent tasks, such as splitting the dataset into training and testing sets, feature extraction, training machine learning algorithm classifier, and evaluating the model's performance. The label encoding is particularly useful for handling categorical variables in the target variable and potentially categorical features, allowing them to be processed by machine learning algorithms that expect numerical inputs.

4.5 Training and Testing Data

The data is then divided into a training set and test set through the process of data splitting by using `train_test_split` function. Data splitting is often done to avoid overfitting. The train set is usually used for training the model, to fully design the optimized model. The test set is used for testing or basically for final use after finish designing the model. The dataset is split into two, 80% for the training data set and 20% left for the testing data set.

4.6 Wireless Sensor Network DDoS attack detection using Multiple Machine Learning algorithms (ML)

Classification for detecting wireless sensor network DDoS attack by using machine learning algorithms that have been chosen, which is SVM (Support Vector Machine), KNN (K-Nearest Neighbors) and ANN (Artificial Neural Network). The algorithms are imported from sklearn libraries. Each classifier will have its own classification report to check the performance of the classifier. The classification will utilize the training dataset and testing dataset.

4.7 Performance Evaluation

After training the model, the evaluation will be done by using the testing data set. The classification performance report will have precision, accuracy, time elapsed and some regression metrics which will be used to evaluate the performance of each classifier of the machine learning algorithms.

4.7.1 Accuracy

A level of accuracy indicates how closely a result or prediction resembles the actual or anticipated value. It's frequently stated as a percentage. Accuracy is defined as the proportion of accurate predictions to all other predictions in the context of machine learning and, if classification tasks. For instance a classifier predicts 90 correctly out of 100 times, its accuracy is 90%.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

*Where TP=True Positive, TN=True Negative, FP=False Positive, FN=False Negative

4.7.2 Precision

The number of relevant instances that are chosen is a measure of precision. It focuses on the classifier's positive predictions and counts the proportion of those that are accurate. The ratio of true positives (instances of positive outcomes that were correctly anticipated) to the total of true positives and false positives (occurrences of positive outcomes that were not correctly predicted) is used to measure precision. Precision is especially helpful when the cost of false positives is significant since it helps evaluate the quality of positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

*Where TP=True Positive, FP=False Positive

4.7.3 Time Elapsed

The term "time elapsed" describes how long it took to finish a particular operation or process. It counts the amount of time that has elapsed between the start and finish of an operation. The amount of time that has passed can be expressed in seconds, minutes, hours, or any other pertinent time unit.

4.7.4 Regression Metrics

Regression metrics is to evaluate the performance of model that predict continuous values. This metrics help to qualify how well a regression model is performing and ability to capture the underlying patterns in data.

i) Mean Absolute Error (MAE)

MAE is a simple and intuitive metric that measures the average absolute differences between predicted and actual values. It gives equal weight to all errors, regardless of their magnitude or direction. A smaller MAE indicates better model performance.

$$\text{Mean Absolute Error (MAE)} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

*Where n=total number of samples, y_i =actual value, \hat{y}_i is the predicted value

ii) Mean Squared Error (MSE)

MSE is another common metric that measures the average of the squared differences between predicted and actual values. Squaring the differences penalizes larger errors more heavily than smaller ones. MSE tends to magnify the impact of outliers due to the squaring operation.

$$\text{Mean Squared Error (MSE)} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

*Where n=total number of samples, y_i =actual value, \hat{y}_i is the predicted value

iii) Mean Absolute Percentage Error (MAPE)

MAPE is a percentage-based metric that measures the average percentage difference between predicted and actual values. It is commonly used where expressing errors as a percentage of the actual values is meaningful. MAPE provides a relative measure of error, making it easier to interpret across different scales.

$$\text{Mean Percentage Error (MAPE)} = \frac{1}{n} \sum_{i=1}^n \left(\frac{|y_i - \hat{y}_i|}{|y_i|} \right) \times 100$$

*Where n=total number of samples, y_i =actual value, \hat{y}_i is the predicted value

4.8 Interface Design

Lastly, to achieve the last objective, a simple interface design will be created by using the same platform, Python on PyCharm. The model performance will be seen whether it successfully detects the wireless sensor network DDoS attacks or not.

4.9 Summary

Chapter 4 outlines the experimental design for the project aimed at detecting wireless sensor network Distributed Denial of Service (DDoS) attacks using machine learning algorithms. The chapter begins with an introduction and then presents a detailed plan for the experiment's design. The experiment involves several steps, including data collection, data preparation and preprocessing, training and testing data, utilizing multiple machine learning algorithms for detection, and performance evaluation.

For data collection, a public dataset from Kaggle with 312107 rows and 18 features is used. The features include various parameters related to the wireless sensor network DDoS attack. The dataset undergoes data cleaning and preprocessing, involving steps such as dropping unnecessary columns, handling missing values, and applying label encoding to categorical variables.

The training and testing data are then split, with 80% used for training and the remaining 20% for testing. Three machine learning algorithms, namely SVM, KNN, and ANN, are employed for wireless sensor network DDoS attack detection. The performance of each classifier is evaluated using classification reports that include precision, accuracy, time elapsed, and regression metrics.

The performance evaluation metrics include accuracy, precision, time elapsed, and regression metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE). These metrics provide a comprehensive assessment of the classifiers' effectiveness in detecting DDoS attacks.

Finally, an interface design using Python on PyCharm is proposed to achieve the project's last objective. The interface aims to showcase the model's performance in detecting wireless sensor network DDoS attacks. Overall, Chapter 4 provides a thorough overview of the experimental design and methodology for the project.

CHAPTER 5

IMPLEMENTATION

5.1 Introduction

This chapter will introduce the experiments that have been implemented so far including data cleaning, preparation and preprocessing and others. All the experiments are implemented by using Python in PyCharm.

5.2 Experiment Implementation

Dataset WSN 1.csv - Excel

Search

TEE YEW CHUN

Share

FileHomeInsertPage LayoutFormulasDataReviewViewAdd-insHelpAcrobat

PasteCutCopyFormat PainterClipboard

Calibri11AaB I U Font

Wrap TextMerge & CenterAlignment

General\$ % & Number

Conditional FormattingFormat as TableCell StylesStyles

InsertDeleteFormatCells

AutoSumFillClearEditing

Sort & FilterFind & Select

POSSIBLE DATA LOSS

Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

Don't show againSave As.

17

fx

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Event	Time	S_Node	Node_id	Rest_Energy	Trace_Level	Mac_Type	P_Source_IP	Des_IP	Port	Packet_Size	TTL	Hop_Count	Broadcast	Dest_Node	Dest_Seq	Src_Node	Src_Seq	N_Class			
2	1	0.1	79	79	600	5	0	79.255	1.255	48	30	1	1	100	0	79	4	normal				
3	2	0.100963	78	78	599.979723	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
4	2	0.100964	76	76	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
5	2	0.100964	75	75	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
6	2	0.100964	118	118	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
7	2	0.100964	117	117	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
8	2	0.100964	116	116	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
9	2	0.100964	74	74	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	Blackhole				
10	2	0.100988	77	77	599.979723	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
11	2	0.100989	39	39	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
12	2	0.100989	119	119	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
13	2	0.100989	38	38	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
14	2	0.100989	37	37	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
15	2	0.100989	36	36	599.979722	5	800	79.255	1.255	48	30	1	1	100	0	79	4	normal				
16	2	0.101383	1	1	599.979639	5	800	0.255	1.255	48	30	1	1	121	0	0	4	normal				
17	2	0.101383	3	3	599.979639	5	800	0.255	1.255	48	30	1	1	121	0	0	4	normal				
18	2	0.101384	40	40	599.979638	5	800	0.255	1.255	48	30	1	1	121	0	0	4	normal				
19	2	0.101384	42	42	599.979638	5	800	0.255	1.255	48	30	1	1	121	0	0	4	normal				
20	2	0.101384	43	43	599.979638	5	800	0.255	1.255	48	30	1	1	121	0	0	4	Forwarding				
21	2	0.101408	2	2	599.979639	5	800	0.255	1.255	48	30	1	1	121	0	0	4	normal				
22	2	0.101409	4	4	599.979638	5	800	0.255	1.255	48	30	1	1	121	0	0	4	normal				
23	2	0.101409	41	41	599.979638	5	800	0.255	1.255	48	30	1	1	121	0	0	4	normal				
24	2	0.101409	5	5	599.979638	5	800	0.255	1.255	48	30	1	1	121	0	0	4	normal				
25	1	0.101417	116	116	599.979722	5	800	116.255	1.255	48	29	2	1	100	0	79	4	normal				
26	1	0.101869	78	78	599.979326	5	800	78.255	1.255	48	29	2	1	100	0	79	4	normal				

Dataset WSN 1

ReadyAccessibility: Unavailable

26°C Partly cloudy09:17 PM05-Jul-23

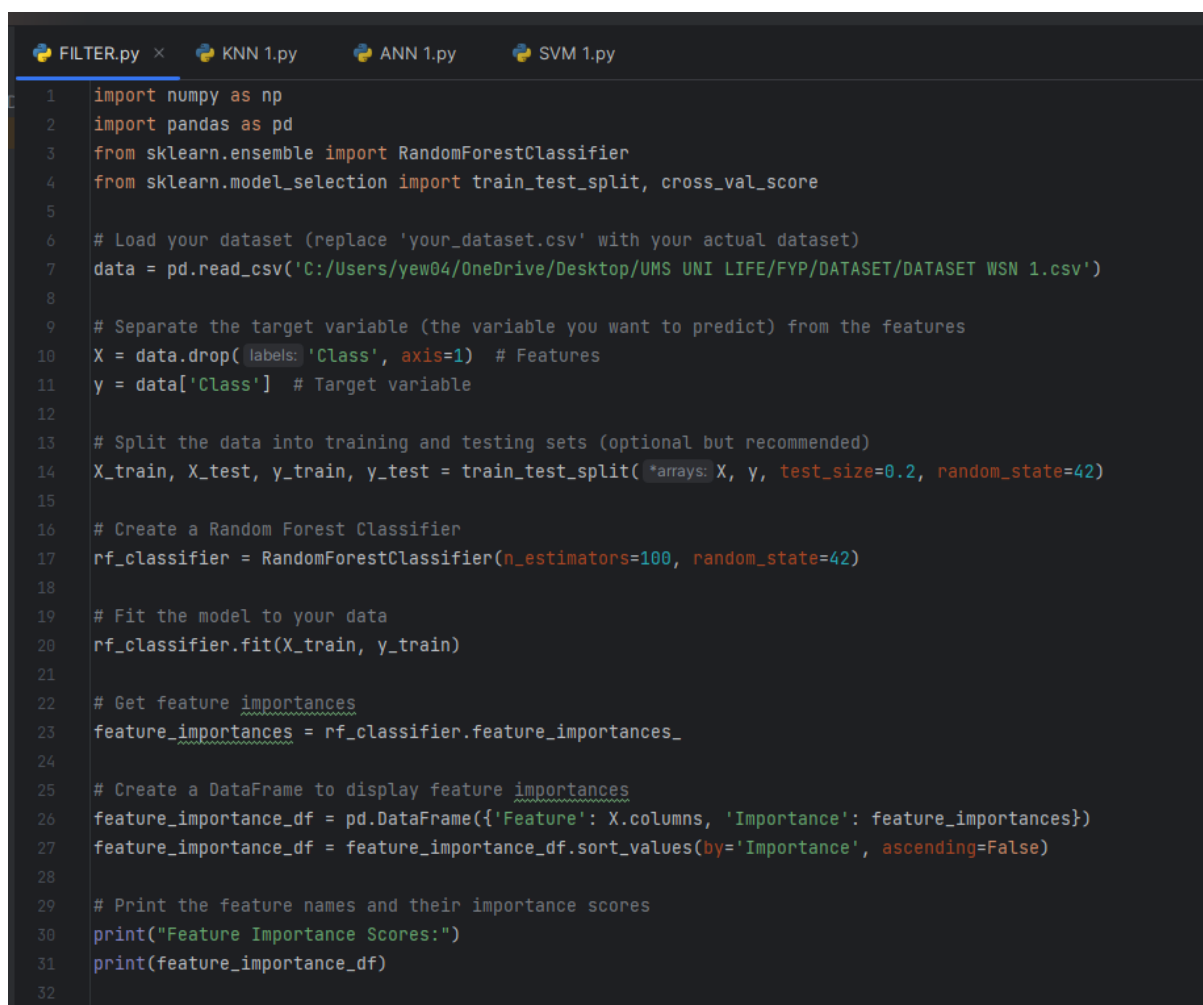
Figure 5.1: Original Wireless Sensor Network DDoS Attack Dataset from Kaggle

