RES MILITARIS
REVUE EUROPEENNE D ETUDES EUROPEAN JOURNAL OF MILITARY STUDIES
Social Science Journal

DEEP LEARNING APPROACH FOR MULTICLASS CLASSIFICATION OF MALWARE IN INDUSTRIAL IOT

A. Manasa[1], Kashish Singh[2], P. Saikumar[2], P. Ajay Kumar[2], G. Ramakrishna Reddy[2]

[1]Assist. Professor, [2]UG Scholar, Department of Computer Science & Engineering (Data Science),

[1,2]Kommuri Pratap Reddy Institute of Technology, Ghatkesar, Hyderabad, Telangana.

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: Malware prediction, Internet of Things, Industrial IoT, Deep learning.

1. INTRODUCTION

The evolution of technology has continually reshaped the industrial landscape, with the advent of the Internet of Things (IoT) marking a significant turning point. As industries embraced IoT technologies to bolster efficiency, productivity, and automation, the concept of Industrial IoT (IIoT) emerged, promising unprecedented connectivity and control over industrial processes. However, this advancement also introduced new challenges, particularly concerning cybersecurity. Historically, industrial systems were largely isolated from external networks, relying on proprietary protocols and closed-loop architectures for operation. However, the integration of IoT devices into industrial environments necessitated connectivity with external networks, exposing these systems to a plethora of cybersecurity threats. Malicious actors quickly recognized the potential vulnerabilities inherent in IIoT systems, leading to an uptick in targeted attacks aimed at disrupting operations, stealing sensitive data, or causing physical damage.

Traditional cybersecurity measures, including firewalls and antivirus software, proved inadequate in safeguarding IIoT systems against evolving threats. Signature-based detection methods, which rely on matching known malware signatures, struggled to keep pace with the rapid proliferation of new and sophisticated malware variants. Furthermore, the static nature of these approaches hindered their ability to adapt to the dynamic and heterogeneous nature of IIoT environments.

Recognizing the pressing need for more robust and adaptive security solutions, researchers began exploring innovative approaches to malware detection and prevention in IIoT systems. Machine learning and artificial intelligence emerged as promising avenues for enhancing cybersecurity capabilities, offering the potential to analyze vast datasets and identify anomalous patterns indicative of malicious activity in real-time.

2. LITERATURE SURVEY

Various studies utilizing static, dynamic, hybrid, and memory analysis methods have been conducted to analyze how malware works and how code flows prior to its detection. Most studies have employed a static analysis method that can check the overall malware structure without executing the malware. However, static analysis has difficulty detecting obfuscated malware using packing and identifying the overall functions of malware, which are drawbacks

To solve this, studies on dynamic analysis methods have been conducted to analyze overall functions of malware and to detect obfuscated malware as well as new and variant malware by executing it [1]-[8] In addition, studies on techniques to transform feature data into images for malware detection by utilizing a large amount of feature data generated by malware have been conducted

A. Malware Detection Utilizing Dynamic Analysis Techinique

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. has the difficulty of manually verifying by the malware analyst in the process of selecting and labeling the representative behavior of the malware.

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 [9]-[14]. 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.

3. PROPOSED SYSTEM

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.

