A ROBUST MACHINE LEARNING-BASED FRAMEWORK TO LEVERAGE CLASSIFICATION OF MALWARE

Lingaraj Sethi author[1*], Dr prof Prashanta Kumar Patra[2]

[1*]Research Scholar, Computer Science and Engineering, Biju Patnaik University of Technology, Rourkela

[2]Dean, SRIC, Computer Science and Engineering,SOA University Bhubaneswar, Odisha

Abstract

Malicious software, or malware, is a growing problem in today's cyber landscape and threatens the availability, confidentiality, and integrity of digital information. An effective malware detection system can be designed with the help of ensemble machine-learning models, according to this research paper. This research makes use of the Canadian Institute for Cybersecurity's CIC-Malmem2022 dataset, which was designed for studies on complicated malware classification. To strengthen the malware detection model's accuracy and resilience, the suggested approach combines Principal Component Analysis, Recursive Feature Eliminator, Decision Trees, Light Gradient Boosting Machine, and Gradient Boosting. Although the ensemble model achieved a high level of accuracy (99.96%) on the test set, the results demonstrate its effectiveness. Model hyperparameter tuning reveals best-practice parameters, and the ensemble confusion matrix delves into classification efficacy. Analyses comparing the proposed approach to current methods show that it is superior at detecting malware. The study finishes with suggestions for a safe environment to deploy the model and for frequent updates to address shifting cybersecurity threats.

Keywords: Malware, Malicious Software, Spyware, Adware, Gradient Boosting.

1. Introduction

In recent years, advancements in computer systems and the Internet have greatly improved human existence. Almost anything can be done online, from socializing to making financial transactions to tracking a person's bodily changes, etc. These advancements entice cybercriminals to commit crimes online instead of in the actual world. Cybercrimes cost the global economy trillions of dollars, according to recent studies from both academia and industry[1]. Malware is a common tool used by cybercriminals to initiate cyberattacks. Malicious software is defined as any program that uses a victim's computer to carry out malicious or suspicious tasks. Threats like viruses, worms, Trojan horses, rootkits, ransomware, and so on fall under several malware categories. Malware comes in many forms, and it may steal sensitive information, launch DDoS attacks, and disrupt computer systems [2]. Malware has evolved to the point that it may evade detection by using tactics like encryption and packing[3]. These new strains were able to propagate because they preyed on people's confidence. Examples of well-known vectors for malware transmission include opening

attachments in emails, downloading malicious apps, and viewing and downloading files from malicious websites.

Identifying malware in its early stages is crucial for preventing system compromise. The term malware detection refers to the steps used to determine if a file is fraudulent or not[4]. There is an additional phase in malware categorization. Malware categorization is identifying the family or category of malware that the file belongs to once it has been determined to be malicious[5]. Malware detection is a three-step process:

- The right tools are used to examine malware files.
- The files that have been evaluated are used to extract both static and dynamic characteristics.
- Features are organized in specific ways to distinguish between harmful and harmless software[6].

1.1 Types of Malware

The term malware describes any piece of software whose express purpose is to cause damage or exploit systems, networks, or data. Many distinct kinds of malicious software exist, each with its own set of traits and objectives. Some examples of malware are discussed below in figure 1:

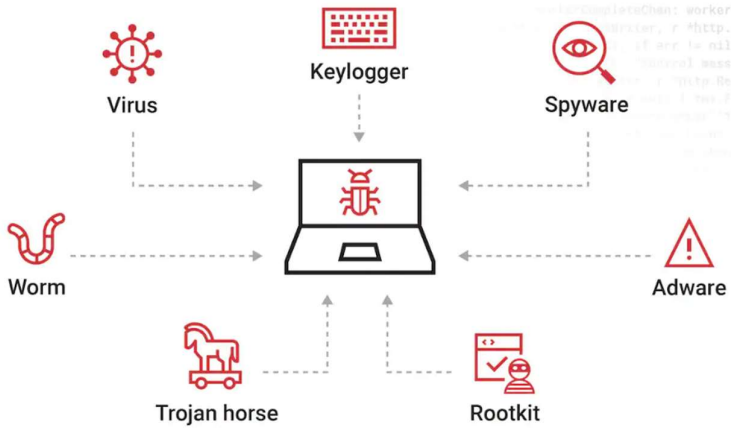

Figure 1: Types of Malware[7]

- **Viruses:**Viruses may infect even perfectly legal executable files and then multiply themselves whenever the infected application is executed. When infected files are shared, they have the potential to infect additional computers and files[8].
- **Worms:**Worms are programs that can replicate themselves and propagate across networks without any human interaction. Their favorite vector for spreading is taking advantage of security holes in software or operating systems.
- **Trojan Horses:**Trojan horses are dangerous programs that masquerade as safe software. Trojans differ from viruses and worms in that they do not self-replicate; instead, they trick people into running them by making themselves appear as innocuous files[9].

