# Predictive Security Measures for Industrial IoT: Machine Learning-Based Malware Detection

S. Krishna Reddy[1*], G. Prajwal[1], Ch. Venkanna Babu[1], A. Abhayanjaneyulu[1], Md. Amer[1]

[1]Department of CSE, Sree Dattha Institute of Engineering and Science, Sheriguda, Hyderabad, Telangana, India

Corresponding E-mail: krishnareddyjboss@sreedattha.ac.in

**Abstract**

As industries increasingly adopt Internet of Things (IoT) technologies to enhance efficiency, productivity, and automation, the security of Industrial IoT (IIoT) systems becomes a critical concern. The integration of IoT devices in industrial settings introduces new attack surfaces and vulnerabilities, making these systems attractive targets for malicious actors. Malware attacks on IIoT systems can have severe consequences, including production disruptions, data breaches, and potential damage to physical infrastructure. The challenge lies in predicting and preventing malware attacks on IIoT systems effectively. Traditional security systems for IIoT often rely on signature-based detection methods, which involve matching known malware signatures to identify and block threats. However, this approach is limited in its ability to detect new and previously unknown malware variants. Additionally, the static nature of signature-based systems may struggle to adapt to the dynamic and complex IIoT environments. Moreover, the unique characteristics of IIoT, such as real-time operation requirements and resource constraints, present additional challenges in implementing effective security measures without impacting system performance. Hence, this project develops an innovative and intelligent malware prediction system for IIoT. The significance of proposed model lies in its ability to provide proactive and adaptive security measures. By leveraging advanced machine learning and classification techniques, the system can analyze the behavior of devices and network traffic in real-time, identifying anomalies indicative of potential malware activity. This proactive approach allows for early detection and mitigation of threats, minimizing the risk of disruptions to industrial operations and ensuring the integrity and confidentiality of sensitive data.

**Keywords:** Internet of Things, Industrial IoT, Malware prediction, Proactive security, Deep learning.

## 1. Introduction

The adoption of Internet of Things (IoT) technologies in industries, known as Industrial IoT (IIoT), has become increasingly prevalent as organizations seek to improve efficiency, productivity, and automation. IIoT involves the integration of sensors, devices, and networks into industrial processes, enabling real-time data collection, analysis, and control. This transformative technology has the potential to revolutionize various sectors, including manufacturing, energy, healthcare, and transportation. The concept of IoT dates back to the early 2000s when Kevin Ashton, a British technologist, coined the term "Internet of Things" to describe the connectivity between physical objects and the internet. However, the practical implementation of IoT gained momentum in the following years with advancements in wireless communication, sensor technologies, and cloud computing. In India, the adoption of IoT in industrial settings has been influenced by global trends and the country's push towards digitalization and Industry 4.0 initiatives.

India's industrial sector is a significant contributor to the country's economy, accounting for a substantial share of GDP and employment. According to the India Brand Equity Foundation (IBEF), the manufacturing sector alone contributes around 16-17% to India's GDP and employs over 12% of the country's workforce. With the government's focus on initiatives such as "Make in India" and "Digital India," there has been a growing emphasis on leveraging IoT and IIoT technologies to enhance industrial competitiveness and drive economic growth. Furthermore, the Indian IoT market is witnessing robust growth, fuelled by factors such as increasing internet penetration, advancements in technology, and government initiatives to promote digitalization. According to a report by NASSCOM, the IoT market in India is expected to reach $15 billion by 2025, with the industrial sector being a significant contributor to this growth. Additionally, industries such as manufacturing, energy, and logistics are increasingly adopting IIoT solutions to improve operational efficiency, optimize resource utilization, and enable predictive maintenance. However, along with the opportunities, IIoT adoption in India also poses challenges related to cybersecurity and data privacy. As IIoT systems become more interconnected and data-driven, there is a growing need for robust security measures to protect against cyber threats and ensure the integrity and confidentiality of sensitive information.

Given the significance of IIoT in India's industrial landscape, there is a pressing need to address security concerns and develop advanced malware prediction systems tailored to the country's specific requirements. By leveraging machine learning and real-time analysis techniques, Indian industries can enhance their cybersecurity posture and mitigate the risks associated with IIoT deployments. Additionally, collaborative efforts between government agencies, industry stakeholders, and cybersecurity experts are essential to promote awareness, build resilience, and foster a secure and sustainable IIoT ecosystem in India.

