Dynamic Malware Detection Using Effective Machine Learning Models with Feature Selection Techniques

Inam Ullah Khan[1], Fida Muhammad Khan[1], Zeeshan Ali Haider[1],Saba Khattak[2],Gulshan Naheed[3],Sana Shaoor Kiani[4]

[1]Department of Computer Science, Qurtuba University of Science & Information, Technology, Peshawar, Pakistan.
[2]Department of Computer Science, University of Science & Technology, Bannu, Pakistan
[3] Higher Education Departments KPK, Peshawar.
[4]Ministry of Law & Justice Pakistan
*** Correspondence.** Inam Ullah Khan. inam1software@gmail.com

Citation | Khan.I.U, Khan,F.M, Haider.Z.A, Khattak.S, Naheed.G, Kiani, S.S "Dynamic Malware Detection Using Effective Machine Learning Models with Feature Selection Techniques", IJIST, Vol. 6 Issue.3 pp 1438-1452, Sep 2024
Received | Aug 13, 2024 **Revised |** Sep 9, 2024 **Accepted |** Sep 13, 2024 **Published |** Sep 16, 2024.

Dynamic Malware is a type of virus that is self-modifying, which makes it difficult to analyze in the course of its operation. It occasionally changes its behavior based on the existing environment and the context of execution. The goal of this study was to identify and detect dynamic malware in Android devices using effective machine-learning models with feature selection techniques. With new malicious software emerging daily, relying solely on manual heuristic analysis has become ineffective. To address this limitation, the study used dynamic detection methods to detect the events of interest using machine learning models. Some of these measures entailed duplication of an environment in which the behavior of malware could be replicated and then come up with reports. The reports were then transformed into sparse vector models so that other machine-learning techniques could then be applied to them. In this research study seven different models, namely, KNN, DT, RF, AdaBoost, SGD, Extra Trees, and Gaussian NB, were used to train an effective malware detection model to predict the dynamic malware in its early stages. The study showed that Random Forest, Stochastic Gradient Descent, Extra Tree, and Gaussian Naive Bayes classifiers achieved the highest accuracy compared to other models. This study endorses the application of machine learning-based automated behavior analysis for malware detection, about the complexities involved in the dynamic behavioral analysis of malicious software.

Keywords. Cyber Security, Machine Learning, Cyber-Attacks, Random Forest, Decision Trees, KNN, Gaussian Naive Bayes (NB), Malicious Threats.

Introduction.

In today's era of global digitization, hackers and cyberterrorists present a significant threat. One of the major challenges with the Internet as a medium for virus transmission is the rapid proliferation of polymorphic, newly developed, and innovative executable types. Traditional antivirus software, which relies on signature-matching mechanisms, often struggles to counter these threats. This limitation allows many forms of malware to bypass standard defenses. Moreover, static analysis based on human evaluation of heuristics has proven to be impractical and ineffective, especially as the diversity and sophistication of malware continue to increase [1].

An alternative approach to malware defense involves enhancing dynamic analysis through automated data mining techniques, including the use of ML tools [2]. In the current landscape, although virus scanners and malware detection programs are widely accessible, the frequency of malware incidents continues to rise. Both static and dynamic methods for malware detection and classification have been proposed. Dynamic analysis offers distinct advantages over static approaches, primarily because malicious behaviors are more difficult to hide during execution than when detected statically [3][4].

In era years, cyber security experts have increasingly turned to ML algorithms to identify malware and predict the behavior of malware families. However, there is no centralized database that systematically evaluates and compares different ML techniques for detecting and classifying malware. To address this gap, we conducted a series of tests to assess various ML approaches for malware detection and dynamic family classification. Figure 1 [5] highlights the constant threat posed by emerging malware, with new variants appearing every second.

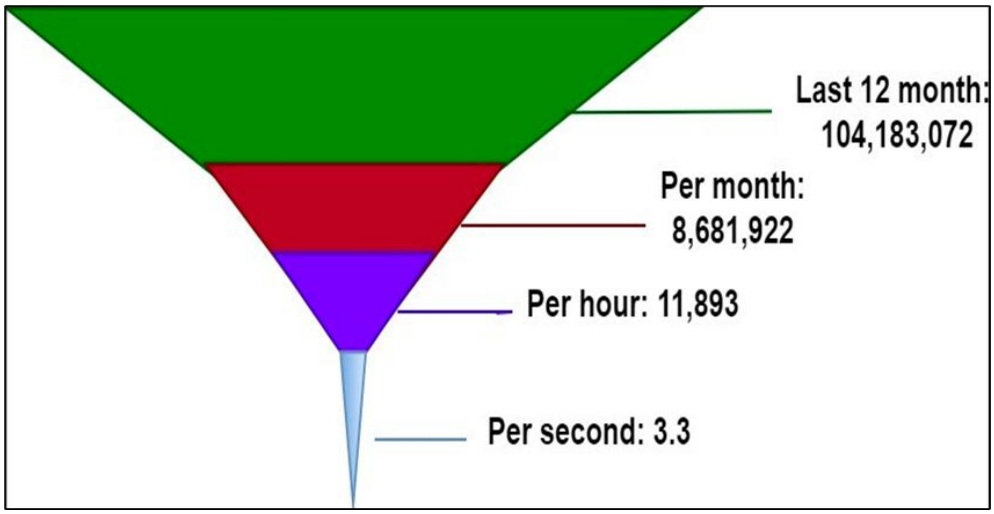

Figure 1. New malware threats every second

To capture malware behavior, a dataset of real malware samples was gathered and run in a sandboxed environment by safe programs from Virus. We then used this data to evaluate machine learning approaches using widely used performance measures. Execution data gathered as JSON reports offer a promising collection of characteristics characterizing a malware sample's behavior. Once processing is finished, malicious files can be distinguished from benign ones using the produced feature set. This work was motivated by the fact that multiple ways have been developed to optimize for a broad range of criteria. Because of this, even under identical situations, they behave in different ways. To address the problem of dynamically identifying malware using machine learning approaches, we also provide recommendations for future research directions. Figure 2 [6] shows the categorization of OS-based risks.

