International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 1, Jan-Feb-2026, ISSN (Online): 2395-566X

Malware Detection Using Machine Learning & Performance Evaluation

I.Sravani[1], D. Lakshmi[2], M.Ushaswini[3], L.Aswini[4], C. Subramanyam[5]
[1]Assistant Professor, Department of Computer Science and Engineering, Sai Rajeswari Institute of Technology.
[2,3,4,5]UG students, Department of Computer Science and Engineering, Sai Rajeswari Institute of Technology

Abstract- Malware is any type of program that is intended to wreak havoc to the computer system and network. Examples of malware are bot, ransomware, adware, keyloggers, viruses, trojan horses, worms and others. The exponential growth of malware is posing a great danger to the security of confidential information. The problem with many of the existing classification algorithms is their low performance in term of their ability to detect and prevent malware from infecting the computer system. There is an urgent need to evaluate the performance of the existing Machine Learning classification algorithms used for malware detection. This will help in creating more robust and efficient algorithms that have the capacity to overcome the weaknesses of the existing algorithms. This study did the performance evaluation of some classification algorithms such as J45, LMT, Naïve Bayes, Random Forest, MLP Classifier, Random Tree, REP Tree, Bagging, AdaBoost, KStar, SimpleLogistic, IBK, LWL, SVM, and RBF Network. The performance of the algorithms was evaluated in terms of Accuracy, Precision, Recall, Kappa Statistics, F-Measure, Matthew Correlation Coefficient, Receiver Operator Characteristics Area and Root Mean Squared Error using WEKA machine learning and data mining simulation tool. Our experimental results showed that Random Forest algorithm produced the best accuracy of 99.2%. This positively indicates that the Random Forest algorithm achieves good accuracy rates in detecting malware.

Keywords – Malware, classification algorithms, Random Forest, AdaBoost, Bagging, Naïve Bayes.

I. INTRODUCTION

The breakthrough in internet technology and computer networking have made high speed shared internet possible. The effect of this development is the daily increase in the number of computer systems that have become susceptible to malware attacks 1, 2. The innovation has made the internet a huge storehouse where resources are virtualized and utilised to the need of users. Despite the immense benefits that the internet revolution has brought, there are numerous challenges that it also poses to the security of computer systems. The conventional computer system is entirely centered on a single host machine running operating system, while several machines connected to the host are running on the guest operating system 1.

The prevalent security threat confronting the users is the attack on a computer system by malicious programs which spread to other computers that have not been infected 3. The threat posed by malware infections has become a major challenge in the field of computer security over the years. The number of new malware on the internet keep on increasing at an alarming rate even as anti-virus companies are making effort to curtail the trend so as to make the vast number of computer user safe. Malware has evolved over time and is becoming more sophisticated than before. It is now more difficult to detect them.

There is therefore the need to invent more efficient techniques that can detect and prevent these attacks. Malware is a malicious program which infringes on the security of a computer system in terms of privacy, reliability, and accessibility of data 3. This trend has made academicians and industry practitioners to move from the conventional static detection techniques 4, 5 to more dynamic, sophisticated and spontaneous methods that applies accumulated malware behaviour to detect malware attacks 6, 7, 8. A malware can simply be defined as a malicious program which the user unsuspectingly install on their machine and later these programs can begin to disrupt the proper operation of the machine or might continue unnoticed and carry out malicious actions without been noticed 9.

When the attacker gains control of the machine, he can then have access to any information stored on the machine. Some of the deceptive approaches used to install malware on the computer system through the internet include repackaging the software, update attack 10 or desire for download 11. The attacker employs any of the methods mentioned before to malicious software by inserting a certain type of malware into

it before uploading it to the internet. Malware can be described as various types of software which have the capacity to wreak havoc on a computer system or illegally make use of this information without the consent of the users 12. Malware can be categorized in various types, for instance, Botnet, Backdoor, Ransomware, Rootkits, Virus, Worms, and Trojan Horse, Spyware, Adware, Scareware and Trapdoor. They are used to attack computer systems and for performing criminal activities such as scam, phishing, service misuse and root access 13.

Objective

- To analyze malware behaviour: by collecting and studying different types of malicious software data.
- To develop a machine learning model: that can automatically detect malware based on extracted features from files or system activities.
- To perform feature extraction and preprocessing: on malware datasets to improve the accuracy of the detection model.
- To train and test multiple machine learning algorithms: such as Decision Tree, Random Forest, Support Vector Machine, or Neural Networks for malware detection.
- To evaluate the performance of the models: using metrics like Accuracy, Precision, Recall, and F1-Score.
- To compare the performance of different machine learning models: and identify the most effective algorithm for malware detection.

Activity Diagram

An activity diagrams illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses some of the same modeling conventions.

Basic Activity Diagram Symbols and Notations

- Action states

Action states represent the non-interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.

- Action Flow

Action flow arrows illustrate the relationships among action states.

- Object Flow

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object.

- Initial State

A filled circle followed by an arrow represents the initial action state.