## 2. Literature Survey

Mohaisen *et al.* [1] proposed an automated and behavior-based malware analysis and labeling (AMAL) system to automatically analyze and classify malware behaviors. AMAL largely consists of AutoMal, which monitors behaviors of the file system, network, and registry, and MaLabel, which classifies similar malware by family based on the monitoring of extracted behaviors. MaLabel classifies specific malware families using the machine learning techniques support vector machine (SVM), decision tree (DT), and K-nearest neighbor (KNN) algorithms. However, the AMAL proposed by Mohaisen [1] has the difficulty of manually verifying by the malware analyst in the process of selecting and labeling the representative behavior of the malware.

Galal *et al.* [2] proposed a behavior analysis method of malware that collects information from application programming interface (API) calls and parameters used by malware through an API hooking technique. It infers unique malware behaviors in the API sequence generated from the extracted API calls and parameters. Although machine learning techniques such as DT, random forest (RF), and SVM algorithms were used to classify malware based on the inferred behaviors, the method had difficulty detecting malware, because the inference of malware behaviors involved the subjective intervention of analyzers.

Phode *et al.* [3] proposed a model to predict malware in execution files by setting the file execution time to a sec unit. The behavior data used to classify and detect malware were continuous data such as the total number of processes, the maximum number of allocated process IDs, or memory usage; which were trained by a recurrent neural network (RNN) to determine the presence of malware before the malware executed the payload, thereby protecting the system from malicious attacks.

Shaid *et al.* [4] proposed a malware behavior image technique that visualizes malware by mapping a color according to the malware intensity of API calls after capturing the calls from behavior data to emphasize the malicious acts of the variant malware. When the malware intensity of API calls is higher, warmer colors are used; cooler colors represent a lower malware intensity of calls.

Trinius *et al.* [5] proposed a treemaps and thread graphs to image malware behaviors to summarize and represent a large number of behavior record reports extracted from CWSandbox. The treemaps extract data about the frequency of API calls and operations performed by malware, conducting the visualization. By contrast, the thread graph converts the behavior data, where individual thread operations of processes are sequentially listed by time into images.

Han *et al.* [6] proposed a method to create images based on the opcode sequence extracted from the execution results of static and dynamic analyses on binary files. The method can measure the similarity between variant malware by comparing the RGB pixel information between the images generated from the binary files.

Cui [7] proposed a method to quickly detect variant malicious code by visualizing the malware through image processing technology. First, after converting the binary file for malware into a gray scale image, CNN was used to automatically extract the features of the generated image. In addition, a data equalization method was applied to the malware image by applying the bat algorithm to solve the overfitting problem caused by the number of different malware families. This malware detection method showed an excellent detection speed, and the accuracy was 94.5%.

The DAIMD proposed in this paper analyzes the overall functions of malware through dynamic analysis to detect well-known IoT malware as well as new and variant IoT malware and compresses and represents feature data by visualizing a large amount of this data. It selects representative features in images through a CNN model and trains them to analyze and detect malware, thereby avoiding the need for the subjective intervention of malware analyzers.

## 4. Proposed Methodology

The Malware Prediction GUI system is designed to provide an innovative and intelligent solution for predicting malware in Industrial IoT systems. It integrates various machine learning models, data preprocessing techniques, and visualization tools to offer users a comprehensive platform for malware detection and analysis as shown in Figure 1.

Dataset Upload and Visualization:

- The system starts by allowing users to upload their datasets through a user-friendly interface. This dataset serves as the foundation for training and testing machine learning models.
- Upon upload, the dataset is displayed within the GUI, offering users transparency and ensuring they are working with the correct data.
- A count plot of class categories is presented, providing insights into the distribution of different classes within the dataset. This visualization aids users in understanding the balance or imbalance among classes, which is crucial for effective model training.

Data Preprocessing and Model Training:

- The uploaded dataset undergoes preprocessing to clean and prepare it for model training. This includes handling missing values, encoding categorical variables, and splitting the dataset into training and testing sets.
- Multiple machine learning models, including Decision Tree Classifier, Random Forest Classifier, and Deep Neural Network (DNN), are trained using the preprocessed dataset. Each model's performance metrics, such as accuracy, precision, recall, and F1-score, are computed and displayed within the GUI.

- ROC curves are plotted for each trained model, illustrating the trade-off between true positive rate and false positive rate. These curves provide users with valuable insights into the models' performance and help them make informed decisions.

