Iraqi Journal of Intelligent Computing and Informatics (IJICI)
Vol. 4, 2. December 2025, pp. 124~133
ISSN: 2791-2868, DOI: 10.52940/ijici.v4i2.113

CLASSIFICATION OF MALICIOUS NODES IN WIRELESS SENSOR NETWORKS USING MACHINE LEARNING TECHNIQUES

Dhuha Kh. Altmemi[1], Noor Habeeb[2], Hussein M. Mohammed [3]
[1,2] University of Basrah, College of Law, Department of Public Law
[3] Directorate General of Education Basrah, Ministry of Education, Basrah, Iraq

Article Info

Article history:

Received May 16, 2025
Revised Jun 28, 2025
Accepted July 20, 2025

Keywords:

Wireless Sensor Networks
Random Forest
K-Nearest Neighbors
Energy Consumption
Support Vector Machine

ABSTRACT

Wireless sensor networks (WSNs) are crucial in domains such as environmental monitoring, smart farming, and healthcare. These sensor node networks, which are dispersed over large regions, collect and transmit data to enable real-time decision-making. Contrarily, WSNs face significant security challenges, particularly from hostile nodes that can compromise data, obstruct communication, or steal confidential information. Malicious nodes significantly impact the reliability and efficiency of wireless sensor networks (WSNs). Addressing this challenge requires developing a method that accurately identifies trusted and vulnerable nodes. In this study, a new machine learning-based method is proposed to classify nodes within sensor networks by analyzing their characteristics such as energy consumption, communication behavior, and others. Machine learning algorithms can effectively detect malicious nodes. Metrics, including precision, recall, and F1 score, are used to evaluate the performance of models. Three prominent algorithms Random Forest (RF), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) are compared. Experimental results indicate that the RF algorithm achieves superior results due to its robustness and reliability in detecting malicious node behavior, while also enhancing the security and energy efficiency of WSNs.

Corresponding Author:

Dhuha Kh. Altmemi
University of Basrah, College of Law, Department of Public Law
Email: dhuha.shayal@uobasrah.edu.iq

1. INTRODUCTION

Wireless Sensor Networks (WSNs) it is are being used in many places now days such like farms and hospital and in military also even in environment things because they are good in collect data automatically and they are very helpful for people who need fast decisions to make. The sensors are be putting in many locations different from each other and them sends the dates to one place where the processing it is happen. But because they depend too much on wireless connect and they are many times in places that is remote or not stable very much, so they are get affected with many kinds of threats that is making the system not work good [1]. Because the WSNs they not strong devices and have small resource and little energy and the system not allow to use security methods normal such as encryption or pass wording because it takes too much compute and battery, and then it makes system slow and not useful. That why machine learnings becoming important to solve these kinds of problems, it looking on nodes and see if them use too much power or send too much and if something strange happening then maybe it is bad node and this technique helping so much to find it [1].

Other peoples in research they also trying different way like using model of infection in biology like SI model where people get sick and infect other one and now, they say maybe same happen in networks when malwares come in and spreading from node to other nodes and then network become all corrupted. But those models have big problem because them don't include things that is important like Media Access Control(MAC)protocol or moving nodes and power low or many attacks same time for example botnets that make chaos in network, so model is not reliable too much in real life [2].

In this research paper what authors they try to do is they use machine learning model name is HGB which is like one of advanced model that it can check nodes how they behavior and what kind of communication they doing and by this way it can know if node is bad or not and if it is bad then we can remove it from the system for protecting other nodes. Also, they use IPFS which is another system that makes the data safer because it is splitting it into pieces and hashing it and put it in many places and this also help for making security better and make sure that the bad node not change or steal the data [3].

And the traditional method that called Proof of Work (PoW) was be used before but it not give good performance and need very much power and time and this is not suitable for WSN because they have low ability, and now there is new thing called Verifiable Byzantine Fault Tolerance(VBFT) that is much better because it use voting system and it do consensus faster and without using too much resource. In the testing what done using WSN-DS dataset they find that HGB model was better than other model like AdaBoost and Gradient Boost and Ridge and Linear Discriminant Analysis (LDA) and it gave good results in different metrics such like precision and accuracy and F1 and also recall. The different was like from 2% until 16% better in some time which is showing how good HGB working [4].

And also, the VBFT showed that it is better than PoW in not only speed but also in storing data and making sure system not losing it or get hacked. It gave 20 to 30% better result in those things which is good evidence that this combination of Machine Learning(ML) and consensus make network safer and more reliable and less chance for breaking down [5]. And it's important too because rogue nodes when they are in system, they make many troubles like stop the communication and use all battery and make data wrong or delete it or break whole system in some cases and we must stop it.

Also, many ways to find those nodes before were simple ones that is based on rules like if something happen then do this, but they are not changing or learning and so not good if attack is new one. But ML model like SVM and RF and KNN they are can learning and adapting to attack that didn't happen before because they see the pattern in network and detect what is strange [6].

