International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 02 February 2024
Page No: 261 - 270
https://irjaeh.com

Ensemble Learning for Robust Malware Detection in the Windows 7 Environment

Gururaja H S[1], Nethra Khandige[2], Ninad N Nayak[3], Raghav Srivatsan[4], Sumedha Athreya N[5]*

[1] *Assistant Professor, Department of ISE, B.M.S. College of Engineering, Bengaluru, India.*

[2,3,4,5] *UG Student, Department of CSE, PES University, Bengaluru, India.*

Email id: *gururajhs.ise@bmsce.ac.in[1], nethrakhandige@gmail.com[2], ninadn511@gmail.com[3], raghavsrivatsan3002@gmail.com[4], sumedhavn@gmail.com[5]*

****Corresponding Author Orcid ID:*** *https://orcid.org/0000-0002-9718-4672*

Abstract

In today's digital era, the exponential growth in the dissemination of malicious software presents a formidable and pervasive threat to individuals and organizations alike. Malware, comprising of computer code or scripts, exhibits a propensity to override computer systems and engage in unauthorized activities, such as the illicit extraction of valuable information. The magnitude of such control is particularly perilous in the contemporary landscape. The rapid and constant evolution of malware compounds this risk, necessitating the development and implementation of sophisticated countermeasures to combat this pressing issue. In response to this challenge, this study has turned to the utilization of ensemble learning techniques as a means of effectively and accurately identifying and detecting malware. Specifically, the investigation focuses on the classification of applications on the MS Windows 7 and 8 operating systems as either malicious or benign, by leveraging static and dynamic features extracted from these applications. The foundational models employed in this study consist of recurrent neural networks (RNNs), trained on the dynamic features of malware, and convolutional neural networks (CNNs), and trained on the static features of malware. This comprehensive approach ensures that no malware goes undetected. Multiple techniques for constructing an ensemble model, such as Boosting, Stacking, and Bagging, are thoroughly examined and analyzed. Ultimately, the Bagging technique is deemed the most suitable and is consequently implemented in this study.

Keywords: *Ensemble model, Malware Detection, RNN, CNN, Boosting, Stacking, Bagging.*

1. Introduction

In the contemporary world we live in, where electronic devices have become the quintessential and most trusted repository for personal information, it would be deemed highly scandalous if these devices were to be compromised without the user's knowledge. The question of whether or not to take the risk of storing confidential information in the present cyberspace realm poses a significant challenge. It is important to note that the utilization of Windows operating system continues to be prevalent on a global scale, which consequently renders it a primary target for cybercriminals. The compromise of a device or System not only shatters the trust placed in it, but it also disrupts the lives of individuals. Furthermore, it has the potential to destabilize economies and even entire nations. Consequently, the detection of malware plays a pivotal role in safeguarding against such catastrophic consequences. The primary objective of this research endeavor is to effectively detect malware specifically in the Windows operating system. The utilization of an ensemble model serves as a key strategy in enhancing the performance of a model, thereby ensuring high accuracy [2]. This is of utmost importance, particularly in the context of malware detection. It is absolutely unacceptable for any form of malware to evade detection. By employing an ensemble model, one can achieve greater stability and reduce the presence of noise in the system. The present study focuses on the creation of an ensemble model by incorporating both static and dynamic features of malware. This approach guarantees that no form of malware goes undetected, thereby fortifying the overall effectiveness of the system.

1.2 Background

In recent times, the prevailing menace of nefarious software (malware) has experienced a significant escalation in the realm of digital technology, thereby presenting grave dangers to both individuals and organizations alike. Windows 7, being an operating system that is widely utilized, has garnered substantial attention from developers of malware owing to its vast user base. The continuous evolution of malware, coupled with its capacity to supersede computer systems and partake in unauthorized activities, necessitates the implementation of an advanced and adaptive approach to detection. Conventional techniques often fall short in keeping up with the ever-changing nature of malware. Ensemble learning, a methodology that entails the amalgamation of multiple models, presents a promising avenue for augmenting the resilience and accuracy of systems designed for the detection of malware. The primary objective of this research undertaking is to explore the intricacies and specific challenges that malware poses within the Windows 7 environment, as well as to leverage the techniques of ensemble learning in order to cultivate a detection mechanism that is more steadfast and efficacious.

