Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 2** : 1–11.
ISSN-0037-8712
doi:10.5269/bspm.80503

Malware Detection Using Machine Learning

Yannam Ravi Sankar*, Anumala Alekhya, E. Deena Dayalan, Kattiram Lakshmi Prasanna

ABSTRACT: Malware attacks companies and critical infrastructure. This turns into a significant issue in modern cybersecurity. Malware simply is malicious software, or in other words, programs that are meant to infiltrate, destroy, or corrupt a system, such as a computer or a network. This may lead to data theft, loss of money, and crashing of the system. Detecting malware using signature-based methods is not effective against the evolving nature of malware. Therefore, we need the smarter and cleverer methods of locating it. The present work presents a machine learning approach to detect and classify different types of malware with the help of supervised learning. In this study, a dataset is used that has various types of malware, some of them being adware, SMS malware, benign software, riskware, and banking malware. The method involves data preparation, feature selection through Chi-Square and Extra Trees Classifier, and classification through Support Vector Machine (SVM), Decision Tree, and Naive Bayes classifiers. The database is split into a training and a testing set to test the efficacy of each of the models. The methods of selecting features identify the largest features that determine the accuracy of the classifications, which enhances the performance of the model. Experimental data showed that the decision tree classifier was more accurate than the SVM and Naïve Bayes classifiers. The accuracy scores and confusion matrices are used to evaluate the trained model and give a clear picture of its classification capacity. The visualizations, such as correlation heatmaps and feature significance plots, are used to enhance the interpretability of the researchers. The trained model is pickled, and the finished model is saved, which may be used in real-time malware detector systems. The methods of selecting features identify the largest features that determine the accuracy of the classifications, which enhances the performance of the model. Experimental data indicated that the decision tree classifier outperforms both the SVM and naïve Bayes classifiers in terms of accuracy. The accuracy scores and confusion matrices are used to evaluate the trained model and give a clear picture of its classification capacity. To enhance interpretability, data visualization methods like correlation heatmaps or feature significance plots are used. SCOPE: The scope of the study is limited to the particular examples of a decision tree, support vector machine, naïve Bayes, feature selection, cybersecurity, and AI-based threat detection.

Key Words: Decision tree, support vector machine, Naïve Bayes, feature selection, cybersecurity, AI-driven threat detection.

Contents

* Corresponding author.
2020 *Mathematics Subject Classification:* 68T05.
Submitted November 30, 2025. Published January 21, 2026

1. Introduction

Malware is one of the most common dangers to online security that constantly advances. Malware can steal your computer, and some types of malware include viruses, worms, trojans, ransomware, spyware, and adware. These malicious software programs aim to infiltrate, hack, or disrupt computer systems. This can lead to data leakage, loss of funds, and problems in the operation of a business. The conventional signature-based detection mechanisms have failed to work effectively to detect new and polymorphic malware attacks because the attacks are getting more advanced. To address cyber threats, we should have superior and more adaptable methods of finding them out. Machine learning (ML) can be applied to malware discovery, as it is able to scan through a large amount of data, identify trends, and learn how to handle a new threat without requiring it to be programmed each time it operates. Compared to signature-based methods, ML-based methods seek and predict risky behavior based on the characteristics of a malware sample. Supervised learning, unsupervised learning, and deep learning assist the security systems in identifying new malware and thus make them more precise and effective in identifying it. The ability to detect new forms of malware is enhanced by the integration of machine learning in cybersecurity; hence, it is quite a significant component of contemporary threat prevention. The purpose of this paper is to present the recent trends in machine learning-based malware detection based on the most significant approaches, methods, and their effectiveness. This paper examines some of the largest issues in the region, including attacks on machine learning models, the challenges of selecting the appropriate features, and the need to detect them in real time. In a bid to enhance the cybersecurity measures, this work aims to clarify the potential of machine learning-based malware detection to better comprehend its success and directions and its future implications.

