SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Mechanics

Journal homepage:
https://semarakilmu.com.my/journals/index.php/appl_mech/index
ISSN: 2289-7895

Secure Communication Method in Wireless Sensor Networks

Tee Yew Chun[1], Salmah Fattah[1,*], Shaliza Hayati A. Wahab[1], Nordin Saad[1], Waleed Abdelrahman Yousif Mohammed[2]

[1] Faculty of Computing and Informatics, Universiti Malaysia Sabah, Jalan UMS, Kota Kinabalu, Sabah 88400 Malaysia
[2] Faculty of Computer Studies and Information Technology, Nile University, Khartoum, Sudan

ARTICLE INFO

Article history:
Received 16 September 2025
Received in revised form 19 October 2025
Accepted 26 October 2025
Available online 8 December 2025

Keywords:
Wireless Sensor Networks (WSNs); security threats; machine learning; detection mechanism; WSN attacks

ABSTRACT

Among the security risks of Wireless sensor networks (WSNs) include unauthorized access, distributed denial of service (DDoS), eavesdropping, node, capture, wormhole assault, Sybil attack. However, the current attack detection methodologies in WSNs should still be further defined. This work intends to address these issues by means of an efficient communication detection method specifically for WSN assaults, investigated and selected, applied, and thoroughly assessed against significant criteria like accuracy, precision, and processing time. Important phases of an experimental approach include data collecting, preprocessing, feature extraction, model training, testing and assessment. The WSN DDoS attacks are identified using Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Artificial Neural Network (ANN) systems. With outstanding accuracy and precision, KNN is the most successful classifier of the ones used in identifying WSN DDoS attacks. KNN specifically obtains an accuracy of 99.63% and a precision of 99.64%, hence demonstrating its great capacity in precisely detecting attacks while lowering false positives. With an average elapsed time of 91.05 seconds to manage large datasets, KNN also shows good processing. The results of the research significantly contribute to strengthening WSN security by providing insightful analysis of the relative performance of several detection techniques. KNN's superiority in terms of accuracy, precision, and computing economy over SVM and ANN emphasizes its potential for practical application in securing wireless sensor networks against a wide range of threats. These results provide vital insights on choosing ideal detection mechanisms to improve the robustness and security resilience of WSNs and underwater sensor communication networks against a dynamic environment of changing security threats, consequently leading researchers and practitioners.

1. Introduction

Specialized sensors dispersed across diverse locations comprise Wireless Sensor Networks (WSNs), monitoring environmental variables and relaying data to central hubs. As capable of measuring parameters like temperature, noise, pollution levels, humidity, and wind, WSNs find

* *Corresponding author.*
E-mail address: salmahf@ums.edu.my

https://doi.org/10.37934/aram.44.1.1127

applications across various fields, including environmental monitoring, industrial automation, threat detection, and supply chain management [1]. From tracking inventory in warehouses to monitoring hospital patient health, WSNs are pivotal in facilitating data-driven decision-making and enhancing operational efficiency. However, WSNs face numerous challenges, including resource limitations and communication unreliability. Despite these obstacles, technological advancements have led to the development of new communication techniques and algorithms aimed at improving the dependability, effectiveness, and scalability of WSNs. Yet, these advancements have also heightened concerns regarding security risks, with unauthorized access, Distributed Denial of Service (DDoS) attacks, eavesdropping, and node capture posing significant threats to network integrity [2]. DDoS attacks pose a significant threat, aiming to disrupt network availability by overwhelming it with unauthorized requests or exploiting vulnerabilities to deplete system resources. Within WSNs, these attacks have the potential to disrupt regular operations, impair network performance, and compromise data integrity, underscoring the critical importance of implementing robust security measures.

Addressing the challenges of secure communication in WSNs requires careful consideration of the unique constraints and vulnerabilities inherent to these networks. Developing effective and efficient secure communication protocols necessitates innovative approaches to ensure the security and reliability of WSNs across various applications, including environmental monitoring, healthcare, industrial automation, and smart agriculture. This article outlines the significant contributions as follows:

1. This study advances classification techniques by identifying and investigating the optimal method for detecting DDoS attacks within wireless sensor networks. Through this exploration, the research illuminates innovative strategies to improve the accuracy and reliability of DDoS detection mechanisms.
2. The research introduces a tailored experimental analysis to detect DDoS attacks within wireless sensor networks. By employing state-of-the-art methodologies, the experimental design effectively captures and analyzes network behaviors, thereby facilitating the development and assessment of robust detection mechanisms.
3. This article delivers valuable insights into the performance of DDoS attack detection mechanisms within wireless sensor networks. Through the evaluation of key performance metrics such as accuracy, precision, and computational efficiency, the study provides a comprehensive understanding of the effectiveness and limitations of existing detection methods.

Despite significant advancements in DDoS detection methods for Wireless Sensor Networks (WSNs), existing research predominantly focuses on individual machine learning algorithms or hybrid models without a comprehensive comparison of multiple approaches under consistent conditions. Additionally, there is a limited exploration of the computational efficiency and scalability of these methods in real-time environments. This study addresses these gaps by evaluating and comparing the performance of KNN, SVM, and ANN in detecting DDoS attacks, emphasizing accuracy, precision, and processing time. The article comprises five sections, each fulfilling a distinct purpose. Section 2 dives into previous research to provide context and background information. Section 3 explains the methods used in the study, breaking down the process step by step. In Section 4, the findings of the study are presented and discussed in detail. Finally, Section 5 summarizes the article by summarizing the main points and suggesting areas for further exploration.

2. Related work

This section reviews several pertinent studies where researchers have proposed the application of machine learning algorithms for Wireless Sensor Network (WSN) attack detection.