1.3 Motivation

The impetus behind the undertaking of this investigation arises from the pressing necessity to tackle the increasingly dire peril of malicious software in the specific context of Windows 7, which happens to be an extensively employed operating system fraught with notable vulnerabilities. The existing techniques utilized for the detection of malware, which often rely on solitary models, find themselves struggling to keep pace with the ever-advancing complexity and swift evolution of such nefarious software. The motivation at hand is to delve into and put into practice ensemble learning techniques, which have garnered recognition for their capacity to enhance accuracy and adaptability when it comes to detection. By directing our attention towards the environment encompassing Windows 7, the primary objective of this research is to provide insights and solutions that are meticulously tailored to the unique characteristics and challenges posed by this particular operating system. Ultimately, the overarching goal is to furnish a malware detection system [7] that is both sturdy and sophisticated, one that not only accurately identifies threats but also effectively adjusts to the ever-shifting landscape of malware within Windows 7. The ramifications stemming from the outcomes of this investigation possess the potential to substantially augment cybersecurity measures, especially for users and organizations that heavily rely on the Windows 7 platform [14].

2. Literature Review

The authors [1] have conducted a comprehensive study on the file monitoring method aimed at detecting malicious files using an ensemble method. This method involves classifying a file into one of the nine types of malwares. To convert the ASM data, the authors have utilized opcode. Furthermore, the byte files have been transformed into png blocks and subsequently normalized. By combining the byte pre-processed data and the ASM pre-processed data using Artificial Neural Networks (ANN), an ensemble model has been constructed. This ensemble model has achieved an impressive accuracy rate of 97% with a loss of 9%. Moreover, the performance of this model surpasses that of conventional Machine Learning (ML) methods as well as deep learning methods, thus clearly demonstrating the effectiveness of ensemble learning. In a study conducted by the authors [2] from George Mason University, a hardware-driven malware detection technique that primarily relies on the execution hardware has been explored. This technique involves assigning scores to features based on their importance and relevance. Through the utilization of feature reduction techniques, closely linked hardware counters are identified and ranked accordingly. It has been observed that a majority of machine learning classifiers perform well before feature reduction, yielding high accuracy rates of over 80%. However, the introduction of ensemble learning has shown to further enhance the performance of these classifiers. The ensemble detector proposed in [3] aims to enhance the resistance to specific evasion techniques. This is achieved by increasing the unpredictability of the analysis process, particularly in terms of detection. Additionally, the suggested methods also assist in improving the detection rates when dealing with unidentified malware families, thus eliminating the necessity to constantly keep up with the evolving nature of malware. Ensemble Droid [4] is an ensemble learning algorithm-based malware detection approach specifically designed for the Android system. It combines both homogeneous stacking and heterogeneous stacking within the ensemble model. Remarkably, this approach has achieved an accuracy rate of 91% for both homogeneous and heterogeneous stacking, showcasing its effectiveness in detecting malware within the Android system. The ensemble model SMASH [5] takes into account both software and hardware features, resulting in the production of feature vectors. To extract the three key features of malware, namely API call sequence, memory dumb grayscale image, and performance counter count, a dynamic sandbox is utilized. The incorporation of hardware features helps to balance out the susceptibility of software feature evasion, while software features compensate for any lack in hardware feature detection precision. As a result, an impressive accuracy rate of 97.8% has been achieved. The authors in [6] have developed an ensemble model that not only aids in the detection of malware but also evaluates the efficiency of different classifiers in characterizing and predicting malware. This model has demonstrated its ability to predict unknown malware with an accuracy rate of 99.8% and a false positive rate of 0.2%, thereby highlighting its effectiveness in the field of malware detection. In general, the features of malware can be gathered through either static or dynamic analysis of malware samples [8]. Static analysis involves analyzing and extracting features from the meta-data of the binary file without executing it. This includes specific string patterns, byte sequences, operation codes, and library calls. However, static analysis alone is not sufficient due to the obfuscation methodologies employed by malware authors. On the other hand, dynamic analysis requires the executable file to be running in order to aggregate generated outcomes for feature extraction. This method involves extracting features such as system calls, registry changes, memory usage, and network behavior.