- **Spyware:**The goal of spyware is to stealthily monitor users and gather their personal information. Without the user's knowledge, it covertly logs their login information, web browsing habits, and keystrokes[10].
- **Adware:**Adware is software that sneaks advertisements onto consumers' devices. Though harmless most of the time, its disruptive behavior could lead to system performance problems[11].
- **Rootkits:**Malicious software, and rootkits, in particular, are created with the express purpose of evading detection by antivirus programs and system administrators. After infiltrating a system, rootkits can remain undetected while granting an attacker complete system privilege[12].
- **Keyloggers:**Keylogger software or hardware secretly captures computer or mobile device keystrokes. Keyloggers may be used to monitor children's internet activity or workplace efficiency, or they can be used maliciously to obtain sensitive data[13].

1.2 Malware Analysis

In addition to behavioral categorization, malware execution privileges should be considered. Malware becomes more damaging and tougher to detect and analyze as it gets right. Unknown binary code analysis may infect. The worst-case scenario is that the code under scrutiny irreparably harms the system without the user even realizing it has happened. A virus could compromise the integrity of reports, cause samples to be incorrectly labeled, or even put the entire company at risk.

A total of four protective rings, numbered zero through three, surround the Intel x86 central processing unit (CPU)[14]. The poll excludes Rings 1 and 2, which Windows does not utilize. More read/write permissions are granted to programs operating at lower protection rings. Protection rings underpin the four privilege levels:

- **User mode (Ring 3): -**User mode allows RAM loading of new process code. Anything that just needs user mode rights is a user mode code. Any user-installed applications and substantial components of Windows (even the Administrator account has user mode rights) are included. Malware may be removed by reversing its modifications or reformatting the system when studying user-mode malware.
- **Kernel mode (Ring 0): -**System resources are managed by the kernel, which is an operating system component. It controls the whole system, and it also has features for talking to the hardware. Ring 0 is where the kernel runs, with root or kernel mode privileges. The operating system kernel and system drivers are the only ones allowed to use this security ring [15]. The OS may then manage resources like CPU time and memory allocation, as well as physical devices, and user mode programs. Kernel mode code can load new code into the kernel whenever necessary, whenever new hardware is connected to the system[16]. The new device may be interacted with using this specific sort of code, which is known as a driver. Malware may also be known as a rootkit if it manages to enter the kernel of the system and execute commands with root capabilities [17].

- **Hypervisor (Ring -1): -**A hypervisor is software that allows many virtual operating systems to run on top of one another on a single piece of hardware. Although it is not a real protection ring, a hypervisor is considered to be operating in Ring -1 since it has higher rights than kernel mode. A hypervisor may be: Multiple operating systems may run simultaneously on type 1 hypervisors. Cybercriminals have been aware of hypervisors' capabilities for quite some time. An exploitable hypervisor, for instance, might be set up to ensnare an OS in a virtual machine and remove its administrative rights; Malicious hypervisors may take over operating systems when they obtain control of the kernel in this manner. Hypervisor code execution is invisible to any operating system-installed analysis tool. Virtual machine-based rootkits are malicious software that installs bad type 1 hypervisor [18,19]. Hypervisors of the type 2 kind enable the operation of virtual machines.
- **Hardware (Ring-3): -**Malware, such as the Ring-3 rootkit, may infect hardware components and then conduct attacks against other devices outside of the CPU without worrying about being detected [20]). Malicious firmware updateis a typical method of gaining access to protected electronics. The firmware is the programming that runs the gadget that is embedded in every physical component. Occasionally, the firmware may be upgraded to address security issues and resolve defects. But if an attacker finds a hole in the update procedure, they may exploit it to install malicious firmware that the CPU cannot detect, giving them access to the machine. One prevalent kind of hardware infection is USB [21], IoT, and medical devices.

2. Related Work

This section provides a thorough evaluation of the research published in A Robust Machine Learning-based Framework to Leverage Classification of Malware. Also considered are the relevant works of many writers.

2.1 Malware DetectionUsing Machine Learning

Hussain et al., (2022)[22]discovered a malware detection system that relies on Machine Learning (ML) to assess the safety or danger of a Portable Executable file by analyzing data extracted from its header. Random Forest, Decision Tree, AdaBoost, Gaussian Naive Bayes (GNB), and Gradient Boosting are among the ML models used to fight the virus after the author does data preprocessing. To choose the best ML model for the given issue, it also compares several models. The Random Forest (RF) achieved the highest accuracy level of 99.44% in detecting malware, according to the trial data. Use this to create a Windows desktop application that checks for malware and lets users customize the scanning process.

Shoaib and Feng, (2022)[23]discovered that to enhance the security of computer networks, it was possible to identify malicious traffic by comparing the results of malware analysis and detection using ML algorithms to determine the difference in correlation symmetry (Naive Byes, SVM, J48, RF, and the proposed method). Out of all the classifiers tested, DT(99%), CNN(98.76%), and SVM (96.41%) had the highest detection accuracy. In a specific dataset, it evaluated the efficacy of DT, CNN, and SVM algorithms in detecting malware using a small FPR. CNN attained 3.97%, SVM 4.63%, and DT 2.01%. Considering how common and advanced malicious software is, these results are significant.

Barath and Venkata, (2020) [24]proposed employing a combination of convolutional and recurrent neural networks as a method for malware software classification. While the proposed ensemble method attains a new benchmark of 99.8 percent overall accuracy, the LSTM network obtains 97.2% accuracy when distinguishing assembly files and 99.4% accuracy when classifying finished files.