Model Comparison and Prediction:

- A comparison graph is generated to visualize the performance metrics across different models. This graph allows users to easily compare the strengths and weaknesses of each model and choose the most suitable one for their specific use case.

Finally, the system enables users to make predictions on test data using the trained models. The predicted results are displayed within the GUI, enabling users to assess the models' performance in real-world scenarios and take appropriate actions.

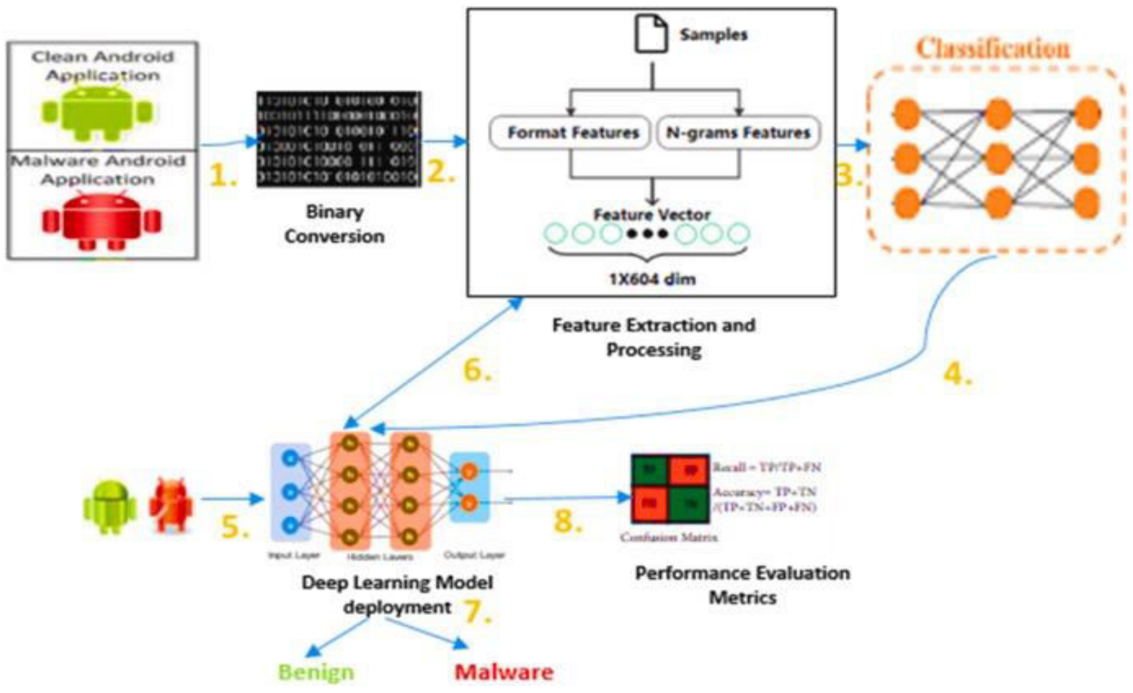
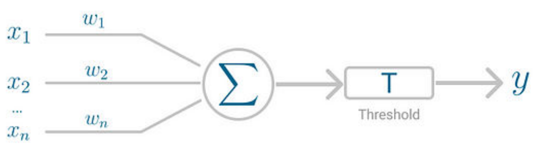

Clean Android Application
Malware Android Application
1.
Binary Conversion
2.
Samples
Format Features
N-grams Features
Feature Vector
1X604 dim
Feature Extraction and Processing
3.
Classification
4.
6.
5.
Deep Learning Model deployment
7.
Benign
Malware
8.
Recall = TP/TP+FN
Accuracy= TP+TN /(TP+TN+FP+FN)
Confusion Matrix
Performance Evaluation Metrics

Figure 1: Proposed system architecture of malware prediction.

## 3.1 ANN Classifier

Although today the Perceptron is widely recognized as an algorithm, it was initially intended as an image recognition machine. It gets its name from performing the human-like function of perception, seeing, and recognizing images. Interest has been centered on the idea of a machine which would be capable of conceptualizing inputs impinging directly from the physical environment of light, sound, temperature, etc. — the “phenomenal world” with which we are all familiar — rather than requiring the intervention of a human agent to digest and code the necessary information. Rosenblatt’s perceptron machine relied on a basic unit of computation, the neuron. Just like in previous models, each neuron has a cell that receives a series of pairs of inputs and weights. The major difference in Rosenblatt’s model is that inputs are combined in a weighted sum and, if the weighted sum exceeds a predefined threshold, the neuron fires and produces an output.



$$y = \begin{cases} 1, \text{if } \overbrace{\sum_i w_i x_i}^{\text{weighted sum}} - \underbrace{T}_{\text{threshold}} > 0 \\ 0, \text{otherwise} \end{cases}$$

Figure 2: Perceptron neuron model (left) and threshold logic (right).