3. Existing System

The current state of malware detection in the Windows 7 environment is confronted with a range of significant challenges. Traditional antivirus solutions, which often rely on signature-based detection methods, find it difficult to keep up with the rapidly evolving landscape of malware. As the sophistication of malicious software continues to increase, these conventional systems are limited in their ability to accurately identify and prevent new threats. Additionally, the dynamic and polymorphic nature of malware allows it to easily evade signature-based approaches, making them less effective over time. Although some heuristic and behavior-based detection methods have been implemented, their adaptability and robustness are not always sufficient to provide comprehensive protection. Given that the Windows 7 environment is a prime target for malware developers, a more advanced and proactive approach is required to combat the diverse array of threats it faces. The inadequacies of the current systems highlight the need for innovative methodologies that can enhance detection accuracy, resilience, and adaptability. The shortcomings of the current systems, coupled with the escalating threat landscape, act as the driving force behind the proposed research. The limitations of single-model detection systems and the urgency to address the evolving nature of malware in the Windows 7 environment emphasize the necessity for the exploration and implementation of ensemble learning techniques, as described in the title and abstract of the research paper. The constraints of the existing system create a compelling motivation to seek novel and more effective approaches to malware detection, ultimately aiming to strengthen cybersecurity measures in the face of an ever-changing threat landscape.

4. Proposed Methodology

4.1 Overview

The process of obtaining the desired dataset involves making modifications to the original dataset. This modification is necessary in order to meet the specific requirements and objectives of the desired dataset. The dataset is first analyzed to identify any shortcomings or limitations that may hinder its usefulness for the intended purpose. Once these shortcomings are identified, various techniques and methodologies are employed to modify the dataset in order to address these limitations. This modification process may involve cleaning the data, removing irrelevant or redundant information, transforming the data into a more suitable format, or even merging multiple datasets together. The aim of this modification process is to ensure that the resulting dataset is of high quality, accurate, and reliable, thereby making it more suitable for the intended purpose. The RNN model is a type of neural network model that is specifically designed to handle dynamic features. These dynamic features refer to variables that change over time or have a temporal aspect to them. The RNN model is particularly useful in applications where the input data is sequential, such as in natural language processing, speech recognition, and time series analysis. The RNN model has the ability to capture and utilize the temporal information present in the input data, which makes it well-suited for tasks that require an understanding of the context and history of the data. The performance of the RNN model is evaluated based on various metrics, such as accuracy, precision, recall, and F1 score. These metrics provide insights into how well the RNN model is able to learn and predict the patterns and trends in the dynamic features of the data. On the other hand, the CNN model is a type of neural network model that is specifically designed to handle static features. These static features refer to variables that do not change over time and are independent of the sequence or order in which they appear. The CNN model is particularly useful in applications where the input data is structured, such as in image recognition, object detection, and sentiment analysis. The CNN model has the ability to extract and learn hierarchical representations of the input data, which makes it well-suited for tasks that require an understanding of the spatial relationships and patterns in the static features. The performance of the CNN model is evaluated based on various metrics, such as accuracy, precision, recall, and F1 score. These metrics provide insights into how well the CNN model is able to learn and classify the patterns and features in the static features of the data. Furthermore, the ensemble model is a technique that combines multiple models in order to improve the overall performance and accuracy of the predictions. The ensemble model is particularly useful in situations where a single model may not be sufficient to capture the complexity and variability of the data. By combining different models, the ensemble model is able to leverage the strengths and mitigate the weaknesses of each individual model, thereby improving the overall performance and accuracy of the predictions. There are various methods and techniques for creating an ensemble model, such as bagging, boosting, and stacking. These methods differ in how they combine the predictions of the individual models and how they handle the diversity and correlation among the models. The most suitable method for creating an ensemble model depends on the specific requirements and characteristics of the data, as well as the objectives and constraints of the task at hand. Therefore, a thorough comparison and evaluation of different ensemble methods is necessary in order to determine the most suitable method for a given problem domain. This comparison and evaluation may involve analyzing the performance of the ensemble models using various metrics, such as accuracy, precision, recall, and F1 score. These metrics provide insights into how well the ensemble models are able to learn and predict the patterns and trends in the data, and can help in selecting the most suitable method for a given problem domain.

4.2 Dataset

The dataset referred to as dataset [15] encompasses a collection of malware samples that have been meticulously analyzed and have had their features systematically extracted from a total of 595 files. It is important to note that these files were specifically obtained from the operating systems MS Window 7 and 8. In order to embark on the creation of a Recurrent Neural Network (RNN) based model that can effectively handle dynamic features, as well as a Convolutional Neural Network (CNN) based model, it is imperative to have two distinct datasets at one's disposal. These datasets, which are indispensable for the creation of the aforementioned models, include the dynamic features dataset and the static features dataset. The available dataset that serves as the foundation for these models effortlessly encompasses both static and dynamic features. In order to ensure the utmost efficiency and organization, this dataset is divided into two separate entities, with each dataset catering to either dynamic or static features respectively. After the original dataset has been successfully partitioned, it is important to take note that the attributes of the malware samples remain intact within each of the resulting datasets. However, it is pertinent to acknowledge that the attributes within each dataset differ significantly from one another. For instance, the dynamic dataset consists of a total of 3550 instances, each of which possesses 23 distinct attributes. Prior to the utilization of this dataset, meticulous data cleaning and preprocessing procedures are carried out in order to ensure its reliability and accuracy. Conversely, the static dataset proudly boasts 3550 instances, with each instance being armed with an impressive 182 attributes. In order to guarantee the utmost accuracy and precision within the datasets, an array of comprehensive data preprocessing and data cleaning techniques are meticulously employed. Notably, redundant data and outliers are promptly identified and subsequently discarded from the datasets. Furthermore, in order to facilitate the data organization process, the filename attribute, which is inherently a character, is ingeniously encoded utilizing the highly effective Label Encoder technique. As a result, this allows for the seamless transformation of the dataset into both training and testing data, thereby enhancing the overall efficacy and efficiency of the models that are to be constructed.