[3] emphasize the critical importance of detecting Distributed Denial-of-Service (DDoS) attacks in Wireless Sensor Networks (WSNs), essential for both remote missions and future applications. Comparing system performance metrics before and after a DDoS attack [4] underscores the significance of DDoS detection and mitigation in IoT-enabled WSN systems. [5] propose a mechanism to identify and thwart DDoS attacks, specifically targeting UDP reflection amplification attacks, achieving high detection rates with minimal false positives and negatives. Addressing WSNs' security challenges, [6] introduce the Spotted Hyena Optimizer with Quantum Neural Network for DDoS Attack Classification (SHOQNN-AC) approach, enhancing detection effectiveness through a QNN classification model and SHO algorithm. Integrating pandemic modeling into WSNs [7] demonstrates that combining decision trees and random forests yields accurate attack identification. Amid the COVID-19 pandemic, [8] proposed algorithms for real-time adaptable DDoS and DRDoS attack detection in WSNs, addressing the surge in cyber threats. [9] present a hybrid approach combining machine learning with fuzzy logic systems for DDoS attack detection, achieving nearly 94% accuracy.

Investigating anomaly detection methods considering IoT device limitations, [10] find k-nearest neighbors (KNN), decision trees (CART), and random forests (RF) outperform other models. [10] focus on MAC layer attack detection using Neural Network (NN) and Support Vector Machine (SVM) techniques, with SVM showing higher precision. [11] utilize NN and SVM for detecting denial-of-service (DoS) attacks on the MAC layer, with NN outperforming SVM. Overall, while previous studies primarily compared two machine learning algorithms, this study aims to compare SVM, KNN, and ANN in terms of accuracy, precision, and processing time. In the following section, this paper will describe the methodology employed, followed by an evaluation of the proposed algorithm.

Feature extraction is a crucial step in machine learning with several key objectives. In essence, feature selection optimises the model's performance, making it more effective, interpretable, and scalable [12]. 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. 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 [13]. Besides that, [14] proposed An Enhanced Intrusion Detection Model Based on Improved KNN in WSNs. The proposed PL-AOA algorithm performs well in the benchmark function evaluation and effectively ensures the enhancement of the kNN classifier. The experimental findings demonstrate that the proposed intrusion detection model has positive effects and practical significance. By highlighting a proposed integrated blockchain and machine learning solution, [15] reviews 164 publications on WSN security to develop a lightweight security framework with two lines of defense: cyberattack detection and prevention. In line with this, [16] proposed the CyberShield Framework, which models attacks and defenses for cybersecurity in the Internet of Things (IoT). Their study highlights the vulnerabilities of IoT environments, emphasizing the need for robust authentication, encryption, network segmentation, and machine learning-driven intrusion detection systems to mitigate cyber threats effectively. The findings reinforce the importance of tailored security solutions to enhance the resilience of both IoT and WSN infrastructures against evolving attack vectors.

The study by [17] proposes a lightweight machine learning (ML) framework for detecting Denial-of-Service (DoS) attacks in Wireless Sensor Networks (WSNs). The methodology involves selecting

key features from network traffic data and evaluating computationally efficient ML models, such as Decision Trees and Random Forest, to balance detection accuracy and resource consumption. The findings highlight the framework's effectiveness in detecting DoS attacks, outperforming traditional methods in both accuracy and efficiency and its scalability for WSNs. Another study [18] explored anomaly detection in Wireless Sensor Networks (WSNs) through machine learning (ML) techniques, showing that algorithms like Decision Trees, Support Vector Machines (SVMs), and Neural Networks deliver high accuracy and low false positive rates in detecting anomalies. This underscores the effectiveness of ML in improving WSN security and reliability. However, the study also points out limitations, including the high computational complexity of some ML models, which may not align with the resource constraints of WSNs, and the dependence on high-quality training data for optimal results. While [19] proposed an ensemble-based machine learning approach for detecting cyber-attacks in Wireless Sensor Networks (WSNs), combining models like Random Forest, Gradient Boosting, and AdaBoost to enhance detection accuracy and robustness. Their findings demonstrate that the ensemble framework outperforms single-model methods, achieving higher accuracy and effectively identifying diverse attack types, such as DoS and spoofing while maintaining computational efficiency. However, the study highlights limitations, including the increased computational complexity of ensemble methods, which may challenge resource-constrained WSNs, and a lack of extensive evaluation in large-scale networks, raising scalability concerns.

3. Methodology

This section introduces the methodology employed for the experiment in detecting attacks in wireless sensor networks.

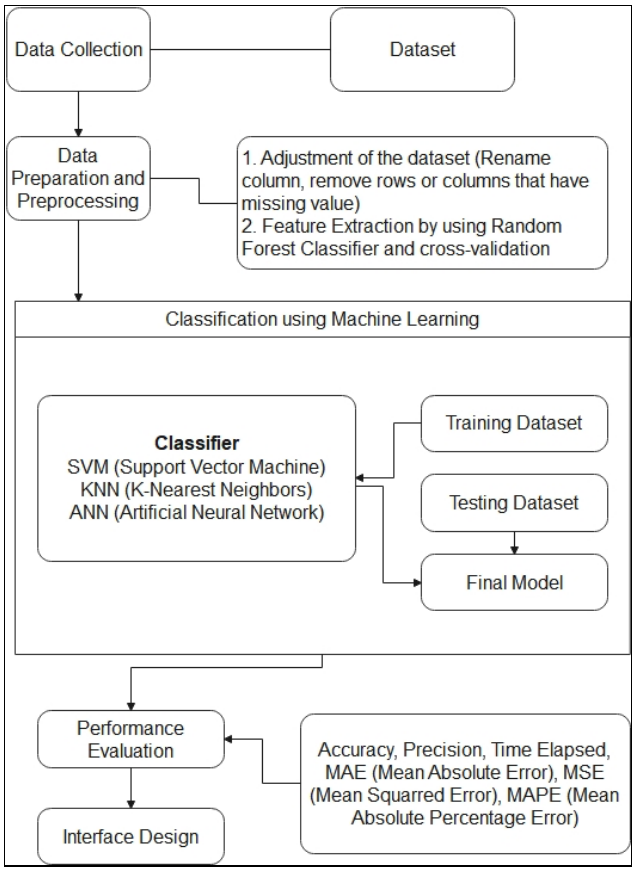

Fig. 1. Overview of the model

Several steps are necessary to achieve the objectives of this study: data collection, data preparation and preprocessing, data training and testing, detection using machine learning algorithms, and performance evaluation. In essence, this study encompasses all the processes depicted in Figure 1.