- Final State

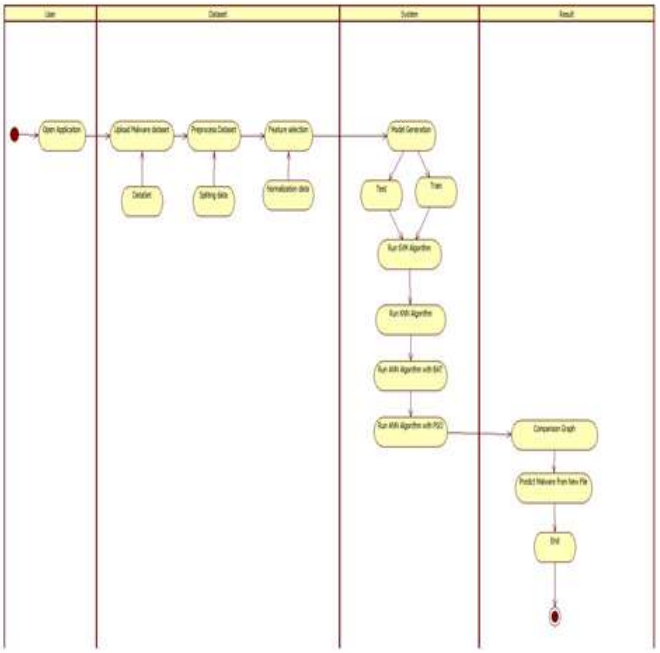
An arrow pointing to a filled circle nested inside another circle represents the final action state.

- Branching

A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else'.

- Synchronization

A synchronization bar helps illustrates parallel transitions. Synchronization is also called for king and joining.



Activity Diagram for overall project

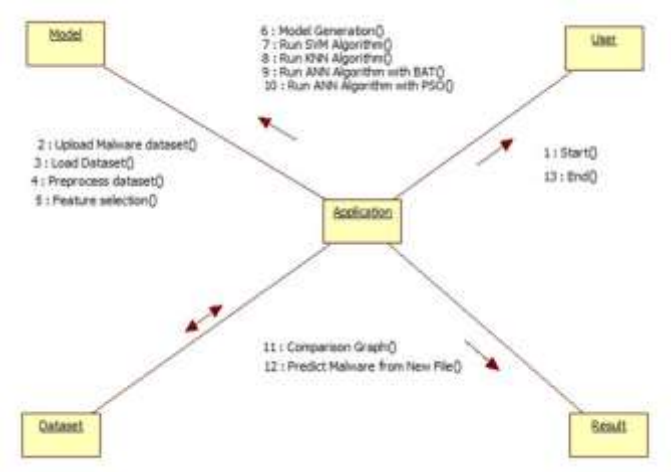
Collaboration Diagram

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram.

The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization. Diagrams of sequence and collaboration are employed, but from a different perspective, to depict the dynamic nature. The interaction diagram's goal is to

- To describe how messages, go across the system.
- To describe the organisational structure of the things.

International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 1, Jan-Feb-2026, ISSN (Online): 2395-566X



Collaboration Diagram for overall project

Encryption and Feature Selection Module

The Encryption and Feature Selection Module plays an important role in improving the security and efficiency of the malware detection system. In this module, encryption techniques are used to protect sensitive data by converting it into an unreadable format, ensuring that the dataset and system information are secure from unauthorized access during storage and transmission.

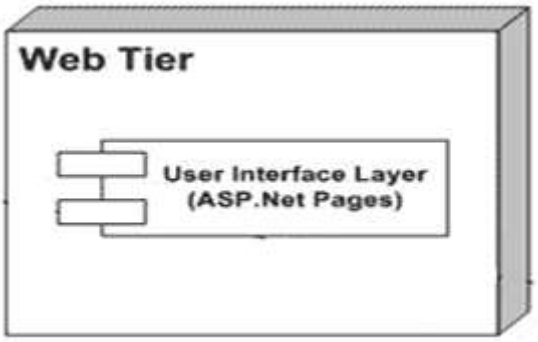
Methods such as AES or RSA can be used to maintain data confidentiality and integrity. After securing the data, feature selection techniques are applied to identify the most relevant attributes from the malware dataset. Since the dataset may contain many features, selecting only the important ones helps reduce data complexity and improves the performance of machine learning algorithms. Techniques like Information Gain, Chi-Square, PCA, or Recursive Feature Elimination are commonly used for this purpose. This module helps enhance the accuracy of the malware detection model, reduces training time, and increases the overall efficiency of the system.

Deployment Diagram

The aim of the diagram is explained by the word "deployment" itself. For depicting the physical components where software components are delivered, deployment diagrams are employed. Deployment diagrams and component diagrams have a lot in common.

UML is primarily made to concentrate on a system's software artefacts. These two diagrams, however, are unique ones that highlight the hardware and software components.
Deployment diagrams are meant to concentrate on the hardware topology of a system, whereas most UML diagrams are used to handle logical components. The system engineers employ deployment diagrams.