4.3 RNN Model

Recurrent Neural Network (RNN), an artificial neural network that operates on sequential data, is characterized by its deep neural architecture [12]. Given its suitability for sequence data, RNN holds potential for detecting malware due to its ability to analyze dynamic features recorded during program execution. The dataset used for the RNN model as depicted in Figure 1 encompasses all dynamic features of Windows 7 and 8 malware that were captured. These dynamic features [11] serve as essential input for the model's training and evaluation. The architecture of the RNN model as depicted in Table 1 and Table 2 comprises a collection of TensorFlow layers that have been incorporated into the Sequential model. Among these TensorFlow layers, the LSTM layer serves as the input layer, while the Dense, Batch Normalization, and additional LSTM layers function as hidden layers, processing the data at each step. Moreover, a dense layer with a sigmoid activation function is employed to capture the output results.

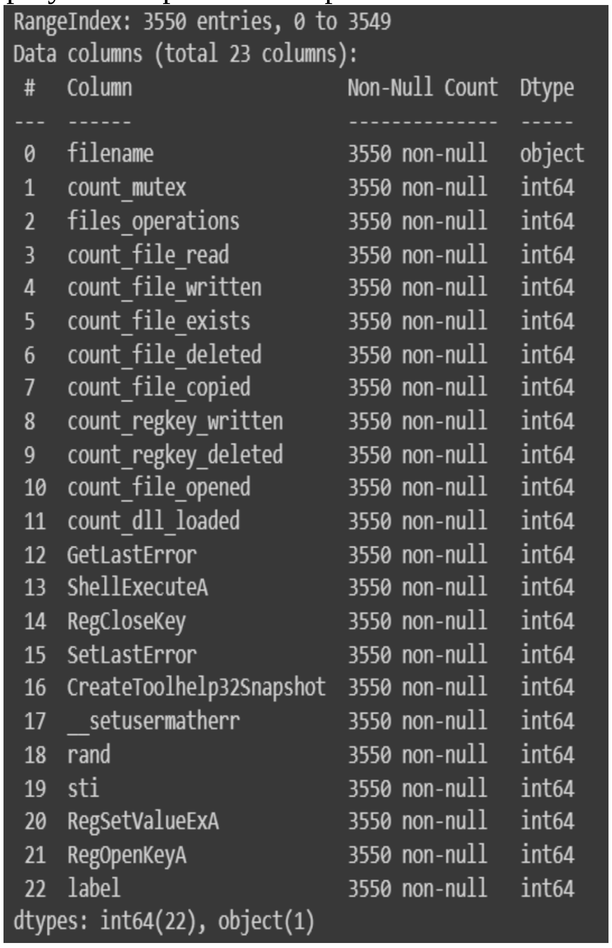
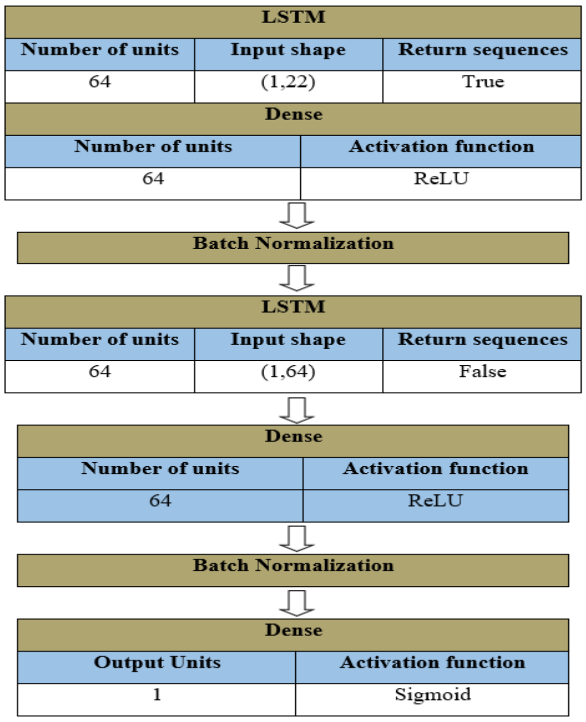

```
RangeIndex: 3550 entries, 0 to 3549
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   filename                  3550 non-null   object
 1   count_mutex               3550 non-null   int64
 2   files_operations          3550 non-null   int64
 3   count_file_read           3550 non-null   int64
 4   count_file_written        3550 non-null   int64
 5   count_file_exists         3550 non-null   int64
 6   count_file_deleted        3550 non-null   int64
 7   count_file_copied         3550 non-null   int64
 8   count_regkey_written      3550 non-null   int64
 9   count_regkey_deleted      3550 non-null   int64
 10  count_file_opened         3550 non-null   int64
 11  count_dll_loaded          3550 non-null   int64
 12  GetLastError              3550 non-null   int64
 13  ShellExecuteA             3550 non-null   int64
 14  RegCloseKey               3550 non-null   int64
 15  SetLastError              3550 non-null   int64
 16  CreateToolhelp32Snapshot  3550 non-null   int64
 17  __setusermatherr          3550 non-null   int64
 18  rand                      3550 non-null   int64
 19  sti                       3550 non-null   int64
 20  RegSetValueExA            3550 non-null   int64
 21  RegOpenKeyA               3550 non-null   int64
 22  label                     3550 non-null   int64
dtypes: int64(22), object(1)
```
Figure 1 Dynamic Features in The Dataset