Threshold $T$ represents the activation function. If the weighted sum of the inputs is greater than zero the neuron outputs the value 1, otherwise the output value is zero.

**Perceptron for Binary Classification**

With this discrete output, controlled by the activation function, the perceptron can be used as a binary classification model, defining a linear decision boundary. It finds the separating hyperplane that minimizes the distance between misclassified points and the decision boundary. The perceptron loss function is defined as below:

$$\underbrace{D(w,c)}_{\text{distance}} = -\sum_{\underbrace{i \in \mathrm{M}}_{\text{misclassified observations}}} \overset{\text{output}}{y_i} \left(x_i w_i + c\right)$$

To minimize this distance, perceptron uses stochastic gradient descent (SGD) as the optimization function. If the data is linearly separable, it is guaranteed that SGD will converge in a finite number of steps. The last piece that Perceptron needs is the activation function, the function that determines if the neuron will fire or not. Initial Perceptron models used sigmoid function, and just by looking at its shape, it makes a lot of sense! The sigmoid function maps any real input to a value that is either 0 or 1 and encodes a non-linear function. The neuron can receive negative numbers as input, and it will still be able to produce an output that is either 0 or 1.   The reason why ReLU became more adopted is that it allows better optimization using SGD, more efficient computation and is scale-invariant, meaning, its characteristics are not affected by the scale of the input. The neuron receives inputs and picks an initial set of weights random. These are combined in weighted sum and then ReLU, the activation function, determines the value of the output.

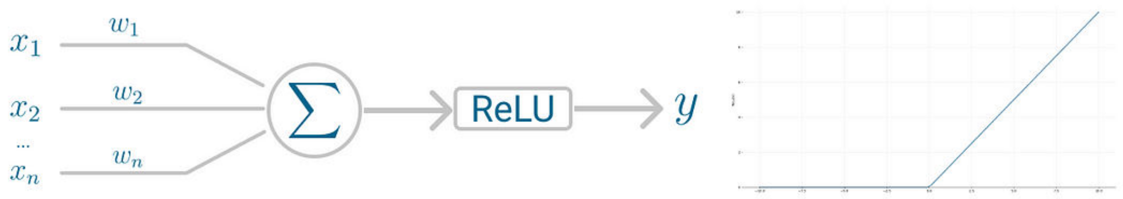


Figure 3: Perceptron neuron model (left) and activation function (right).

Perceptron uses SGD to find, or you might say learn, the set of weight that minimizes the distance between the misclassified points and the decision boundary. Once SGD converges, the dataset is separated into two regions by a linear hyperplane. Although it was said the Perceptron could represent any circuit and logic, the biggest criticism was that it couldn't represent the XOR gate, exclusive OR, where the gate only returns 1 if the inputs are different. This was proved almost a decade later and highlights the fact that Perceptron, with only one neuron, can't be applied to non-linear data.

## 4. Results and Discussion

Figure 4 displays the Receiver Operating Characteristic (ROC) curve of the Random Forest Classifier (RFC) model. The ROC curve is a graphical representation of the true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold settings. By plotting the ROC curve for the RFC model, users can assess its performance across different threshold values. A model with superior performance typically exhibits an ROC curve that is closer to the top-left corner of the plot, indicating higher sensitivity and lower false positive rate. This visualization aids users in evaluating the RFC model's discriminative ability and determining its suitability for malware prediction tasks.