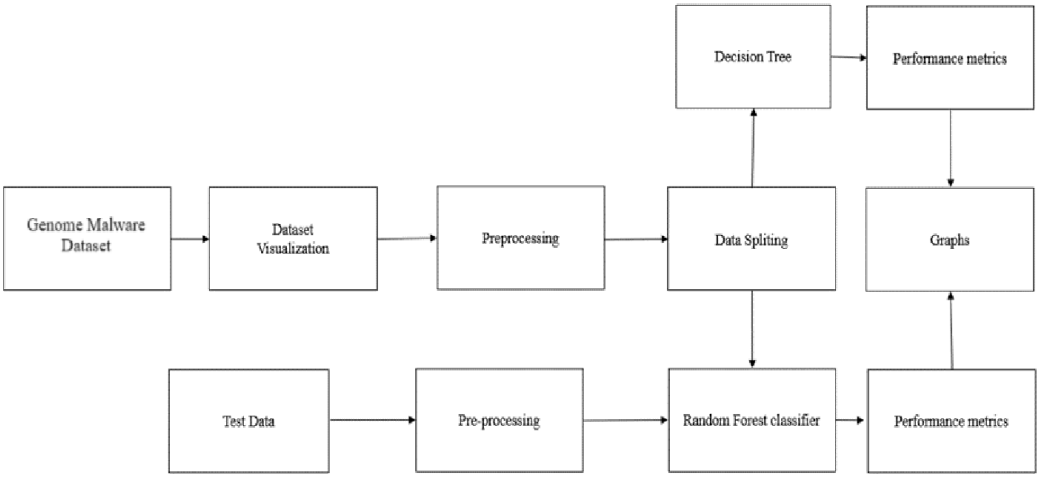

Fig 1: Block diagram of proposed diagram.

3.2 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

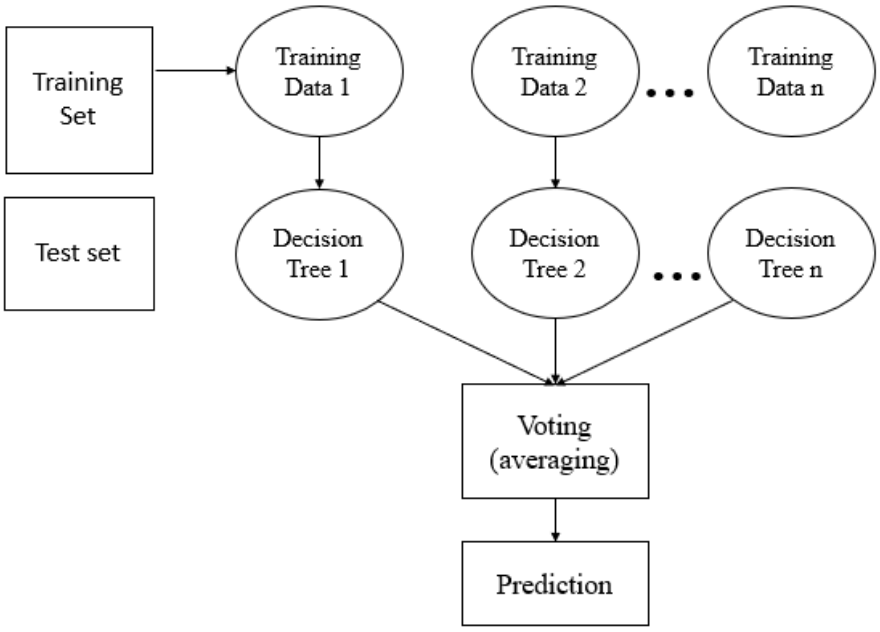

Fig. 2: Random Forest algorithm.

3.2.1 Random Forest algorithm

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

3.2.2 Important Features of Random Forest

- **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
- **Immune to the curse of dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
- **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
- **Train-Test split**- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
- **Stability**- Stability arises because the result is based on majority voting/ averaging.

4. RESULTS

Figure 3 showcases the dataset that has been uploaded into the Malware Prediction GUI. This dataset serves as the foundation for training and testing various machine learning models for malware detection in Industrial IoT systems. The dataset is displayed within the GUI, providing users with transparency and ensuring they are working with the correct data. By presenting the uploaded dataset, users can verify the data they intend to use for model training and analysis. Figure 4 presents a count plot of the class categories within the uploaded dataset. This plot offers insights into the distribution of different classes within the dataset, specifically pertaining to malware and non-malware instances. By visualizing the class distribution, users can assess the balance or imbalance among different classes. This information is crucial for understanding the dataset's characteristics and can guide decisions regarding model training strategies, such as handling class imbalance through sampling techniques or adjusting class weights during model training.