Table 1 Parameters of the RNN model

Parameter	Value
Optimizer	Adam
Loss function	Binary Cross-entropy
Batch size	30
Epochs	50

To ensure optimal performance, the model is compiled with binary Cross-entropy as the loss function and Adam as the chosen optimizer. Training the model involves iterating over 35 epochs, allowing the model to learn from the data and improve its performance over time.

Table 2 Parameters of Layers in the RNN Model



LSTM		
Number of units	Input shape	Return sequences
64	(1,22)	True

Dense	
Number of units	Activation function
64	ReLU

Batch Normalization

LSTM		
Number of units	Input shape	Return sequences
64	(1,64)	False

Dense	
Number of units	Activation function
64	ReLU

Batch Normalization

Dense	
Output Units	Activation function
1	Sigmoid

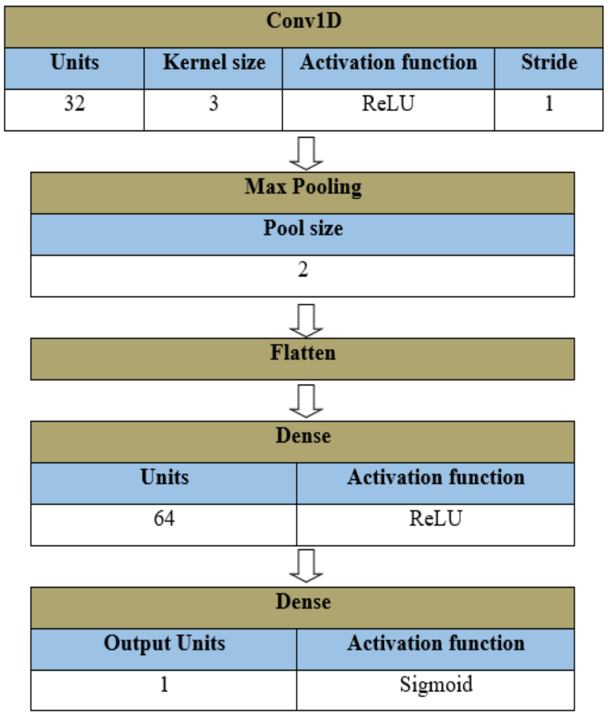
4.4 CNN Model

The Convolutional Neural Network (CNN) is a computational model inspired by the structure and functionality of the human brain. It is an artificial neural network that is specifically designed for recognizing complex patterns in various types of data. In the context of malware detection, the CNN can be utilized to identify patterns that are indicative of malicious behavior. By analyzing the static features of a program or application, such as its properties and characteristics, the CNN can effectively detect the presence of malware.