The datasets used for this project named as “WSNBFSFDataset” (Wireless Sensor Network Blackhole Flooding Selective Forwarding) which consist of 312107 rows that was obtained when the raw data set was prepared after the necessary pre-processing was performed. The

dataset consists of 4 different traffics: Blackhole, Flooding, Selective Forwarding Attack and Normal Traffic. It is a public dataset obtained from Kaggle. All libraries required such as pandas, numpy and sklearn are imported before loading the dataset. After that, the dataset is imported into the csv. File. The data set consists of 312107 rows and 18 features which is *Event*, *Time*, *S_Node*, *Node_id*, *Rest_Energy*, *Trace_Level*, *Mac_Type_Pckt*, *Source_IP_Port*, *Des_IP_Port*, *Packet_Size*, *TTL*, *Hop_Count*, *Broadcast_ID*, *Dest_Node_Num*, *Dest_Seq_Num*, *Src_Node_ID*, *Src_Seq_Num* and *Class*. To avoid any error in the experiments, any missing value in the dataset is removed.

5.3 Feature Extraction

An experiment is set up to filter the features of the dataset. The experiment is set up by using Python and Random Forest to get the feature importance score.



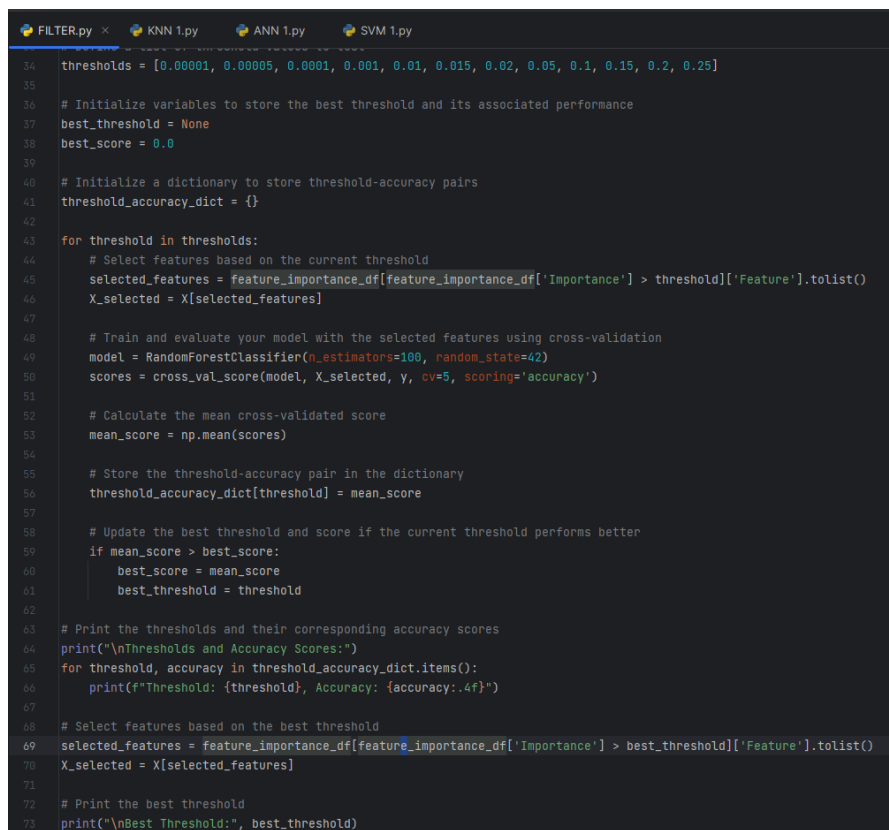
```
1 import numpy as np
2 import pandas as pd
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.model_selection import train_test_split, cross_val_score
5
6 # Load your dataset (replace 'your_dataset.csv' with your actual dataset)
7 data = pd.read_csv('C:/Users/yew04/OneDrive/Desktop/UMS UNI LIFE/FYP/DATASET/DATASET WSN 1.csv')
8
9 # Separate the target variable (the variable you want to predict) from the features
10 X = data.drop(labels='Class', axis=1) # Features
11 y = data['Class'] # Target variable
12
13 # Split the data into training and testing sets (optional but recommended)
14 X_train, X_test, y_train, y_test = train_test_split(*arrays: X, y, test_size=0.2, random_state=42)
15
16 # Create a Random Forest Classifier
17 rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
18
19 # Fit the model to your data
20 rf_classifier.fit(X_train, y_train)
21
22 # Get feature importances
23 feature_importances = rf_classifier.feature_importances_
24
25 # Create a DataFrame to display feature importances
26 feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importances})
27 feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
28
29 # Print the feature names and their importance scores
30 print("Feature Importance Scores:")
31 print(feature_importance_df)
32
```

Figure 5.2: Experiment Setup to get the Feature Importance Score

Based on Figure 5.2, this figure demonstrates the implementation of a Random Forest Classifier for a classification task using the scikit-learn library. The code first loads the dataset into a Pandas Data Frame, separating the features (x) from the target variable (y). It then splits the data into training and testing sets using the `train_test_split` function. A Random Forest Classifier is instantiated with 100 trees (`n_estimators=100`) and fitted to the training data using the fit method.

The script calculates the feature importance score using the `feature_importances` attribute of the trained Random Forest model. These importance scores are organised into a Pandas DataFrame, which is sorted in descending order based on feature importance scores. Finally, the code prints the names of features along with their corresponding importance scores, providing insights into the significance of each feature in the classification process. This feature importance analysis can be valuable for understanding the contribution of different variables to the model's predictions and potentially for guiding feature selection decisions.

After getting the feature importance score, the score is used for further analysis to select the best features.



```

34 thresholds = [0.00001, 0.00005, 0.0001, 0.001, 0.01, 0.015, 0.02, 0.05, 0.1, 0.15, 0.2, 0.25]
35
36 # Initialize variables to store the best threshold and its associated performance
37 best_threshold = None
38 best_score = 0.0
39
40 # Initialize a dictionary to store threshold-accuracy pairs
41 threshold_accuracy_dict = {}
42
43 for threshold in thresholds:
44     # Select features based on the current threshold
45     selected_features = feature_importance_df[feature_importance_df['Importance'] > threshold]['Feature'].tolist()
46     X_selected = X[selected_features]
47
48     # Train and evaluate your model with the selected features using cross-validation
49     model = RandomForestClassifier(n_estimators=100, random_state=42)
50     scores = cross_val_score(model, X_selected, y, cv=5, scoring='accuracy')
51
52     # Calculate the mean cross-validated score
53     mean_score = np.mean(scores)
54
55     # Store the threshold-accuracy pair in the dictionary
56     threshold_accuracy_dict[threshold] = mean_score
57
58     # Update the best threshold and score if the current threshold performs better
59     if mean_score > best_score:
60         best_score = mean_score
61         best_threshold = threshold
62
63 # Print the thresholds and their corresponding accuracy scores
64 print("\nThresholds and Accuracy Scores:")
65 for threshold, accuracy in threshold_accuracy_dict.items():
66     print(f"Threshold: {threshold}, Accuracy: {accuracy:.4f}")
67
68 # Select features based on the best threshold
69 selected_features = feature_importance_df[feature_importance_df['Importance'] > best_threshold]['Feature'].tolist()
70 X_selected = X[selected_features]
71
72 # Print the best threshold
73 print("\nBest Threshold:", best_threshold)

```

Figure 5.3: Experiment Setup to get the Important Features

Based on Figure 5.3, the next process is performing feature selection based on a range of threshold values for feature importance scores obtained from a Random Forest Classifier. The script initialises a list of threshold values to test and sets up variables to store the best threshold and its associated performance. A dictionary, `threshold_accuracy_dict`, is initialised to store threshold-accuracy pairs. The script then iterates through each threshold value, selecting features with importance scores above the current threshold. For each set of selected features, a Random Forest model is trained and evaluated using cross-validation (5-fold). The mean cross-validated accuracy score is calculated and stored in the dictionary.

After iterating through all threshold values, the script prints the thresholds and their corresponding accuracy scores. It identifies the best-performing threshold based on the highest mean accuracy score. Finally, the code selects features using the best threshold, prints the selected features, and displays the overall best threshold.

This process aids in determining an optimal threshold for feature selection, considering the balance between the number of selected features and the model's accuracy. The selected features can then be used for further model training and analysis.

After the datasets have been prepared and pre-processed, testing and training will be run.

5.4 Testing and Training of Dataset

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import LabelEncoder
5 from sklearn.svm import SVC
6 from sklearn.metrics import accuracy_score, precision_score
7 import time
```

Figure 5.4: Importing Libraries

In this section, the necessary libraries are imported. `pandas` is imported for data manipulation, `numpy` for numerical operations, `train_test_split` for splitting the dataset, `LabelEncoder` for label encoding, `SVC` for the SVM classifier, `accuracy_score` and `precision_score` for performance evaluation, and `time` for tracking the execution time.

```
9 # Load the dataset
10 df = pd.read_csv("C:/Users/NotTee/Desktop/UMS UNI LIFE/FYP/DATASET/DATASET WSN 1.csv")
```

Figure 5.5: Importing .csv Dataset

This line loads the dataset from a CSV file. Make sure to provide the correct file path to your dataset.

```
12 # Extract the features and the target variable from the dataset
13 X = df.drop('Class', axis=1) # Exclude the 'Class' column
14 y = df['Class']
```

Figure 5.6: Extracting Features

Here, the features and the target variable are extracted from the dataset. The features are stored in the variable `X`, and the target variable is stored in `y`. The `Class` column is dropped from `X` using `drop()`.

```
16 # Apply label encoding to the target variable
17 le = LabelEncoder()
18 y = le.fit_transform(y)
```

Figure 5.7: Applying Label Encoding

The target variable `y` is label encoded using `LabelEncoder()`. This is necessary when the target variable contains categorical values that need to be converted to numerical values for model training.