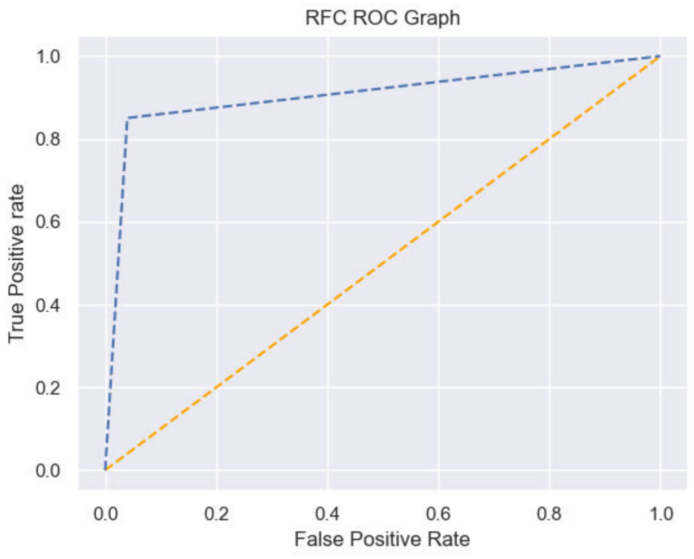


Figure 4: ROC curve of RFC model.

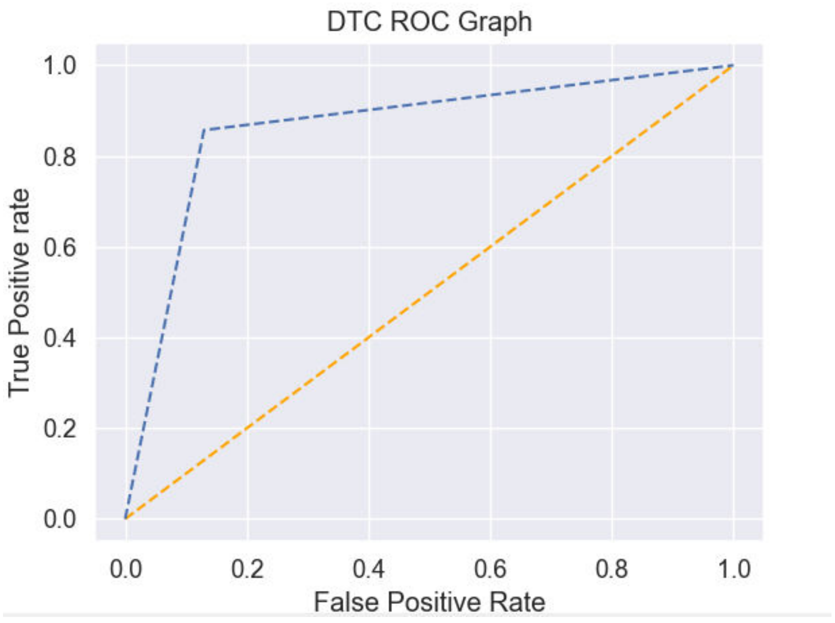


Figure 5: ROC curve of DTC model.

Figure 5 showcases the ROC curve of the Decision Tree Classifier (DTC) model. Similar to Figure 4, the ROC curve provides insights into the DTC model's performance in distinguishing between malware and non-malware instances. By analyzing the ROC curve, users can assess the model's sensitivity and specificity across different threshold settings. Comparing the ROC curves of different models, such as RFC and DTC, enables users to make informed decisions regarding model selection and deployment in real-world scenarios.

Figure 6 presents the ROC curve of the Deep Neural Network (DNN) model. As with Figures 4 and 5, the ROC curve visualizes the DNN model's performance in classifying malware and non-malware instances. By examining the ROC curve, users can evaluate the DNN model's ability to balance sensitivity and specificity, ultimately determining its efficacy in detecting malware in Industrial IoT systems. Comparing the ROC curves of various models aids users in selecting the most suitable model based on their performance requirements and constraints.

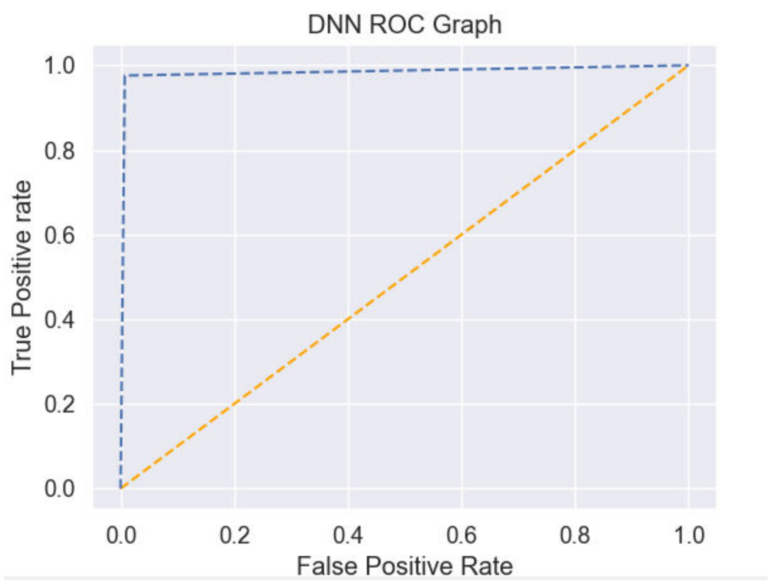


Figure 6: ROC curve of DNN model.