Sumitand Amol, (2020) [25]presented implementations of convolutional neural networks (CNNs) and hybrid CNNs (CNNs+SVMs). Using CNNs as an automated feature extractor is more efficient than the existing methods. Evaluate this in comparison to other CNN models that are currently available: VGG16 (96.06%), ResNet50 (97.11%), InceptionV3 (97.22%), and Xception (97.56%). Among them, the proposed model's 98.03 percent accuracy is the most impressive.

Kumar et al., (2019) [26]utilizedanRF classifier, reached a static technique and got the maximum classification accuracy of 97.95%. The second reason the author utilized a dynamic method was that static analysis isnot always enough to decipher malware that has been encoded or packaged. At its peak, the RF classifier allowed us to get a classification accuracy of 99.13%. Finally, what it calls the Hybrid technique combines static and dynamic methods to get around the problems with each. Using Random Forest, our studies obtained a maximum classification accuracy of 99.74% in the first four seconds of malware operation, allowing us to categorize it into kinds.

2.2 Malware Detection Using Deep Learning

Alomari et al., (2023)[27]displayed an effective system for detecting malware using deep learning and feature selection. To detect malware and differentiate it from harmless actions, two databases are used. Using preprocessed datasets for feature selection based on correlation creates new datasets. Improving deep learning models using feature selection and LSTM on different versions of feature-selected datasets. Several metrics, including recall, accuracy, precision, and F1-score, are used to evaluate models following training. Feature selection can preserve the original dataset performance in some cases, according to the study. The degree to which performance varies is variable across datasets. The feature reduction ratios in the first dataset range from 18.18% to 42.42%, and the performance degradation is between 0.07% and 5.84%. The second set of data shows a reduction rate between 81.77% and 93.5% and a performance degradation between 3.79% and 9.44%.

Masum et al., (2022)[28] provided a framework for ransomware detection and prevention that is based on feature selection and uses various ML methods, such as designs based on neural networks, to categorize security levels. Several ML methods were used for ransomware classification, including Decision Tree (DT), RF, Naïve Bayes (NB), Logistic Regression (LR), and classifiers based on Neural Networks (NN). To test our methodology, the author only used one ransomware dataset.

Sitaula et al., (2022)[29]evaluated thirteen distinct DL models that have already been trained to identify monkeypox. First, the author analyzes the outcomes using four well-established metrics: Accuracy, Precision, Recall, and F1-score. Then, it fine-tunes them by adding universal custom layers to each of them. The authoruses a majority vote on the probabilistic

outputs derived from the best-performing DL models to ensemble them and increase overall performance. Using a publically accessible dataset, it conducts tests that demonstrate the effectiveness of our suggested ensemble strategy. The results show an average Precision of 85.44%, Recall of 85.47%, F1-score of 85.40%, and Accuracy of 87.13%. The suggested strategy may be useful for health practitioners doing mass screenings, according to these promising findings that surpass the state-of-the-art methodologies.

He and Dong, (2019) [30]created a malware detection system that transforms malware files into visual representations and then classes those representations using CNNs. CNNs are trained with spatial pyramid pooling layers (SPP) to handle inputs of varying sizes. The author ran our system on both original and modified data to find out how effectively SPP and picture color space (greyscale/RGB) function. Duplicate API injection can be prevented with greyscale imaging, and the results show that memory restrictions make naïve SPP implementation impractical.

Yuxin, and Zhu, (2019) [31]utilized a deep belief network (DBN) to detect malware by describing its opcodes, which are sequences of instructions. Three baseline malware detection models are evaluated about DBNs in terms of performance: one that utilizes decision trees (DT), one that employs support vector machines, and the third that uses the k-nearest neighbor algorithm. When compared to the baseline models, the DBN model achieves better detection accuracy, according to the experiments.

3. Research Methodology

In this part, the CIC-Malmem2022 dataset for malware classification is presented. Various tools, including PCA for feature extraction and RFE for feature selection, as well as ensemble approaches, which involve DTs,LightGBM, and Gradient Boosting, are explored further in the discussion of malware detection issues.

3.1 Dataset Description

The Canadian Institute for Cybersecurity has released CIC-Malmem2022, an academic dataset designed for studies on malware classification, with a focus on obfuscated malware in particular. Produced by running a feature extraction procedure on memory dumps, this dataset is structured. There are 29,298 benign records and 29,298 malicious records in the 58,596-record dataset.

3.2 Technique Used

In this part, information is provided on the tools that were employed.

- **Principal Component Analysis**

Malware detection feature extraction is based on Principal Component Analysis (PCA). This method employs PCA on a massive dataset consisting of malware samples. By lowering the number of features, PCA decreases the data set, which might increase the detection algorithm's overall performance [32]. Primary component analysis (PCA) reduces the number of components involved in a function by evaluating factors including file size, API calls, system calls, opcode sequences, and byte n-grams [33]. By concentrating on the main components, PCA helps remove noise and capture vital information, allowing for the differentiation of

benign from malicious files. Using PCA to simplify malware data allows for the creation of a detection system that is both simple and effective [34].