```
20 # Split the dataset into training and testing dataset
21 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 5.8: Splitting Dataset

The dataset is split into training and testing sets using `train_test_split()`. The training set contains 80% of the data, while the testing set contains the remaining 20%. The features and target variables are split into `X_train`, `X_test`, `y_train`, and `y_test`, respectively.

```
23 # Create an instance of the SVC classifier
24 svm = SVC()
```

```
23 # Create an instance of the KNN classifier
24 knn = KNeighborsClassifier()
```

```
23 # Create an instance of the ANN classifier
24 ann = MLPClassifier()
```

Figure 5.9: Creating instance for the classifier

Instance of the SVM, KNN and ANN classifier is created using `SVC()`, `KNeighborsClassifier` and `MLPClassifier`. This initializes the classifier with different parameters.

```
26 # Variables to track training progress
27 total_epochs = 10 # Specify the total number of epochs
28 train_losses = []
29 val_losses = []
30 train_accuracies = []
31 val_accuracies = []
32 train_precisions = []
33 val_precisions = []
```

Figure 5.10: Create Variables to track training progress

These variables are initialized to track the training progress. `total_epochs` specifies the total number of epochs (iterations over the training dataset) for training the model. Lists are created to store the training and validation losses, accuracies, and precisions for each epoch.

```
35 # Training loop
36 start_time = time.time()
37 for epoch in range(total_epochs):
38     # Train the model using the training data
39     nn.fit(X_train, y_train)
40
41     # Training evaluation
42     train_pred = nn.predict(X_train)
43     train_loss = 1 - accuracy_score(y_train, train_pred)
44     train_accuracy = accuracy_score(y_train, train_pred)
45     train_precision = precision_score(y_train, train_pred, average='weighted', zero_division=1)
46     train_losses.append(train_loss)
47     train_accuracies.append(train_accuracy)
48     train_precisions.append(train_precision)
49
50     # Validation evaluation
51     val_pred = nn.predict(X_test)
52     val_loss = 1 - accuracy_score(y_test, val_pred)
53     val_accuracy = accuracy_score(y_test, val_pred)
54     val_precision = precision_score(y_test, val_pred, average='weighted', zero_division=1)
55     val_losses.append(val_loss)
56     val_accuracies.append(val_accuracy)
57     val_precisions.append(val_precision)
```

```
59 print(
60     f"Epoch {epoch + 1}/{total_epochs} - Train Loss: {train_loss:.4f} - Train Accuracy: {train_accuracy:.4f} - Train Precision:
61
62 end_time = time.time()
```

Figure 5.11: Training Loop

The code then enters a training loop that runs for the specified number of epochs (`total_epochs`). In each epoch:

- For example for the SVM classifier is trained on the training data using `svm.fit(X_train, y_train)`.
- Training evaluation metrics such as loss, accuracy, and precision are calculated using the training data.
- Validation evaluation metrics are calculated using the testing data (`X_test`).
- The calculated metrics are appended to the respective lists for later analysis.

After the training loop, the model is evaluated on the testing set. Predictions are obtained using for example svm `svm.predict(X_test)`, and the accuracy and precision scores are calculated using `accuracy_score()` and `precision_score()`.

```
71 # Calculate total validation accuracy and precision
72 val_accuracy_total = accuracy_score(y_test, val_pred)
73 val_precision_total = precision_score(y_test, val_pred, average='weighted', zero_division=1)
74 print(f"Total Validation Accuracy: {val_accuracy_total:.4f}")
75 print(f"Total Validation Precision: {val_precision_total:.4f}")
```

Figure 5.12: Calculating Total Accuracy and Precision

Total validation accuracy and precision are calculated using the predictions obtained during the last epoch.

```
77 # Calculate average training and validation losses
78 avg_train_loss = np.mean(train_losses)
79 avg_val_loss = np.mean(val_losses)
80 print(f"Average Training Loss: {avg_train_loss:.4f}")
81 print(f"Average Validation Loss: {avg_val_loss:.4f}")
82
83 # Calculate average training and validation accuracies
84 avg_train_accuracy = np.mean(train_accuracies)
85 avg_val_accuracy = np.mean(val_accuracies)
86 print(f"Average Training Accuracy: {avg_train_accuracy:.4f}")
87 print(f"Average Validation Accuracy: {avg_val_accuracy:.4f}")
88
89 # Calculate average training and validation precisions
90 avg_train_precision = np.mean(train_precisions)
91 avg_val_precision = np.mean(val_precisions)
92 print(f"Average Training Precision: {avg_train_precision:.4f}")
93 print(f"Average Validation Precision: {avg_val_precision:.4f}")
```

Figure 5.13: Calculating Average Training and Validation Losses

Average training and validation losses, accuracies, and precisions are calculated using ``np.mean()`` on the respective lists.

```
95 # Calculate total time elapsed
96 total_time = end_time - start_time
97 print(f"Total Time Elapsed: {total_time:.2f} seconds")
```

Figure 5.14: Calculating Total Time Elapsed

The total time elapsed for model training and evaluation is calculated and printed.

This code performs the training of an SVM (Support Vector Machine), KNN (K-Nearest Neighbors), ANN (Artificial Neural Network) classifier, evaluates its performance on the testing set, and provides various metrics such as accuracy, precision, losses, and time elapsed. The training loop allows tracking the progress and performance over multiple epochs.

5.5 Summary

Chapter 5 discusses the implementation of the experiments, focusing on data cleaning, preparation, preprocessing, and feature extraction. The experiments are conducted using Python in PyCharm, and various steps are illustrated with code snippets. The original dataset, obtained from Kaggle, is named "WSNBFSFDataset" and contains 312107 rows with 18 features, including information related to Blackhole, Flooding, Selective Forwarding Attacks, and Normal Traffic.

Feature extraction is performed using a Random Forest Classifier to obtain feature importance scores. The experiment sets up threshold values for feature selection, iterates through them, and evaluates the model's performance, helping identify optimal features for further analysis.

The testing and training of the dataset involve importing necessary libraries, loading the dataset, extracting features, applying label encoding, and splitting the dataset into training and testing sets. The chapter provides code snippets for importing libraries, loading the dataset, extracting features, applying label encoding, and splitting the dataset into training and testing sets.

The implementation includes creating instances for SVM, KNN, and ANN classifiers and a training loop to iterate through epochs, calculating training and validation metrics. Finally,

the chapter calculates total accuracy, precision, average training and validation losses, and total time elapsed for model training and evaluation.

In summary, Chapter 5 details the practical implementation of the experiments, providing insights into the steps involved in preparing, processing, and training the dataset for wireless sensor network DDoS attack detection.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Introduction

In this chapter, the experiment results after conducting the experiments will be shown and explained. After the data preparing and preprocessing is conducted, the classifications method chosen are applied. The overall report of the classification performance has been analysed.

6.2 Experiment Result of Feature Extraction

6.2.1 Feature Importance

The feature importance score is calculated using the `feature_importances` attribute of the trained Random Forest model instantiated with 100 trees (`n_estimators=100`) and fitted to the training data using the fit method.

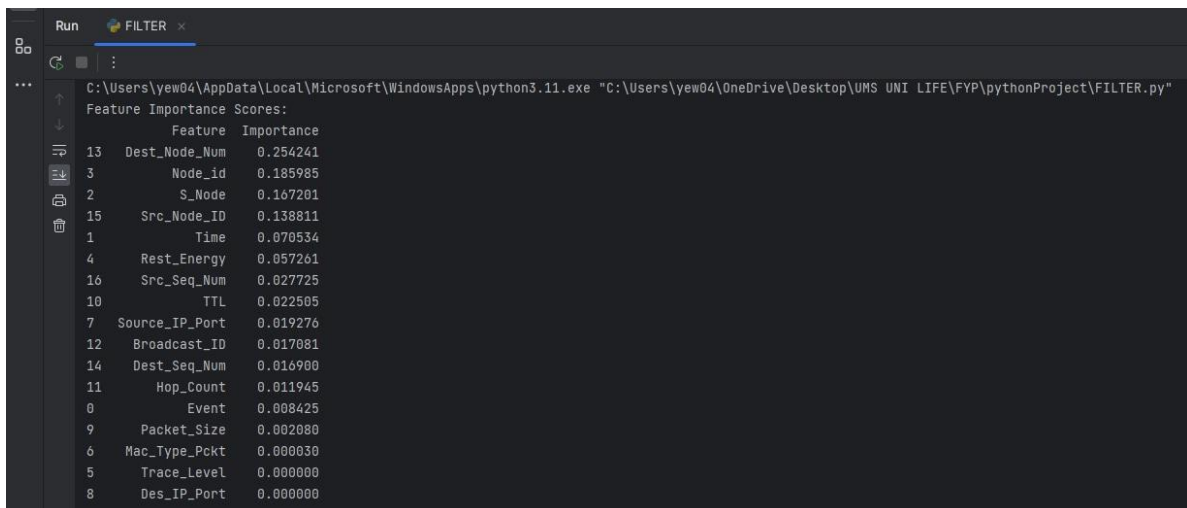


Figure 6.1: Outcome Feature Importance Score

The Feature Importance Score of each feature is calculated. For instance, `Dest_Node_Num` has feature importance score of 0.254241 which means that it has 25.4241% of contribution

in the dataset. This feature importance analysis can be valuable for understanding the contribution of different variables to the model's predictions and potentially for guiding feature selection decisions.

6.2.2 Feature Extraction

After getting the feature importance score, the score is used for further analysis to select the best features. A range of threshold values will be test and sets up variables to store the best threshold and its associated performance. `threshold_accuracy_dict`, is initialised to store mean cross-validated accuracy score that is calculated.

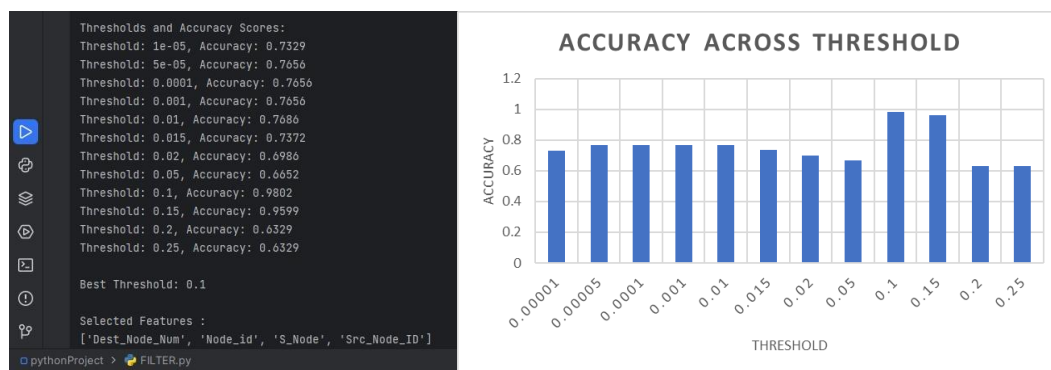


Figure 6.2: Outcome Important Features

After iterating through all threshold values, the script prints the thresholds and their corresponding accuracy scores which is threshold with 0.1. Finally, the code selects features using the best threshold, prints the selected features which is `Dest_Node_Num`, `Node_id`, `S_Node` and `Src_Node_ID`.

6.3 Experiments Results of Machine Learning Algorithms

The classifiers that have been choose and used are SVM (Support Vector Machine), KNN (K-Nearest Neighbors) and ANN (Artificial Neural Network).

6.3.1 Accuracy

The accuracy is calculated using the `accuracy_score` function in the Python for PyCharm from the `sklearn.metrics` module. The `accuracy_score` function compares the predicted labels (`y_pred`) with the actual labels (`y_test`) and calculates the accuracy as the ratio of correct predictions to the total number of predictions. Accuracy calculation for training evaluation and validation evaluation:

```
44 train_accuracy = accuracy_score(y_train, train_pred)
```

```
53 val_accuracy = accuracy_score(y_test, val_pred)
```

It calculates the accuracy of the training and validation predictions (`train_pred`) and (`val_pred`) by comparing them with the actual training labels and validation labels (`y_train`) and (`y_test`).

For the total testing accuracy to be calculated:

```
66 accuracy = accuracy_score(y_test, y_pred)
```

It calculates the accuracy of the final testing predictions (y_pred) by comparing them with the actual testing labels (y_test).

6.3.2 Precision

On the other hand, precision is calculated using the `precision_score` function from the `sklearn.metrics` module. The `precision_score` function computes the precision of the model's predictions.

Precision is defined as the ratio of true positives (correctly predicted positive samples) to the sum of true positives and false positives. It measures how well the model identifies positive samples correctly.