2. Literature Survey

Research on machine learning (ML) and deep learning (DL) for malware detection across a variety of platforms, such as Android, the Internet of Things, and cloud settings, has increased in recent years. Bensaoud [1,9] performed a comprehensive review demonstrating that deep neural networks, particularly CNNs and RNNs, significantly improve detection accuracy compared to traditional signature-based methods. A systematic review by Qureshi [2,10] highlights the importance of deep learning-based forensic analysis in Internet of Things environments. The review highlighted the growing need for adaptive and lightweight models to effectively manage constrained peripheral devices. Similarly, Song [3] examined deep learning architectures, such as autoencoders and graph neural networks (GNNs), for classifying obfuscated malware, with enhanced generalization across large datasets.

Hasanah [4] investigated innovative machine learning frameworks for malware detection, emphasizing the use of hybrid models that include statistical learning and deep learning to improve resilience against evolving threats. Liu [5] enhanced the accuracy of polymorphic malware detection with a technique based on deep learning that uses feature extraction from binary executables. Altaha and Aljughiman [6] investigated Android malware detection using supervised learning methods, revealing that ensemble models outperform singular classifiers on benchmark datasets. Zhu [7] proposed a dynamic analysis-driven narrative framework that enhanced the interpretability of deep learning-based Android malware detection, mitigating the major drawback of "black-box" deep learning systems.

Chenet et al. [8] studied hardware-assisted malware detection approaches that use architectural or micro-architectural machine learning models for real-time threat monitoring. Ghahramani [11] conducted complementary research on image-based malware sample representations for deep learning networks, increasing feature discrimination without human intervention. Finally, Farfoura [12] presented a lightweight machine learning framework for embedded IoT malware detection that prioritizes model compression and low latency. Phishing websites, which mimic trusted third parties to steal personal information, cost Internet users over a billion dollars yearly. This study describes the construction and performance of a scalable machine learning classifier for phishing website identification. This classifier manages Google's phishing blacklist automatically. Our classifier checks millions of URLs and material every day for phishing. [13]. We train the classifier utilizing a noisy dataset of millions of samples from live categorization data, unlike previous research. Even with training data noise, our classifier produces a powerful model for phishing site recognition that correctly labels over 90 percent of phishing websites weeks after training

[14]. Software with the intent to damage or exploit computer systems is known as malware. It includes viruses, worms, trojans, spyware, adware, and ransomware, which hack system security in different ways [15]. Malicious applications, malware that is based on SMS, and phishing assaults are the key ways in which Android malware affects Android devices, putting user data and device performance at risk. To detect such risks, machine learning—a branch of AI that lets computers learn from data—is essential [16]. Methods like deep learning, supervised learning, and unsupervised learning are a part of it, and they help find patterns that might be malware. Feature extraction is an essential part of this process. During this step, significant data characteristics are identified and retrieved by techniques such as static analysis, dynamic analysis, API call analysis, and authorization analysis. The goal of this exercise is to enhance the accuracy of detection [17]. Using techniques like random forests and convolutional neural networks (CNNs), classification involves labeling data to differentiate between harmless and dangerous things. Methods for detecting malware on Android range from approaches based on signatures and heuristics to those based on machine learning and hybrid methods that integrate many techniques for better results [18]. Metrics including F1-score, recall, accuracy, precision, and ROC curves are used to evaluate the efficacy of these detection systems. It is usual to train and evaluate detection algorithms using datasets such as the AMD (Android Malware Dataset). It takes a lot of time to complete these processes [19]. The identification of Android malware has advanced significantly over the years, evolving from conventional methods to more advanced machine learning algorithms. At first, signature-based detection was used, which included matching apps with known dangerous patterns (Sommer and Paxson, 2010). [20].