- **Recursive Feature Elimination**

A well-liked feature selection method in machine learning, Recursive Feature Elimination (RFE) identified the most critical process features [35]. In malware detection, RFE allows us to choose the most important traits that differentiate between false positives and real samples [36]. The significance of RFE in detecting functionality is crucial since it reveals patterns of harmful activity. The feature-oriented strategy of RFE improves the model's generalizability and prevents overfitting [37]. File size, API calls, device calls, opcode sequences, and byte-n-grams are some of the essential data that RFE emphasizes, which improves the model's ability to identify malware [38]. When RFE is involved in feature set refining, a detection program may accurately and efficiently identify many different kinds of malware strains [39].

- **Decision Tree**

When it comes to detecting and classifying malware, DTis an effective and definable way of doing so [40]. Malware detection often uses DT as classifiers. DT can classify files or system function types as risky or unhealthy by extracting attributes when trained on labeled data sets with instances of both types of software [41]. To improve the accuracy and simplicity of the model in general so has increased, cluster methods combine multiple DTs [42]. When it comes to malware detection, this can improve the generalizability of the model and its ability to detect different types of malicious behavior [43].

- **Light Gradient Boosting Machine**

Large data sets with high dimensionality [44] are valid for LightGBMs. In general, negative research databases have more negative samples than non-negative ones [45]. A framework for dealing with imbalanced datasets in LightGBM helps improve model performance in smaller classes [46]. The ensemble learning approach supported by LightGBM allows for the integration of multiple simple models into one complex one [47]. This can be useful in improving the accuracy of malware detection algorithms in general [48].

- **Gradient Boosting**

Gradient Boosting is an MLmethod that has been successful in several fields including virus detection. Some weak learners such as DT can have their predictions combined to form stronger learners using this ensemble learning technique. As an ensemble approach, many weak learners are used to produce a single strong one [49]. Overall, this ensemble technique enhances both model accuracy and generalizability. Gradient Boosting is resistant to various types of viruses and makes use of the predictions of many DTs [50]. It is this way that it can capture complex data relationships [51].

4. Proposed Methodology

Figure 2shows how the research will be done and then its workflow will be detailed in subsequent sections. This section deals with the main steps for developing a model for detecting malware. Several significant stages are involved inthe creation of a system for the detection of malware. The complete dataset can be generated by gathering many samples from different

families and subtypes of malware. Later on,PCA is used to extract opcode sequences, byte n-grams, file size, API calls, and system calls. When data is missing or partial, it is important to normalize and standardize the attributes to make sure that samples are uniform. When it comes time for feature selection, dimensionality reduction techniques like Wrapper RFE are employed to streamline the dataset while preserving all the relevant information. DT, Support Vector Machines, and CNN are some of the top machine-learning methods to consider in the fourth stage. Ensemble methods enhance accuracy by combining predictions from several models. Using a validation set, the model may be fine-tuned and made accurate after training. The model's accuracy, precision, recall, F1 score, and ROC curve analysis are assessed on a test set in the last phase. Confusion matrices are used to analyze malware categories. In adversarial testing, the model is trained to be more resistant to malicious samples that are intentionally changed. The last step in making sure the model can identify new types of malwares is to either release it into a test environment or provide an API for it to use. To keep the model up-to-date and effective against new malware threats, data must be updated often.

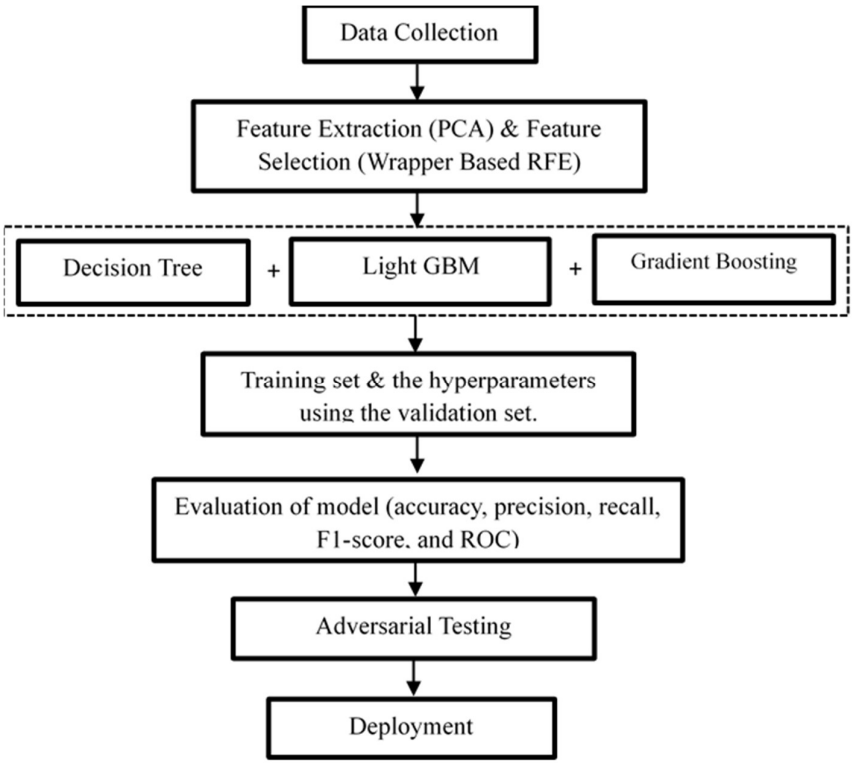

Figure 2. Proposed Methodology

4.1 Proposed Algorithm

Start

1. Data Collection:

Obtain a collection of malware samples representing different families and subtypes.