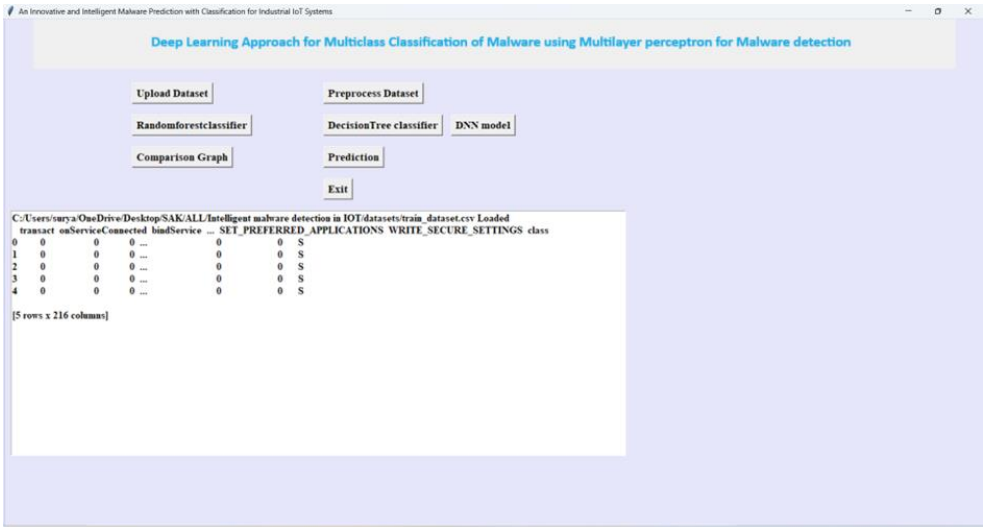

Figure 3: Presents uploaded dataset in the Malware Prediction GUI.

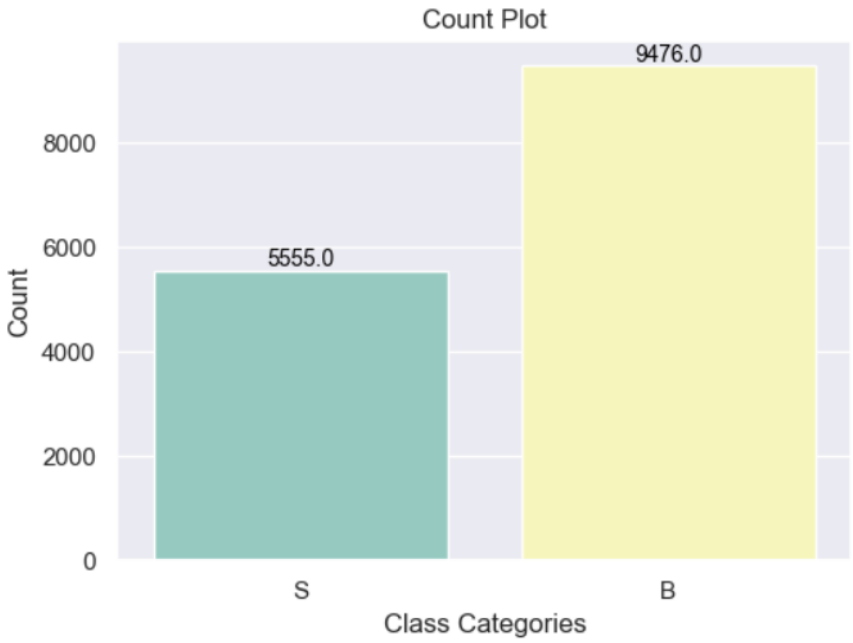

Figure 4: Displays the count plot of Class Categories.