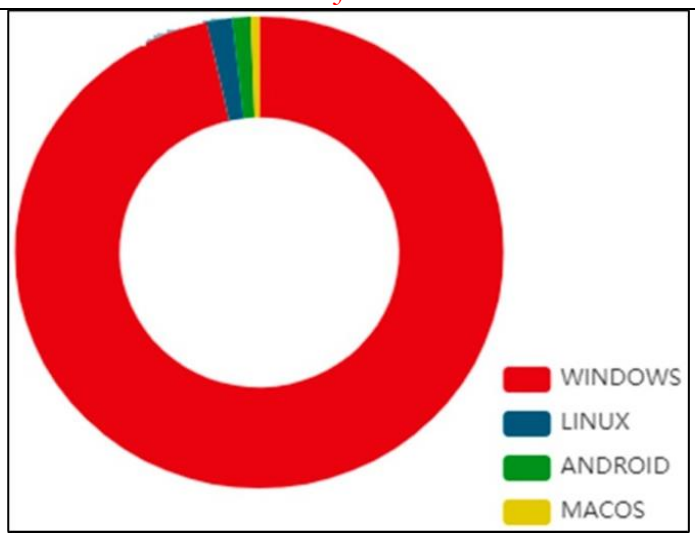

Figure 2. Threat classification of OS-based malware

A growing number of individuals are accessing the Internet through an expanding range of devices, from embedded systems to desktop PCs [7][8]. This increase in Internet usage has brought significant benefits, including faster communication and broader accessibility for more people [9]. With constant Internet connectivity, users can access and use their preferred online services at any time. However, the rise in Internet usage has also contributed to the proliferation of malware. Given the lucrative nature of the malware market for illicit software developers, it is unsurprising that its prevalence has surged. Figure 3 highlights the exponential growth in malware detection that has occurred in just the past few years [10][11][12].

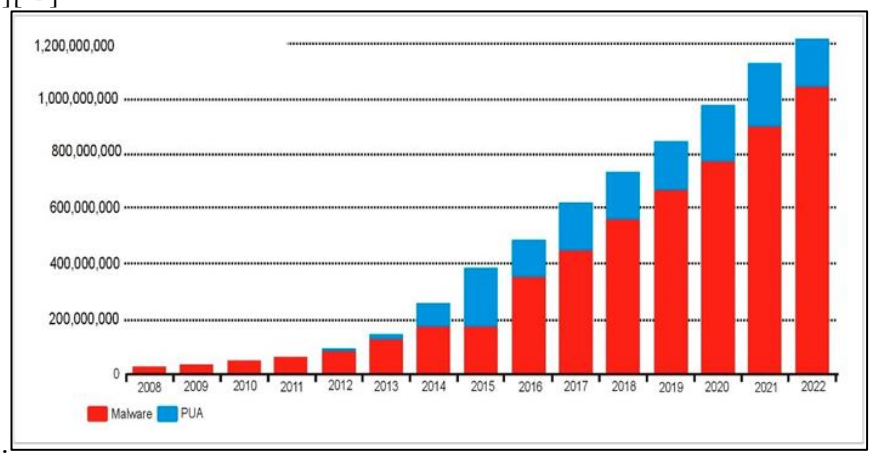

Figure 3. The total quantity of PUAs

Anti-malware software, intrusion detection systems, and other detection methods have been developed in response to the damage that malware may inflict. Nonetheless, given the dynamic methods used by malicious applications and the pervasiveness of security holes in well-known programs, several pressing concerns need to be addressed right away [13][14]. Numerous methods from various fields have been put forth for effective malware detection. Dynamic approaches outperform static ones because it is more challenging to conceal malware's damaging behavior while it is operating. Researchers' attention has migrated away from conventional malware detection approaches as they have come to appreciate the advantages of dynamic and automated methodologies [15].

Although there are antivirus programs, log file analysis, and interactive monitoring that help to detect undesirable behavior, cybercriminals do not cease creating new, dangerous malware. Traditional detection techniques such as signature-based and both static and dynamic

analysis are difficult to use to detect new and evolving malware. The primary problems are non-evolutionary threat detection ability, high false positive ratio, and longer time duration. Therefore, it is of essential importance to develop approaches that impact the number of used features and take care of the malware samples that are completely unknown to any previously created model.

- To optimize the algorithms for malware detection that would be an improvement of the existing ones with an emphasis on false positives absence and short processing time to detect the dynamic malware in its early stages.
- To embark on a new study to discover and define a smaller set of features that can be used to enhance the efficiency of the detection of malware to simplify the process.
- To describe and contrast techniques and strategies for identifying novel and unidentified malware, utilizing cutting-edge technology like machine learning and dynamic analysis to expand the detection space.

Related Work.

Trinius (2016) introduced a new visualization called MIST that is used to trace the activities of dangerous applications. The representation has been optimized by data mining and machine learning to facilitate effective behavior analysis. When looking into malware, it can be collected automatically through software that tracks behavior or manually from behavior reports that already exist. By utilizing shared characteristics, Rieck (2018) aims to classify dangerous programs [16]. One can deduce the authors' goals by looking for recurring behavioral patterns among malware variants, according to Patil (2020). They use a three-step process 1) examine malware behavior in a sandbox 2) use an antivirus application to annotate a corpus of malware, and 3) analyze the findings shown in Figure 4 [17].

An explanation for the classifications made is provided by prioritizing the most distinctive features of the behavior models, which are trained using learning techniques. A machine learning-based mechanism is proposed by Rieck (2018) to automatically analyze malware activities [18].

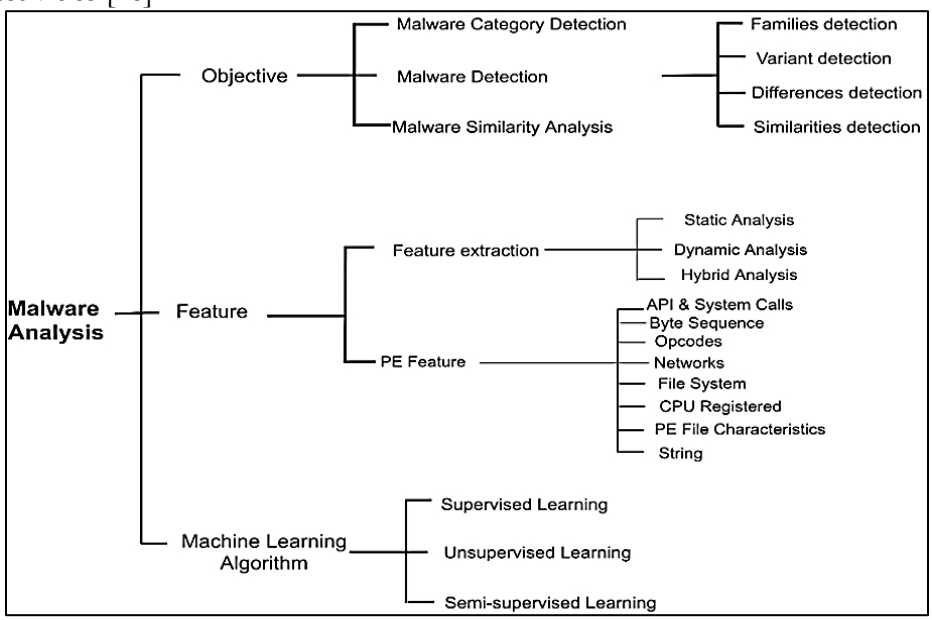

Figure 4. Techniques for analyzing malware [17].