Let $\boldsymbol{M} = \{\boldsymbol{m_1}, \boldsymbol{m_2}, \dots, \boldsymbol{m_n}\}$represent the collected malware samples.

2. Feature Extraction (Principal Component Analysis):

Apply PCA to profile malware samples.

Let$\boldsymbol{F} = \{\boldsymbol{f_1}, \boldsymbol{f_2}, \dots, \boldsymbol{f_k}\}$act as representations of the features that were extracted, which comprise byte n-grams, opcode sequences, system calls, API calls, and file size.

3. Normalization and Standardization:

Make sure the extracted features are uniform by normalizing and standardizing them.

Let's$\boldsymbol{N} = \{\boldsymbol{n_1}, \boldsymbol{n_2}, \dots, \boldsymbol{n_k}\}$stand in for the homogenized and uniform characteristics.

4. Feature Selection (Wrapper RFE):

Use Wrapper RFE to reduce dimensionality and choose features.

Let $\boldsymbol{S} = \{\boldsymbol{s_1}, \boldsymbol{s_2} \dots, \boldsymbol{s_m}\}$symbolize the chosen attributes following dimensionality reduction.

5. Model Selection:

Select the most effective ML algorithms, such as DT, Support Vector Machines, or Convolutional Neural Networks.

Represent the chosen ML model as the Model.

6. Ensemble Methods:

The selected ML model should be denoted as a Model.

Let **EnsembleModel**represent the combination of models.

7. Training:

Generate training set **TrainSet**, validation set **ValSet**, and test set **TestSet**.

Train the model on **TrainSet**and tune hyperparameters on **ValSet**.

Let **TrainedModel**represent the trained model.

8. Evaluation:

Apply several metrics to the test set to assess the model's performance.

Determine the F1 score, recall, accuracy, and precision; use confusion matrices to conduct ROC curve analysis.

Let **EvaluationMetrics**represent the evaluation results.

9. Adversarial Testing:

Evaluate the model's robustness against deliberate modifications in malware samples by conducting adversarial testing.

Improve the model's resilience through adversarial training.

Let **AdversarialTestResults** represent the outcomes of adversarial testing.

10. Deployment:

Create an API for malware classification or deploy the model in a safe environment.

Maintain the model's efficacy against changing threats by regularly updating it with new data.

Let **DeployedModel**represent the deployed malware detection model.

End

5. Results and Implementation

An ensemble ofMLmodels was used to experiment, including PCA, Wrapper RFE,and the proposed DT, XGBoost, and LightGBMmodels.Resultsfromtheseapproacheswereevaluatedusingthetesting and trainingdataset.

5.1 Hyperparameter using validation set

Table 1 shows the hyperparameters of a model and its performance on the validation set and test set.The model achieved an accuracy of 0.95 on the validation set and 0.95 on the test set. The best hyperparameters for the model were {'n_estimators': 200}.

Table 1: Hyperparameters

Parameters	Values
Accuracy on the validation set	0.95
Best Hyperparameters	{‘n_estimators’:200}
Best accuracy on the validation set	0.95
Accuracy on the test set	0.9996

5.2 Ensemble Confusion Matrix

The ensemble confusion matrix in Figure 3 shows the performance of an ensemble classifier on a four-class classification task. The ensemble classifier consists of multiple base classifiers, and the final prediction is made by combining the predictions of the base classifiers in some way.

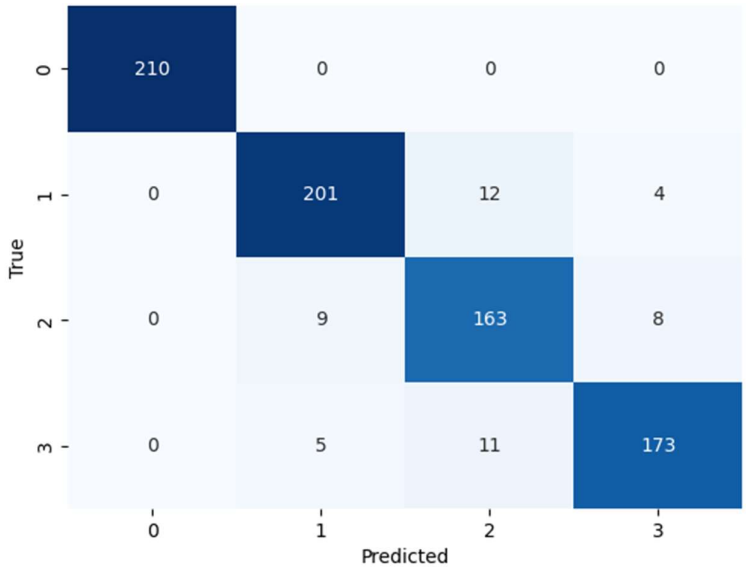

Figure 3: Ensemble Confusion Matrix

5.3 Performance of Machine Learning Classifiers

Table 2 shows the performance of four ML classifiers: Decision Tree, XGBoost, Light GBM, and Ensemble. It compares them based on four metrics: Precision, Recall, F1-score, and Accuracy.