Some people also saying that if we watch network carefully and look at energy used and how many time node sends and if it sends weird or too much or maybe not at all, then it maybe means that the node is not good. So, combining rule system with ML system can be working very effective because ML is smart and rule is fast and together, they can make network more protected.

WSNs is used in many places where failure is dangerous like hospital or military and even small problem can be big damage if data is changed or wrong or missing. That's why it is very important to have good system to detect rogue nodes before something bad happen. And that is what this research doing. It tested 3 algorithms, RF and SVM and KNN and found that Random Forest is better one. In paper they divide into parts that is intro and related works and data collection and testing and results and the last one is conclusion that saying machine learning is good solution for WSN to make it safer and better to use.

2. RELATED WORK

The nodes in WSNs don't have much energy, like they are small and can't last long. If we don't do saving energy, they will stop working very fast. Also, the places where these nodes go are not always good, especially in army areas or far away, which make it hard. The nodes can break easy too, so they not strong at all. That's why hackers can take them over [7].

So, people tried ideas to fix it. One way is to put nodes in groups by clustering. That make the groups work better and save power more. Then network works better and live more longer. A new thing is called ESMCH, it means something like secure hybrid detection system. This adds one moving node that goes around and helps in finding attacks like when attacker watching secretly or dropping data in black hole. The ESMCH was tested and it made things better like security and energy [7].

Other things were added to WSN like IoT so nodes can talk better. In [8], they use method to make data smaller and used models like HMM and GMM, which choose the good things to train on. Their model got 92.18% accuracy on special dataset with attacks and normal data. Also, there's ICRPFDM, which use WSN with IoT and special routing protocol. This method make routing better. Even if the direct route is slow, IoT makes it faster and channel more stable [9].

Collecting data in WSN is hard because nodes are small and can't do a lot. They also can't keep big data. If bad nodes are inside, they can change the data and make everything wrong. So, the data collecting get worse. To fix this, they made SDAP. It helps by putting nodes in tree shape and choosing some nodes to mix the data from others. This is safer and faster. It also makes data move better and more secure. So, it is good option for places where fast and secure is needed [10].

Other researchers did another method using fuzzy logic, like ANFIS. It watches the node behavior and sees if it's bad or not by some trust score. It is good especially in mobile networks like MANET. Bad nodes here make big problems like using energy and breaking connection. The study showed that their

method is better than other ones like OEERP or LEACH or BCDCP. It gave good results in delivery and less energy used [11].

There is also TRS&EP, which looks at bad nodes who do bad things like not sending or changing messages. This method mixes machine learning and check things like energy and who is nearby. Then it uses a function with Gaussian name to guess who is good and who is not. It finds bad ones early and removes them so network stay fine and not crash [12].

In another study [13], they used reinforcement learning to copy the bad attacks like when nodes don't send things right. Then they bring new idea called DT-DPC which find weird nodes using voting and cluster stuff. Even if attacker is smart, DT-DPC still can catch them. They tested it and saw 4% better performance and only 1% false result and 10% missing detection, which is low.

In the paper [14], a malicious node detection strategy called FTM-ABC is proposed, which combines a fuzzy trust model (FTM) with the artificial bee colony (ABC) algorithm. The FTM calculates indirect trust, and the ABC algorithm optimizes this model to detect dishonest recommendation attacks. The fitness function, which includes recommended deviation and interaction index deviation, enhances the effectiveness of detection. Simulation results show that FTM-ABC maintains a high detection rate and low false-positive rate, even when up to 50% of nodes are dishonest. In the paper [15], the authors propose a malicious node identification strategy for WSNs based on a Time Reputation Model and Environmental Parameters Optimization (TRM-EPO). The approach calculates a comprehensive reputation that combines direct and indirect (recommended) reputations. It also constructs an environmental parameters matrix that considers factors such as node energy, data volume, number of adjacent nodes, and node sparsity. The strategy predicts the next cycle's trust based on this matrix and the comprehensive reputation data. The similarity between actual reputation and predicted trust is then compared with an adaptive threshold to identify malicious nodes. Experimental results show that TRM-EPO enhances the security and reliability of sensor nodes in complex environments, outperforming comparison algorithms. In the paper [16], a novel method for detecting malicious nodes is proposed, based on an online learning algorithm. The process begins by calculating the credibility of each path in the network using collected packets. The online instruction algorithm is then used to model the path reputation, and each node's dependability in the IoT environment is assessed. A clustering algorithm is used to identify malicious nodes. To enhance the model's performance in small-scale networks and produce a more successful detection strategy, the network's structure will be processed using the generic online learning detection technique. According to experimental results, the proposed method exhibits high stability and performance and reliably detects malicious nodes.

3. IMPORTANCE OF DETECTING MALICIOUS WSNS

WSNs are everywhere now. You'll find them in smart farms checking on plants, in hospitals helping monitor patients, and even in forests to track weather or pollution. These networks are made of tiny sensors that collect information like temperature, humidity, or light levels and send it to a central place. That central system takes all the data and helps make smart decisions. It sounds great, right? But there's a problem not all parts of the network are safe.