Here's how precision is calculated in the training and validation evaluation:

```
45 train_precision = precision_score(y_train, train_pred, average='weighted', zero_division=1)
```

```
54 val_precision = precision_score(y_test, val_pred, average='weighted', zero_division=1)
```

It calculates the precision of the training predictions and validation predictions (`train_pred`) and (`val_pred`) by comparing them with the actual validation labels (`y_train`) and (`y_test`). The `average='weighted'` argument is used to calculate precision for each class and weight them based on the number of samples in each class.

For total testing precision:

```
67 precision = precision_score(y_test, y_pred, average='weighted', zero_division=1)
```

It calculates the precision of the final testing predictions (`y_pred`) by comparing them with the actual testing labels (`y_test`). The `average='weighted'` argument ensures

that precision is calculated for each class and weighted by the number of samples in each class.

The `accuracy_score` function and `precision_score` functions returns a value between 0 and 1, where 1 represents perfect precision (all positive predictions are correct) and 0 represents no precision (all positive predictions are incorrect). The accuracy is a common evaluation metric used in classification tasks to assess the performance of a model while precision is commonly used as an evaluation metric in classification tasks, especially when class imbalance is present or when the cost of false positives is high.

6.3.3 Time Elapsed

Moreover, the elapsed time is calculated using the `time` module from the Python standard library. Before the training loop starts, the current time is captured using `time.time()` and stored in the variable `start_time`:

```
36 start_time = time.time()
```

After the training loop completes, the current time is captured again using `time.time()` and stored in the variable `end_time`:

```
62 end_time = time.time()
```

The total time elapsed is then calculated by subtracting the `start_time` from the `end_time`:

```
95 # Calculate total time elapsed
96 total_time = end_time - start_time
97 print(f"Total Time Elapsed: {total_time:.2f} seconds")
```

The `total_time` variable represents the total time elapsed between the start and end of the training loop, measured in seconds and the total time elapsed will be displayed in seconds with two decimal places.

6.3.4 Mean Squared Error (MSE)

```
103 # Calculate Mean Squared Error (MSE)
104 mse = mean_squared_error(y_test, y_pred)
105 print(f"Mean Squared Error (MSE): {mse:.4f}")
```

'mean_squared_error' is a function used to measure the average squared difference between the actual and predicted values in a regression problem. 'y_test' represents the true values or ground truth from test set while 'y_pred' represents the predicted values obtained from model. The result is stored in the variable 'mse' and then displays the MSE value with four decimal places.

6.3.5 Mean Absolute Error (MAE)

```
107 # Calculate Mean Absolute Error (MAE)
108 mae = mean_absolute_error(y_test, y_pred)
109 print(f"Mean Absolute Error (MAE): {mae:.4f}")
```

'mean_absolute_error' is a metric that calculates the average absolute differences between the actual and predicted values. 'y_test' and 'y_pred' are the same as in the MSE calculation. The MAE result is stored in the variable 'mae' and it displays the MAE value with four decimal places.

6.3.6 Mean Absolute Percentage Error (MAPE)

```
111 # Calculate Mean Absolute Percentage Error (MAPE)
112 mape = np.mean(np.abs((y_test - y_pred) / np.maximum(y_test, 1))) * 100
113 print(f"Mean Absolute Percentage Error (MAPE): {mape:.4f}%")
```

MAPE calculates the average percentage difference between the actual and predicted values. 'np.mean' computes the average of the absolute percentage errors. 'np.abs' is used to get the absolute values of the errors. 'np.maximum' is used to avoid division by zero by ensuring that the denominator is at least 1. The result is multiplied by 100 to get the percentage and stored in the variable 'mape' and displays the MAPE value with four decimal places.

6.3.7 Confusion Matrix

```
115 # Calculate confusion matrix
116 conf_matrix = confusion_matrix(y_test, y_pred)
117 plt.figure(figsize=(8, 6))
118 sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
119 plt.xlabel('Predicted')
120 plt.ylabel('True')
121 plt.title('Confusion Matrix')
122 plt.show()
```

'confusion_matrix' is a function that computes the confusion matrix to evaluate the accuracy of a classification. 'y_test' represents the true class labels and 'y_pred' represents the predicted class labels obtained from model. The confusion matrix is stored in the variable 'conf_matrix'. 'plt.figure' initializes a new figure for the heatmap with a specified size. 'sns.heatmap' creates a heatmap of the confusion matrix with annotations, using a blue color map while 'plt.show()' displays the heatmap.

These components are commonly used in evaluating the performance of machine learning models, providing insights into how well the model is making predictions.

6.4 Experiment Result for SVM (Support Vector Machine)

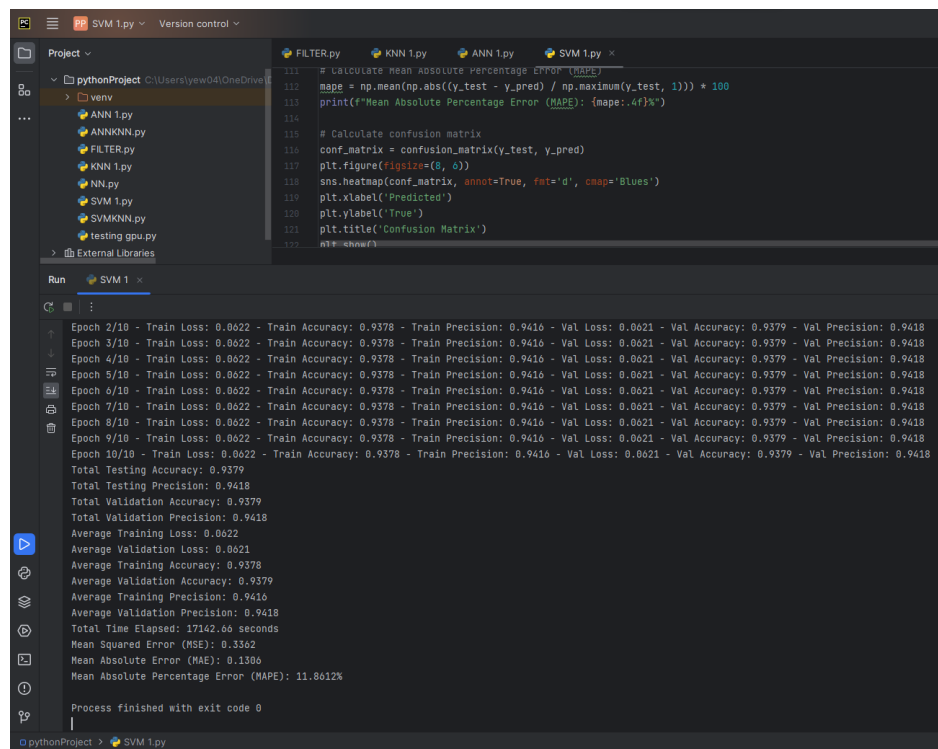


Figure 6.3: One of the experiment result screenshots of SVM

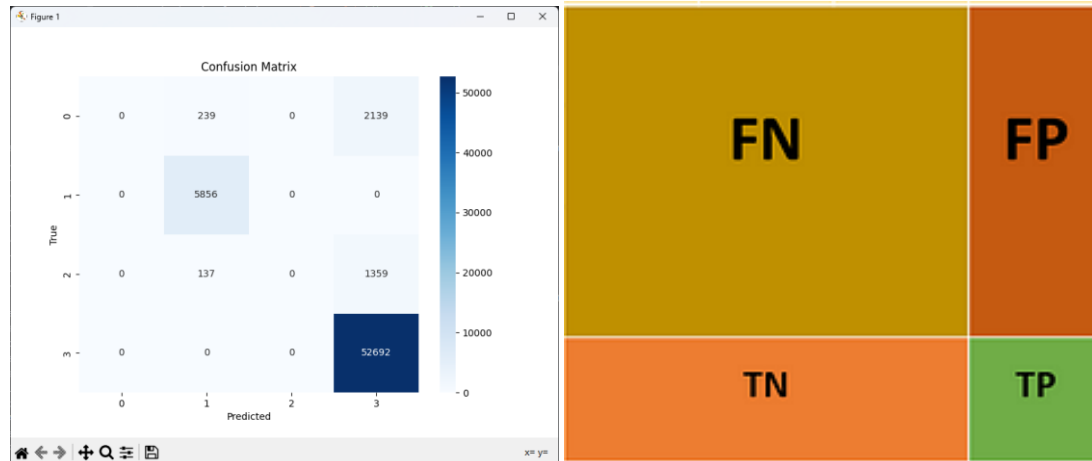


Figure 6.4: Confusion Matrix of SVM

SVM								
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	10	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	17142.66
2	10	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	20077.77
3	10	0.3361	0.1306	11.8612	0.0621	0.9418	0.9379	19159.44
	10	0.336167	0.1306	11.8612	0.0621	0.9418	0.9379	18793.29
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	20	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	36492.3
2	20	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	36196.52
3	20	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	35925.85
	20	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	36204.89
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	30	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	53942.44
2	30	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	54351.84
3	30	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	53530.6
	30	0.3362	0.1306	11.8612	0.0621	0.9418	0.9379	53941.6267

Figure 6.5: Experiment result for SVM

Based on Figure 6.2, the confusion matrix produced is used to evaluating the performance of SVM, it provides a provides a detailed breakdown of the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by SVM. Refer to figure 6.3, after training the dataset with SVM (Support Vector Machine), the average result is quite stable, which is having 0.3362 MSE (Mean Squared Error), 0.1306 MAE (Mean Absolute Error) and 11.86% of MAPE (Mean Absolute Percentage Error). While the average loss is 0.0621, average validation precision is 0.9418 and average accuracy is 0.9379. SVM takes longest time to run the whole dataset, which is around 18793.29 seconds for 10 epochs, 36204.89 seconds for 20 epochs and 539412.63 seconds for 30 epochs which is around 15 hours.