Table 2: Performance of Machine Learning Classifiers

Models	Precision	Recall	f1-score	Accuracy
Decision tree	94.14 %	92.00 %	91.9 %	94.6 %
XGBoost	95.21 %	93.00 %	93.15 %	95.87 %
Light GBM	96.00 %	92.87 %	94.05 %	97.12 %
Ensemble	**99.87 %**	**98.54 %**	**99.85 %**	**99.96 %**

6. Comparison Analysis

Table 3 shows the results of a classification task using three distinct machine-learning approaches. Out of the three methods, the suggested approach gets the best F1-score (99.85) and accuracy (99.96). On the other hand, compared to the other two methods, its recall is slightly lower at 98.54 instead of 100. As applied to this classification position, the suggested strategy seems to have good potential. Its accuracy and F1-score are higher than those of the other two methods.Theproposed method was chosen because it is well-suited to the current categorization job and can achieve the desired results while maintaining a reasonable balance between recall and precision.

Table 3: Comparison Analysis

Author	Technique	Precision	Recall	F1-score	Accuracy
Talukder et al., (2023) [52]	Random Forest, ANN	99 %	99 %	99 %	100 %
Smith et al., (2023) [53]	ADABoost, RandomForest, Decision Tree	100 %	100 %	100 %	99.95 %
Proposed Method	Ensemble (Decision Tree, XGBoost, and light GBM)	**99.87 %**	**98.54 %**	**99.85 %**	**99.96 %**

7. Conclusion

As a conclusion, the increasing prevalence of malware is a major problem in today's cyber environment that threatens the security, privacy, and availability of data stored digitally. This study argues that ensemble machine-learning models should be used to create a more sophisticated malware detection system. The suggested method uses a strategic mix of PCA, RL, DT, XGBoost, and light GBM to improve accuracy and resilience; it leverages the full CIC-Malmem2022 dataset from the Canadian Institute for Cybersecurity.The study's findings highlight the effectiveness of the ensemble model, which achieved a remarkable accuracy rate of 99.96% on the test set. If you want your model to work as well as possible, you should tune its hyperparameters and examine the confusion matrix in depth. The proposed strategy is proven to be superior in identifying malware in comparison to existing methods. At the end of the article, recommended practices are suggested for a safe deployment environment, and the significance of regular upgrades to address changing cybersecurity risks is emphasized.

Overall, this study shows how powerful ensemble ML models can be against malware and gives useful advice on how to build and keep up a reliable malware detection system in the dynamic cybersecurity industry.

Reference

1. Gupta, Ruchika, and S. P. Agarwal. "A comparative study of cyber threats in emerging economies." *Globus: An International Journal of Management & IT* 8, no. 2 (2017): 24-28.
2. R. Komatwar and M. Kokare, ‘‘A survey on malware detection and classification,’’ J. Appl. Secure. Res., pp. 1–31, Aug. 2020.
3. Aslan, Ömer Aslan, and Refik Samet. "A comprehensive review on malware detection approaches." *IEEE Access* 8 (2020): 6249-6271.
4. S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, ‘‘Intelligent vision-based malware detection and classification using deep random forest paradigm,’’ IEEE Access, vol. 8, pp. 206303–206324, 2020.

5. M. Nisa, J. H. Shah, S. Kanwal, M. Raza, M. A. Khan, R. Damaševičius, and T. Blažauskas, ‘‘Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features,’’ Appl. Sci., vol. 10, no. 14, p. 4966, 2020

6. Ö. Aslan, M. Ozkan-Okay, and D. Gupta, ‘‘A review of cloud-based malware detection system: Opportunities, advances, and challenges,’’ Eur. J. Eng. Technol. Res., vol. 6, no. 3, pp. 1–8, Mar. 2021.

7. https://www.akamai.com/glossary/what-is-malware

8. Alenezi, Mohammed N., Haneen Alabdulrazzaq, Abdullah A. Alshaher, and Mubarak M. Alkharang. "Evolution of malware threats and techniques: A review." *International Journal of Communication Networks and Information Security* 12, no. 3 (2020): 326-337.

9. Aboaoja, Faitouri A., Anazida Zainal, Fuad A. Ghaleb, Bander Ali Saleh Al-rimy, Taiseer Abdalla Elfadil Eisa, and Asma Abbas Hassan Elnour. "Malware detection issues, challenges, and future directions: A survey." *Applied Sciences* 12, no. 17 (2022): 8482.

10. Wazid, Mohammad, Ashok Kumar Das, Joel JPC Rodrigues, Sachin Shetty, and Youngho Park. "IoMT malware detection approaches analysis and research challenges." *IEEE Access* 7 (2019): 182459-182476.

11. Andrade, Eduardo de O., José Viterbo, Cristina N. Vasconcelos, Joris Guérin, and Flavia Cristina Bernardini. "A model based on LSTM neural networks to identify five different types of malware." *Procedia Computer Science* 159 (2019): 182-191.

12. Matrosov, Alex, Eugene Rodionov, and Sergey Bratus. *Rootkits and boot kits: reversing modern malware and next generation threats*. No Starch Press, 2019.