3.1 Data Preparation and Preprocessing

Following data collection, the dataset undergoes meticulous data preparation and preprocessing. Initially, data cleaning is performed, involving the removal of unnecessary columns, renaming of columns for clarity, and elimination of rows or columns containing missing values to prevent errors during experiments. Subsequently, the dataset undergoes feature extraction using a Random Forest Classifier and cross-validation to enhance data quality and compatibility. This involves instantiating a Random Forest Classifier with 100 trees (n estimators=100) and calculating the feature importance for each feature. Feature importance analysis aids in understanding the contribution of different variables to the model's predictions and guiding feature selection decisions. Feature selection is then conducted using various threshold values for feature importance scores obtained from the Random Forest Classifier. Through iterative threshold selection and model evaluation with cross-validation, the optimal threshold is identified based on the highest accuracy score, and features are selected accordingly. This iterative process achieves an optimal balance between the number of selected features and model accuracy for further analysis and training.

Subsequently, the preprocessing steps involve loading the dataset from a CSV file using the read CSV() function, extracting features and the target variable, and assigning them by selecting columns from the dataset. Label encoding is applied to the target variable using an instance of the LabelEncoder class to convert categorical values into numerical labels. Additionally, label encoding is applied to categorical features by looping through each feature specified in categorical features and encoding the corresponding columns in the feature matrix. These preprocessing steps aim to prepare the data for subsequent tasks, including splitting the dataset into training and testing sets, feature extraction, training machine learning algorithm classifiers, and evaluating model performance. Label encoding is particularly advantageous for handling categorical variables in the target variable and potentially categorical features, enabling their processing by machine learning algorithms that expect numerical inputs.

3.2 Training and testing data

The dataset is divided into a training set and a test set using the train test split function. Data splitting is a common practice aimed at preventing overfitting. The training set is utilized to train the model, allowing for the development of an optimized model. Conversely, the test set is reserved for final evaluation after model design completion. The dataset is partitioned into two subsets, with 80% allocated for the training dataset and the remaining 20% reserved for the testing dataset.

3.3 Wireless Sensor Network DDoS attack detection using Multiple Machine Learning algorithms

For detecting wireless sensor network DDoS attacks, classification was conducted using selected machine learning algorithms: SVM (Support Vector Machine), KNN (K-Nearest Neighbors), and ANN (Artificial Neural Network). These algorithms were imported from the sklearn library. Each classifier generated its own classification report, allowing for performance assessment. The classification process leveraged both the training and testing datasets. KNN is a distance-based algorithm known

for its simplicity and effectiveness in datasets with limited size. Its low computational cost makes it ideal for real-time WSN applications. SVM is effective for high-dimensional data and provides robust classification. However, its computational complexity increases with larger datasets. ANNs are powerful for modeling complex patterns and non-linear relationships, but they require extensive training data and computational resources to achieve stability.

3.4 Performance Evaluation

After the model was trained, evaluation was conducted using the testing dataset. The classification performance report included metrics such as precision, accuracy, elapsed time, and various regression metrics. These metrics served as benchmarks for evaluating the performance of each classifier within the machine learning algorithms.

3.4.1 Accuracy

Accuracy represents the degree to which a result or prediction aligns with the actual or expected value, often expressed as a percentage. In the context of machine learning and classification tasks, accuracy is defined as the ratio of correct predictions to the total number of predictions. For example (refer to Eq. (1)), if a classifier makes 90 correct predictions out of 100, its accuracy is 90%.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where TP=True Positive, TN=True Negative, FP=False Positive, FN=False Negative.

3.4.2 Precision

Precision (Eq. (2)) measures the number of relevant instances correctly identified by a classifier. It emphasizes the accuracy of positive predictions by calculating the ratio of true positives (instances of positive outcomes correctly anticipated) to the sum of true positives and false positives (occurrences of positive outcomes incorrectly predicted). Precision is particularly valuable in situations where the cost of false positives is substantial, as it assesses the reliability of positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

where TP=True Positive, FP=False Positive.

3.4.3 Time Elapsed

Time elapsed refers to the duration taken to complete a specific operation or process. It measures the duration between the initiation and completion of an operation. The elapsed time can be quantified in seconds, minutes, hours, or any other relevant unit of time.

3.4.4 Regression Metrics

Regression metrics are essential tools for evaluating the performance of models that predict continuous values including Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE). These metrics play a crucial role in determining how effectively a regression model performs and its ability to capture the inherent patterns present in the dataset.

3.4.5 Interface Design

An interface is developed using Python on PyCharm. The interface showcases the performance of the model, indicating whether it effectively detects wireless sensor network DDoS attacks.

4. Results and discussion

In this section, this paper present and discuss the results of the experiments conducted. After completing the data preparation and preprocessing steps, the study applied the selected classification methods. The analysis includes the evaluation of the overall performance.

4.1 Experiment Result of Feature Extraction

The feature importance score is computed using the 'feature importances' attribute of the trained Random Forest model, which was instantiated with 100 trees (n_estimators=100) and fitted to the training data using the fit method. This score reflects the importance of each feature in the dataset. For example, 'Dest Node Num" has a feature importance score of 0.254241, indicating that it contributes 25.4241% to the dataset. Conducting feature importance analysis is crucial for comprehending the significance of various variables in the model's predictions and can aid in guiding feature selection decisions.

Once the feature importance score is obtained, it is utilized for further analysis to select the best features. A range of threshold values is tested, and variables are established to store the optimal threshold along with its corresponding performance metrics. 'threshold_accuracy_dict' is initialized to store the mean cross-validated accuracy score calculated during this process. Upon completing the iteration through all threshold values, the script prints the thresholds and their corresponding accuracy scores, as depicted in Figure 2. Subsequently, utilizing the best threshold, the code selects features and displays the chosen ones, which include Dest_Node_Num, Node_id, S_Node, and Src_Node_ID.

4.2 Experiments results of machine learning algorithms

The classifiers selected and utilized for this study are SVM (Support Vector Machine), KNN (K-Nearest Neighbors), and ANN (Artificial Neural Network).