6.5 Experiment Result for KNN (K-Nearest Neighbors)

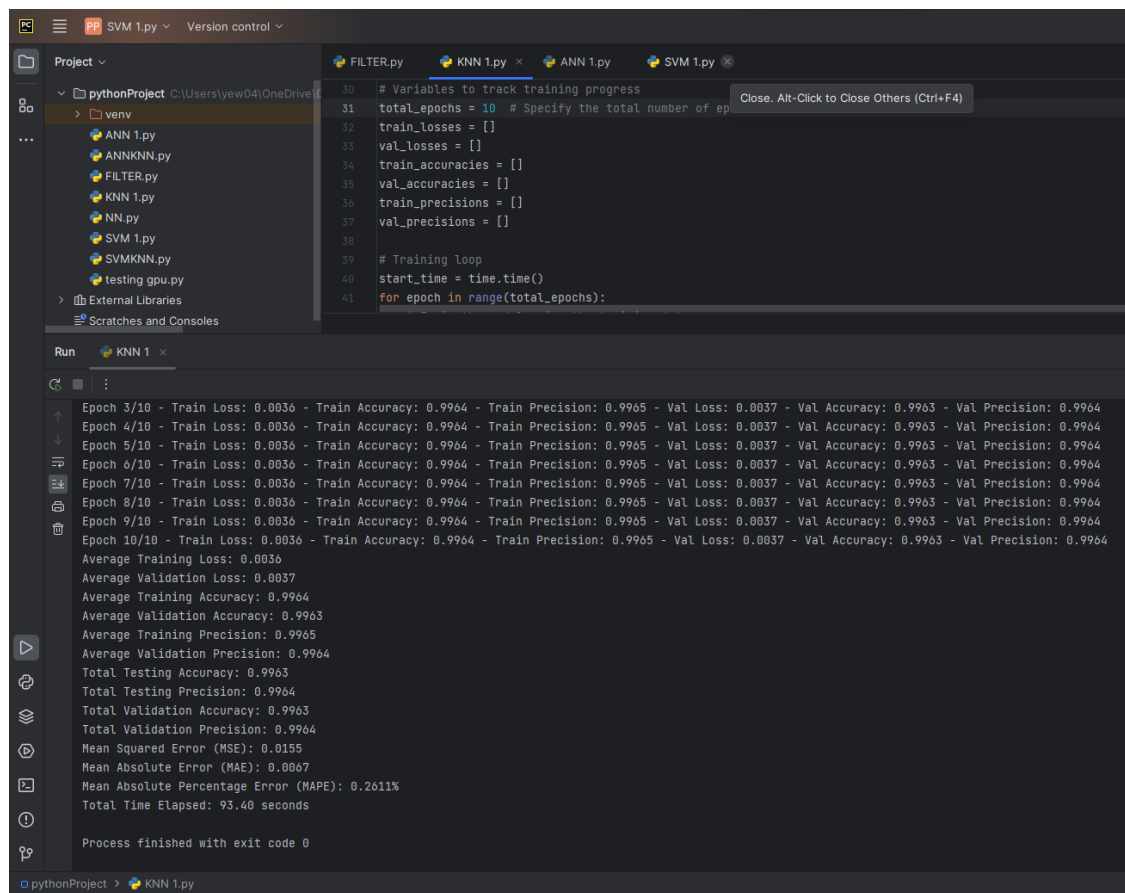


Figure 6.4: One of the experiment result screenshots of KNN

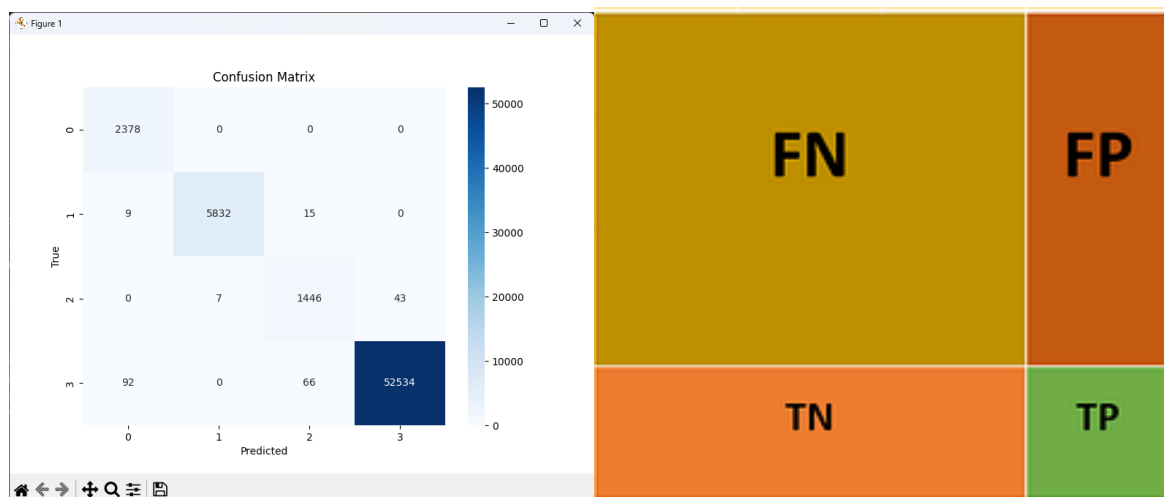


Figure 6.7: Confusion Matrix of KNN

KNN								
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	10	0.0155	0.0067	0.2611	0.0037	0.9964	0.9963	93.4
2	10	0.0155	0.0067	0.2611	0.0037	0.9964	0.9963	90.83
3	10	0.0155	0.0067	0.2611	0.0037	0.9964	0.9963	88.91
	10	0.0155	0.0067	0.2611	0.0037	0.9964	0.9963	91.0466667
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	20	0.0155	0.0067	0.2611	0.0037	0.9964	0.9963	182.88
2	20	0.0155	0.0067	0.2611	0.0037	0.9964	0.9963	177.4
3	20	0.0155	0.0067	0.2611	0.0037	0.9694	0.9963	174.1
	20	0.0155	0.0067	0.2611	0.0037	0.9874	0.9963	178.126667
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	30	0.0155	0.0067	0.2611	0.0037	0.9964	0.9963	256.51
2	30	0.0155	0.0067	0.2611	0.0037	0.9964	0.9963	260.69
3	30	0.0155	0.0067	0.2611	0.0037	0.9964	9963	275.03
	30	0.0155	0.0067	0.2611	0.0037	0.9964	3321.6642	264.076667

Figure 6.8: Experiment Result of KNN

Based on Figure 6.5, the confusion matrix produced is used to evaluating the performance of KNN, it provides a provides a detailed breakdown of the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by KNN. Figure 6.6, after training the dataset with KNN (K-Nearest Neighbors), the experiment result is more stable. It reaches 0.0155 of MSE (Mean Squared Error), 0.0067 of MAE (Mean Absolute Error) and 0.2611 of MAPE (Mean Absolute Percentage Error). Besides, the average loss is 0.0037, average precision is 0.9964 and the average accuracy is 0.9963. KNN only takes a short time to run which is 91.05 seconds for 10 epochs, 178.13 seconds for 20 epochs and 264.08 seconds for 30 epochs which is only around 4 minutes.

6.6 Experiment Result for ANN (Artificial Neural Network)

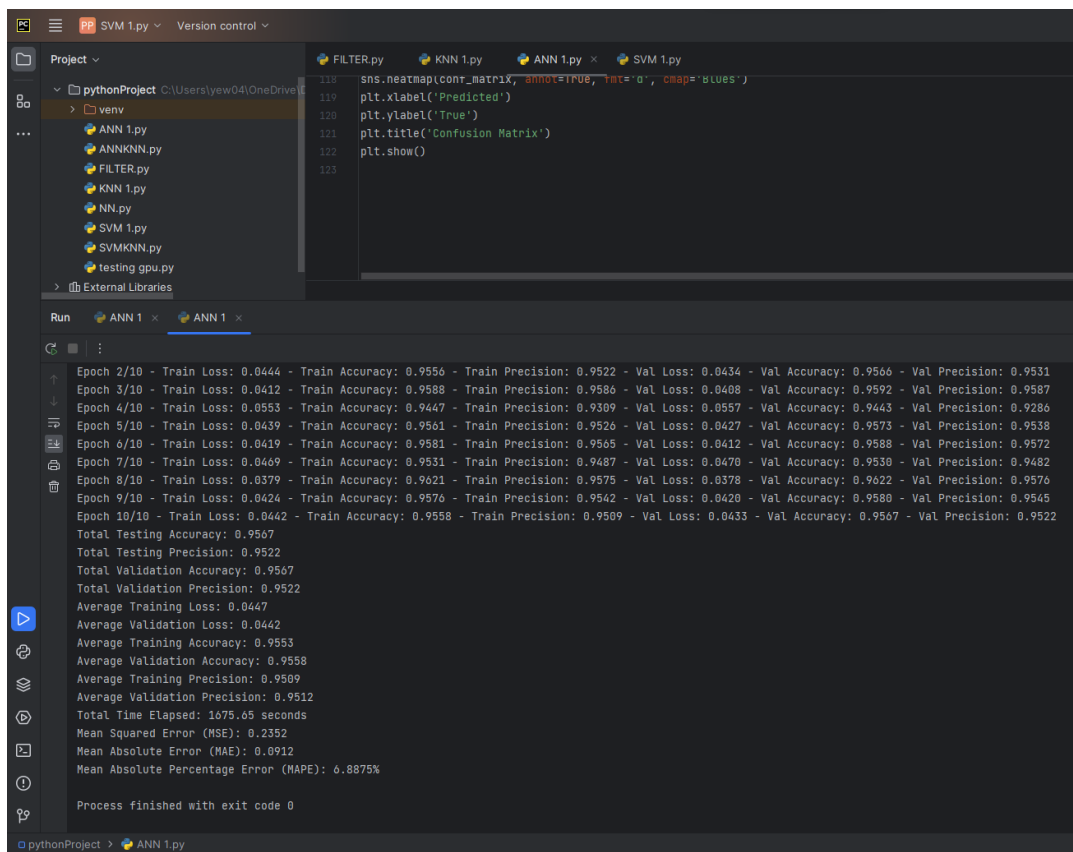


Figure 6.9: One of the experiment result screenshots of ANN

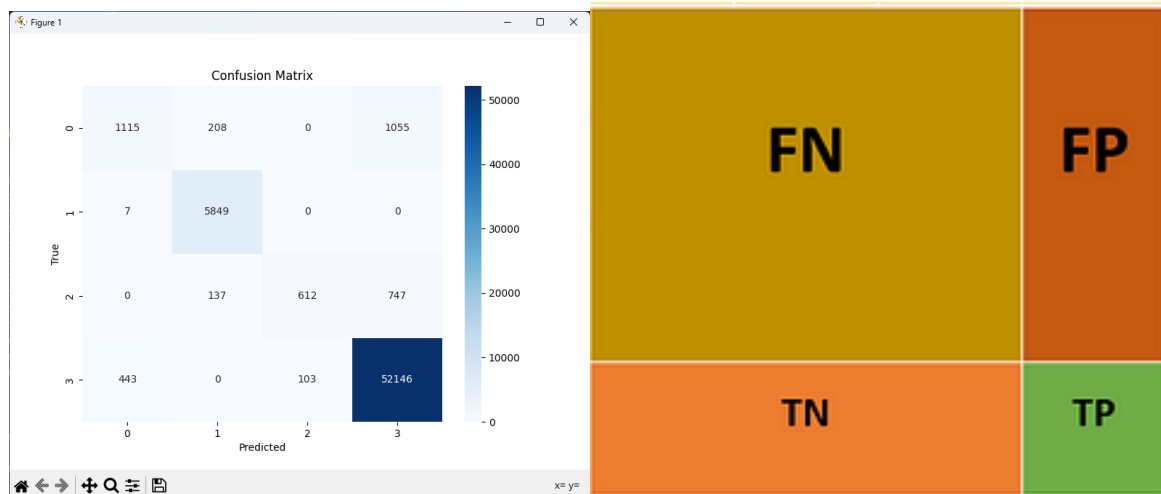


Figure 6.10: Confusion Matrix of ANN

ANN								
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	10	0.2352	0.0912	6.8875	0.0442	0.9512	0.9558	1675.65
2	10	0.2757	0.1068	8.0332	0.0436	0.9528	0.9564	1426.43
3	10	0.2332	0.0864	6.7396	0.0404	0.9576	0.9596	1389.19
10		0.248033	0.0948	7.2201	0.042733	0.953866667	0.95726667	1497.09
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	20	0.2338	0.0893	6.8437	0.0405	0.9572	0.9595	3454.04
2	20	0.2142	0.0842	7.3051	0.0415	0.9554	0.9585	2858.05
3	20	0.2308	0.0883	6.7706	0.0417	0.9551	0.9583	2866.56
20		0.226267	0.08727	6.9731333	0.041233	0.9559	0.95876667	3059.55
Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse
1	30	0.2652	0.0978	9.0481	0.041	0.9551	0.959	4055.94
2	30	0.2379	0.0966	7.1113	0.0424	0.9544	0.9576	4291.47
3	30	0.218	0.0834	6.5244	0.0404	0.9564	0.9596	4691.34
30		0.240367	0.0926	7.5612667	0.041267	0.9553	0.95873333	4346.25

Figure 6.11: Experiment result of ANN

Based on Figure 6.8, the confusion matrix produced is used to evaluating the performance of ANN, it provides a provides a detailed breakdown of the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by ANN. Refer to figure 6.9, after training the dataset with ANN (Artificial Neural Network), the result is quite unstable, it achieved average 0.955 of accuracy and average loss is 0.04. Besides, ANN with 20 epochs have the highest precision with 0.2263 MSE (Mean Squared Error), 0.0873 MAE (Mean Absolute Error) and 6.9731 of MAPE (Mean Absolute Percentage Error). The time elapse increased when the epoch increased.