The proposed framework categorizes undiscovered malware based on their behaviors into groups that have already been identified. In 2018, Christodorescu proposed a method that detects potential threats by comparing the execution patterns of known malware with those of safe applications. The authors identify and extract malicious components from known viruses

that are not present in safe programs which malware detectors can use to detect newly emerging malware [19]. Machine learning algorithms enhance feature quality through techniques such as engineering, selection, and representation. By training on data that represents the characteristics of malicious and legitimate software, the model establishes a plane to distinguish between goodware and malware [20]. Effective feature selection and engineering rely heavily on domain expertise.

A weakness of traditional machine learning-based malware detection systems may reverse-engineer and mimic the features that the model uses. Machine learning algorithms also require large datasets for training. However, due to privacy and security concerns, high-quality public datasets for malware research are scarce. Many researchers now generate their datasets for analysis, following data science methodologies [21]. Reviewing the content from Ye (2017) would be a labor-intensive task. These challenges complicate the development of a real-time malware detection system based on machine learning [22].

Modern AI systems employ deep learning models—advanced neural networks—across various tasks in fields such as natural language processing and robotics. These models learn from errors during training and store comprehensive representations of features in hidden layers. Neelam (2020) explored research on deep learning systems for malware analysis [23]. Figure 4 highlights various malware analysis methods [17]. Machine learning techniques are applied to develop a malware behavior classifier, focusing on the most distinctive behavioral traits to explain the classifications. Rieck (2018) introduced an ML-based method for automatically analyzing malware behavior, enabling the classification of previously unknown malware based on their activities[18]. Christodorescu (2018) presented a method comparing known malware execution with safe programs to detect potential threats, extracting malicious components absent from a batch of safe software. Malware detectors can then use this method's output to identify new malware [19].

Machine learning algorithms prioritize improving the quality of features through development, selection, and representation. By training on data that represents the attributes of each class, the model can malicious software [20]. Domain expertise is critical for effective feature engineering and selection. However, ML-based malware detection systems are vulnerable if attackers manage to reverse-engineer and understand the features the model uses. These algorithms also require vast amounts of data for training. Due to privacy and security concerns, obtaining high-quality public datasets suitable for malware research is difficult. As a result, many researchers now create their datasets for analysis [21]. Examining Ye's (2017) data would require significant effort, adding to the challenges of developing a real-time machine learning-based malware detection system [22].

Deep learning models, an advanced form of neural networks, are used in modern AI systems for various tasks in NLP and robotics. These models attempt to store a comprehensive, learning from mistakes as they go. Neelam (2020) examined studies employing deep-learning models for malware analysis [23]. In 2015, Microsoft introduced a malware classification challenge on Kaggle, providing a dataset of around 20,000 malware samples, which amounted to nearly half a terabyte. Ronen (2015) reviewed published and suggested research using this dataset [24]. Souri (2020) conducted an extensive literature review on different malware detection methods, comparing behavior-based and signature-based approaches. The review concluded that hybrid methods, which combine static and dynamic analysis, are more effective than using either approach alone [17].

Yanfang (2018) reviewed prior research on feature extraction, classification methods, cloud-based malware detection, and malware development trends, with a particular focus on hybrid, dynamic, and static analysis methods. However, these reviews are limited to studies conducted before 2017, suggesting the need for more current research [25][26]. In 2017, Ucci systematically reviewed machine learning approaches for identifying malicious PE files in the

Windows operating system, considering factors such as objectives, methodologies, and sample characteristics in each study [16][27][28][29].

Methodology.

Modern computer infrastructure is increasingly susceptible to the spread of more sophisticated viruses. As malware samples multiply exponentially, traditional signature-based detection methods are becoming less effective. Researchers have demonstrated that machine learning can accurately detect and classify malicious data. In this study, we aim to reduce the dataset size to minimize computational overhead, using feature selection techniques to identify the most critical attributes. This approach will further enhance the accuracy of machine-learning models for malware detection.

The objective of this research is to apply a ML-based model to improve the efficiency and accuracy of malware detection and classification. We utilized the Cuckoo sandbox for dynamic analysis, which allows malware to execute in an isolated environment and generates a report detailing its behavior. Additionally, we proposed a module for feature extraction and selection. Our experiment results showed that this approach improves detection and classification accuracy compared to state-of-the-art techniques. The strategy is composed of three key steps including feature selection for the model, identification of advanced malware, and finding suitable datasets to train the classifier. A detailed explanation of the research methodology is outlined below, with the proposed methodology illustrated in Figure 5.

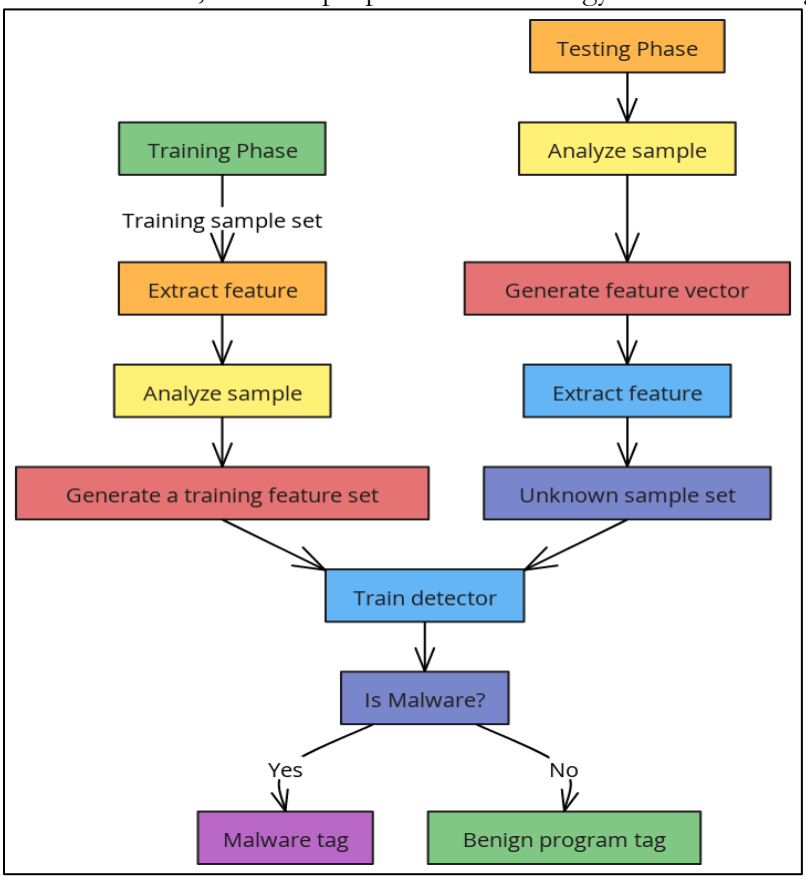

Figure 5. Proposed malware detection methodology.