4.2.1 Accuracy

The accuracy is determined through the 'accuracy_score' function available in the Python for PyCharm's 'sklearn.metrics' module. This function compares the predicted labels ('y_pred') against the actual labels ('y_predtest'), ultimately computing accuracy as the ratio of correct predictions to

the total number of predictions made. Specifically, accuracy is computed for both training evaluation (Eq. (3)) and validation evaluation (Eq. (4)).

$$train\_accuracy = accuracy\_score(y\_train - train\_pred) \quad (3)$$

$$val\ accuracy = accuracy\ score(y\ test - val\ pred) \quad (4)$$

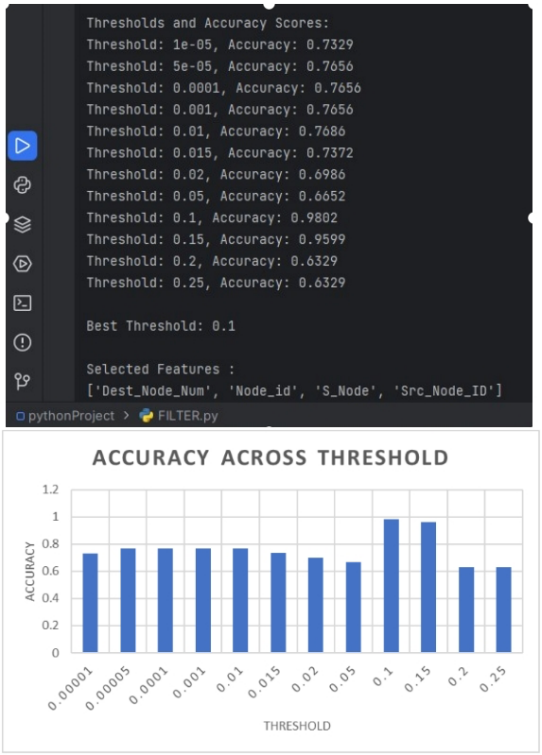

Fig. 2. Outcome important features

By comparing the predicted labels (train_pred and val_pred) with the actual training and validation labels (y_train and y_test), it computes the accuracy for both the training and validation sets. To determine the overall testing accuracy (refer to Eq. (5)):

$$accuracy = accuracy\_score(y\_test - y\_pred) \quad (5)$$

By comparing the final testing predictions (y_pred) to the actual testing labels (y_test), Eq. (5) calculates the accuracy.

4.2.2 Precision

The precision score function from the sklearn.metrics module computes the precision of the model's predictions, measuring how well the model identifies positive samples correctly by quantifying the ratio of true positives (correctly predicted positive samples) to the sum of true positives and false positives. Eq. (6) and (7) calculates precision during both the training and validation evaluations.

$$train\_precision = precision\_score(y\_train, train\_pred, average = 'weighted', zero\ division = 1) \quad (6)$$

$$val\_precision = Precision\_score\ (y\_test, val\_pred, average = 'weighted', zero\_division = 1) \quad (7)$$

To calculate precision, it compares the training predictions and validation predictions (train_pred and val_pred), respectively with the actual validation labels (y_train and y_test). It utilizes the average = 'weighted' argument to compute precision for each class and weight them according to the number of samples in each class. Eq. (8) calculates the total testing precision.

$$precision = precision\ score(y\ test, val\ pred, average = 'weighted', zero\ division = 1) \quad (8)$$

Eq. (8) 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.

Accuracy score function and precision score function return 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.

4.2.3 Time Elapsed

Additionally, the elapsed time is computed utilizing the time module from the Python standard library. Prior to commencing the training loop, the current time is acquired with time.time() and then stored in the variable start time (Eq. (9)).

$$start\_time = time.time() \quad (9)$$

Upon completion of the training loop, the current time is once again retrieved using time.time() and then stored in the variable end time (Eq. (10)).

$$\text{end\_time} = \text{time.time}() \quad (10)$$

The total time variable denotes the overall duration between the initiation and conclusion of the training loop, expressed in seconds. The total elapsed time will be presented in seconds with two decimal places (Eq. (11)).

$$total\_time = end\_time - start\_time \quad (11)$$

4.2.4 Mean Squared Error (MSE)

The function 'mean_squared_error' is employed to gauge the average squared variance between the actual and predicted values in a regression scenario. 'y_test' symbolizes the true values or ground truth from the test set, while 'y_pred' signifies the predicted values derived from the model. The outcome is preserved in the variable 'mse', which then exhibits the MSE value with four decimal places (Eq. (12)).

$$mse = mean\_squared\_error(y\_test, y\_pred) \quad (12)$$

4.2.5 Mean Absolute Error (MAE)

The metric 'mean absolute error' computes the average absolute variances between the actual and predicted values. 'y test' and 'y pred' retain their roles as in the MSE calculation. The resulting MAE is stored in the variable 'mae', which presents the MAE value with four decimal places (Eq. (13)).

$$mae = mean\_absolute\_error(y\_test, y\_pred) \quad (13)$$

4.2.6 Mean Absolute Percentage Error (MAPE)

MAPE determines the average percentage deviation between the actual and predicted values. Utilizing 'np.mean', the average of the absolute percentage errors is computed, with 'np.abs' employed to obtain the absolute values of the errors. To prevent division by zero, 'np.maximum' ensures that the denominator is at least 1. The resulting value is multiplied by 100 to express it as a percentage, then preserved in the variable 'mape', displaying the MAPE value with four decimal places (Eq. (14)).

$$mape = np.mean(np.abs(((y\_test - y\_pred)/np.maximum(y\_test, 1))) * 100 \quad (14)$$

4.2.7 Confusion Matrix

The function confusion matrix calculates the confusion matrix, assessing the classification accuracy. 'y_test' denotes the true class labels, while 'y_pred' signifies the predicted class labels from the model (Eq. (15)). These components are commonly utilized in evaluating the performance of machine learning models, offering insights into the model's prediction efficacy.

$$conf\_matrix = confusion\_matrix(y\_test, y\_pred) \quad (15)$$

4.3 Experiment Result for SVM (Support Vector Machine)

In reference to Figure 3, the confusion matrix serves to evaluate the performance of the Support Vector Machine (SVM), providing a detailed breakdown of the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by SVM. Referring to Figure 3, upon training the dataset with SVM, the results demonstrate notable stability, with a mean squared error (MSE) of 0.3362, a mean absolute error (MAE) of 0.1306, and a mean absolute percentage error

(MAPE) of 11.86%. Additionally, the average loss is 0.0621, the average validation precision is 0.9418, and the average accuracy is 0.9379.SVM exhibits the longest runtime among the algorithms tested, requiring approximately 18,793.29 seconds for 10 epochs, 36,204.89 seconds for 20 epochs, and a substantial 539,412.63 seconds for 30 epochs, which translates to around 15 hours.