Sometimes, there are bad nodes and small devices in the network that are controlled by hackers or attackers. These devices look like normal sensors, but they're not. They're on the network to cause problems, and they can do a lot of different things to mess everything up.

a. Messing with the Data

A bad node can change the data that it sends or receives. So, if the real temperature is 25°C, the bad node might say it's 45°C. That can cause big problems, especially in systems where decisions are based on that data. For example, in a smart greenhouse, it could make the system think the crops are overheating and turn on the cooling system for no reason. That wastes energy and money.

b. Breaking the Routes

In WSNs, data moves from one sensor to another until it reaches the main station. But if there's a malicious node, it might stop the data from going through or send it in the wrong direction. That means the data might never reach its destination. It's like trying to deliver a letter, but someone in the middle keeps throwing it away.

c. Flooding the Network (DoS Attacks)

Some bad nodes just send too much data—more than the network can handle. This is called a Denial of Service (DoS) attack. It overloads the system, making it slow or even causing it to shut down. And since these sensors often run on batteries, all that extra work uses up energy really fast.

d. Stealing Data

WSNs often carry sensitive information. In healthcare, for example, they might carry patient heart rates or other personal data. A malicious node can read that information and send it to someone else without permission. It's like spying from the inside.

e. Pretending to Be a Trusted Node

Some malicious nodes are sneaky. They pretend to be normal, trusted parts of the network. They act helpfully and follow the rules—at first. But once they gain trust, they start causing problems. This is called an impersonation attack, and it's hard to detect because everything looks fine on the surface.

f. Teamwork Between Bad Nodes (DDoS)

Sometimes it's not just one bad node but many of them are all bad together. This is called DDoS which is a big attack. They all send lots and lots of fake messages to the network and it can't take it and then it stops working and crash.

Finding and deleting bad nodes fast is very important because they can do many bad things. Like they steal info or make data not go right or use battery fast or just break the system. Even if data is just a little wrong, it can make bad choices happen, especially in health stuff or when watching the environment.

Also, when bad nodes stay for a long time, they can join together and do a bigger attack that is harder to stop. So, we have to stop them early before they do more damage. This way, the network stays good, works longer, and the data will be better and not wrong.

4. THE PROPOSED APPROACH

This paper talks about using machine learning to find bad nodes in mobile WSNs. The idea is to check if a node is bad or not by looking at what it does, and then try to make the network safer and not break. The way they do it is by using some known algorithms like RF and KNN and SVM. These ones are used a lot and people say they are good because they can find stuff in data.

They tested the models by checking things like accuracy and recall and F1 and precision. These things help show if the model is good or bad, like if it misses a bad node or not. The goal is to make the wrong guesses as little as possible and find the bad guys correctly.

There is a figure in the paper (Figure 1) that shows all the stuff like collecting data and fixing it and testing it and how the whole thing works from start to end.

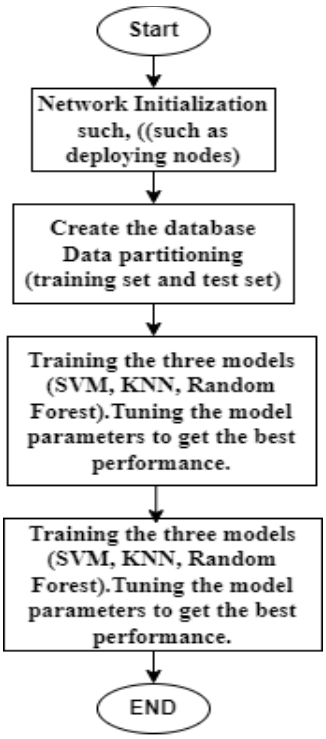

Figure1 proposed methods

a. Random Forest (RF)

RF is one of the machine learning stuffs that people use a lot. It helps in knowing what will happen later or putting things into groups. The main idea in Random Forest is to use many small trees that decide on their own. Each tree makes its own choice first.

Then, at the end, all the trees vote. If it's for classification, the group that gets the most votes wins. If it's for prediction, they just take the average of what the trees say. So, the result is made after all trees say something, and then they put the answers together.

First, important features such as energy usage and packet transmission data are chosen from the data. These features are then prepared by cleaning up any mistakes, like missing data or noise. After that, the data is divided into two parts: one part to train the model, and the other part to test it. Each tree in the forest is built using a method called bootstrap sampling, which means that the data is randomly selected with replacement. In each tree, a random selection of features is used at each step to make sure the trees are different from each other, which helps to avoid overfitting (when a model is too specific to the training data and doesn't work well with new data). The decision nodes in the trees are split using something like Gini Impurity, which measures how "impure" or mixed the data is at that point. The tree stops growing based on conditions like the maximum depth or minimum samples per leaf. After all the trees are created, the results of all the trees are combined to get the final prediction.

b. Support Vector Machine (SVM)