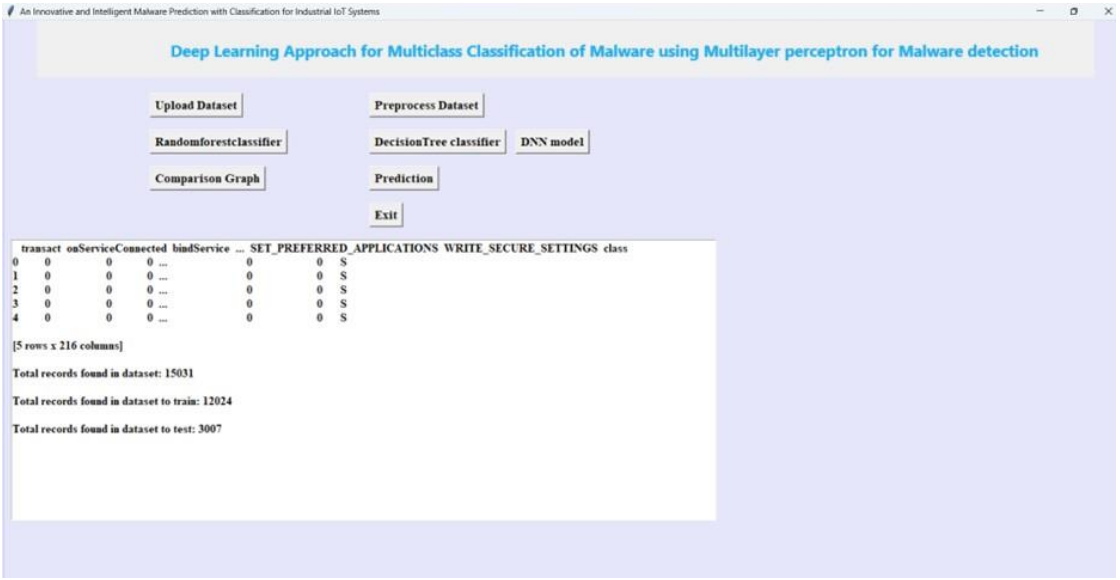

Figure 5: Presents the dataset Preprocessing.

Figure 5 illustrates the preprocessing steps applied to the uploaded dataset within the Malware Prediction GUI. Preprocessing is a critical phase in machine learning workflows, involving tasks such as handling missing values, encoding categorical variables, and splitting the dataset into training and testing sets. By presenting the dataset preprocessing, users gain visibility into the data preparation steps undertaken to ensure the dataset is suitable for model training. This transparency enhances users' understanding of the data processing pipeline and facilitates reproducibility of results. Figure 6 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.

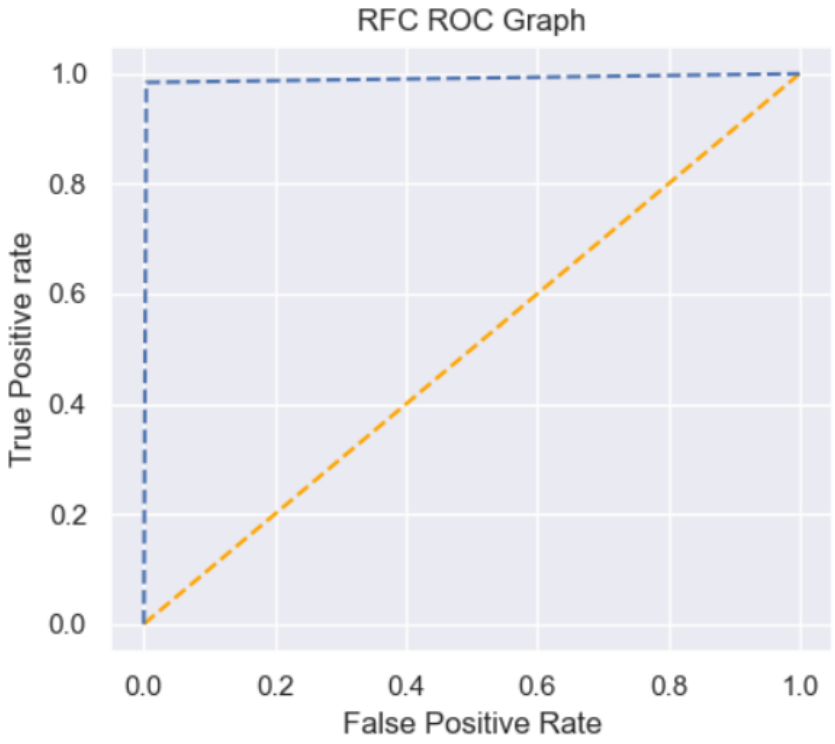

Figure 6: ROC curve of RFC model.