Table 1
Experiment result for SVM

Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse (seconds)
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	**0.3362**	**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 (seconds)**
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 (seconds)**
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**	**53941.63**

4.4 Experiment Result for KNN (K-Nearest Neighbors)

The confusion matrix produced is used to evaluate the performance of KNN. It provides a detailed breakdown of the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by KNN, as shown in Figure 4. After training the dataset with KNN (K-Nearest Neighbors), the experimental results are more stable. The model achieves a Mean Squared Error (MSE) of 0.0155, a Mean Absolute Error (MAE) of 0.0067, and a Mean Absolute Percentage Error (MAPE) of 0.2611. Additionally, the average loss is 0.0037, the average precision is 0.9964, and the average accuracy is 0.9963. KNN only takes a short time to run, with 91.05 seconds for 10 epochs, 178.13 seconds for 20 epochs, and 264.08 seconds for 30 epochs, totaling approximately 4 minutes.

Table 2
Experiment result for KNN

Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse (seconds)
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**	**91.05**
Experiment	**Epoch**	**MSE**	**MAE**	**MAPE (%)**	**Ave_Loss**	**Ave_Precision**	**Ave_Accuracy**	**Time Elapse (seconds)**
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**	**178.13**

Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse (seconds)
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**	**264.08**

4.5 Experiment Result for ANN (Artificial Neural Network)

The efficacy of the Artificial Neural Network (ANN) is assessed by utilizing the confusion matrix generated in Figure 5. It provides a detailed breakdown of the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made by the ANN. According to Figure 5, after training the dataset with the ANN, the results are quite unstable, with an average accuracy of 0.955 and an average loss of 0.04. Notably, the ANN with 20 epochs achieves the highest precision, with a Mean Squared Error (MSE) of 0.2263, a Mean Absolute Error (MAE) of 0.0873, and a Mean Absolute Percentage Error (MAPE) of 6.9731. Additionally, the time required for training increases with the number of epochs.

Table 3
Experiment result for ANN

Experiment	Epoch	MSE	MAE	MAPE (%)	Ave_Loss	Ave_Precision	Ave_Accuracy	Time Elapse (seconds)
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	**0.2480**	**0.0948**	**7.2201**	**0.0427**	**0.9539**	**0.9573**	**1497.09**
Experiment	**Epoch**	**MSE**	**MAE**	**MAPE (%)**	**Ave_Loss**	**Ave_Precision**	**Ave_Accuracy**	**Time Elapse (seconds)**
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	**0.2263**	**0.0873**	**6.9731**	**0.0412**	**0.9559**	**0.9588**	**3059.55**
Experiment	**Epoch**	**MSE**	**MAE**	**MAPE (%)**	**Ave_Loss**	**Ave_Precision**	**Ave_Accuracy**	**Time Elapse (seconds)**
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	**0.2404**	**0.0926**	**7.5613**	**0.0413**	**0.9553**	**0.9587**	**4346.25**

4.6 Discussion

From the data presented in Figures 6, 7, 8, and 9, it is evident that among the three classifiers—KNN (K-Nearest Neighbors), SVM (Support Vector Machine), and ANN (Artificial Neural Network)—KNN with 10 epochs performs the best for wireless sensor network detection. KNN achieves the highest accuracy of 99.63% and the highest precision of 99.64%. It also has the shortest processing time of 91.05 seconds, approximately 1.52 minutes, for a large dataset. Additionally, KNN shows the lowest Mean Squared Error (MSE) at 0.0155, the lowest Mean Absolute Error (MAE) at 0.0067, and the lowest Mean Absolute Percentage Error (MAPE) at 0.26%. These metrics demonstrate that KNN outperforms the other classifiers in terms of accuracy, precision, and efficiency, making it the optimal

choice for wireless DDoS attack detection. The result is supported by [20], indicating that KNN is an effective model for intrusion detection.