Another effective classification approach is SVM, which is particularly useful for complicated or non-linear data. It is often employed in network environments for tasks like traffic classification and intrusion detection. SVM attempts to identify the optimal border (referred to as a hyperplane) that divides the data into distinct groups. The distance between the various classes should be as large as feasible. Radial Basis Function (RBF) kernels and polynomial kernels are two examples of kernel functions that SVM utilizes to handle non-linear data. SVM may be used to analyze node activity in WSNs, such as energy consumption or data transmission, in order to determine whether or not a node is malevolent. SVM has a significant drawback, though, in that it is computationally costly and demands a lot of processing power, particularly when dealing with large datasets. For real-time systems, when we require prompt training and predictions, this renders it less beneficial.

c. K-Nearest Neighbours (KNN)

KNN is more simple than other ones like RF and SVM, but it still works okay for classifying networks. People used KNN for things in wireless like checking traffic or finding bad nodes. It looks at a new point and then checks what points are near it and picks the group that most of them are in.

KNN uses things like how much energy the node uses or how often it sends packets. One good thing is that it doesn't really need training like others do. So, it can change fast when the network changes or moves, like in mobile WSNs.

But there is a problem. KNN has to look at all the old points and see how far they are from the new one, and that can be very slow when there’s a lot of data. Some fixes include doing the search faster or reducing the features so it doesn’t have to check too much stuff.

5. PERFORMANCE EVALUATION

The method used for the simulation is not very strong or powerful, but it still can get the job done somehow. The computer system that runs the simulation has an Intel Core i5 processor, which is the 7th generation of this kind. This means it is not the newest or the fastest processor out there, but it can still handle some work. The CPU runs at a speed of 2.60 GHz, which is okay for normal tasks but not very fast compared to newer processors. Also, the computer has seven cores in its processor, and having multiple cores means it can do several things at the same time, which is called multitasking. Multitasking is useful especially when running simulations because the simulation might need to do many processes at once. However, if the simulation is very big or very complicated, this computer might start to slow down or take longer to finish. It was made more for regular, everyday tasks, not for heavy or complex simulations, so it can struggle if the work is too big or hard.

An interesting feature of this system is called dual boot. This means that there are two operating systems installed on the same machine, and the user can choose which one to use when they turn on the computer. The options here are Windows 10 and Windows 8. This is helpful because sometimes one operating system might work better with certain software or programs than the other one. So, if one OS doesn’t run a program properly, you can switch to the other OS and try it there. While having dual boot is useful and gives flexibility, it is not something very critical if you mostly use only one system and don’t switch often.

Talking about memory, the computer has 4 GB of RAM. This is the part of the computer that helps it run programs quickly by temporarily storing data that the CPU uses. Four gigabytes are okay for simple tasks and small simulations. But if you want to work with big datasets or run more complex simulations, 4 GB might not be enough. When the system runs out of memory, it can become slow, freeze, or stop responding. So, for heavy or large simulations, this memory size can be a bottleneck and cause problems. But if the tasks are not too demanding, 4 GB can be enough to get things done reasonably.

For the simulations themselves, the programming language used is Python. Python is very popular in scientific and research communities because it is flexible and can handle different types of data and

problems. Many researchers use Python because it is easy to learn and use, and it has many libraries that help with data analysis and simulations. Python is stable and can run experiments without much trouble. However, Python is an interpreted language and sometimes it can be slower compared to other languages like C++ or Java, especially when the tasks are very big or complicated. Still, for most common scientific tasks and simulations, Python works fine and is a good choice.

Now, the evaluation of the models is very important in this kind of work. If you don't evaluate the models properly, you won't know if they are actually working well or not. The models here are designed to find bad or malicious nodes in Wireless Sensor Networks (WSNs), so it's necessary to test them thoroughly to be sure they can do their job right. You cannot just assume the models work; you must check and verify them with proper tests. There are many different ways and metrics to check how well the models perform. If a model makes too many mistakes, it's not useful. Making wrong predictions can lead to false information and cause problems. The models should be able to find malicious nodes correctly without making errors. They need to be precise, meaning they find the right bad nodes and not mistakenly mark good nodes as bad. There are many metrics like accuracy, precision, recall, and F1-score that help measure the performance of these models. If the model is not accurate or reliable, then it is not helpful at all. So, proper evaluation is necessary to know if the model can work well in real-world situations and be trusted for practical use.

a. Accuracy

Accuracy is one of the most used performance metrics. It determines the proportion of all projections that are accurate, including both true positives and true negatives. This is how it is calculated.

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

In general, accuracy is a helpful metric, but it is insufficient in imbalanced datasets, meaning that there are many more benign nodes than dangerous nodes. In these circumstances, relying just on accuracy might give an inaccurate impression of the model's functionality.

b. True Positive Rate (TPR)

The percentage of real malicious nodes that the model accurately detects as malicious is called the True Positive Rate (TPR), often referred to as recall. In security-sensitive systems, identifying as many rogue nodes as possible is critical; a high TPR indicates that fewer harmful nodes are overlooked.

$$TPR = \frac{TP}{TP+FN} \quad (2)$$

c. False Positive Rate (FPR)

FPR measures the percentage of benign nodes that are incorrectly classified as hostile. A low FPR is essential to avoid unnecessary disruptions from real nodes being mistakenly removed from the network or misconstrued as malevolent. This is how it is calculated.

$$FPR = \frac{FP}{FP+FN} \quad (3)$$

d. False Negative Rate (FNR)

FNR displays the proportion of malicious nodes that are incorrectly classified as benign. A low FNR is necessary to ensure that no dangerous conduct is missed by the system. This is how it is calculated.