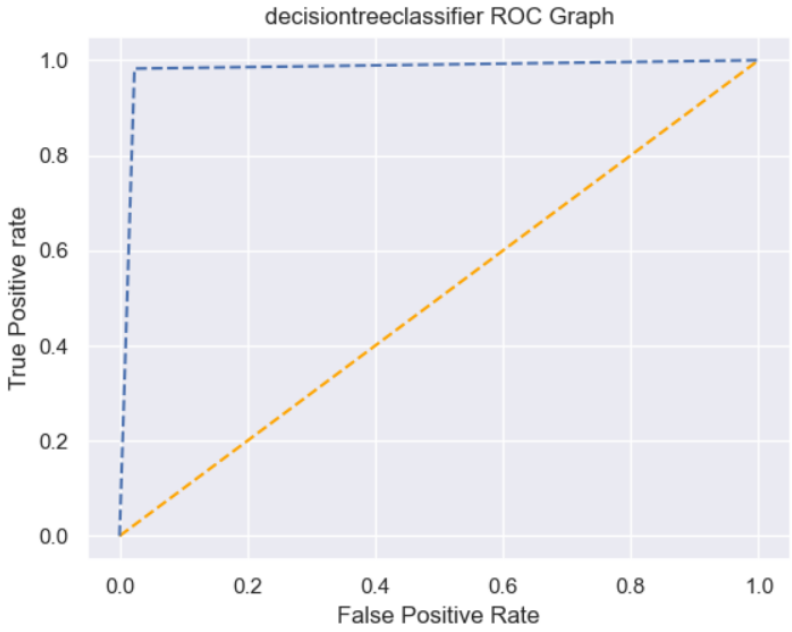

Figure 7: ROC curve of DTC model.

Figure 7 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 8 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.

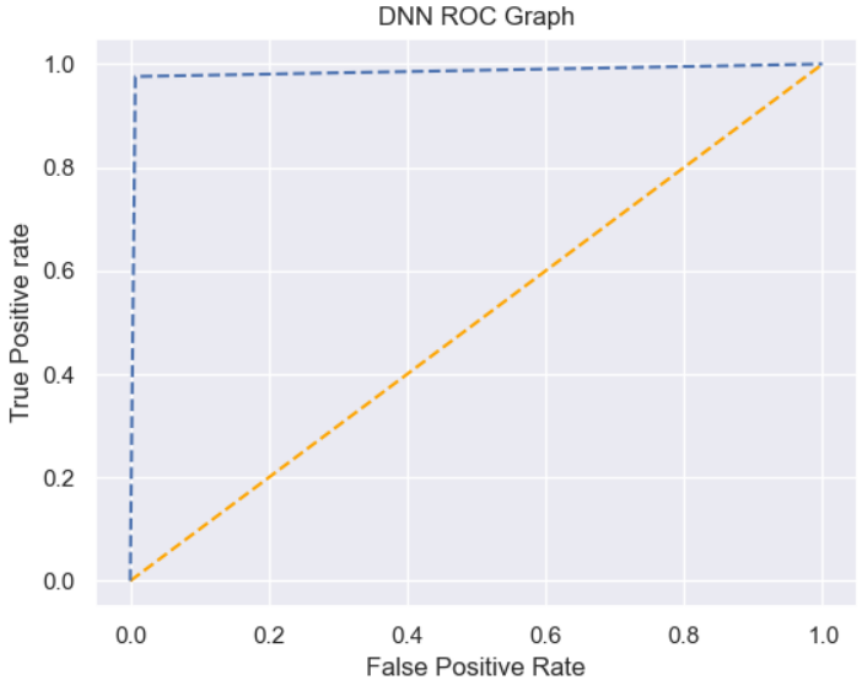

Figure 8: ROC curve of DNN model.