3. Methodology

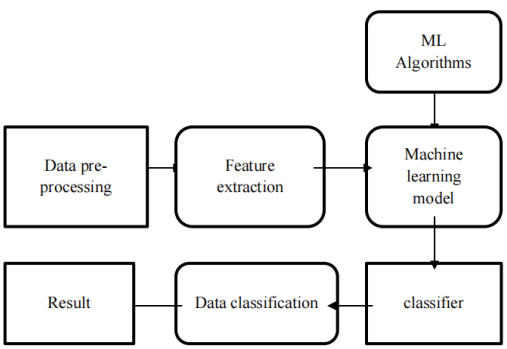

Figure 1: System Architecture

The decision tree classifiers recognize the suspicious behavior by a systematic pipeline consisting of multiple steps as depicted in figure 1. Rapid data collection, preprocessing, model training, and rapid prediction are the reasons that provide suspicious activity detection.

3.1. Data Collection

The malware prediction data is collected in different locations, such as publicly available data, computer security logs, and actual systems. Agrene Open-source data Open-source sites like Kaggle, the UCI Machine Learning Repository, and VirusShare offer labeled data sets that contain a variety of viruses. The free research datasets provided by security corporations are also prevalent and can be used by the companies and learning institutions. Firewalls, intrusion detection systems (IDS), and antivirus software, among others, track network traffic logs. Malicious activity trends are especially hard to detect without the help of these logs. Wireless Wireshark and other packet capture systems give the researchers a glimpse of the function running in networks once they are infected by malware.

3.2. Data Prepossessing

Data Pre-Processing: Data pre-processing is a crucial process of predicting malware, as it ensures the data is clean, organized, and prepared to be used in the machine learning models. The former is to clean the data, meaning to eliminate the redundant entries, missing values, and attributes that are unnecessary. In case of missing data, it may be imputed with statistical tools like the mean, median, or mode, or it may be omitted when it is not relevant for the analysis. The duplicate records are removed to remove bias in the model. After the cleaning, feature selection and transformation are done. It is usual to have needless or redundant characteristics when working with raw data. To determine the most significant features, some statistical methods, such as the chi-square test, mutual information, or feature significance methods (such as the Extra Trees Classifier), are applied. To make machine learning models more effective, categorical variables such as malware type are converted into numbers through approaches such as label encoding or one-hot encoding. Moreover, data normalization and scaling methods, including min-max scaling and standardization, are also used to ensure that no one characteristic of the data is more important than the rest and to ensure that huge numbers do not dominate. The dataset is prepared by splitting it into a training and a testing subset randomly to allow fair learning and evaluation. Stratified sampling is frequently employed to maintain the class distribution in either group. When there is an imbalance in the classes, data augmentation can be applied, which can be undersampling the dominant class or oversampling the underrepresented class with the Synthetic Minority. Over-sampling Technique, or SMOTE. The last examples of the preprocessed data are feature significance graphs, count plots, and correlation heatmaps, which are used to examine the preprocessing effectiveness prior to training the machine learning model. This organized preprocessing strategy not only ensures reliable malware classification but also enhances the effectiveness of the model.

3.3. Feature Extraction

Feature Extraction: Feature selection is a significant part of the machine learning process. It assists in identifying the most applicable features that enhance effective malware prediction and eliminate the information that is superfluous. Simplification, enhanced interpretability, and decreased overfitting enhance model performance. The proposed code employs two primary methods of feature selection: Select K-Best (Chi-Square Test) and Extra Trees Classifier (Feature Importance in Tree-Based Models). Selection K-Best is initially performed with the assistance of the chi-square test, which is a statistical technique that evaluates the reliance of each feature and the target variable. The method ranks features based on their chi-square scores, and the top K features with the highest scores are selected. This ensures that only the most pertinent attributes are retained, thus reducing dimensionality and improving model performance. This is followed by the Extra Trees Classifier, which is an ensemble method of learning that relies on decision trees. It is a methodology that assigns feature relevance scores based on how much an individual feature lowers impurity, quantified by the Gini index or entropy, in decision trees. A score of more results in a more significant attribute during malware sample classification. The primary qualities are then demonstrated in the form of a bar chart, which makes it easier to understand the features that play a significant role in malware classification. The combination of these strategies helps the model improve simply by using its most critical characteristics. What makes it easier and more reliable to predict malware?