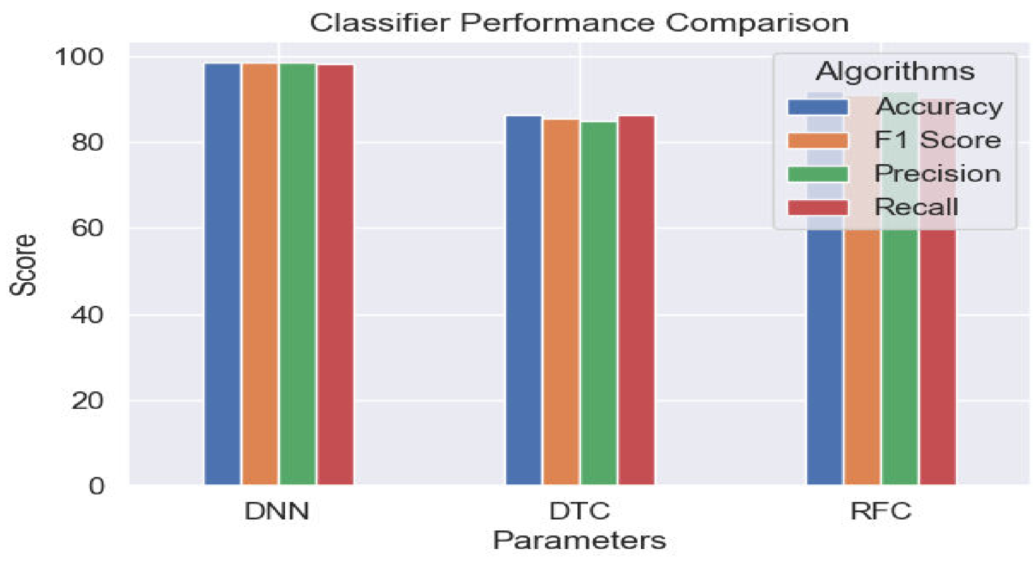


Figure 7: Presents the all-model comparison graph of performance metrices.

Figure 7 illustrates a model comparison graph depicting the performance metrics of different machine learning models. These metrics may include accuracy, precision, recall, and F1-score, among others. By presenting the performance metrics in a graphical format, users can easily compare the strengths and weaknesses of each model. This visualization facilitates data-driven decision-making by highlighting the models that excel in specific performance criteria. Users can leverage the model comparison graph to identify the most effective model for malware prediction tasks in Industrial IoT systems.

Table 1 summarizes the performance metrics of three different machine learning models: Random Forest (RF), Decision Tree, and Deep Neural Network (DNN).

- **Random Forest (RF)**: The RF model shows high precision, recall, F1-score, and accuracy, all above 99%. This indicates that the RF model performs exceptionally well in both precision and recall, meaning it makes very few false positives and false negatives, and overall, it classifies instances with high accuracy.
- **Decision Tree**: The Decision Tree model also demonstrates strong performance with precision, recall, F1-score, and accuracy values ranging around 98%. However, these metrics are slightly lower compared to the RF model, suggesting that the Decision Tree may not generalize as well as RF, possibly due to overfitting or other limitations of the decision tree algorithm.
- **Deep Neural Network (DNN)**: The DNN model exhibits performance metrics similar to the Random Forest, with precision, recall, F1-score, and accuracy around 98%. This suggests that the DNN model performs comparably to the Random Forest in terms of classification accuracy, making it a viable alternative for the given task.