Table 4
Experiment result for SVM, KNN and ANN

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

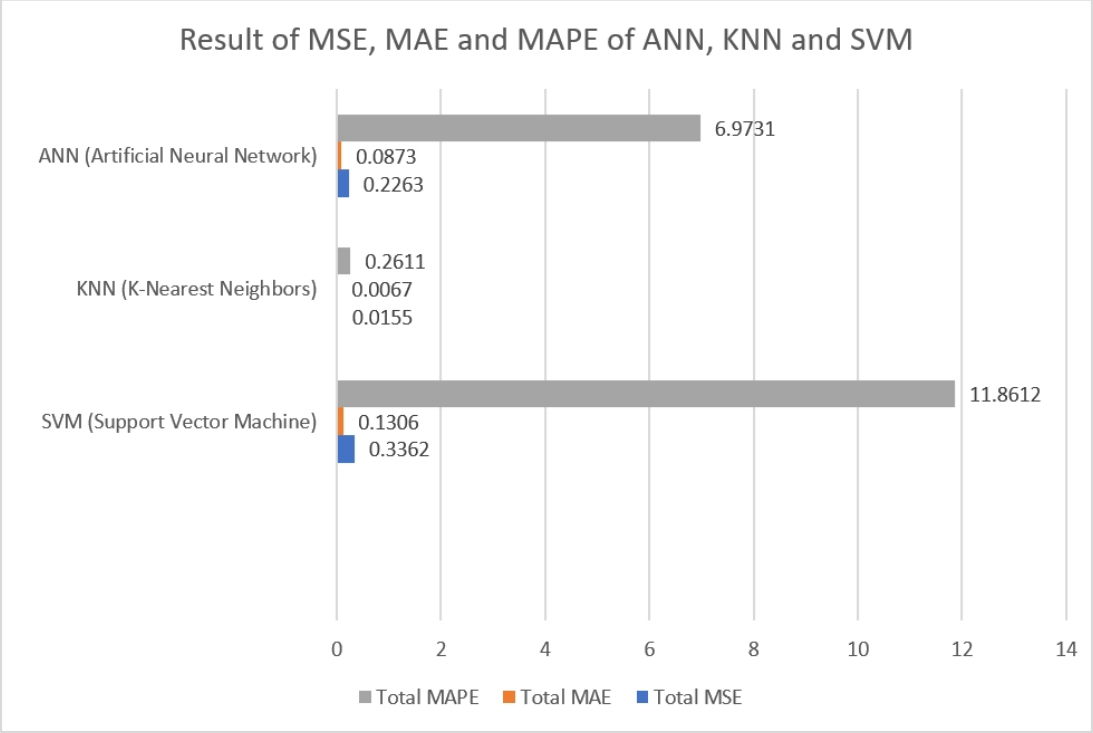

Fig. 3. Result of MSE, MAE and MAPE of ANN, KNN and SVM

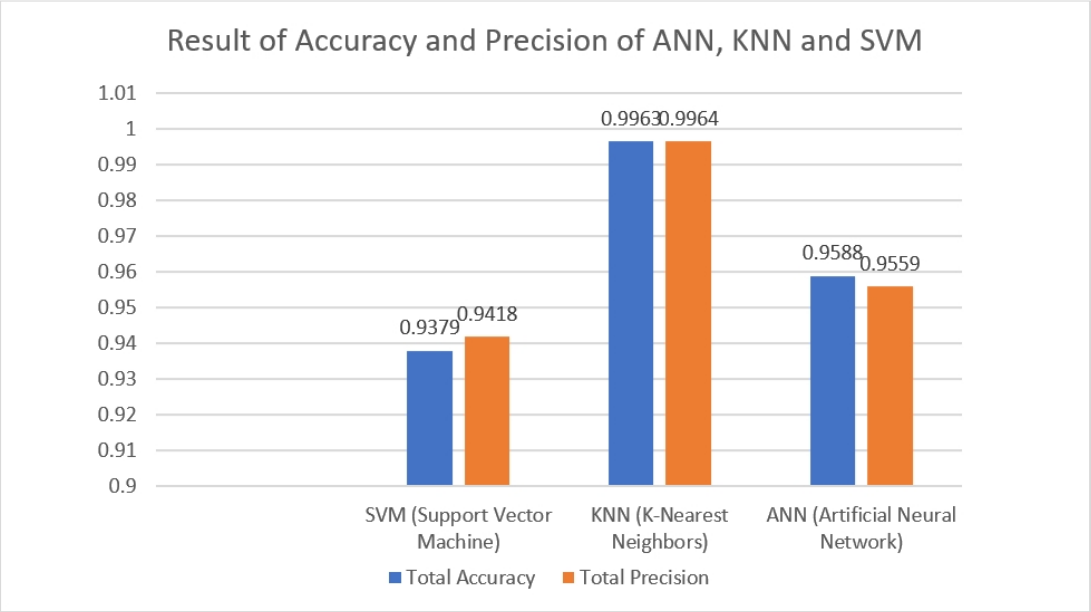

Fig. 4. Result of Accuracy and Precision of ANN, KNN and SVM

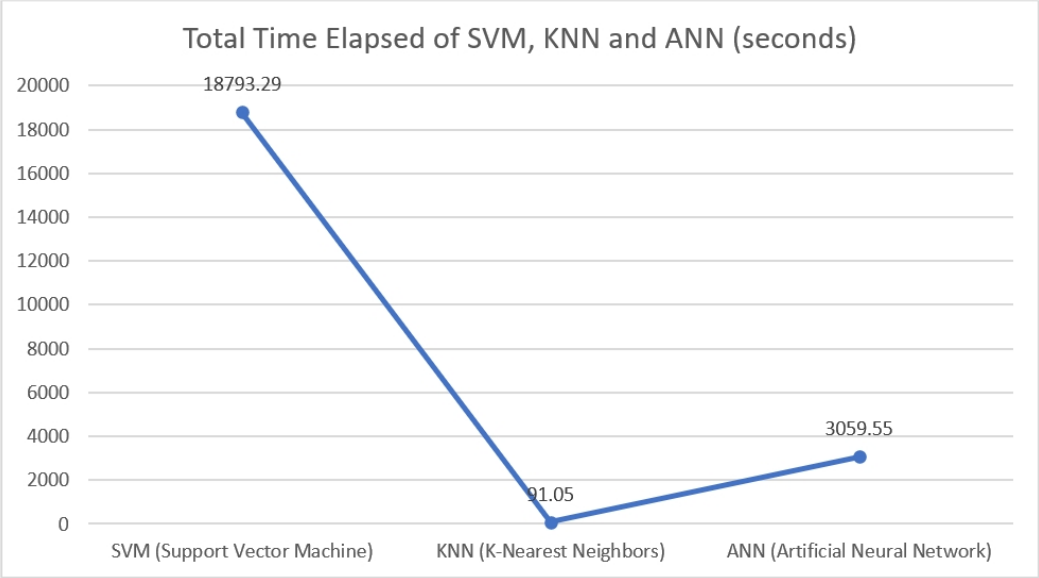

Fig. 5. Total Time Elapsed of SVM, KNN and ANN

After identifying KNN (K-Nearest Neighbors) as the best classifier among KNN, SVM (Support Vector Machine), and ANN (Artificial Neural Network), we developed an interface for users to perform Wireless Sensor Network (WSN) DDoS attack predictions using KNN. The application provides an overview of the dataset, displaying key information such as the total number of rows and features (refer to Figure 6). This initial overview helps users understand the scope and structure of their data before proceeding with the analysis. In addition to the initial dataset overview, the dashboard offers several analytical tools and visualizations to help users gain insights into potential WSN DDoS attacks. The application leverages KNN's high accuracy and efficiency, ensuring reliable predictions and quick processing times, making it a valuable tool for detecting and mitigating WSN DDoS attacks [21].