Dataset.

The dataset used in this study was sourced from the Kaggle repository. By integrating both native and non-native features extracted from Windows programs, we constructed a training set. The dataset contained a total of 373 samples, of which 301 were identified as malicious and 72 as non-malicious. There were 531 features in total, labeled F1 through F531,

along with a label column indicating the file's risk level. The Kaggle dataset was the sole data source utilized for the study. Many files in the dataset contained log data that had been manipulated by various malware programs, which enabled us to train multiple models using the recovered log data. The samples were infected by five distinct malware families. The dataset included over 198,063 unique data points gathered from various sources. Table 1 provides an overview of the malware dataset, which consists of 373 rows and 531 columns.

Table 1. A preview of the dataset

		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	...	F-522	F-523	F-524	F-525	F-526	F_52
0	non-malicous	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0
1	non-malicous	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0
2	non-malicous	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0
3	non-malicous	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0
4	non-malicous	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0

In contemporary datasets, features with tens of thousands or more are typical. As a machine learning model's characteristic count rises, the problem becomes increasingly apparent.

Selection of Features.

Selecting which features to employ comes next, following the discovery of new features through the feature extraction process. Feature selection is the process of picking features from a group of recently identified attributes. It helps to reduce overfitting, streamline the model, and increase accuracy. Researchers have employed a range of feature classification techniques to detect malicious software. The feature rank approach is often used in this work since creating models to identify malware is its main goal. Figure 6 shows that 78% of the data points in our dataset were classified as malicious, while 22% of the data points were classified as non-malicious.

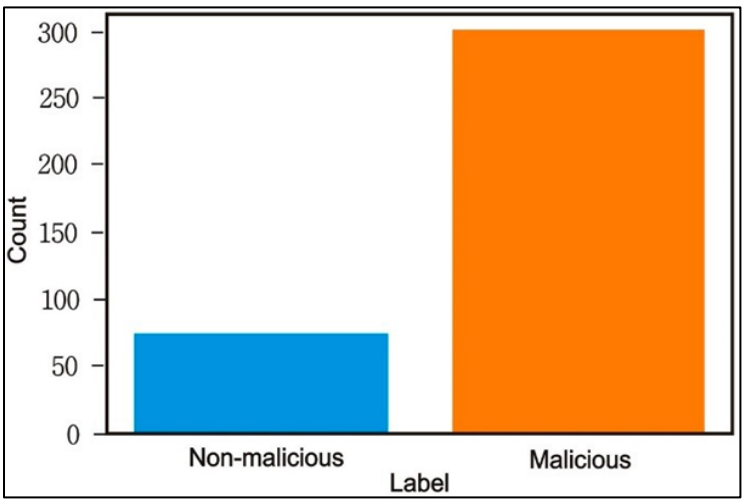

Figure 6. feature selection, the counts of malicious and non-malicious data points.

Experimental Results.

In this section, we present the experimental findings, including an evaluation of the classification models' effectiveness and key insights drawn from the data. The results are assessed using performance metrics such as accuracy, precision, recall, and the F1 score, offering a thorough evaluation of the classifier's performance. The discussion interprets these metrics, emphasizing the strengths and potential limitations of the models, and explores how the results align with or deviate from expectations. Furthermore, we analyze the broader implications of these findings to the research objectives and practical applications.

Training and testing are critical components of any successful classification strategy. It is essential to use both malicious and benign data during the training phase to ensure balanced learning. Machine learning techniques enable classifiers to generate high-quality predictions automatically. With the addition of more labeled data, classifiers like random forests, (SGD), extra trees, and Gaussian classifiers exhibit improved performance. During the validation process, the classifier must accurately identify which files belong to each category when presented with a new set of files, some of which are malicious and some benign.

Model Training Using Random Forest.

Table 2 presents the classification report for the Random Forest model. According to this report, Random Forest achieved the highest accuracy compared to other models. Figure 7 shows the confusion matrix of the Random Forest Model.

Table 2. Random Forest Classification Report

Type of Classification / Average	Support	F1-Score	Precision	Recall
Classification Type. Malicious	61	1.00	1.00	1.00
Classification Type. Non-Malicious	14	1.00	1.00	1.00
Accuracy		1.00%		
Average Type. Macro Average	75	1.00	1.00	1.00
Average Type. Weighted Average	75	1.00	1.00	1.00

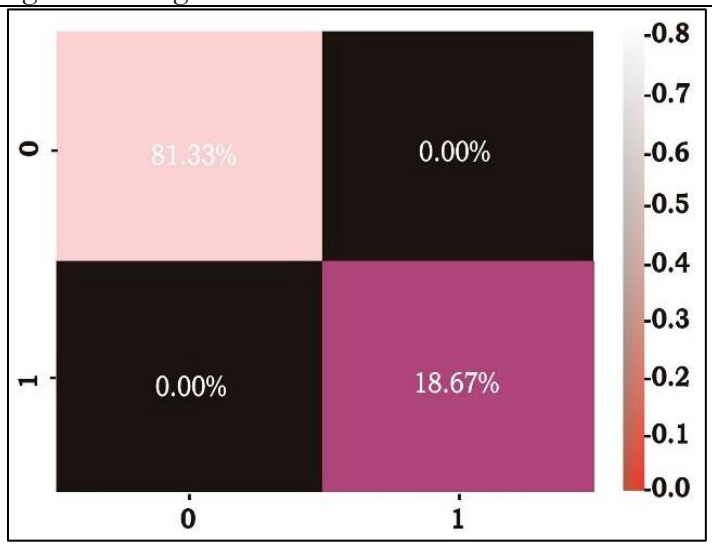

Figure 7. Confusion Matrix of Random Forest.

Model Training Using Decision Tree.

Table 3 shows the classification report of the Decision Tree Model. According to this report, Decision Tree achieved 98.76% accuracy. Figure 8 shows the confusion matrix of the Decision Tree Model.