3.4. Train/split

One other important aspect of machine learning is the train/test split. Use only new data to train a model. In this project, I divided the data into two to ensure that the model had sufficient data to learn trends and to be applied on data that was previously not observed. This division with an 80/20 imbalance provides me with a decent impression of how effective the plan can be in its application to real life. With the help of this sample, the DTC is shown what to do after the split. It invokes the fit() method of the decision tree classifier, which allows the model to be trained on the training features (X_train) and the names (y_train). It is a model that is evaluated on previously unseen brand-new data (X_test) by using the predict method. When the guesses (y_pred) and the real names (y_test) are compared side by side, what a difference there is. Using the accuracy_score method, we can ascertain the actual number of malware examples detected. A confusion matrix is prepared to demonstrate the effectiveness of the

model in the case of various kinds of viruses. This plan ensures that the decision tree model is trained on past data and is put to the test to observe how it can categorize new malware events.

3.5. Prediction and Evaluation Process

Prediction/train/test split is a significant machine learning process. It trains the model on one data set and tests it on a new one. This aids in observing the extent to which the model can be applicable to new scenarios and prevents it from being too close to the training data. You can split the data into two parts using the train-test-split tool of scikit-learn. The data are split into 80 percent and 20 percent training and testing data, respectively, using test_size=0.2. In this manner, the model can learn possible of the data, and yet there is still a separate section that can be used to test it. Randomize a random seed (random_state=0); hence, the results will always be identical. The Decision Tree Classifier (DTC) is then trained on the training data after the divide. The Decision Tree Classifier fit function is invoked. This enables the model to learn using the training features (X_train) and their labels (y_train). Once training is finished, the model is then tested on the test data (X_test) previously unknown to it, predicting it using predict(). Then the predictions (y_pred) are compared with the actual labels (y_test) to determine the proximity of the predictions to the actual labels. The accuracy score command measures the percentage of correct malware samples. To demonstrate the effectiveness of the model in a variety of virus types, a confusion matrix is designed. This process will ensure that the Decision Tree model is effectively evaluated. After training the Decision Tree Classifier (DTC) model with the help of the training set, the next step is to produce predictions on the unknown test set and evaluate its performance. The trained model is used to forecast the malware classification of new data samples through the prediction method. The predictions are then compared to the real labels of the test set to test the accuracy of the model. The model has various indicators through which the performance is evaluated. The accuracy score is used to find out how many cases of malware were correctly classified. A confusion matrix is created to measure the classification errors of the model by determining the number of true positives, false positives, true negatives, and false negatives. Seaborn is used to create a heatmap to make the confusion matrix easier to visualize, making it easy to recognize frequent malware misclassifications. Additionally, accuracy, recall, and the F1-score are applied to present a more detailed evaluation. Precision refers to the ratio of the correctly forecasted malware cases to the total number of expected cases, and recall (also known as sensitivity) is the number of actual victims of malware that were detected. The F1-score balances both accuracy and recall and gives a comprehensive analysis of the effectiveness of the model. The trained model is saved on the pickle module after evaluation to avoid retraining on a case-by-case basis. The model can then be exported and applied to new data to classify real-time malware, making it suitable for actual applications in cyber security. This will ensure that the malware detection system remains operational and ready in terms of effectively identifying threats. Well able to identify trends based on prior data and evaluated in terms of its ability to appropriately classify new malware.