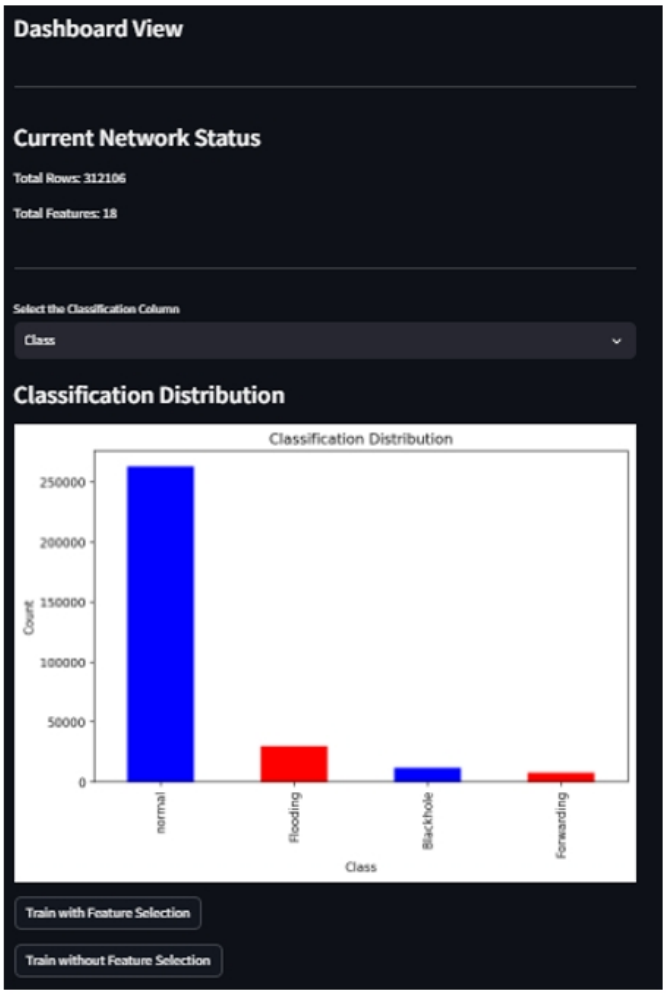

Fig. 6. WSN DDoS Attack Prediction Dashboard

The application provides users with the flexibility to train a DDoS attack prediction model either with or without feature selection. Here's a detailed breakdown of both options:

1. With Feature Selection:
   - Initial Training: A Random Forest classifier is trained on the dataset.
   - Evaluation: Cross-validation is performed to assess the model's performance.
   - Feature Selection: Important features are identified using the SelectFromModel method.
   - Subsequent Training: A K-Nearest Neighbors (KNN) classifier is then trained using these selected features.
   - Prediction Results: Results are visualized with a scatter plot.
   - Metrics Displayed: The app shows various metrics including accuracy, total true attacks, total predicted attacks, total lost attacks, and their respective percentages.
2. Without Feature Selection:
   - Direct Training: A KNN classifier is trained directly on the entire dataset without prior feature selection.
   - Prediction Results: Results are visualized in a manner like the feature selection case.
   - Feature Selection: Important features are identified using the SelectFromModel method.
   - Metrics Displayed: Relevant metrics such as accuracy, total true attacks, total predicted attacks, total lost attacks, and their percentages are displayed for evaluation.

This dual-approach functionality enhances user experience by offering: Flexibility: Users can choose the approach that best suits their data and objectives. Insightful Visualizations: Both options provide visual tools to better understand the prediction results and feature importances. Comprehensive Metrics: Displaying detailed metrics helps users evaluate the performance and reliability of the model in detecting DDoS attacks. By leveraging Streamlit's capabilities, the WSN DDoS Attack Prediction Dashboard ensures an intuitive and efficient process for users to upload datasets, train models, and analyze prediction results, thereby aiding in the effective detection of WSN DDoS attacks [22][23].

5. Conclusions

Three machine learning algorithms—KNN (K-Nearest Neighbors), SVM (Support Vector Machine), and ANN (Artificial Neural Network)—were utilized to predict and detect DDoS attacks in wireless sensor networks. Each algorithm underwent rigorous training and testing, with performance evaluated based on metrics such as accuracy, precision, mean squared error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and time elapsed. The results revealed that KNN is a highly promising algorithm for detecting DDoS attacks in wireless sensor networks, especially when feature extraction is applied. It demonstrated high accuracy and precision while maintaining relatively low computational time. Choosing the correct algorithm involves considering both performance metrics and computational efficiency, particularly in real-time applications where rapid detection is essential. This research advances the understanding of machine learning applications in wireless sensor network security and lays the groundwork for further exploration and optimization. Insights from feature importance analysis can guide future research in refining the selection of relevant features, leading to enhanced model performance. Overall, the study underscores the potential of KNN for effective DDoS attack detection, highlighting its balance of accuracy, precision, and computational efficiency, which are critical for maintaining robust security in wireless sensor networks.

This study focuses on the evaluation of three machine learning algorithms—KNN, SVM, and ANN—for detecting DDoS attacks in Wireless Sensor Networks (WSNs). While the findings demonstrate the superior performance of KNN in terms of accuracy, precision, and computational efficiency, the study is limited to a single dataset and specific types of attacks. Additionally, the algorithms have not been tested in real-world WSN deployments, which may present unique challenges such as resource constraints, dynamic network conditions, and varying attack patterns. These limitations suggest the need for further research to generalize the results and validate the proposed methods in diverse and practical settings.

Acknowledgement
This research was not funded by any grant.