13. Dwivedi, Aarushi, Krishna Chandra Tripathi, and M. L. Sharma. "Advanced keylogger-a stealthy malware for computer monitoring." *Asian Journal For Convergence In Technology (AJCT) ISSN-2350-1146* 7, no. 1 (2021): 137-140

14. Cadden, James, Thomas Unger, Yara Awad, Han Dong, Orran Krieger, and Jonathan Appavoo. "SEUSS: skip redundant paths to make serverless fast." In *Proceedings of the Fifteenth European Conference on Computer Systems*, pp. 1-15. 2020.

15. Sebastio, Stefano, Eduard Baranov, Fabrizio Biondi, Olivier Decourbe, Thomas Given-Wilson, Axel Legay, Cassius Puodzius, and Jean Quilbeuf. "Optimizing symbolic execution for malware behavior classification." *Computers & Security* 93 (2020): 101775.

16. Or-Meir, Ori, Nir Nissim, Yuval Elovici, and Lior Rokach. "Dynamic malware analysis in the modern era—A state of the art survey." *ACM Computing Surveys (CSUR)* 52, no. 5 (2019): 1-48.

17. Li, Hongcheng, Jianjun Huang, Bin Liang, Wenchang Shi, Yifang Wu, and Shilei Bai. "Identifying parasitic malware as outliers by code clustering." *Journal of Computer Security* 28, no. 2 (2020): 157-189.

18. Tian, Donghai, Rui Ma, Xiaoqi Jia, and Changzhen Hu. "A kernel rootkit detection approach based on virtualization and machine learning." *IEEE Access* 7 (2019): 91657-91666.

19. Liu, Zhifeng, Desheng Zheng, Xinlong Wu, Jixin Chen, Xiaolan Tang, and Ziyong Ran. "VABox: A virtualization-based analysis framework of virtualization-obfuscated packed executables." In *Advances in Artificial Intelligence and Security: 7th International Conference, ICAIS 2021, Dublin, Ireland, July 19-23, 2021, Proceedings, Part III 7*, pp. 73-84. Springer International Publishing, 2021.

20. Smith, S. E. *The Geek and the Sheikh*. Montana Publishing, 2023.

21. Mamchenko, Mark, and Alexey Sabanov. "Exploring the taxonomy of USB-based attacks." In *2019 Twelfth International Conference" Management of large-scale system development"(MLSD)*, pp. 1-4. IEEE, 2019.

22. Hussain, Abrar, Muhammad Asif, Maaz Bin Ahmad, Toqeer Mahmood, and M. Arslan Raza. "Malware detection using machine learning algorithms for Windows platform." In Proceedings of International Conference on Information Technology and Applications: ICITA 2021, pp. 619-632. Singapore: Springer Nature Singapore, 2022.

23. Akhtar, Muhammad Shoaib, and Tao Feng. "Malware Analysis and Detection Using Machine Learning Algorithms." Symmetry 14, no. 11 (2022): 2304.

24. Narayanan, Barath Narayanan, and Venkata Salini Priyamvada Davuluru. "Ensemble malware classification system using deep neural networks." *Electronics* 9, no. 5 (2020): 721.

25. Lad, Sumit S., and Amol C. Adamuthe. "Malware classification with improved convolutional neural network model." *International Journal of Computer Network & Information Security* 12, no. 6 (2020): 30-43.

26. Kumar, Nitesh, Subhasis Mukhopadhyay, Mugdha Gupta, Anand Handa, and Sandeep K. Shukla. "Malware classification using early stage behavioral analysis." In *2019 14th Asia Joint Conference on Information Security (AsiaJCIS)*, pp. 16-23. IEEE, 2019.

27. Alomari, Esraa Saleh, Riyadh RahefNuiaa, Zaid Abdi AlkareemAlyasseri, Husam Jasim Mohammed, Nor Samsiah Sani, Mohd Isrul Esa, and Bashaer Abbuod Musawi. "Malware detection using deep learning and correlation-based feature selection." Symmetry 15, no. 1 (2023): 123.

28. Masum, Mohammad, Md Jobair Hossain Faruk, Hossain Shahriar, Kai Qian, Dan Lo, and Muhaiminul Islam Adnan. "Ransomware classification and detection with machine learning algorithms." In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0316-0322. IEEE, 2022.

29. Sitaula, Chiranjibi, and Tej Bahadur Shahi. "Monkeypox virus detection using pre-trained deep learning-based approaches." Journal of Medical Systems 46, no. 11 (2022): 78.

30. He, Ke, and Dong-Seong Kim. "Malware detection with malware images using deep learning techniques." In *2019 18th IEEE International Conference on trust, security, and privacy in computing and communications/13th IEEE International Conference on big data science and engineering (TrustCom/BigDataSE)*, pp. 95-102. IEEE, 2019.

31. Yuxin, Ding, and Zhu Siyi. "Malware detection based on a deep learning algorithm." *Neural Computing and Applications* 31 (2019): 461-472.

32. Kwon, Young-Man, Jae-Ju An, Myung-Jae Lim, Seongsoo Cho, and Won-Mo Gal. "Malware classification using smash encoding and PCA (MCSP)." *Symmetry* 12, no. 5 (2020): 830.