Table 3 Parameters of the CNN Model

Parameter	Value
Optimizer	Adam
Loss function	Binary Cross-entropy
Epochs	50

Table 4 Parameters of Layers in the CNN Model



Conv1D			
Units	Kernel size	Activation function	Stride
32	3	ReLU	1

Max Pooling
Pool size
2

Flatten

Dense	
Units	Activation function
64	ReLU

Dense	
Output Units	Activation function
1	Sigmoid

The architecture of the CNN model as depicted in Table 3 and Table 4. The dataset used for training the CNN model consists of the extracted static features of malware. These features are obtained without actually executing the program or running the application, allowing for a comprehensive analysis of the program's properties. By utilizing these static features, the CNN model can learn to distinguish between normal and malicious software. In order to prepare the training data and testing data for the CNN model, they need to be reshaped to match the input requirements of the model. This reshaping process ensures that the data is in a format that can be effectively processed by the CNN. The CNN model is built on the concept of adding multiple TensorFlow layers to a sequential model. TensorFlow is a popular open-source machine learning framework [9] that provides a range of tools and functionalities for building and training neural networks. The Tensor Flow layers used in the CNN model include a Conv1D layer with 32 filters and a kernel size of 3. This layer is responsible for performing convolutional operations on the input data, extracting relevant features. The activation function used in this layer is the Rectified Linear Unit (ReLU), which introduces non-linearity into the model. This helps in capturing complex patterns and relationships in the input data. Other layers used in the CNN model include MaxPooling1D, Flatten, and Dense layers. The MaxPooling1D layer is used to down sample the input data, reducing its spatial dimensions. The Flatten layer is then used to convert the multidimensional data into a one-dimensional vector, which can be processed by the subsequent dense layers. The Dense layers are fully connected layers that apply a linear transformation to the input data. The activation function used in these layers is also ReLU, which helps in introducing non-linearity and capturing complex patterns. The output layer of the CNN model consists of a dense layer with the sigmoid activation function. This activation function is used to maintain non-linearity in the output, allowing for the modeling of complex relationships between the input data and the output labels. The sigmoid function also ensures that the output values are in the range of 0 to 1, representing the probability of a given input belonging to a certain class.

5. Results and Discussion

5.1 Confusion Matrix for RNN and CNN Models for Malware Detection

A confusion matrix is a table that is used to assess the performance of an algorithm used for classification. It is mainly used to evaluate metrics like accuracy, precision, recall and F1 score.

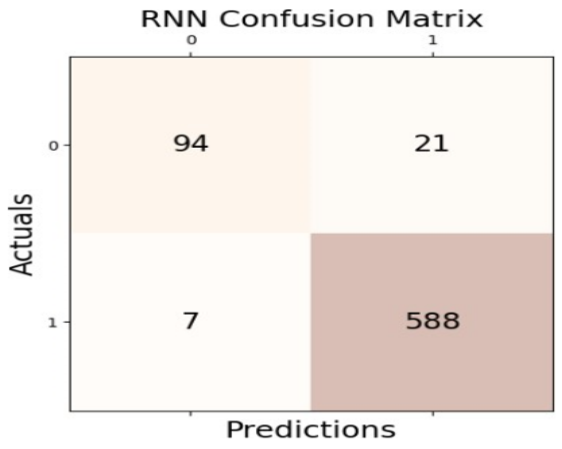


Figure 2 Confusion Matrix for RNN Model

The confusion matrix for the RNN model is depicted in Figure 2. Evaluating the RNN model gives an accuracy rate of **0.838**. In summary, the RNN model's architecture encapsulates various TensorFlow layers, each playing a crucial role in processing the sequential data. By training the model using the provided dataset and optimizing it with suitable loss functions and optimizers, the model can effectively detect malware based on the dynamic features recorded during program execution.

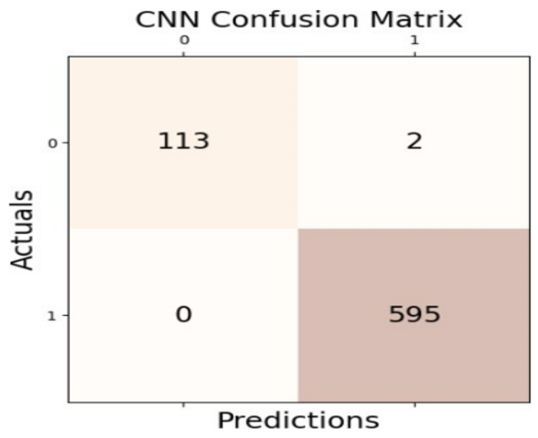


Figure 3 Confusion Matrix for CNN Model

The confusion matrix for the CNN model is depicted in Figure 3. Evaluating the CNN model gives an accuracy rate of **0.9408**. In summary, the architecture of the CNN model for malware detection involves the utilization of Conv1D, MaxPooling1D, Flatten, and Dense layers, each with specific functionalities and activation functions. These layers collectively enable the model to effectively analyze the static features of malware and make accurate predictions. The CNN model, built on the TensorFlow framework, provides a powerful tool for detecting and combating the threat of malware.