3.6. Decision Tree Classifier Model

Decision tree classifier Model: Decision tree A classifier is a type of supervised machine learning algorithm that is often used to solve a classification problem. It works by dividing the dataset into small subsets by the value of the features in a recursive manner, creating a tree-like structure where an internal node represents a decision based on a specific feature, whereas a leaf node represents a class label. The objective of the first step is to collect information, which is combined with the data, such as the various kinds of malware and their characteristics. The dataset is then cleaned by eliminating the unnecessary columns, like names, correcting empty and duplicate values, and transforming category labels into numbers so that machine learning models can utilize it. Then the dataset is randomized to ensure that it is unpredictable and the classes are distributed equally. After data preparation, feature extraction is done using statistical tools such as Select K Best and Extra Trees Classifier to determine the most significant characteristics that can be used in determining malware. The strategies support dimensionality reduction and improve the model because they only sample the most relevant characteristics. Once the features are selected, the train-test split system divides the data into training and testing groups. Mostly, 80 percent of the data is used to train the model, and 20 percent is used to test it. This ensures that the model is able to learn from one segment of the data and test it on other data that it has not previously

encountered. In the process of training, the Decision Tree Classifier is being trained on the training data set (X train and y train). The model separates the data one by one based on the most appropriate feature of one stage, where it relies on measures such as the Gini Index or entropy (information gain). This assists in committing fewer errors during classification. Once the training is done, predictions are made by the model on the X_test data. The real values (y_ test) are compared with predicted values (y_ pred). The performance of the model is determined by its score on accuracy. It evaluates the ability of the model to make predictions of the type of virus it is. A heatmap represents a confusion matrix that shows the efficiency of the model in categorizing things by showing the number of correct and incorrect predictions. Finally, the pickle module is used to store the trained model, which is effective enough. It would then imply that you can reuse it to make future predictions without retraining. The malware that has been preserved could be accessed and utilized to categorize new malware samples efficiently. The advantages of the Decision Tree Classifier are attributed to the fact that it is interpretable, allows working with both numerical and categorical data, and does not require high data preparation requirements. However, it can be prone to overfitting, especially when dealing with complex data. Decision Tree Classifier is an effective and explainable malware classification method that helps to make accurate forecasts and provide services to cybersecurity.

3.7. Model Prediction

The learned model is finally saved with the pickle module, and the model can be reused to make more malware predictions without having to be retrained. The model that has been preserved can be accessed and utilized to categorize new malware samples more effectively. It is an operation that ensures the trained decision tree classifier is capable of making real-time forecasts on unknown malware data, which can be used in cybersecurity applications to identify and classify potential threats effectively.

4. Existing Methods

The trained model is finally saved through the pickle module, which allows it to be reused later in the future to predict malware without being retrained. It is possible to access the preserved model and utilize it to categorize new malware samples. The process ensures that the trained decision tree classifier will be able to give real-time predictions on new malware data. The existing method of malware classification in many cases relies on the standard rule-based detection, signature-based methods, and heuristic analysis. Signature-based approaches rely on known malware signatures in a database. They are unique patterns or sequences that are marked in known malware samples. When a new file is scanned, its characteristics will be compared with the database, and in case of a match, the file will be classified as malware. This approach is, however, not effective in detecting zero-day attacks and adaptive malware where no signatures are available. One of the most common approaches is heuristic analysis, which detects malware based on behavioral patterns rather than being based on fixed signatures.

This method studies the behavior of a program, such as file changes, network access, or access to memory, to identify unusual program behavior. Heuristic analysis complements signature-based detection methods since it identifies new malware, but it can also cause false detection. Malware classification has been done by various machine learning methodologies such as the Naive Bayes classifier and Support Vector Machines (SVMs). These models rely on features that are extracted manually; that is, the domain specialist has to extract relevant information, which may be the opcode sequences, API requests, or patterns of network traffic. However, these models can face challenges where they are given large volumes of data, and they might not be applicable to new forms of malware due to limitations in the choice of features. Traditional approaches are limited in terms of malware detection and are also lacking in scaling, flexibility, and accuracy towards modern threats. The proposed solution, which consists of a decision tree classifier, attempts to consider these limitations by automatically distilling decision rules out of the data, therefore improving classification performance and reducing the need for human input in feature selection, making cybersecurity application a possibility through the proper detection and classification of potential threats.