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

e. Area Under the Receiver Operating Characteristic Curve (AUC-ROC)

One way to check if a model is good for classifying stuff is AUC-ROC. It shows how well the model can tell bad nodes from good ones. They make a graph with True Positive Rate (TPR) and False Positive Rate (FPR). TPR is how many bad nodes the model finds right, and FPR is how many good nodes it says are bad by mistake. The AUC number goes from 0 to 1. If it's close to 1, the model is good. If it's near 0.5, it's not better than guessing. So higher AUC means better model. This helps to see which model is better when choosing between them.

In WSNs, it's important to check models because if they make mistakes, it can cause trouble. Like saying good nodes are bad (false positives) wastes time and resources. Or missing bad nodes (false negatives)

lets bad stuff happen. That's why people use other numbers too, like accuracy, precision, recall, and F1 score. These show how well the model works in different ways.

Using these helps researchers know if the model works well for real life. Also, network people use these to pick the best way to protect WSNs. They want the model that finds bad nodes without too many mistakes to keep the network safe and working well.

6. RESULTS

They compared three machine learning models: SVM, KNN, and Random Forest. The goal was to see which one can find bad nodes and good nodes in WSNs. Random Forest was better than the others. It was good at telling bad nodes from good ones. The study said Random Forest solved the problem better than the other two. It works well because it knows how to handle data right.

Random Forest did well because it found bad nodes with good accuracy. This is important for network security because you want to know if a node is bad or not. Another reason Random Forest is good is that it makes fewer mistakes. Some models say good nodes are bad by accident, and that causes problems for the network. But Random Forest makes fewer of these mistakes because it uses many decision trees, so it stays more correct.

The system was also good at finding bad nodes early. This matters a lot for WSNs because you want the network to be safe and working right. If you find bad nodes fast, you stop problems before they happen. When we look at SVM and KNN, they worked okay, but not as good as Random Forest. For example, SVM missed some bad nodes sometimes. It didn't catch all threats. Its accuracy was okay but not perfect.

Then there's KNN, which struggled with false positives. It sometimes flagged benign nodes as malicious, which could cause unnecessary disruptions. Although KNN did an okay job in several areas, it couldn't reconcile the issue of false positives while still performing comparably to SVM. In the end, the Random Forest model clearly outperformed the others because it offered a much more balanced and reliable solution for detecting malicious nodes, making it more effective in real-world applications. Random Forest was suitable for applications requiring comprehensive and well-rounded performance because of its notable ability to maintain equilibrium across several assessment metrics. Although Random Forest has shown adaptability in addressing the various node classification challenges, the other models tended to outperform it in certain metrics at the price of others.

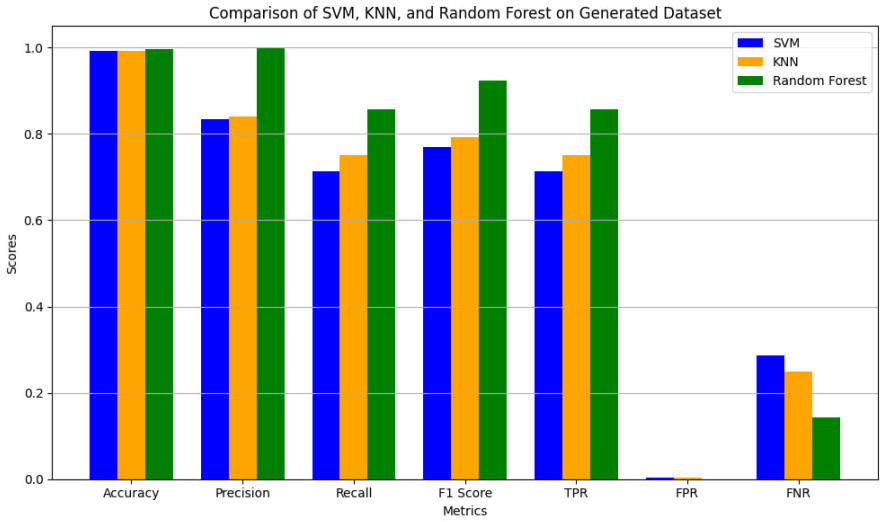

Figure 2 Results of The Measurements for The Three Models

This study emphasizes the value of cutting-edge machine learning methods, such as Random Forest, in improving WSN security and efficiency and strengthening their resistance to security risks. The measurements for the three models are displayed in Figure 2, and the proportional results of the three model measures are displayed in Figure 3. Because of its supremacy, Random Forest is the recommended option for real-world applications that need accurate node categorization in WSNs. Furthermore, the model is an effective instrument for guaranteeing network security due to its strong architecture and capacity to manage various dynamic datasets.