Table 3. Classification Report of Decision Tree

Type of Classification	Support	F1-Score	Precision	Recall
Classification Type. Malicious	61.00	0.99	0.98	1.00
Classification Type. Non-Malicious	14.00	0.96	1.00	0.93

Accuracy. 98.76%

Support. Total instances evaluated. 75

Type of Average	Support	F1-Score	Precision	Recall
Average Type. Macro Average	75	0.98	0.99	0.96
Average Type. Weighted Average	75	0.99	0.99	0.99

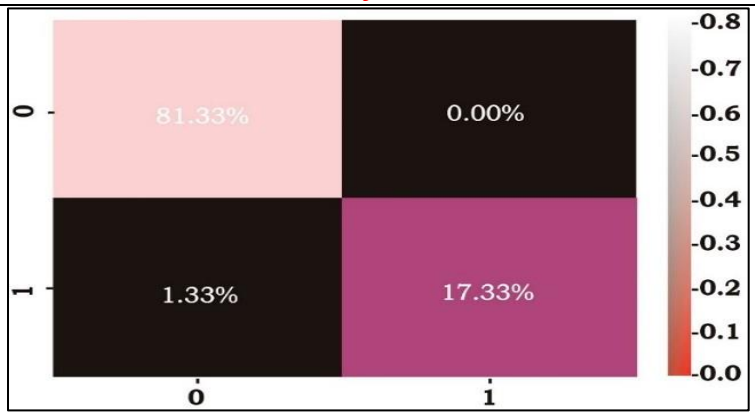

Figure 8. Confusion Matrix of Decision Tree

Model Training Using KNN.

Table 4 presents the classification report for the KNN model, which achieved an accuracy of 98.69%. Figure 9 illustrates the confusion matrix for the KNN model.

Table 4. KNN Classification Report

Type of Classification	Support	F1-Score	Precision	Recall
Classification Type. Malicious	61	0.99	0.98	1.00
Classification Type. Non-Malicious	14	0.96	1.00	0.93

Accuracy. 98.69%

Support. Total instances evaluated. 75

Type of Average	Support	F1-Score	Precision	Recall
Average Type. Macro Average	75	0.98	0.99	0.96
Average Type. Weighted Average	75	0.99	0.99	0.99

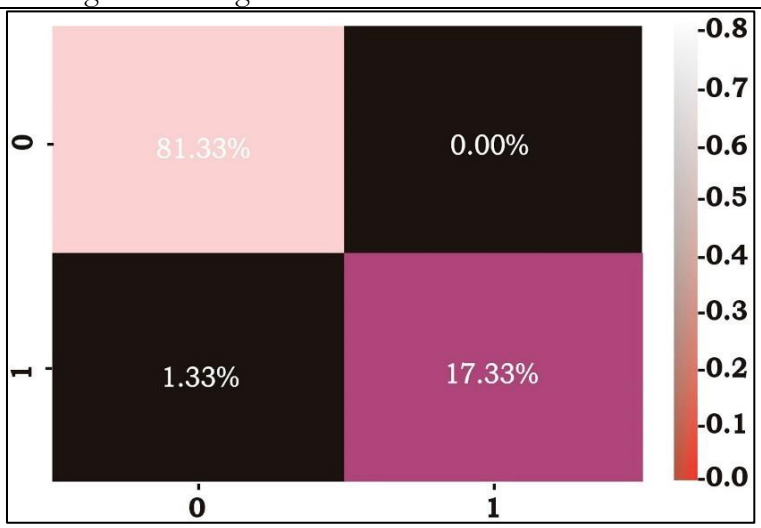

Figure 9. Confusion Matrix of KNN Model

Model Training Using AdaBoost.

Table 5 presents the classification report for the AdaBoost model, which achieved an accuracy of 98.71%. Figure 10 displays the confusion matrix for the AdaBoost model.

Table 5. AdaBoost Classification Report

Type of Classification	Support	F1-Score	Precision	Recall
Classification Type. Malicious	61	0.99	0.98	1.00
Classification Type. Non-Malicious	14	0.96	1.00	0.93

Accuracy. 98.71%

Support. Total instances evaluated. 75

Type of Average	Support	F1-Score	Precision	Recall

Average Type. Macro Average	75	0.98	0.99	0.96
Average Type. Weighted Average	75	0.99	0.99	0.99

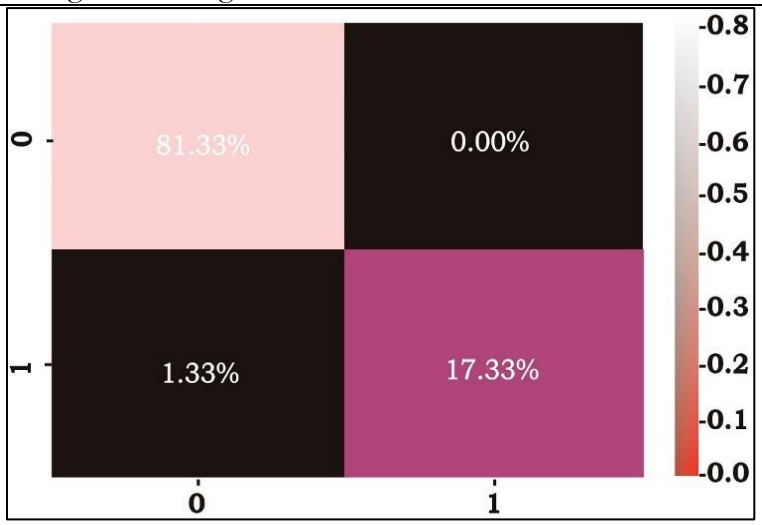

Figure 10. Confusion Matrix of AdaBoost Modle

Model Training Using Gradient Boosting.

Table 6 presents the classification report for the Gradient Boosting model, which achieved an accuracy of 100%. Figure 11 displays the confusion matrix for the Gradient Boosting model.

Table 6. Gradient Boosting Classification Report

Type of Classification	Support	F1-Score	Precision	Recall
Classification Type. Malicious	61	1.00	1.00	1.00
Classification Type. Non-Malicious	14	1.00	1.00	1.00

Accuracy. 100%

Support. Total instances evaluated. 75

Type of Average	Support	F1-Score	Precision	Recall
Average Type. Macro Average	75	1.00	1.00	1.00
Average Type. Weighted Average	75	1.00	1.00	1.00