6.7 Discussion

Table 6.1: Summary of Results of classifier

Classifier	Total MSE	Total MAE	Total MAPE	Total Accuracy	Total Precision	Total Time Elapsed
SVM (Support Vector Machine)	0.3362	0.1306	11.8612	0.9379	0.9418	18793.29 seconds
KNN (K-Nearest Neighbours)	0.0155	0.0067	0.2611	0.9963	0.9964	91.05 seconds
ANN (Artificial Neural Network)	0.2263	0.0873	6.9731	0.9588	0.9559	3059.55 seconds

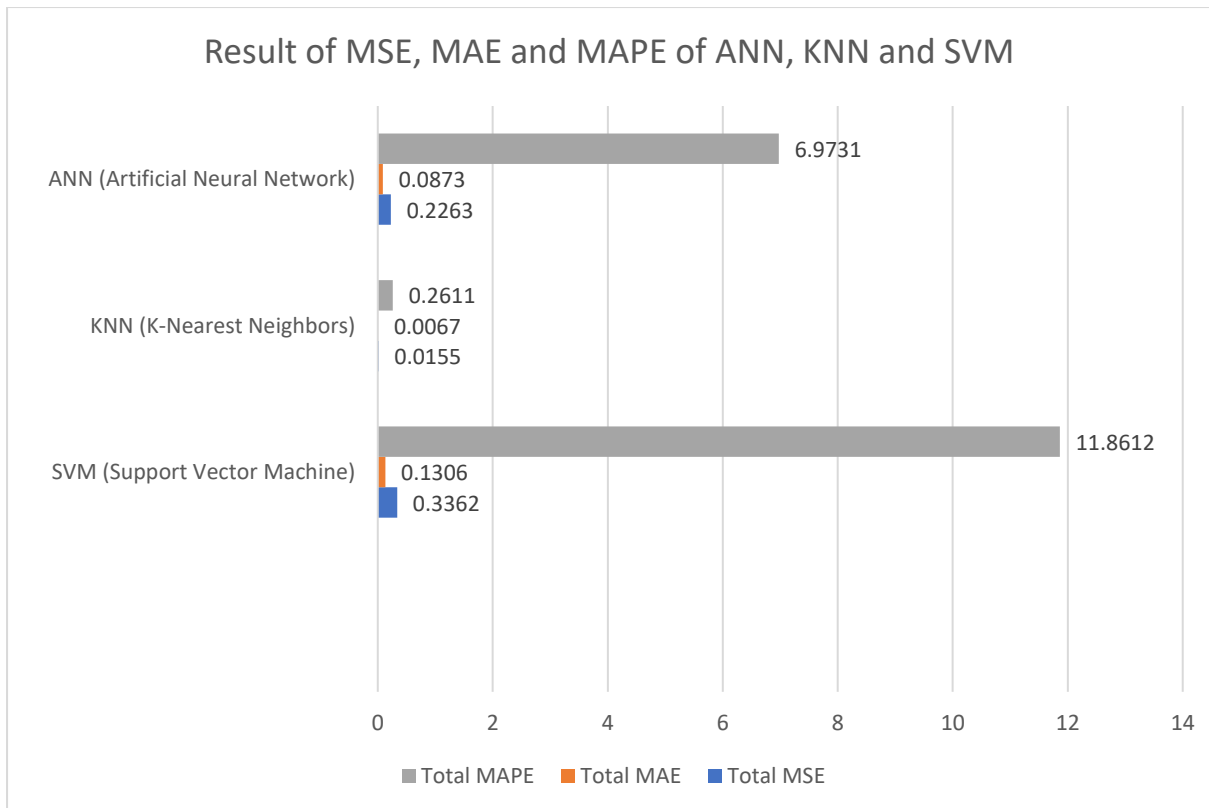


Figure 6.12: Result of MSE, MAE and MAPE of ANN, KNN and SVM

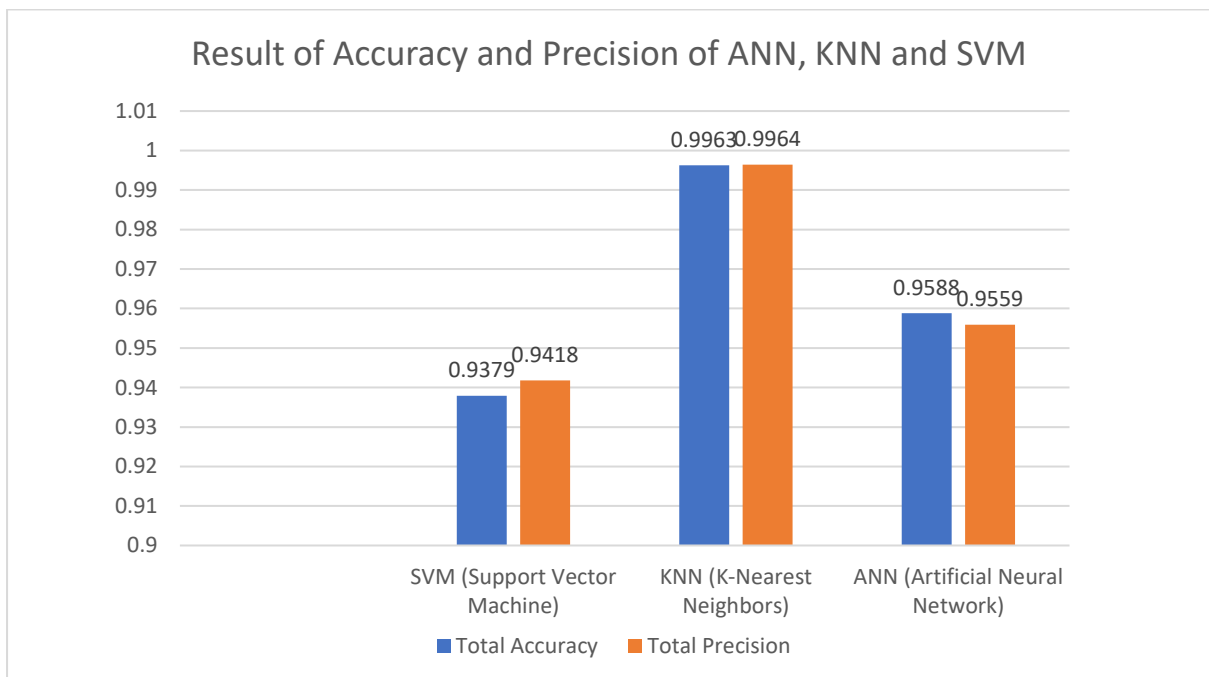


Figure 6.13: Result of Accuracy and Precision of ANN, KNN and SVM

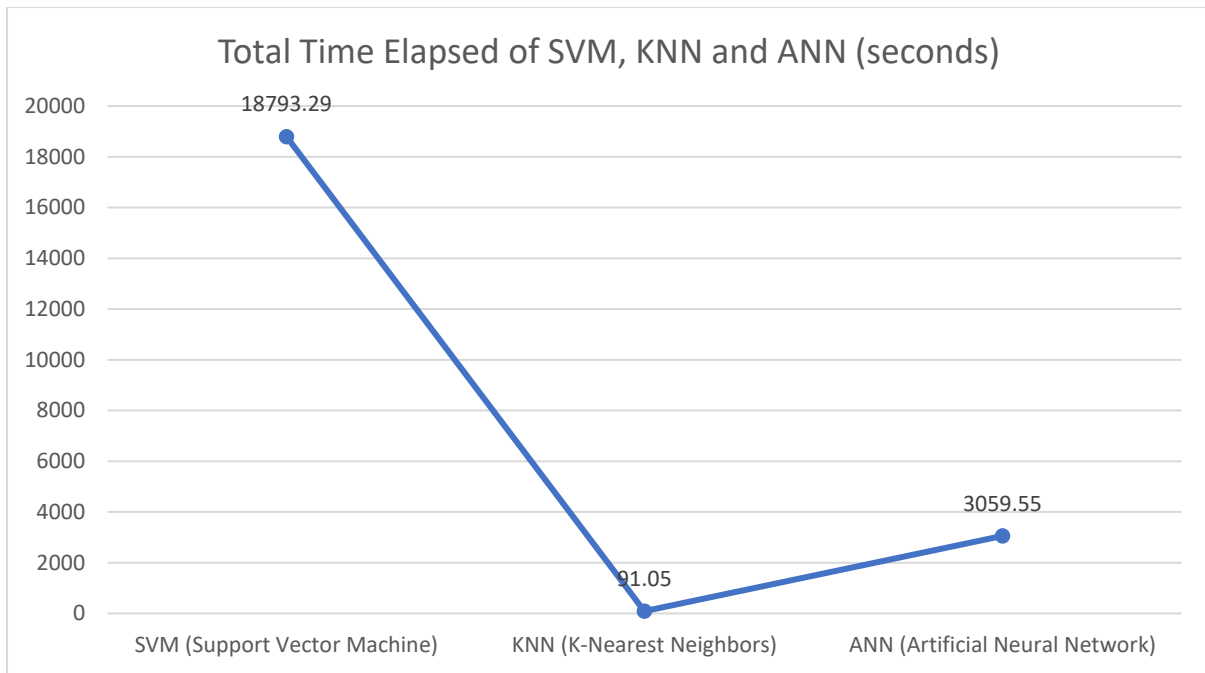


Figure 6.14: Total Time Elapsed of SVM, KNN and ANN

Refer to table 6.1, Figure 6.12, Figure 6.13 and Figure 6.14, based on the results from the experiments from the three classifiers, KNN (K-Nearest Neighbors) with 10 epochs is the best in wireless sensor network detection among SVM (Support Vector Machine) and ANN (Artificial Neural Network) because it have a highest accuracy of 99.63%, highest precision of 99.64% and a shortest time elapsed of 91.0591.05 seconds which is around 1.52 minutes only for a such huge dataset. Moreover, it also having the lowest 0.0155 MSE (Mean Squared Error), 0.0067 MAE (Mean Absolute Error) and 0.26% MAPE (Mean Absolute Percentage Error) which means that KNN have a better model performance than others. KNN is more suitable to be used for Wireless DDoS attack detection.