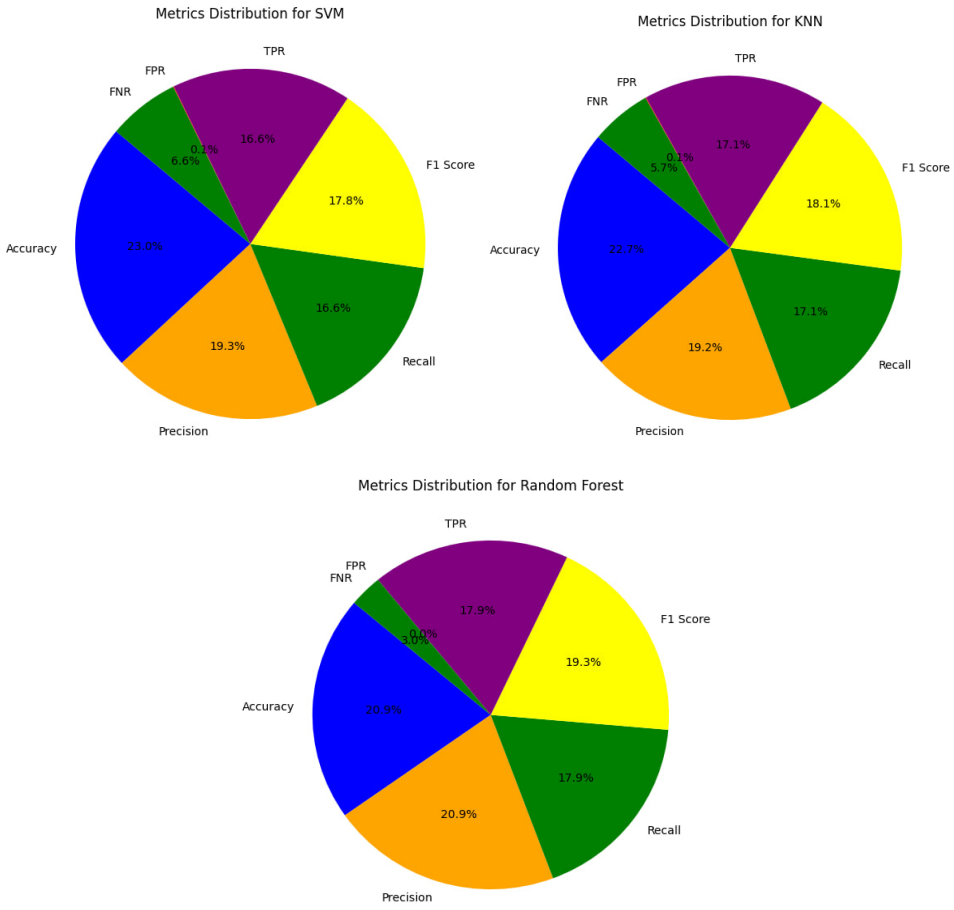

Figure 3 Results of The Three Model Measures by Proportions.

The energy consumption of machine learning models is a crucial consideration when evaluating their suitability for real-world applications in the context of WSNs. Since nodes in WSNs are usually battery-powered and positioned in isolated or challenging-to-reach areas, energy efficiency is a top priority. Models such as SVM, KNN, and Random Forest are evaluated based on their energy efficiency in addition to their detection and accuracy. Figure 4 displays each model's energy usage.

a. Energy Consumption in SVM

Support Vector Machines, especially when using a kernelized version like the Radial Basis Function (RBF) kernel, are computationally intensive. The energy consumed by SVM primarily depends on the size of the dataset and the kernel computation. Training SVM can be resource-demanding because it involves solving a quadratic optimization problem, particularly with large datasets or high-dimensional features. However, in deployment, SVM's prediction process is relatively efficient, as it only involves calculating the distance of a sample to the support vectors. Despite this, the energy consumption during training may render SVM less favorable in energy-constrained systems unless the model is trained offline and only deployed in the WSN.

b. Energy Consumption in KNN

K-Nearest Neighbors is a non-parametric model, meaning it does not build a predictive model during training but instead stores the training data for reference. While this makes KNN energy-efficient during training, its prediction process is computationally expensive. Every time a new node needs to be classified, KNN calculates the distance from the test sample to all stored samples, leading to high energy consumption during inference. This is particularly problematic in WSNs where nodes have limited processing capabilities. The larger the dataset, the higher the energy requirements for KNN's classification, making it less suitable for scenarios with frequent or real-time predictions.

c. Energy Consumption in Random Forest

Random Forest balances energy economy and computing complexity. Creating several decision trees is necessary for Random Forest training, which can be computationally demanding. However, this procedure is

usually carried out offline in environments with plenty of resources. The energy consumption of Random Forest during deployment in WSNs is significantly reduced since it uses a computationally light inference procedure that traverses a limited number of decision trees. Because of this, Random Forest uses a lot less energy during the running duration than KNN. Furthermore, robust is Random Forest's ensemble nature, which ensures accurate classifications without consuming a significant amount of resources.