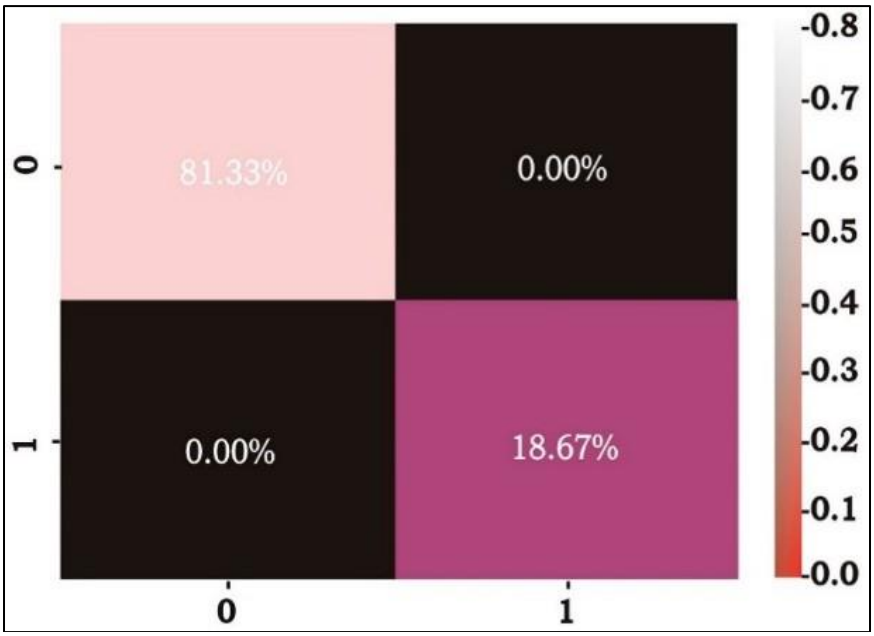

Figure 11. Confusion Matrix of SGD Model

Model Training Using Extra Trees. Table 7 presents the classification report for the Extra Trees model, which achieved 100% accuracy. Figure 12 displays the confusion matrix for the Extra Trees model.

Table 7. Extra Trees Classification Report

Classification Type	Support	F1-Score	Precision	Recall
Classification Type. Malicious	61	1.00	1.00	1.00
Classification Type. Non-Malicious	14	1.00	1.00	1.00

Accuracy. 100%

Support. Total instances evaluated. 75

Average Type	Support	F1-Score	Precision	Recall
Average Type. Macro Average	75	1.00	1.00	1.00
Average Type. Weighted Average	75	1.00	1.00	1.00

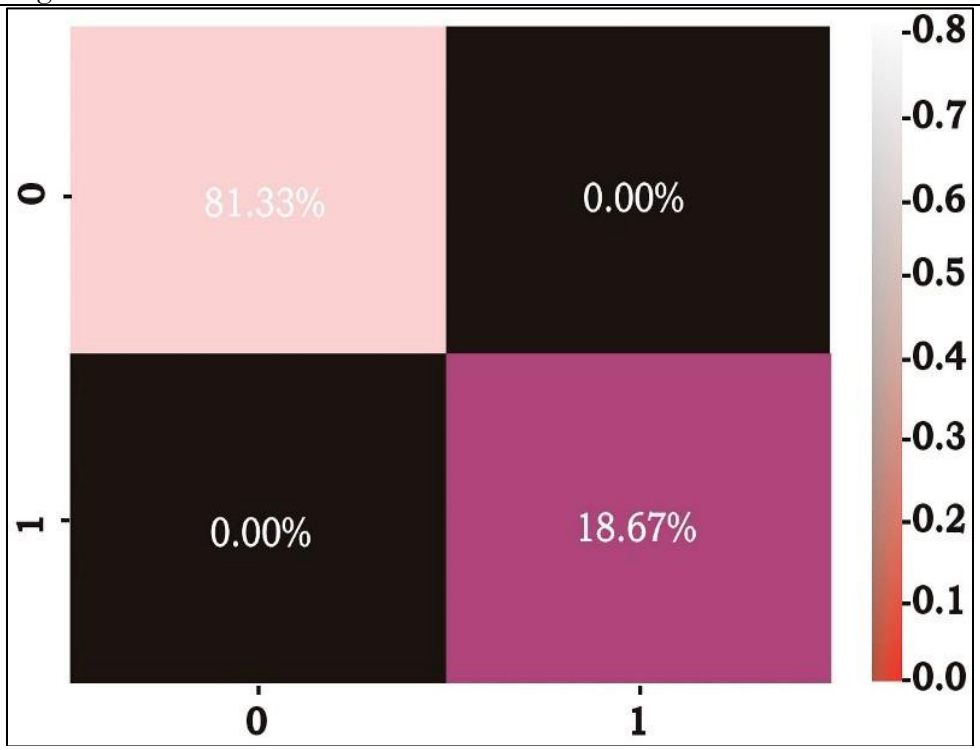

Figure 12. Confusion Matrix of Extra Tree Model

Model Training Using Naïve Bayes.

Table 8 presents the classification report for the Naïve Bayes model, which achieved 100% accuracy. Figure 13 displays the confusion matrix for the Naïve Bayes model.

Table 8. Naïve Bayes Classification Report

Classification Type	Support	F1-Score	Precision	Recall
Classification Type. Malicious	61	1.00	1.00	1.00
Classification Type. Non-Malicious	14	1.00	1.00	1.00

Accuracy. 100%

Support. Total instances evaluated. 75

Average Type	Support	F1-Score	Precision	Recall
Average Type. Macro Average	75	1.00	1.00	1.00
Average Type. Weighted Average	75	1.00	1.00	1.00

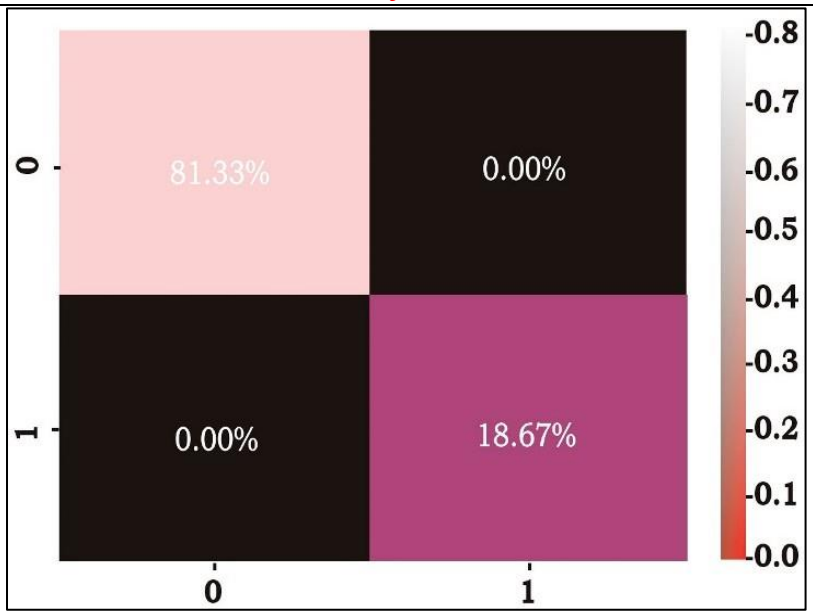

Figure 13. Confusion Matrix of Naïve Bayes Model

Accuracies Comparison of Proposed Models.