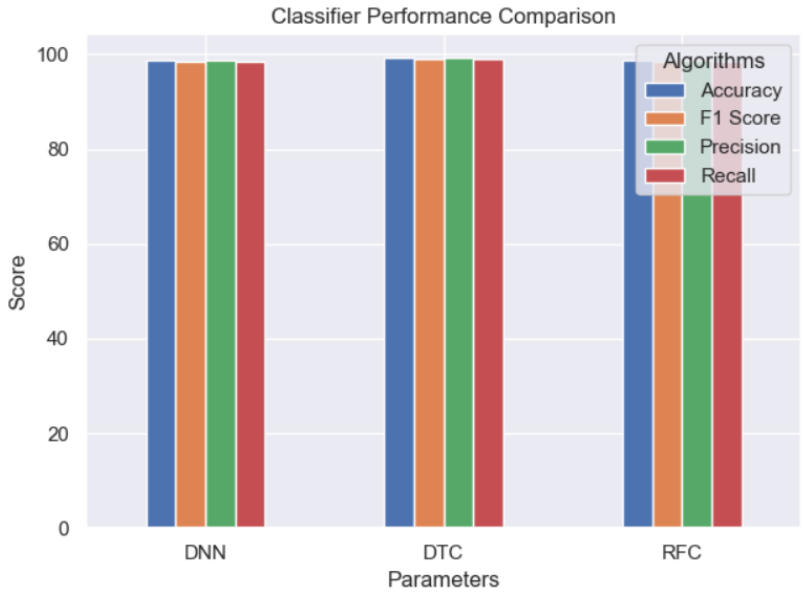

Figure 9: Presents the all-model comparison graph of performance metrices.

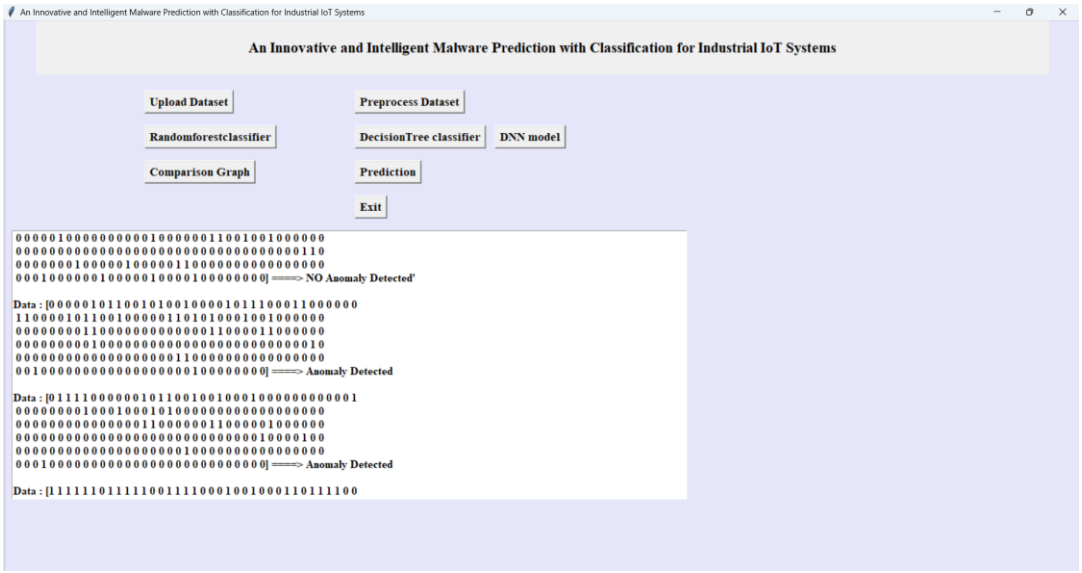

Figure 10: Displays the model prediction on test data.