6.8 Interface Design

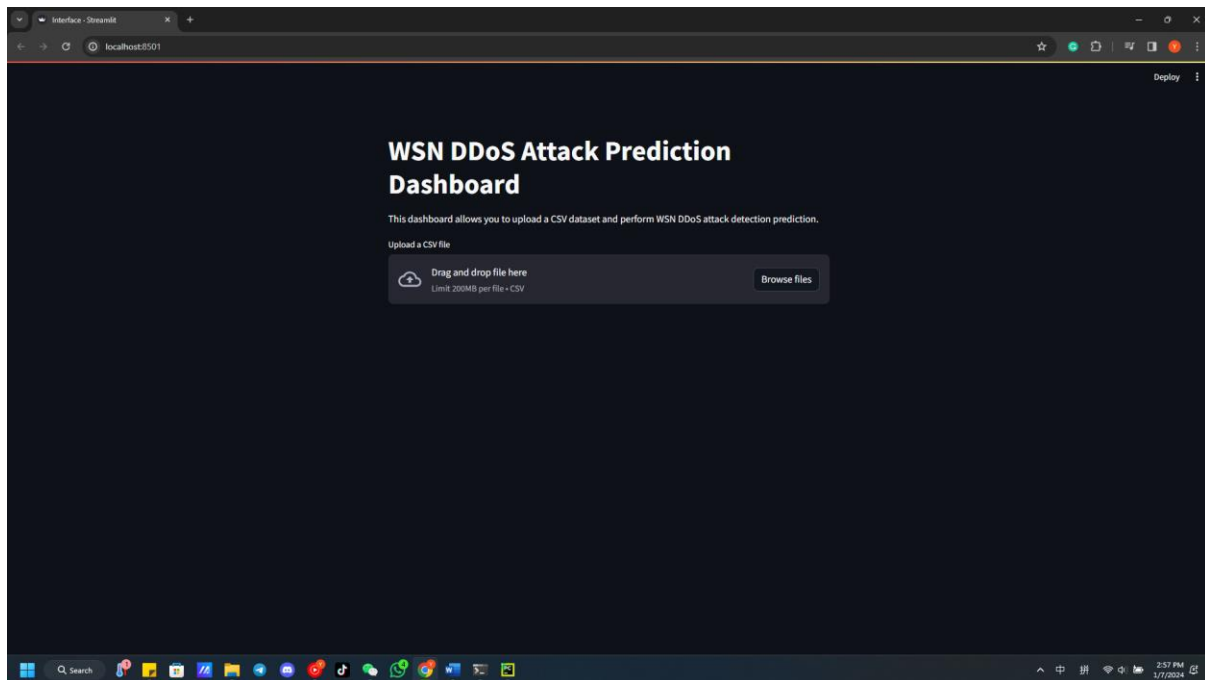


Figure 6.15: WSN DDoS Attack Prediction Dashboard

After we get the best classifier between KNN (K-Nearest Neighbors), SVM (Support Vector Machine) and ANN (Artificial Neural Network), an interface will be developed to let users to do Wireless Sensor Network DDoS attack prediction by using KNN. The WSN DDoS Attack Prediction Dashboard was researched and developed by using Streamlit, which allows a structured flow for users to upload a CSV dataset and analyze WSN DDoS attack predictions.

The app begins with a file upload widget, allowing users to upload a CSV file. Once uploaded, the dataset is loaded into a Pandas DataFrame (``df``). The app then displays basic information about the dataset, such as the total number of rows and features, providing users with an initial overview.

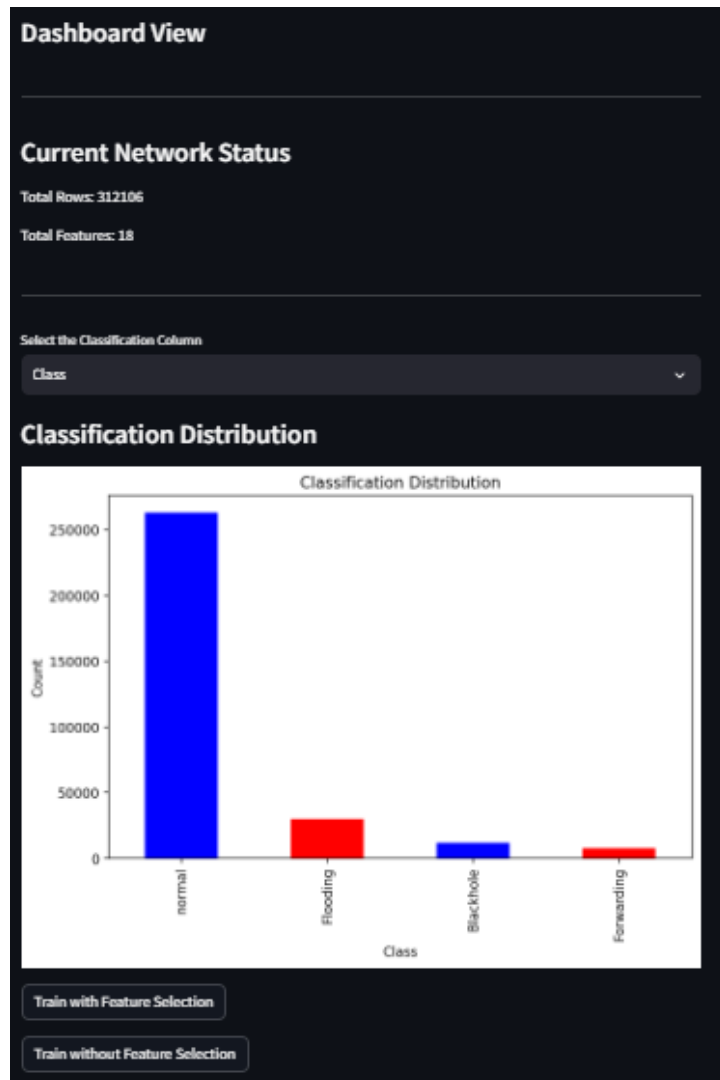


Figure 6.16: Dashboard view of WSN DDoS Attack Prediction Dashboard

Moving on to the "Dashboard View" section, users can choose the target classification column from the uploaded dataset. The class distribution of the selected classification column is visualized using a bar chart, giving users insights into the distribution of different classes in the dataset.

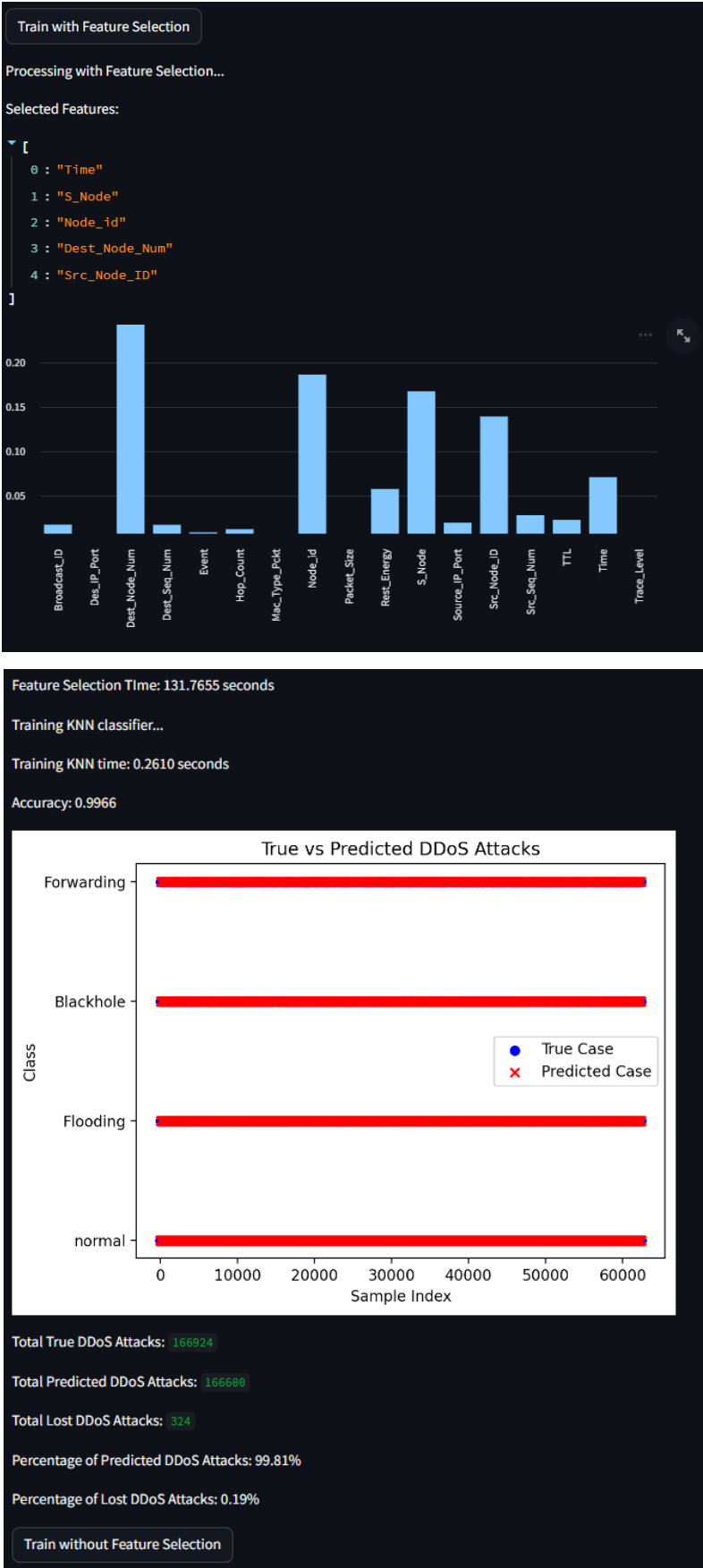


Figure 6.17: Train with Feature Selection

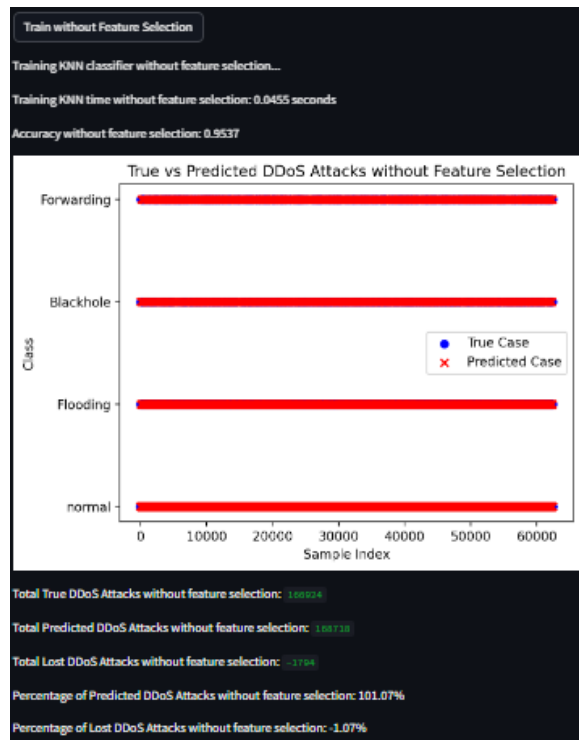


Figure 6.18: Train without Feature Selection

The app provides an option for users to train a DDoS attack prediction model with or without feature selection. If the user opts for feature selection, a Random Forest classifier is trained on the dataset. Cross-validation is performed to evaluate the model, and important features are selected using the `SelectFromModel` method. The selected features are displayed, along with a bar chart showing the feature importances. Subsequently, a K-Nearest Neighbors (KNN) classifier is trained on the selected features. Prediction results are visualized using a scatter plot, and various metrics such as accuracy, total true attacks, total predicted attacks, total lost attacks, and percentages are displayed.

If the user chooses to train without feature selection, a KNN classifier is trained directly on the dataset. Prediction results are visualized similarly to the feature selection case, and relevant metrics are displayed for evaluation.

Throughout the app, there are checks and warnings to guide users, such as prompting them to select a classification column before proceeding with training. The use of Matplotlib and Streamlit's `st.pyplot` enables visualizations, providing an interactive and informative experience for users exploring DDoS attack predictions.

In summary, this Streamlit app offers a comprehensive and interactive platform for users to upload, explore, and analyze DDoS attack predictions, incorporating feature selection and model training with visualization of results.

6.9 Summary

Chapter 6 presents the preliminary results of experiments conducted for wireless sensor network DDoS attack detection. The chapter begins by showcasing the outcome of feature extraction, specifically the feature importance score and the subsequent selection of important features based on a threshold. It then discusses the results of experiments with machine learning algorithms, including SVM, KNN, and ANN.

For each algorithm, the chapter provides details such as accuracy, precision, time elapsed, and additional regression metrics like MSE, MAE, and MAPE. Confusion matrices are also presented to evaluate the performance of the classifiers. A discussion section compares the results across the three classifiers, emphasizing the strengths and weaknesses of each.

An interface design section illustrates the development of a DDoS attack prediction dashboard using Streamlit. The dashboard allows users to upload a CSV dataset, visualize class distribution, choose target classification, and train a DDoS attack prediction model with or without feature selection. The interface provides an interactive and informative experience for users exploring DDoS attack predictions.