5.2 Results of Ensemble Models for Malware Detection

The study analyses different ensemble methods [10] and arrives at one most suitable model for malware detection. In machine learning, ensemble learning is an approach which combines multiple easy to build and computationally inexpensive models which ensures that the predictive power is as strong as a single strong predictive model [13-15]. The comparison is made between Boosting, Stacking and Bagging models and the comparison of results are provided in Figure 4.

- **Boosting:** This algorithm assigns equal weight to each data sample and makes prediction based on it. There are 2 types of boosting, Adaptive and Gradient Boosting. Gradient Boosting is where equal weights are not given to incorrectly classified samples. Gradient boosting is used in this model. It optimizes the loss function by generating base learners sequentially so that present base learner is always more effective than the previous one. The results of this model are presented in Table 5.
- **Stacking:** Stacking considers many heterogeneous weak learners, learns them in parallel, and combines them by training a Meta learner to out a prediction based on different weak learner's prediction. The weak learners used in the model are Decision Tree and Random Forest. The algorithm takes the outputs of sub models as input and attempts to learn how to best combine the input predictions to make a better output prediction. The results of this model are presented in Table 6.
- **Bagging:** Bagging also knows as Bootstrap aggregation is mainly used in high noise dataset to reduce variance and increase accuracy. The model is set up on ensemble of 100 base estimators using a maximum of 80 percent of training data and calculates out-of-bag (OOB) score using Decision Tree as base estimator. The results of this model are presented in Table 7.

Table 5 Results of Boosting model

Metrics	Value
Accuracy	0.9788732394366197
Precision	0.9785346209566838
Recall	0.9788732394366197
F1 score	0.978343865642811

Table 6 Results of Stacking model

Metrics	Value
Accuracy	0.9690140845071423
Precision	0.9643435980551054
Recall	1.0
F1 score	0.9818481848184819

Table 7 Results of Bagging model

Metrics	Value
Accuracy	0.9830985915492958
Precision	0.9821138211382114
Recall	0.9983193277310924
F1 score	0.9900826446280991

Upon engaging in a meticulous and exhaustive examination and development of numerous ensemble models, it has been ascertained that the bagging model showcases an exceedingly commendable accuracy rate of 98%. Moreover, it surpasses alternative model methodologies such as boosting and stacking in relation to the comprehensive efficacy it demonstrates. The comparison of results for various ensemble models is provided in Figure 4.

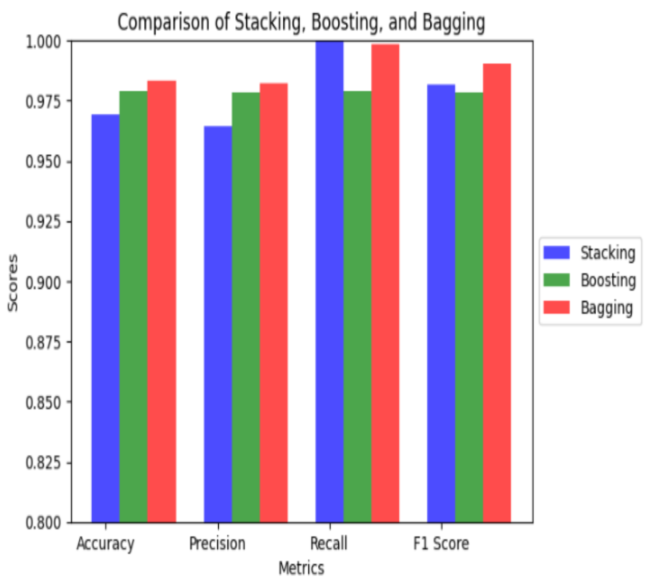


Figure 4 Comparison of Results of Ensemble Models

Conclusion and Future Work

This specific research endeavor effectively utilizes the ensemble learning approach in order to identify and detect instances of malicious software within Windows operating systems. This is achieved through the utilization of both static and dynamic features associated with malware. It is important to acknowledge that relying solely on a single model cannot always ensure that all malware is effectively detected and mitigated. Hence, the development of an ensemble model that incorporates both static and dynamic features of malware is imperative. The escalating issue of operating systems being vulnerable to malware attacks is a concerning trend, particularly given that Windows remains the primary operating system in use. Consequently, this study serves a crucial purpose in facilitating the identification and detection of malware. Through the utilization of the ensemble approach and a comprehensive dataset consisting of various features related to malware, instances of malicious software within the system are effectively detected. The ensemble model employed in this research incorporates both a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN), which are responsible for classifying malware based on their static and dynamic features, respectively.