Comparison and Perspectives

1. Training Phase: Although training is usually conducted offline, SVM and Random Forest often consume more energy than KNN. While Random Forest creates several trees for training, SVM enhances support vectors. KNN utilizes extremely minimal energy during training because it just needs data storage.
2. Inference Phase: KNN consumes the greatest energy as it has to calculate distances for every test instance. SVM has a comparatively low inference energy consumption and classifies data using support vectors. Random Forest consumes the least amount of energy during deployment as it just evaluates a subset of trees for predictions.
3. Scalability: Random Forest handles scalability better because more data has little impact on the energy required to traverse decision trees, whereas KNN's energy consumption rises linearly with dataset size and becomes unaffordable; SVM's energy requirements rise with the number of support vectors and kernel complexity.

Useful Consequences for WSNs

Given WSN energy constraints, Random Forest is the most practical choice for deployment. Its energy-efficient inference phase allows nodes to perform classifications without rapidly depleting their energy storage. Even though SVM saves energy when it works, its training part needs a lot of power, so it can't be used in places where there isn't much energy. KNN uses a lot of energy when it tries to classify, so it's not good for fast or many classifications in WSNs. But sometimes, if it's okay to guess randomly, KNN can be used. Random Forest uses energy better than the others, so WSNs can last longer and still work well. This means you don't have to fix things or change batteries too often. Because of this, Random Forest is the best choice for WSNs that need to save energy.

Table 1 shows the good and bad points of the three models based on how well they did.

.

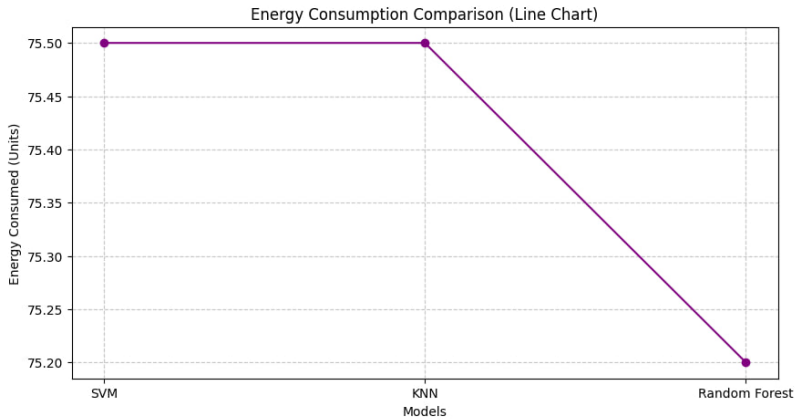

Figure 4 Energy Consumed In Each Model.

Table 1: Advantages and disadvantages of the three models.

Model	Advantages	Disadvantages
SVM	1. Effective in binary classification. 2. Performs well in high-dimensional spaces. 3. Works well with non-linear data when using kernels.	1. Long training time for large datasets. 2. Sensitive to noise and overlapping data.
KNN	1. Simple and easy to implement. 2. No training process required. 3. Performs well on small to medium datasets.	1. Affected by large datasets. 2. Requires large memory. 3. Sensitive to unbalanced data.
Random Forest	1. Robust and reliable with good performance in most cases. 2. Handles large and diverse data. 3. Reduces overfitting risk.	1. Long training time for large datasets. 2. Hard to interpret due to multiple trees.

7. CONCLUSION

This research talks about how important it is to keep the network safe and working well. It made a strong way to find bad and good nodes in WSNs using machine learning. They looked at three models: SVM,

KNN, and Random Forest. They compared them to see which one is more accurate, faster, and uses less energy. Random Forest was the best because it was good at accuracy, speed, and saving energy. The results showed that Random Forest is a good choice for WSNs that need to last long and work well. KNN used more energy, so it was not very good at saving power when classifying. SVM was okay but not the best, and it used resources badly.

This work shows that Random Forest can find bad nodes right and keep the network working. It proves that machine learning is important to make WSNs more secure. The system they made can help with future security ideas, especially for places with few resources. This could help make smart networks and industries better in the future.

REFERENCES

[1] Rabbani, Mahdi, et al. "A review on machine learning approaches for network malicious behavior detection in emerging technologies." *Entropy* 23.5 (2021): 529,doi: 10.3390/e23050529 .

[2] Kumar, Mohit, et al. "Improved deep convolutional neural network based malicious node detection and energy-efficient data transmission in wireless sensor networks." *IEEE Transactions on Network Science and Engineering* 9.5 (2021): 3272-3281,doi: 10.1109/TNSE.2021.3098011.

[3] Nwokoye, ChukwuNonso H., and V. Madhusudanan. "Epidemic models of malicious-code propagation and control in wireless sensor networks: An indepth review." *Wireless personal communications* 125.2 (2022): 1827-1856,doi: 10.1007/s11277-022-09636-8.