Table 1: Performance metrics in a tabular form

| **Model** | **Precision** | **Recall** | **F1-Score** | **Accuracy** |
|---|---|---|---|---|
| Random Forest (RF) | 91.2988 | 89.7350 | 90.4236 | 91.3535 |
| Decision Tree | 85.1450 | 86.37382 | 85.6555 | 86.5646 |
| Deep Neural Network (DNN) | 98.6733 | 98.4302 | 98.549 | 98.6697 |

## 5. Conclusion

In conclusion, the development of an innovative and intelligent malware prediction system for Industrial IoT represents a significant step towards enhancing cybersecurity in industrial environments. By leveraging advanced machine learning and classification techniques, the proposed system offers proactive and adaptive security measures capable of preemptively detecting and mitigating malware attacks. Looking ahead, the future scope of this research includes further refinement and optimization of the malware prediction algorithms to improve detection accuracy and reduce false positives. Additionally, integrating threat intelligence feeds and collaborative defense mechanisms can enhance the system's capabilities in identifying and responding to emerging cyber threats. Furthermore, exploring the potential integration of blockchain technology for securing IIoT communications and data integrity presents an exciting avenue for future research. By leveraging blockchain's decentralized and immutable nature, it may be possible to enhance the resilience and trustworthiness of IIoT systems against malicious attacks.

## REFERENCES

[1] H. Sun, X. Wang, R. Buyya and J. Su, "CloudEyes: Cloud-based malware detection with reversible sketch for resource-constrained Internet of Things (IoT) devices", Softw. Pract. Exper., vol. 47, pp. 421-441, Mar. 2017.

[2] M. Noor, H. Abbas and W. B. Shahid, "Countering cyber threats for industrial applications: An automated approach for malware evasion detection and analysis", J. Netw. Comput. Appl., vol. 103, pp. 249-261, Feb. 2018.

[3] S. Sharmeen, S. Huda, J. H. Abawajy, W. N. Ismail and M. M. Hassan, "Malware threats and detection for industrial mobile-IoT networks", IEEE Access, vol. 6, pp. 15941-15957, 2018.

[4] O. A. Waraga, M. Bettayeb, Q. Nasir and M. A. Talib, "Design and implementation of automated IoT security testbed", Comput. Secur., vol. 88, pp. 1-17, Jan. 2020.

[5] R. Kumar, X. Zhang, R. U. Khan and A. Sharif, "Research on data mining of permission-induced risk for Android IoT devices", Appl. Sci., vol. 9, no. 2, pp. 1-22, Jan. 2019.

[6] P. K. Sharma, J. H. Park, Y.-S. Jeong and J. H. Park, "SHSec: SDN based secure smart home network architecture for Internet of Things", Mobile Netw. Appl., vol. 24, no. 3, pp. 913-924, Jun. 2019.

[7] Y.-S. Jeong and J. H. Park, "IoT and smart city technology: Challenges opportunities and solutions", J. Inf. Process. Syst., vol. 15, no. 2, pp. 233-238, Apr. 2019.

[8] T. Lei, Z. Qin, Z. Wang, Q. Li and D. Ye, "EveDroid: Event-aware Android malware detection against model degrading for IoT devices", IEEE Internet Things J., vol. 6, no. 4, pp. 6668-6680, Aug. 2019.

[9] K. Gafurov and T.-M. Chung, "Comprehensive survey on Internet of Things architecture security aspects applications related technologies economic perspective and future directions", J. Inf. Process. Syst., vol. 15, no. 4, pp. 797-819, Aug. 2019.

[10] S.-Y. Choi, C. G. Lim and Y.-M. Kim, "Automated link tracing for classification of malicious Websites in malware distribution networks", J. Inf. Process. Syst., vol. 15, no. 1, pp. 100-115, Feb. 2019.

[11] N. Y. Kim, S. Rathore, J. H. Ryu, J. H. Park and J. H. Park, "A survey on cyber physical system security for IoT: Issues challenges threats solutions", J. Inf. Process. Syst., vol. 14, no. 6, pp. 1361-1384, Dec. 2018.

[12] A. Nieto and R. Rios, "Cybersecurity profiles based on human-centric IoT devices", Hum.-Centric Comput. Inf. Sci., vol. 9, no. 1, pp. 1-23, Nov. 2019.

[13] T. A. Alghamdi, "Convolutional technique for enhancing security in wireless sensor networks against malicious nodes", Hum.-Centric Comput. Inf. Sci., vol. 9, no. 1, pp. 1-10, Oct. 2019.

[14] P. K. Sharma, J. H. Ryu, K. Y. Park, J. H. Park and J. H. Park, "Li-Fi based on security cloud framework for future IT environment", Hum.-Centric Comput. Inf. Sci., vol. 8, no. 1, pp. 1-13, Aug. 2018.