Figure 14 displays the accuracy comparison of the models used in this study. After generating a dataset containing both malware and cleanware, the data was utilized for testing. The accuracy achieved by the proposed models Gaussian (NB), (RF), Extra Trees, Stochastic Gradient Descent (SGD), and KNN—is summarized in Table 9. This study exemplifies the growing interest among researchers in applying machine learning techniques for malware detection. We explored three different methodologies to determine which machine learning technique was the most efficient for malware detection. The results indicated that the Gaussian Naive Bayes (NB), Random Forest (RF), Extra Trees, and Stochastic Gradient Descent (SGD) models achieved the highest accuracy in malware detection, as illustrated in the figures.

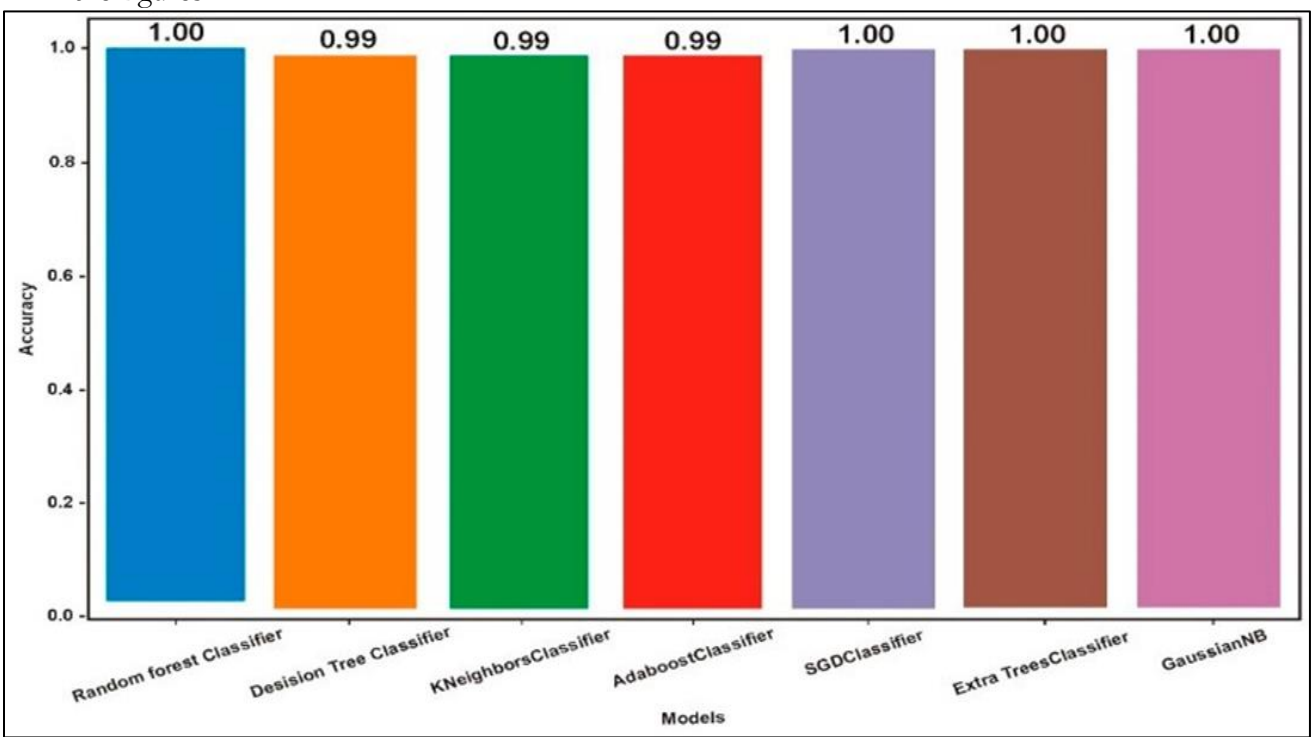

Figure 14. Accuracies Comparison of Models

Table 9. Accuracy comparison table

Model	Accuracy
Random Forest	1.00
Decision Tree	0.99
K-Nearest Neighbors	0.99
AdaBoost	0.99
Stochastic Gradient Descent	1.00
Extra Tree Classifier	1.00
Gaussian Naive Bayes	1.00

Table 9 shows that the accuracies for RF, Gradient Descent, Extra Tree, and Gaussian Naive Bayes have 100% accuracy, 100% precision, and 100% recall along with a flawless F1 score. Malware is one of the top concerns when it comes to internet safety. The majority of online problems, including DDoS assaults and spam emails, are caused by malware. That is, compromised computers are often connected to bigger networks known as botnets, and these hostile, attacker-controlled networks are used to carry out a lot of attacks. Fourthly, it talks about the main issues and challenges that researchers face. We paid special attention to the problems associated with adversarial learning and the problem of concept drift. In addition, it examines the problem of class disparity and the quality of the standards that the scientific community is currently using to assess how effective their methods are.

Conclusions.

This research study addresses the limitations of manual feature engineering and current learning methods by combining Random Forests (RF), Adversarial Sample Generation, Extra Trees, and Gaussian Naive Bayes (NB) models to develop an innovative ensemble deep neural network for malware detection. The Gaussian NB, Extra Trees, and Adversarial Sample Generation models delivered exceptional results, achieving 100% in F1 score, accuracy, precision, and recall. During both training and testing, these models demonstrated near-perfect performance, significantly improving the accuracy of malware detection. By integrating these models, it becomes possible to capture long-term dependencies, and model sequences, and detect spatially local correlations. Future research will focus on increasing the dataset size to enhance model generalization and discover new insights in adversarial sample generation. Additionally, training large language models (LLMs) will be explored to further improve the model's accuracy and robustness against advanced adversarial attacks. These efforts aim to advance the development of more secure and resilient AI systems.

References.

[1] M. Alazab, R. A. Khurma, D. Camacho, and A. Martín, "Enhanced Android Ransomware Detection Through Hybrid Simultaneous Swarm-Based Optimization," Cognit. Comput., pp. 1–15, Jun. 2024, doi. 10.1007/S12559-024-10301-4/METRICS.

[2] M. S. Akhtar and T. Feng, "Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time," Symmetry 2022, Vol. 14, Page 2308, vol. 14, no. 11, p. 2308, Nov. 2022, doi. 10.3390/SYM14112308.

[3] J. Zhang, "DeepMal. A CNN-LSTM model for malware detection based on dynamic semantic behaviors," Proc. - 2020 Int. Conf. Comput. Inf. Big Data Appl. CIBDA 2020, pp. 313–316, Apr. 2020, doi. 10.1109/CIBDA50819.2020.00077.

[4] M. T. Nguyen, V. H. Nguyen, and N. Shone, "Using deep graph learning to improve dynamic analysis-based malware detection in PE files," J. Comput. Virol. Hacking Tech., vol. 20, no. 1, pp. 153–172, Mar. 2024, doi. 10.1007/S11416-023-00505-X/METRICS.