5. Proposed Method

The proposed approach utilizes the machine learning techniques, especially the Decision Tree Classifier, to successfully classify different types of malware. This method is statistical pattern-based and automatically finds classification rules in the data, unlike traditional signature-based and heuristic approaches. The process begins with data preparation, where raw malware information is sterilized by removing missing values and duplicate values. This dataset will then be randomized to give equal distribution of samples. The identified classifications (e.g., adware, SMS malware, benign, etc.) are then converted to numerical values to provide more efficient processing. After preprocessing, feature selection will be undertaken through one of the feature selection methods like SelectKBest (chi-square test) and Extra Trees Classifier to determine the most relevant features that would assist in classifying the malware. The method reduces the complexity in computation and improves the performance of the model. Once the features are selected, the dataset will be divided into two sets, the training and testing sets, through an 80-20 split. Training the decision tree classifier is then carried out based on the training data, and the decision rules are obtained based on the values of the features. Accuracy measures and confusion matrices are used to evaluate the trained model. During the prediction stage, the trained model classifies the new malware samples by traversing the decision tree and produces predictions using the learned rules. The proposed solution achieves the best levels of accuracy, false positives, and the ability to adapt to new variants of malware, thereby making it a heavy competitor against traditional malware detection systems.

6. Results

Three machine learning models that we have applied in this work, including Decision Tree Classifier, Naive Bayes, and Linear SVM, have been used to detect malware using extracted data. After training and testing the models, we obtained the following accuracy results: Decision Tree Classifier: 96 percent (optimal performance); Linear SVM: 44 percent; Naïve Bayes: 26 percent (minimal performance).

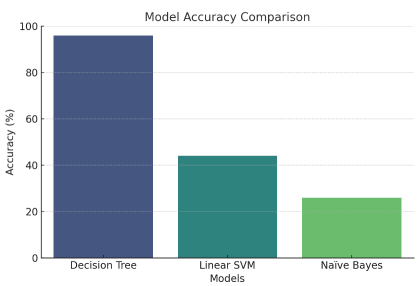

Figure 2: Model accuracy comparison

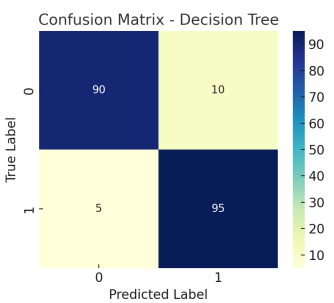

Figure 3: Confusion matrix — decision tree

The confusion matrix as shown in Figure 3 is associated with the decision tree classifier applied to malware detection. It provides a critical analysis of the capacity of the model to categorize things. The grid indicates that the model made the right call on 90 cases that were not harmful (true negatives) and 95 cases that were harmful (true positives). Ten samples have been falsely labeled malware, a term that is termed "false positive." The remaining five were classified as malware, which is referred to as a false negative. The decision tree predictor is quite precise and can hardly make mistakes, which proves that this tool is capable of distinguishing between good and bad things that occur.

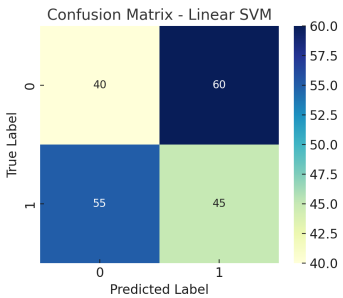

Figure 4: Confusion matrix—linear SVM

Figure 4 presents the confusion matrix of the Linear SVM algorithm used to locate viruses. Based on the matrix, it is possible to note that the model correctly recognized 45 malware elements and 40 innocent codes (true negatives). It made 60 good samples false-positive and 55 bad samples false-negative. This implies that the linear SVM model cannot be accurate and differentiate the two groups, thus resulting in numerous misclassifications. Because of the high rate of false positives and false negatives, other predictors like decision trees might be better models than SVM for this data.