33. Tiwari, Suman R., and Ravi U. Shukla. "An android malware detection technique using optimized permission and API with PCA." In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 2611-2616. IEEE, 2018.

34. Rami, Khyati, and Vinod Desai. "Malware Detection Framework Using PCA Based ANN." In *Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, March 26–27, 2020, Revised Selected Papers 1*, pp. 298-313. Springer Singapore, 2020.

35. Mahmoud, Baffa Sani, and Ahmad Baita Garko. "A Machine Learning Model for Malware Detection Using Recursive Feature Elimination (RFE) For Feature Selection and Ensemble Technique."

36. Al Sarah, Neamat, Fahmida Yasmin Rifat, Md Shohrab Hossain, and Husnu S. Narman. "An efficient android malware prediction using Ensemble machine learning algorithms." *Procedia Computer Science* 191 (2021): 184-191.

37. Gunduz, Hakan. "Malware detection framework based on graph variational autoencoder extracted embeddings from API-call graphs." *PeerJ Computer Science* 8 (2022): e988.

38. Kornyo, Oliver, Michael Asante, Richard Opoku, Kwabena Owusu-Agyemang, Benjamin Tei-Partey, Emmanuel Kwesi Baah, and Nkrumah Boadu. "Botnet Attacks Classification in AMI Networks with Recursive Feature Elimination (RFE) and Machine Learning Algorithms." *Computers & Security* (2023): 103456.

39. Manzil, HashidaHaidros Rahima, and Manohar S. Naik. "COVID-Themed Android Malware Analysis and Detection Framework Based on Permissions." In *2022 International Conference for Advancement in Technology (ICONAT)*, pp. 1-5. IEEE, 2022.

40. Kumar, Rajesh, and S. Geetha. "Malware classification using XGboost-Gradient boosted decision tree." *Adv. Sci. Technol. Eng. Syst* 5 (2020): 536-549.

41. Galen, Colin, and Robert Steele. "Empirical measurement of performance maintenance of gradient boosted decision tree models for malware detection." In *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 193-198. IEEE, 2021.

42. Ullah, Faizan, Qaisar Javaid, Abdu Salam, Masood Ahmad, Nadeem Sarwar, Dilawar Shah, and Muhammad Abrar. "Modified decision tree technique for ransomware detection at runtime through API calls." *Scientific Programming* 2020 (2020).

43. Mustafa Hilal, Anwer, Siwar Ben Haj Hassine, Souad Larabi-Marie-Sainte, Nadhem Nemri, Mohamed K. Nour, Abdelwahed Motwakel, Abu Sarwar Zamani, and Mesfer Al Duhayyim. "Malware Detection Using Decision Tree Based SVM Classifier for IoT." *Computers, Materials & Continua* 72, no. 1 (2022).

44. Al-Kasassbeh, Mouhammd, Mohammad A. Abbadi, and Ahmed M. Al-Bustanji. "LightGBM algorithm for malware detection." In *Intelligent Computing: Proceedings of the 2020 Computing Conference, Volume 3*, pp. 391-403. Springer International Publishing, 2020.

45. Gao, Yun, Hirokazu Hasegawa, Yukiko Yamaguchi, and Hajime Shimada. "Malware detection using LightGBM with a custom logistic loss function." *IEEE Access* 10 (2022): 47792-47804.

46. Abbadi, M., M. Al-Bustanji, and Mouhammd Al-Kasassbeh. "Robust Intelligent malware detection using LightGBM Algorithm." *International Journal of Innovative Technology and Engineering* 9, no. 6 (2020): 1253-1263.

47. Zhang, ZheMing. "Microsoft Malware Prediction Using LightGBM Model." In *2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pp. 41-44. IEEE, 2022.

48. Ghourabi, Abdallah. "A security model based on light gum and transformer to protect healthcare systems from cyberattacks." *IEEE Access* 10 (2022): 48890-48903.

49. Thosar, Keshav, Pranay Tiwari, Revanth Jyothula, and Dayanand Ambawade. "Effective malware detection using gradient boosting and convolutional neural network." In *2021 IEEE Bombay Section Signature Conference (IBSSC)*, pp. 1-4. IEEE, 2021.

50. Yousefi-Azar, Mahmood, Vijay Varadharajan, Len Hamey, and Shiping Chen. "Mutual Information and Feature Importance Gradient Boosting: automatic byte n-gram feature reranking for Android malware detection." *Software: Practice and Experience* 51, no. 7 (2021): 1518-1539.

51. Turnip, ToguNovriansyah, Amsal Situmorang, Ayu Lumbantobing, Josua Marpaung, and Samuel IG Situmeang. "Android malware classification based on permission categories using extreme gradient boosting." In *Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology*, pp. 190-194. 2020.

52. Talukder, Md Alamin, KhondokarFida Hasan, Md Manowarul Islam, Md Ashraf Uddin, Arnisha Akhter, Mohammad Abu Yousuf, Fares Alharbi, and Mohammad Ali Moni. "A dependable hybrid machine learning model for network intrusion detection." *Journal of Information Security and Applications* 72 (2023): 103405.

53. Smith, Daryle, Sajad Khorsandroo, and Kaushik Roy. "Supervised Feature Selection to Improve the Accuracy for Malware Detection." (2023).