Figure 9 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.

Figure 10 displays the model predictions generated by the trained machine learning models on test data. The predictions indicate whether individual instances in the test dataset are classified as malware or non-malware by the models. By presenting the model predictions within the GUI, users can assess the models' performance in real-world scenarios and validate their effectiveness in detecting malware. This visualization enables users to gain insights into the models' predictive capabilities and make informed decisions regarding their deployment in production environments.

Table 1: Performance metrics in a tabular form

Model	**Precision**	**Recall**	**F1-Score**	**Accuracy**
Random Forest (RF)	99.2338	99.0294	99.1302	99.2019
Decision Tree	97.52	98.64	98.08	98.17
Deep Neural Network (DNN)	98.6733	98.4302	98.5498	98.6698

- **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.

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]. A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques", Hum.-Centric Comput. Inf. Sci., vol. 8, no. 1, pp. 1-22, Jan. 2018.

[2] B. Yu, Y. Fang, Q. Yang, Y. Tang and L. Liu, "A survey of malware behavior description and analysis", Frontiers Inf. Technol. Electron. Eng., vol. 19, no. 5, pp. 583-603, May 2018.

[3] Z. Bazrafshan, H. Hashemi, S. M. H. Fard and A. Hamzeh, "A survey on heuristic malware detection techniques", 5th Conf. Inf. Knowl. Technol, May 2013.

[4] R. Sihwail, K. Omar and K. A. Z. Ariffin, "A survey on malware analysis techniques: Static dynamic hybrid and memory analysis", Int. J. Adv. Sci. Eng. Inf. Technol., vol. 8, no. 2, pp. 1662-1671, 2018.

[5] C.-W. Tien, J.-W. Liao, S.-C. Chang and S.-Y. Kuo, "Memory forensics using virtual machine introspection for malware analysis", IEEE Conf. Depend. Secure Comput, Aug. 2017.

[6] S. Wu, P. Wang, X. Li and Y. Zhang, "Effective detection of Android malware based on the usage of data flow APIs and machine learning", Inf. Softw. Technol., vol. 75, pp. 17-25, Jul. 2016.

[7] D. Ucci, L. Aniello and R. Baldoni, "Survey of machine learning techniques for malware analysis", Comput. Secur., vol. 81, pp. 123-147, Mar. 2019.

[8]. A. Mohaisen, O. Alrawi and M. Mohaisen, "AMAL: High-fidelity behavior-based automated malware analysis and classification", Comput. Secur., vol. 52, pp. 251-266, Jul. 2015.

[9] H. S. Galal, Y. B. Mahdy and M. A. Atiea, "Behavior-based features model for malware detection", J. Comput. Virol. Hacking Techn., vol. 12, no. 2, pp. 59-67, May 2016.

[10] M. Rhode, P. Burnap and K. Jones, "Early-stage malware prediction using recurrent neural networks", Comput. Secur., vol. 77, pp. 578-594, Aug. 2018.

[11] S. Z. M. Shaid and M. A. Maarof, "Malware behaviour visualization", J. Teknol., vol. 70, no. 5, pp. 25-33, Sep. 2014.

[12]. P. Trinius, T. Holz, J. Gobel and F. C. Freiling, "Visual analysis of malware behavior using treemaps and thread graphs", 6th Int. Workshop Vis. Cyber Secur, Oct. 2009.

[13] K. Han, B. Kang and E. G. Im, "Malware analysis using visualized image matrices", Sci. World J., vol. 2014, pp. 1-15, Jul. 2014.

[14]. Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang and J. Chen, "Detection of malicious code variants based on deep learning", IEEE Trans. Ind. Informat., vol. 14, no. 7, pp. 3187-3196, Jul. 2018.