- Deployment diagrams serve the following purposes:
- Visualize a system's hardware topology.
- What hardware elements are needed to install software components?
- Describe the nodes that handle runtime processing.



Deployment Diagram for overall project

Test Plan

Test plan is a general document for entire project, which defines the scope, approach to be taken and the personal responsible for different activities of testing. The inputs for forming test plan are following.

- Project plan.
- Requirements document.

Test Case Specification

Although there is one test plan for entire project test cases have to be specified separately for each test case. Test case specification gives for each item to be tested. All test cases and outputs expected for those test cases.

Test Case Execution and Analysis

The steps to be performed for executing the test cases are specified in separate document called test procedure specification. This document specifies any specify requirements that exist for setting the test environment and describes the methods and formats for reporting the results of testing.

II. IMPLEMENTATION

Below are some facts about Python. Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQtetc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

III. MATERIALS AND METHODS

Three stages were involved in the performance evaluation of the various Machine Learning classifiers considered in this study. The phases are Dataset Preparation, Pre-Processing and Application of different Machine Learning algorithms on the ClaMP (Classification of Malware with PE headers) dataset files 34. The dataset has a total of 5184 instances, which contain 2683 Malware, and 2501 Benign.

The dataset has 55 features. The ClaMP dataset 36 is converted into .arff format (a format compatible for the file) supported by the WEKA Machine Learning simulation environment for input data that was used for the analysis. To do a satisfactory classification of the ClaMP dataset 36 , J45, LMT, Naïve Bayes, Random Forest, MLP Classifier, Random Tree, REP Tree, Bagging, AdaBoost, KStar, SimpleLogistic, IBK, LWL, SVM, and RBF Network were utilized and a 10 folds cross-validation was employed in this study. The reason for opting for 10 folds was because of outputs generated from extensive tests on different datasets with erratic learning modus operandi that have proved beyond reasonable doubt that 10 is the most appropriate number of folds needed to obtain the optimal estimate of error 35.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

The Experimental Results and Discussion section presents the outcomes obtained after implementing the malware detection system using machine learning algorithms. In this phase, the trained models are tested using the prepared dataset to evaluate their effectiveness in identifying malicious and benign files. Various performance metrics such as accuracy, precision, recall, and F1-score are used to measure the efficiency of the models.

The results show how well the selected features and machine learning algorithms perform in detecting malware. A comparison of different algorithms is also carried out to determine which model provides the best performance. The discussion part explains the obtained results, analyzes the strengths and limitations of the proposed system, and highlights how feature selection and encryption contribute to improving the overall detection accuracy and security of the system.

DRBA Confusion Graph

V. CONCLUSION

This paper presents a comparative study of malware detection using fifteen different Machine Learning algorithms. Some of the state-of-the-art models such as J45, LMT, Naïve Bayes, Random Forest, MLP Classifier, Random Tree, REP Tree, Bagging, AdaBoost, KStar, SimpleLogistic, IBK, LWL, SVM, and RBF Network were used in the study and their statistical results presented.

From the experimental results obtained from running the various classification using 10-fold cross-validation and 66% split test, it has been demonstrated that some unpopular algorithms perform relatively well on the ClaMP dataset 36 on WEKA. It becomes apparent from our study that Random Forest is the best classifier among the fifteen (15) classifiers considered. Experimental results indicated that even with less feature selection used, the Random Forest classifier with 0.992 performs comparatively better in malware classification, much better than the popular classification algorithms such as SVM with 0.956 accuracy, AdaBoost with accuracy of 0.922, Bagging with 0.978, J48 with 0.978, Naïve Bayes with 0.652, and Multilayer Perceptron classifier with 0.973.

We recommend that more publicly available malware datasets be used to evaluate the performance of other Machine Learning algorithms using different Data Mining and Machine Learning tools such as RapidMiner.

REFERENCES

1. Sanjay Chakrabortya and Lopamudra Dey. A rule-based probabilistic technique for malware code detection. Multiagent and Grid Systems – An International Journal, IOS Press, 12, 2016, pp. 271–286 271. DOI 10.3233/MGS-160254
2. Y. Zhou, Z. Wang, W. Zhou, and X. Jiang. Hey, you, get off of my market: Detecting malicious apps in official and

alternative android markets. in NDSS, vol. 25, no. 4, 2012, pp. 50–52.
3. D. Keragala. Detecting malware and sandbox evasion techniques, SANS Institute InfoSec Reading Room, 2016. URL: https://www.sans.org/reading-room/whitepapers/forensics/detecting-malware-sandbox-evasion-techniques-36667.
4. Sharif, M., Yegneswaran, V., Saidi, H., Porras, P., and Lee, W. Eureka: A framework for enabling static malware analysis. In Computer security-ESORICS 2008, pages 481- 500. Springer.
5. Moser, A., Kruegel, C., and Kirda, E. Limits of static analysis for malware detection. In Computer security applications conference, ACSAC 2007. Twenty-third annual, 2007, pages 421-430.