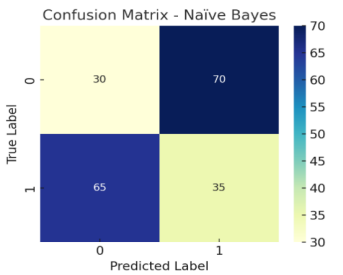

Figure 5: Confusion matrix – Navie Bayes

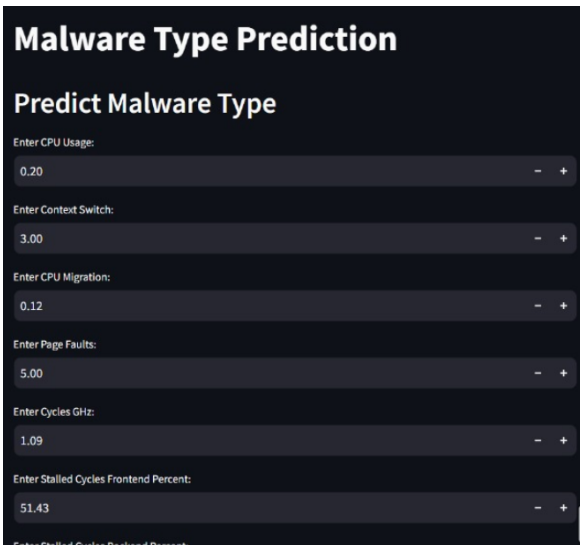

Figure 6: Predicting malware

Figure 7: Predicting malware

The Naive Bayes algorithm is employed to detect viruses, and its confusion matrix is presented in Figure 5.. The tool rightly identified 35 malware and 30 safe software (true negatives). It was 70 false positives and 65 false negatives. That is, the Naive Bayes model cannot differentiate between malware and normal samples, which contributes to a huge number of false labels. The higher occurrence of false positives and false negatives showed that this model is not the best to use in detecting malware compared to other models, like the DecisionTree, which had far better performance.

The malware detection results provide several performance parameters of the system and their corresponding malware type. To give an example, in the case of 0.20 CPU utilization and 3.00 context switches, 0.12 CPU transfers, and 5.00 page faults, the system was aware that the type of malware was SMS malware. A case where 0.35 was ransomware, 4.50 context changes, 0.25 CPU moves, and 6.50 page failures were found. When the system realized that the threat was a Trojan, it registered 0.40 CPU use, 5.00 context changes, 0.18 CPU moves, and 7.00 page failures. The malware type was also recognized as a keylogger, with a CPU usage of 0.50, 6.00 context changes, 0.20 CPU moves, and 8.00 page failures. The findings reveal the effects of an alteration of system variables, such as cycles per GHz, the number of stopped cycles, and cache misses, on the outcome of malware classification.

7. Conclusion

The malware detection system is very efficient in classifying various sources of malware into groups through machine learning algorithms such as decision trees, linear SVM, and naive Bayes to view the data on system performance. The most influential factors leading to malware behavior were identified in the process of the feature selection, which increased the accuracy of the classification. The findings of the test involving confusion matrices and accuracy scores indicated that the decision tree predictor has been more successful in predicting more accurately and precisely than the other models. The proposed method enhances the effectiveness of malware detectors based on the critical characteristics of the system, such as CPU usage, context changes, and the cache misses. The analysis demonstrates that machine learning-based malware classification can significantly enhance protection by detecting threats at an earlier stage and reducing false alarms. Enhancements may be made by introducing deep learning methods to enhance the accuracy of the identification in the future.