[5] S. Hosseini, A. E. Nezhad, and H. Seilani, "Android malware classification using convolutional neural network and LSTM," J. Comput. Virol. Hacking Tech., vol. 17, no. 4, pp. 307–318, Dec. 2021, doi. 10.1007/S11416-021-00385-Z/METRICS.

[6] D. Čeponis and N. Goranin, "Investigation of Dual-Flow Deep Learning Models LSTM-FCN and GRU-FCN Efficiency against Single-Flow CNN Models for the Host-Based Intrusion and Malware Detection Task on Univariate Times Series Data," Appl. Sci. 2020, Vol. 10, Page 2373, vol. 10, no. 7, p. 2373, Mar. 2020, doi. 10.3390/APP10072373.

[7] C. Avci, B. Tekinerdogan, and C. Catal, "Analyzing the performance of long short-term memory architectures for malware detection models," Concurr. Comput. Pract. Exp., vol. 35, no. 6, pp. 1–1, Mar. 2023, doi. 10.1002/CPE.7581.

[8] A. Mahindru and A. L. Sangal, "FSDroid.- A feature selection technique to detect malware from Android using Machine Learning Techniques. FSDroid," Multimed. Tools Appl., vol. 80, no. 9, pp. 13271–13323, Apr. 2021, doi. 10.1007/S11042-020-10367-W/TABLES/21.

[9] M. Al-Kasassbeh, S. Mohammed, M. Alauthman, and A. Almomani, "Feature Selection Using a Machine Learning to Classify a Malware," Handb. Comput. Networks Cyber Secur. Princ. Paradig., pp. 889–904, Jan. 2020, doi. 10.1007/978-3-030-22277-2_36.

[10] M. S. Akhtar and T. Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," Symmetry 2022, Vol. 14, Page 2304, vol. 14, no. 11, p. 2304, Nov. 2022, doi. 10.3390/SYM14112304.

[11] S. Il Bae, G. Bin Lee, and E. G. Im, "Ransomware detection using machine learning algorithms," Concurr. Comput. Pract. Exp., vol. 32, no. 18, p. e5422, Sep. 2020, doi. 10.1002/CPE.5422.

[12] A. Mahindru and A. L. Sangal, "MLDroid—framework for Android malware detection using machine learning techniques," Neural Comput. Appl., vol. 33, no. 10, pp. 5183–5240, May 2021, doi. 10.1007/S00521-020-05309-4/TABLES/37.

[13] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," J. Syst. Archit., vol. 112, p. 101861, Jan. 2021, doi. 10.1016/J.SYSARC.2020.101861.

[14] O. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," IEEE Access, vol. 9, pp. 87936–87951, 2021, doi. 10.1109/ACCESS.2021.3089586.

[15] "Machine Learning for Malware Detection." Accessed. Sep. 18, 2024. [Online]. Available. https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf

[16] P. Singhal and N. Raul, "Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks," Int. J. Netw. Secur. Its Appl., vol. 4, no. 1, 2012, doi. 10.5121/ijnsa.2012.4106.

[17] B. Cuan, A. Damien, C. Delaplace, and M. Valois, "Malware Detection in PDF Files using Machine Learning," pp. 578–585, Sep. 2018, doi. 10.5220/0006884705780585.

[18] S. I. Rimon and M. M. Haque, "Malware Detection and Classification Using Hybrid Machine Learning Algorithm," Lect. Notes Networks Syst., vol. 569 LNNS, pp. 419–428, 2023, doi. 10.1007/978-3-031-19958-5_39.

[19] X. Jin, X. Xing, H. Elahi, G. Wang, and H. Jiang, "A malware detection approach using malware images and autoencoders," Proc. - 2020 IEEE 17th Int. Conf. Mob. Ad Hoc Smart Syst. MASS 2020, pp. 631–639, Dec. 2020, doi. 10.1109/MASS50613.2020.00009.

[20] A. A. Darem, F. A. Ghaleb, A. A. Al-Hashmi, J. H. Abawajy, S. M. Alanazi, and A. Y. Al-Rezami, "An Adaptive Behavioral-Based Incremental Batch Learning Malware Variants Detection Model Using Concept Drift Detection and Sequential Deep Learning," IEEE Access, vol. 9, pp. 97180–97196, 2021, doi.

10.1109/ACCESS.2021.3093366.

[21] T. Huang, R. Zhao, L. Bi, D. Zhang, and C. Lu, "Neural Embedding Singular Value Decomposition for Collaborative Filtering," IEEE Trans. Neural Networks Learn. Syst., vol. 33, no. 10, pp. 6021–6029, Oct. 2022, doi. 10.1109/TNNLS.2021.3070853.

[22] "Performance Evaluation of Machine Learning Algorithms for Detection and Prevention of Malware Attacks." Accessed. Sep. 18, 2024. [Online]. Available. https.//www.researchgate.net/publication/333004518_Performance_Evaluation_of_Machine_Learning_Algorithms_for_Detection_and_Prevention_of_Malware_Attacks

[23] "Introduction to Simple Imputer Class", [Online]. Available. https.//scikitlearn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

[24] J. Mcgiff, W. G. Hatcher, J. Nguyen, W. Yu, E. Blasch, and C. Lu, "Towards Multimodal Learning for Android Malware Detection," 2019 Int. Conf. Comput. Netw. Commun. ICNC 2019, pp. 432–436, Apr. 2019, doi. 10.1109/ICCNC.2019.8685502.

[25] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A Review of Android Malware Detection Approaches Based on Machine Learning," IEEE Access, vol. 8, pp. 124579–124607, 2020, doi. 10.1109/ACCESS.2020.3006143.

[26] U. D. Atmojo, G. Ögmundsdóttir, R. Bejarano, D. Dowling, and V. Vyatkin, "A Digital Twin Model for an Educational Turbocharger Demonstrator," SSRN Electron. J., Feb. 2022, doi. 10.2139/SSRN.4072612.

[27] J. A. Herrera-Silva and M. Hernández-Álvarez, "Dynamic Feature Dataset for Ransomware Detection Using Machine Learning Algorithms," Sensors, vol. 23, no. 3, p. 1053, Feb. 2023, doi. 10.3390/S23031053/S1.

[28] S. Saad, W. Briguglio, and H. Elmiligi, "The Curious Case of Machine Learning in Malware Detection," Int. Conf. Inf. Syst. Secur. Priv., pp. 528–535, 2019, doi. 10.5220/0007470705280535.

[29] M. R. Keyvanpour, M. Barani Shirzad, and F. Heydarian, "Android malware detection applying feature selection techniques and machine learning," Multimed. Tools Appl., vol. 82, no. 6, pp. 9517–9531, Mar. 2023, doi. 10.1007/S11042-022-13767-2/METRICS.