Upon conducting an extensive analysis and formulation of various ensemble models, it has been determined that the bagging model exhibits an impressive accuracy rate of 98%. Furthermore, it outperforms other model methodologies such as boosting and stacking in terms of overall performance. However, it is worth exploring the potential benefits of incorporating alternative algorithms for weak learners, as this may further enhance the performance of the model. There exists significant potential for the utilization of other machine learning algorithms in the field of malware detection. The incorporation of diverse approaches and methodologies will undoubtedly contribute to the development of a more robust and effective ensemble model.

References

[1] T. Vignesh, S. Reddy, S. Kumar, A. Chourey, and C. P. Chavan, "Malware Detection Using Ensemble Learning and File Monitoring," presented at the ICSTSN, 2023.

[2] H. Sayadi, N. Patel, S. M. P.D., A. Sasan, S. Rafatirad, and H. Homayoun, "Ensemble Learning for Effective Run-Time Hardware-Based Malware Detection: A Comprehensive Analysis and Classification," in 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), 2018, pp. 16, doi: 10.1109/DAC.2018.8465828.

[3] M. Ficco, "Malware Analysis by Combining Multiple Detectors and Observation Windows," in IEEE Transactions on Computers, doi: 10.1109/TC.2021.3082002.

[4] S. Guan and W. Li, "EnsembleDroid: A Malware Detection Approach for Android System based on Ensemble Learning," 2022, doi: 10.1109/URTC56832.2022.10002213.

[5] Y. Dai, H. Li, Y. Qian, R. Yang, and M. Zheng, "SMASH: A Malware Detection Method Based on Multi-Feature Ensemble Learning," in IEEE Access, vol. 7, pp. 112588-112597, 2019, doi: 10.1109/ACCESS.2019.2934012.

[6] E. Amer and I. Zelinka, "An Ensemble-Based Malware Detection Model Using Minimum Feature Set," MENDEL, 2019, pp. 1-10, doi: 10.13164/mendel.2019.2.001.

[7] A. Damodaran et al., "A comparison of static, dynamic, and hybrid analysis for malware detection," Journal of Computer Virology and Hacking Techniques, vol. 13, no. 1, pp. 1–12, 2017.

[8] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," Journal of Information Security, vol. 5, pp. 56–64, 2014, doi: 10.4236/jis.2014.52006.

[9] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," Computers & Security, vol. 81, pp. 123–147, 2019.

[10] H.S. Gururaja and M. Seetha, "Analysis of an Ensemble Model for Network Intrusion Detection," in Artificial Intelligence-Enhanced Software and Systems Engineering, vol. 1, Springer, Cham, 2022, pp. 32.

[11] D. Rabadi, K. W. Fok, Z. Dai, and S. G. Teo, "Malicious Behavior Detection based on Extracted Features from APIs for Windows Platforms," in Proc. DYNAMICS 2019: 2019 Workshop on DYnamic and Novel Advances in Machine learning and Intelligent Cyber Security, December 2019, doi: 10.1145/3464458.3464461.

[12] N. Mathappan, S. Rs, B. Vishnupriya, and T. R. M, "DeepNet: Dynamic Detection of Malwares Using Deep Learning Techniques," in Wireless Communication for Cybersecurity, pp. 21-54, October 2023, doi: 10.1002/9781119910619.ch2.

[13] J. Yan, Y. Qi, and Q. Rao, "Detecting Malware with an Ensemble Method Based on Deep Neural Network," in Security and Communication Networks, vol. 2018, no. 1, pp. 1-16, March 2018, doi: 10.1155/2018/7247095.

[14] M. Belaoued, N. Chekkai, A. Derhab, and C. Ramdane, "Deep Learning for Windows Malware Analysis," in Cyber Malware, pp. 119-164, January 2024, doi: 10.1007/978-3-031-34969-0_6.

[15] "Malware static and dynamic features VxHeaven and Virus Total," UCI Machine Learning Repository, 2019. [Online]. Available: https://doi.org/10.24432/C58K6H.