[4] Nouman, Muhammad, et al. "Malicious node detection using machine learning and distributed data storage using blockchain in WSNs." *IEEE Access* 11 (2023): 6106-6121,doi: 10.1109/ACCESS.2023.3236983.

[5] Kumar, K., and M. Khari. "Graph Neural Network-Based Malicious Node Detection to Improve the Security of Wireless Sensor Networks." *International Conference on Advances in Information Communication Technology & Computing*. Singapore: Springer Nature Singapore, 2024,doi: 10.1007/978-981-97-6106-7_22.

[6] Ramasamy, Lakshmana Kumar, et al. "Blockchain-based wireless sensor networks for malicious node detection: A survey." *IEEE Access* 9 (2021): 128765-128785,doi 10.1109/ACCESS.2021.3111923.

[7] Morsi, Asmaa M., Tamer M. Barakat, and Ahmed Ali Nashaat. "An efficient and secure malicious node detection model for wireless sensor networks." *International Journal of Computer Networks & Communications (IJCNC) Vol* 12 (2020),doi: 10.5121/ijcnc.2020.12107.

[8] Moundounga, Anselme Russel Affane, et al. "Malicious attack detection based on continuous Hidden Markov Models in Wireless sensor networks." *Microprocessors and Microsystems* 101 (2023): 104888,doi: 10.1016/j.micpro.2023.104888.

[9] Rani, K. Sasi Kala, and R. Vijayalakshmi. "Experimental evaluations of malicious node detection on wireless sensor network environment." *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2021,doi: 10.1109/ICICCS51141.2021.9432131.

[10] Gomathi, S., and C. Gopala Krishnan. "Malicious node detection in wireless sensor networks using an efficient secure data aggregation protocol." *Wireless Personal Communications* 113.4 (2020): 1775-1790,doi: 10.1007/s11277-020-07291-5.

[11] Subburayalu, Gopalakrishnan, et al. "Cluster-based malicious node detection system for mobile ad-hoc network using ANFIS classifier." *Journal of Applied Security Research* 18.3 (2023): 402-420,doi: 10.1080/19361610.2021.2002118.

[12] Teng, Zhijun, et al. "Malicious node identification strategy with environmental parameters." *IEEE Access* 8 (2020): 149522-149530,doi: 10.1109/ACCESS.2020.3013840.

[13] Ding, Jingze, Haozhen Wang, and Yuanming Wu. "The detection scheme against selective forwarding of smart malicious nodes with reinforcement learning in wireless sensor networks." *IEEE Sensors Journal* 22.13 (2022): 13696-13706,doi: 10.1109/JSEN.2022.3176462.

[14] Pang, Baohe, et al. "A malicious node detection strategy based on fuzzy trust model and the ABC algorithm in wireless sensor network." *IEEE wireless communications letters* 10.8 (2021): 1613-1617,doi: 10.1109/LWC.2021.3070630.

[15] Teng, Zhijun, et al. "A malicious node identification strategy with environmental parameters optimization in wireless sensor network." *Wireless Personal Communications* 117 (2021): 1143-1162,doi: 10.1109/ACCESS.2020.3013840.

[16] Li, Bohan, et al. "A detection mechanism on malicious nodes in IoT." *Computer Communications* 151 (2020): 51-59,doi: 10.1016/j.comcom.2019.12.037.

[17] Khaleefah, Alaa Dhahi, and Haider M. Al-Mashhadi. "Detection of iot botnet cyber attacks using machine learning." *Informatica* 47.6 (2023),doi: 10.31449/inf.v47i6.4668.

[18] Harahsheh, Khawlah M., and Chung-Hao Chen. "A survey of using machine learning in IoT security and the challenges faced by researchers." *Informatica* 47.6 (2023),doi: 10.31449/inf.v47i6.4635.

[19] Mohammed, Hanan Abbas, and Idress Mohammed Husien. "A Deep Transfer Learning Framework for Robust IoT Attack Detection." *Informatica* 48.12 (2024),doi: 10.31449/inf.v48i12.5955.

[20] Al-Shareeda, Mahmood A., et al. "Sadetection: Security mechanisms to detect slaac attack in ipv6 link-local network." *Informatica* 46.9 (2023),doi: 10.31449/inf.v46i9.4441.

[21] Jaint, Bhavnesh, et al. "Malicious node detection in wireless sensor networks using support vector machine." *2019 3rd international conference on recent developments in control, automation & power engineering (RDCAPE)*. IEEE, 2019,doi: 10.1109/RDCAPE47089.2019.8979125.

[22] Liu, Gaoyuan, et al. "An enhanced intrusion detection model based on improved kNN in WSNs." *Sensors* 22.4 (2022): 1407,doi: 10.3390/s22041407.

[23] Indira, K., and U. Sakthi. "A hybrid intrusion detection system for sdwsn using random forest (RF) machine learning approach." *International Journal of Advanced Computer Science and Applications* 11.2 (2020),doi: 10.14569/IJACSA.2020.0110236,