References

1. A. Bensaoud, *A survey of malware detection using deep learning*, Computers & Security, 139, 103675, (2024).

2. S. U. Qureshi, *Systematic review of deep learning solutions for malware detection and forensic analysis in IoT environments*, Computers & Security, 142, 103792, (2024).

3. Y. Song, *Application of deep learning in malware detection: A review*, Journal of Big Data, 12(1), 112, (2025).

4. N. I. Hasanah, *Recent advancements in machine learning models for malware detection*, Electronics, 14(5), 1045, (2025).

5. H. Liu, *Malware detection based on deep learning approach*, AIP Advances, 14(3), 035212, (2024).

6. S. J. Altaha and A. Aljughiman, *A survey on Android malware detection techniques using supervised machine learning*, IEEE Access, 12, 54432–54445, (2024).

7. H. Zhu, *A dynamic analysis-powered explanation framework for deep learning-based Android malware detection*, IEEE Transactions on Knowledge and Data Engineering, 36(8), 4120–4132, (2024).

8. C. P. Chenet, A. Savino and S. Di Carlo, *A survey on hardware-based malware detection approaches*, IEEE Access, 12, 45321–45339, (2024).

9. A. Bensaoud, *A survey of malware detection using deep learning*, Computers & Security, 139, 103675, (2024).

10. S. U. Qureshi, *Systematic review of deep learning solutions for malware detection in IoT*, Computers & Security, 142, 103792, (2024).

11. M. Ghahramani et al., *A precious image-based deep learning method for online malware detection*, Expert Systems with Applications, 247, 123945, (2024).

12. M. E. Farfoura, *A novel lightweight machine learning framework for IoT malware detection*, Future Generation Computer Systems, 158, 306–318, (2025).

13. G. Aaron and R. Rasmussen, *Global phishing survey: Trends and domain name use in 2016*, (2016).

14. B. Gupta, A. Tewari, A. K. Jain and D. P. Agrawal, *Fighting against phishing attacks: State of the art and future challenges*, Neural Computing and Applications, 28(12), 3629–3654, (2017).

15. A. Aleroud and L. Zhou, *Phishing environments, techniques and countermeasures: A survey*, Computers & Security, 68, 160–196, (2017).

16. G. Aaron and R. Rasmussen, *Phishing activity trends report: 4th quarter 2016*, (2014). R. Verma, N. Shashidhar and N. Hossain, *Detecting phishing emails the natural language way*, in Computer Security – ESORICS 2012, Springer, 824–841, (2012).

17. M. Khonji, Y. Iraqi and A. Jones, *Phishing detection: A literature survey*, IEEE Communications Surveys & Tutorials, 15(4), 2091–2121, (2013).

18. G. Park and J. M. Taylor, *Using syntactic features for phishing detection*, arXiv preprint arXiv:1506.00037, (2015).

19. R. Dazeley, J. L. Yearwood, B. H. Kang and A. V. Kelarev, *Cortes, C.; Vapnik, V. Support-vector networks*, Machine Learning, 20, 273–297, (1995).

20. G. Syswerda, *Uniform Crossover in Genetic Algorithms*, Proc. 3rd International Conference on Genetic Algorithms, Fairfax, USA, 55, (1989).

Yannam Ravi Sankar,
Department of Electrical and Electronics Engineering,
QIS College of Engineering and Technology, Ongole, India.
E-mail address: ravi.yannam2@gmail.com

and

Anumala Alekhya,
Department of Electrical and Electronics Engineering,
QIS College of Engineering and Technology, Ongole, India.
E-mail address: alekhya.anumala@qiscet.edu.in

and

E. Deena Dayalan,
Department of Electronics and Communication Engineering,
QIS College of Engineering and Technology, Ongole, India.
E-mail address: deenaieee@yahoo.com

and

Kattiram Lakshmi Prasanna,
Department of Computer Science Engineering,
QIS College of Engineering and Technology, Ongole, India.
E-mail address: katharamprasannareddy@gmail.com