In summary, Chapter 6 provides a comprehensive overview of the experiment results, highlighting the performance of SVM, KNN, and ANN in wireless sensor network DDoS attack detection. The interface design offers a practical tool for users to interact with and analyze DDoS attack predictions based on the trained models.

CHAPTER 7

Conclusion

7.1 Introduction

In this research, the focus was on the design and implementation of a Wireless Sensor Network (WSN) DDoS attack detection using machine learning algorithms. The experiment involved several key steps, including data collection, data preparation and preprocessing, feature extraction, and training and testing using KNN (K-Nearest Neighbors), SVM (Support Vector Machine), and ANN (Artificial Neural Network) algorithms. The performance of these algorithms was evaluated using various metrics.

7.2 Objectives Achievement

The dataset used in the experiment was obtained from Kaggle and consisted of 312107 rows and 18 features. The data underwent thorough preparation and preprocessing steps, including cleaning, feature extraction using Random Forest Classifier, and splitting into training and testing sets. Feature importance analysis helped identify key features for model training.

Three machine learning algorithms, namely KNN (K-Nearest Neighbors), SVM (Support Vector Machine) and ANN (Artificial Neural Network), were employed for the predict detection of wireless sensor network DDoS attacks. Each algorithm underwent training and testing, and their performance was evaluated using metrics such as accuracy, precision, mean squared error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and time elapsed.

For preliminary result, feature extraction indicated that certain features, such as `Dest_Node_Num`, `Node_id`, `S_Node`, and `Src_Node_ID`, played a crucial role in the detection process. SVM achieved an accuracy of 93.79%, with a time elapsed of approximately 15 hours for 30 epochs. While KNN outperformed other algorithms with an accuracy of 99.63%

for 10 epochs and a significantly shorter time elapsed of around 91.05 seconds. ANN showed a relatively less stable performance with an average accuracy of 95.88% and a time elapsed of around 3059.55 seconds for 30 epochs. The evaluation metrics, including MSE, MAE, MAPE, accuracy, and precision, provided comprehensive insights into the performance of each algorithm. KNN consistently demonstrated superior performance across various metrics.

The findings of this research suggest that KNN after feature extraction is a promising algorithm for wireless sensor network DDoS attack detection, offering high accuracy and precision with relatively low computational time. The choice of algorithm should consider both performance metrics and computational efficiency, especially in real-time applications where rapid detection is crucial.

7.3 Project Constraint

The dataset used in this research, while substantial, may still have limitations in capturing diverse scenarios. Future work could explore larger datasets for enhanced model generalization. Besides, further optimization of algorithms and hyperparameter tuning could improve the overall performance of the detection system. Other than that, the current research focused on offline training and testing. Future work could involve real-time implementation and testing in a live wireless sensor network environment.

7.4 Future Work

The research contributes to the understanding of machine learning applications in wireless sensor network security and lays the foundation for further exploration and optimization. The insights gained from feature importance analysis can guide future research in refining the selection of relevant features for improved model performance.

7.5 Summary

In conclusion, the research provides valuable insights into the design and implementation of a wireless sensor network DDoS attack detection system using machine learning algorithms. The findings underscore the importance of algorithm selection and optimization for efficient and accurate detection, with KNN emerging as a promising candidate in this context.

REFERENCE

- Ageyev, D., Radivilova, T., Bondarenko, O., & Mohammed, O. (2021a). Traffic monitoring and abnormality detection methods for IOT. *2021 IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT)*.
<https://doi.org/10.1109/aict52120.2021.9628954>
- Ageyev, D., Radivilova, T., Bondarenko, O., & Mohammed, O. (2021b). Traffic monitoring and abnormality detection methods for IOT. *2021 IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT)*.
<https://doi.org/10.1109/aict52120.2021.9628954>
- Aysa, M. H., Ibrahim, A. A., & Mohammed, A. H. (2020). IOT Ddos attack detection using machine learning. *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. <https://doi.org/10.1109/ismsit50672.2020.9254703>
- Chaitanya, K., & Narayanan, S. (2023). Security and privacy in wireless sensor networks using intrusion detection models to detect DDOS and Drdos attacks: A survey. *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*. <https://doi.org/10.1109/sceecs57921.2023.10063057>
- Chandan, Kumar, S., & Sinha, S. (2022). Machine learning based robust techniques to detect ddos attacks in WSN. *2022 International Conference on Intelligent Innovations in Engineering and Technology (ICIET)*.
<https://doi.org/10.1109/iciiet55458.2022.9967543>
- Chen, R.-C., Dewi, C., Huang, S.-W., & Caraka, R. E. (2020). Selecting critical features for data classification based on machine learning methods. *Journal of Big Data*, 7(1).
<https://doi.org/10.1186/s40537-020-00327-4>
- Chen, T., Huang, H., Chen, Z., Wu, Y., & Jiang, H. (2015). A secure routing mechanism against wormhole attack in IPv6-based Wireless Sensor Networks. *2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*. <https://doi.org/10.1109/paap.2015.30>

G. Amudha, & K. Ramkumar. (2023). Protecting the WSN by Detecting and Disabling the Affected Sensor Nodes Using NN and SVM Approaches. *Mathematical Statistician and Engineering Applications*, 72(1), 74–89.

<https://doi.org/https://doi.org/10.17762/msea.v72i1.1671>

Huljanah, M., Rustam, Z., Utama, S., & Siswantining, T. (2019). Feature selection using random forest classifier for predicting prostate cancer. *IOP Conference Series: Materials Science and Engineering*, 546(5), 052031. <https://doi.org/10.1088/1757-899x/546/5/052031>

K.NIRMALA, & Dr.CH.D.V.SUBBA. (2021). SURVEY ON SECURITY OF WIRELESS SENSOR NETWORKS USING MACHINE LEARNING TECHNIQUES.

International Journal of Mechanical Engineering, 6(3), 3432–3437.

<https://doi.org/ISSN: 0974-5823>

Kaur, T., Saluja, K. K., & Sharma, A. K. (2016). DDOS attack in WSN: A Survey. 2016 *International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*. <https://doi.org/10.1109/icraie.2016.7939566>

Keerthika, M., & Shanmugapriya, D. (2022). A systematic survey on various distributed denial of service (ddos) attacks in wireless sensor networks (WSN). 2022 *IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*. <https://doi.org/10.1109/icraie56454.2022.10054309>

Kumavat, K. S., & Gomes, J. (2023). Performance evaluation of IOT-enabled WSN system with and without ddos attack. 2023 *International Conference for Advancement in Technology (ICONAT)*. <https://doi.org/10.1109/iconat57137.2023.10080467>

Liu, G., Zhao, H., Fan, F., Liu, G., Xu, Q., & Nazir, S. (2022). An enhanced intrusion detection model based on improved KNN in wsns. *Sensors*, 22(4), 1407. <https://doi.org/10.3390/s22041407>

Santhosh Kumar, B. J., & Sanketh Gowda, V. S. (2022). Detection and prevention of UDP reflection amplification attack in WSN using cumulative sum algorithm. 2022 *IEEE International Conference on Data Science and Information System (ICDSIS)*. <https://doi.org/10.1109/icdsis55133.2022.9915994>

- Seema Taranum, & Mrs. Nalina S.B. (2021). Comparison of K-Means and KNN Algorithms in Data Accumulation and Clustering in WSN. *International Research Journal of Engineering and Technology (IRJET)*, 08(10), 46–52. <https://doi.org/ISO 9001:2008>
- Sohraby, K., Minoli, D., & Znati, T. (2007). *Wireless Sensor Networks: Technology, Protocols, and applications*. John Wiley & Sons.
- Ullo, S. L., & Sinha, G. R. (2020). Advances in smart environment monitoring systems using IOT and sensors. *Sensors*, 20(11), 3113. <https://doi.org/10.3390/s20113113>
- Vimal Kumar Stephen, & Robin Rohit Vincent and Mohammed Tauqeer Ullah. (2021). SECURING AND OPTIMIZING SENSOR NETWORK USING DEEP LEARNING ALGORITHMS. *ARPJ Journal of Engineering and Applied Sciences*, 16(19), 2004–2011. <https://doi.org/ISSN 1819-6608>
- X, S. (2023, February 7). *Wireless sensor networks have potential in health care and agriculture, says study*. Tech Xplore - Technology and Engineering news. <https://techxplore.com/news/2023-02-wireless-sensor-networks-potential-health.html>

APPENDIX


1) List of Revision

1. The FYP title should be in capital letters.	Done correction for the mistake.
2. The Workflow Methodology (page 19), is not related to the project. Most probably, it is for data mining research methodology. Should put the relevant methodology.	Have done modification and correction for the correct machine learning algorithms methodology related to my project.
<ul style="list-style-type: none"> - Inadequate explanation of problem statements - How do you select the most secure protocol for objective 1? - In what kind of environment the protocol will be developed and implemented? Simulation or real-time situation? - In objective 3, why did you not evaluate the protocol's security, but rather its energy usage and latency? It should reflect your project title. - The methodology does not relate to the research that will be conducted. 	<ul style="list-style-type: none"> - Have done modification explanation for the project statements. - I will be using datasets in Kaggle to run the algorithms and check for the performance of the secure communication method in terms of accuracy detection of the attacks. - Corrections have been made for objectives. I should put more focus on the security performance of the secure communication method in terms of accuracy. - Have done modification and correction for the correct machine learning algorithms methodology related to my project.

2) Meeting Log

UMS FYP Unit : FKI Programs -Student

TEE YEW CHUN [\[Sign Out\]](#)

 No Image
TEE YEW CHUN
[My Profile](#)

[What To Do](#)
[Announcement](#)
[Inbox\(0\)](#)
[Task To Do](#)
[Search Project Title](#)
[Apply FYP!](#)
[Presentation Listing](#)

Meeting Log FYP

[FYP 2 Application](#)

FYP Ref :11282 FYP :2 Session :2023/24-SEM1 Date :03-Oct-2023 Status :ACCEPTED

Project Title:

Project Code	Title	Supervisor
7134	Secure Communication Method in Wireless Sensor Networks	SALMAH BINTI FATTAH

Student Particular:

Matrix	Name	Program
BI20110050	TEE YEW CHUN	UH6481002

*Notes: 1)click [open] to view log detail for further actions:edit, accept, reject.
2)student please make sure supervisor accepts your meeting log.

[Refresh](#) [Add Log](#)

Records: 5
Page 1 of 1

Any problem please contact me at
Jika anda menghadapi sebarang masalah sila hubungi saya di
kheau@ums.edu.my
www.ums.edu.my

3) Plagiarism Report

Feedback Studio - Google Chrome

ev.turnitin.com/app/carta/en_us/?o=2127268082&lang=en_us&u=1139298168&s=&student_user=1

tee yewchun | FYP WEEK 14

SECURE COMMUNICATION
METHOD IN WIRELESS SENSOR
NETWORKS.

TEE YEW CHUN

FACULTY OF COMPUTING AND INFORMATICS
UNIVERSITI MALAYSIA SABAH
2023

SECURE COMMUNICATION
METHOD IN WIRELESS SENSOR
NETWORKS.

Match Overview

25%

2	eprints.ums.edu.my Internet Source	1%	>
3	kalaharijournals.com Internet Source	1%	>
4	www.mdpi.com Internet Source	1%	>
5	M. Keerthika, D. Shanm... Publication	1%	>
6	Kavita S. Kumavat, Joa... Publication	1%	>
7	C. Murugesh, S. Murug... Publication	1%	>
8	link.springer.com Internet Source	1%	>
9	Murat Dener, Celil Okur,... Publication	1%	>
10	cps-vo.org Internet Source	1%	>
11	openaccess.altinbas.e... Internet Source	1%	>
12	www.arpnjournals.org Internet Source	1%	>