References
[1] Conant, Gavin C., and Kenneth H. Wolfe. "Turning a hobby into a job: how duplicated genes find new functions." Nature Reviews Genetics 9, no. 12 (2008): 938-950. https://doi.org/10.1038/nrg2482
[2] Ohno, Susumu. Evolution by gene duplication. Springer Science & Business Media, 2013. https://doi.org/10.1007/978-3-642-86659-3
[3] Keerthika, M., and D. Shanmugapriya. "A Systematic Survey on Various Distributed Denial of Service (DDoS) Attacks in Wireless Sensor Networks (WSN)." In *2022 IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, vol. 7, pp. 59-64. IEEE, 2022. https://doi.org/10.1109/icraie56454.2022.10054309

[4] Kumavat, Kavita S., and Joanne Gomes. "Performance Evaluation of IoT-enabled WSN system With and Without DDoS Attack." In *2023 International Conference for Advancement in Technology (ICONAT)*, pp. 1-5. IEEE, 2023. https://doi.org/10.1109/iconat57137.2023.10080467

[5] Kumar, BJ Santhosh, and VS Sanketh Gowda. "Detection and prevention of UDP reflection amplification attack in WSN using cumulative sum algorithm." In *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, pp. 1-5. IEEE, 2022. https://doi.org/10.1109/icdsis55133.2022.9915994

[6] Chaitanya, Kosaraju, and Sankara Narayanan. "Security and Privacy in Wireless Sensor Networks Using Intrusion Detection Models to Detect DDOS and Drdos Attacks: A Survey." In *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pp. 1-8. IEEE, 2023. https://doi.org/10.1109/sceecs57921.2023.10063057

[7] Aysa, Mahdi Hassan, Abdullahi Abdu Ibrahim, and Alaa Hamid Mohammed. "Iot ddos attack detection using machine learning." In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1-7. IEEE, 2020. https://doi.org/10.1109/ismsit50672.2020.9254703

[8] Kumar, Sachin, and Somnath Sinha. "Machine Learning based Robust Techniques to Detect DDoS Attacks in WSN." In *2022 International Conference on Intelligent Innovations in Engineering and Technology (ICIIET)*, pp. 294-300. IEEE, 2022.

[9] Ageyev, Dmytro, Tamara Radivilova, Oleg Bondarenko, and Othman Mohammed. "Traffic Monitoring and Abnormality Detection Methods for IoT." In 2021 IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT), pp. 250-254. IEEE, 2021.

[10] Amudha, G., and K. Ramkumar. "Protecting the WSN by Detecting and Disabling the Affected Sensor Nodes Using NN and SVM Approaches." *Mathematical Statistician and Engineering Applications* 72, no. 1 (2023): 74-89.

[11] Stephen, Vimal Kumar, Robin Rohit Vincent, and Mohammed Tauqeer Ullah. "SECURING AND OPTIMIZING SENSOR NETWORK USING DEEP LEARNING ALGORITHMS." (2006).

[12] Chen, Rung-Ching, Christine Dewi, Su-Wen Huang, and Rezzy Eko Caraka. "Selecting critical features for data classification based on machine learning methods." *Journal of Big Data* 7, no. 1 (2020): 52. https://doi.org/10.1186/s40537-020-00327-4

[13] Huljanah, Mia, Zuherman Rustam, Suarsih Utama, and Titin Siswantining. "Feature selection using random forest classifier for predicting prostate cancer." In IOP Conference Series: Materials Science and Engineering, vol. 546, no. 5, p. 052031. IOP Publishing, 2019. https://doi.org/10.1088/1757-899x/546/5/052031

[14] Liu, Gaoyuan, Huiqi Zhao, Fang Fan, Gang Liu, Qiang Xu, and Shah Nazir. "An enhanced intrusion detection model based on improved kNN in WSNs." *Sensors* 22, no. 4 (2022): 1407. https://doi.org/10.3390/s22041407

[15] Ismail, Shereen, Diana W. Dawoud, and Hassan Reza. "Securing wireless sensor networks using machine learning and blockchain: A review." Future Internet 15, no. 6 (2023): 200. https://doi.org/10.3390/fi15060200

[16] Alrowaily, Mohammed, Freeh Alenezi, and Zhuo Lu. "Effectiveness of machine learning based intrusion detection systems." In *Security, Privacy, and Anonymity in Computation, Communication, and Storage: 12th International Conference, SpaCCS 2019, Atlanta, GA, USA, July 14–17, 2019, Proceedings 12*, pp. 277-288. Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-24907-6_21

[17] Rizvi, Fizza, Ravi Sharma, Nonita Sharma, Manik Rakhra, Arwa N. Aledaily, Wattana Viriyasitavat, Kusum Yadav, Gaurav Dhiman, and Amandeep Kaur. "An evolutionary KNN model for DDoS assault detection using genetic algorithm based optimization." *Multimedia Tools and Applications* (2024): 1-24.

[18] Richards, Tyler. *Getting Started with Streamlit for Data Science: Create and deploy Streamlit web applications from scratch in Python*. Packt Publishing Ltd, 2021. https://doi.org/10.1007/978-1-4842-8111-6_1

[19] Khorasani, Mohammad, Mohamed Abdou, and Javier Hernández Fernández. "Streamlit use cases." In *Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework*, pp. 309-361. Berkeley, CA: Apress, 2022. https://doi.org/10.1007/978-1-4842-8111-6_11

[20] Mahmoud Ahmed, M. A., Idris, S. A., Md Yunus, N. A., Mohd Ali, F., & Sofian, H. (2024). CyberShield Framework: Attacks and Defense Modelling for Cybersecurity in Internet of Things (IoT). *International Journal of Computational Thinking and Data Science*, *4*(1), 30–39. https://doi.org/10.37934/ctds.4.1.3039

[21] Elsadig, M. A. (2023). Detection of denial-of-service attack in wireless sensor networks: A lightweight machine learning approach. *IEEe Access*, *11*, 83537-83552. https://doi.org/10.1109/access.2023.3303113

[22] Ifzarne, S., Tabbaa, H., Hafidi, I., & Lamghari, N. (2021). Anomaly detection using machine learning techniques in wireless sensor networks. In Journal of Physics: Conference Series (Vol. 1743, No. 1, p. 012021). IOP Publishing.

[23] Ismail, S., El Mrabet, Z., & Reza, H. (2022). An ensemble-based machine learning approach for cyber-attacks detection in wireless sensor networks. Applied Sciences, 13(1), 30. https://doi.org/10.3390/app13010030