International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(22s) 2026 pp. 1102-1112
https://doi.org/10.70917/ijcisim-2026-5504
Article

Optimized malware Detection Model Employing Data Balancing and Ensemble Machine Learning Approaches

Dr. GNV Vibhav Reddy (M.Tech, Ph.D)[1], Suguru Shreya[2]

[1] Associate Professor, Department of computer science and Engineering, Sree Dattha Institute of Engineering and sciences, Sheriguda, Hyderabad, India. Email: vibhav.gnv@gmail.com
[2] Student, Department of computer science and Engineering, Sree Dattha Institute of Engineering and sciences, Sheriguda, Hyderabad, India. Email: shreyasuguru@gmail.com

Abstract: The rapid proliferation of malware presents serious challenges to digital security, especially for resource constrained devices where efficiency and memory usage are critical. This study proposes an efficient machine learning based malware detection framework using the Kaggle Malware Detection dataset. Preprocessing includes missing value handling, categorical label encoding, numeric feature standardization, and five fold cross validation. Feature selection is performed using the Extra Trees Classifier based on Gini impurity, followed by data balancing through under sampling and over sampling techniques. A wide range of algorithms is evaluated, including Logistic Regression, Support Vector Machine, K Nearest Neighbors, Gaussian and Bernoulli Naive Bayes, Decision Tree, Random Forest, XGBoost, Gradient Boosting, LightGBM, AdaBoost, CatBoost, Histogram based Gradient Boosting, and Extra Trees. Ensemble strategies such as Voting and Stacking classifiers are also explored. Experimental results show Random Forest achieves 98.97% accuracy without feature selection, while the balanced Stacking Classifier attains 99.90% accuracy. Explainable AI techniques, LIME and SHAP, provide feature level insights. A Flask based web application enables secure user interaction, real time preprocessing, prediction visualization, and classification of inputs as legitimate or malware.
Keywords: Malware detection, random forest classifier, feature selection, extra tree classifier, resource constrained device, execution time”

1. Introduction

Malware, also known as the short term of malicious software (M) refers to software designed to attack, interrupt or unlawfully acquire access to computer networks, systems, and interconnected computers, networks, and other devices. The rapid growth of the Internet of Things (IoT) and ubiquitous digital networks have led to the systems receiving increasingly complex cyber threats [2]. Common malware types—viruses, trojans, spyware, adware, ransomware, and fileless malware—exploit system vulnerabilities and evade conventional defenses [3], causing data breaches, service interruptions, and unauthorized system control [4].

Traditional malware detection methods, which include signature-based malware detection systems, behavioral-based, and heuristic systems have been applied widely [5]. The signature-based methods are good at detecting familiar malware, but are often ineffective when met with new or polymorphic ones [6]. Behavioral methodologies improve the process of identifying unknown threats however it can consume considerable computing power, so it cannot be implemented on low resource devices [7]. Heuristic methods are flexible however complex or zero-day attacks are challenging [8]. These limitations highlight a major failure in achieving rapid, accurate and comprehensive malware detection in modern network systems.

This research work gives a comprehensive approach to enhance malware detection and protection in various systems including IoT systems with limited resources. It prioritizes continuous monitoring, improving the level of

transparency about network utilization, and helping to quickly identify suspicious activities. The approach targets high frequency and major types of malware to offer adaptive security to improve system resilience and reduce risks of breaches without necessarily relying on stagnant or narrowly defined security solutions [9].

It is important as this approach could help to improve cybersecurity in complex digital ecosystems. An effective malware detection can prevent disruptions in its operations, secure sensitive information, and ease financial and reputational losses [10]. Immediate and dynamic security measures are essential in IoT and edge environments, where the huge number of interconnected devices and vulnerability to attack threats make significant contributions to unstable reliability, trust, and service availability. This approach makes modern networked infrastructures stronger and more reliable by transcending the boundaries of older solutions and delivering scalable security.

2. Related Work

Approaches to malware detection based on heuristics have been read into extensively to identify known and unfamiliar threats. Bazrafshan et al. [11] reviewed heuristic detection methods, focusing on their ability to detect zero-day malware based on behavioral analyses of programs and not based on the reliance on signatures. Being effective against new variations, these strategies can often require considerable computer resources, and can have problems with complex obfuscation. Alzarooni [12] highlighted the importance of prioritizing flexibility in detecting malware variants, but the studies have mostly focused on desktop and server environments, thus ignoring the problem of resource-constrained IoT devices.

Virus uniqueness to the Internet of Things has become an area of increased attention. Ngo et al. [13] also investigated the detection of IoT malware with the use of the static features and provided a categorization of resource-constrained devices, but did not focus on dynamic and real-time malware detection that is necessary in cases of new threats. Sasi et al. [14] took this further to categorize the threats of the IoT and study the detection systems and highlighted the limitations regarding scalability and flexibility. Such studies demonstrate that solutions tailored to a variety of low-resource based IoT environments are required.

Malware detection by machine learning techniques has proven to be possible. Azeem et al. [15] benchmarked a number of techniques and found that there were improved detection rates on complex data; however, handling obfuscated malware or malware residing in memory remained a challenge. Open data such as Sys4VPN, Meraz'18 [16], or others used to benchmark, often do not include real-time or moving malware examples. Hossain and Islam [17] studied the obfuscated malware in memory dumps and showed promising detection but appreciating the trade-off between accuracy and processing efficiency.

Behavioral and dynamic analysis has been used to enhance detection of malware. Dynamic Scan Bhatia et al. [18] used the methods of forensic analysis of dynamism to reveal attack patterns and malware presence, but they can be time-consuming and potentially unscalable to large IoT networks. Mohd et al. [19] and Kumar et al. [20] focussed on hybrid and monitored machine learning-based intrusion detection systems, achieving high detection accuracy in different cases in a network. However, there are challenges still to finding flexible, zero-day malware as well as incorporating lightweight monitoring that may be suitable in IoT devices.

3. Materials And Methods

The proposed system develops an efficient malware detection framework tailored for resource-constrained devices using a comprehensive set of [21] machine learning algorithms, including Logistic Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Gaussian Naive Bayes (GNB), Bernoulli Naive Bayes (BNB), Decision Tree (DT), Random Forest (RF), Extreme Gradient Boosting (XGB), Gradient Boosting (GB), LightGBM (LGBM), AdaBoost (AB), CatBoost (CB), Histogram-based Gradient Boosting (HGB), and Extra Trees (ET). The Extra Trees Classifier is used to perform feature selection in order to decrease the aspects of classification, and class imbalances are eliminated using the methods of data balancing (under-sampling and over-sampling). Cross-validation is rigorously measured by using a quintuple cross-validation. The interpretability is provided by ensemble-based learning methods, e.g., Voting Classifier and Stacking Classifier, and explainable AI methods, e.g., LIME and SHAP.

The solution applies a real time, intuitively designed interface which has been written in Flask alongside utilizing malware datasets to comprehensively evaluate in a variety of contexts.

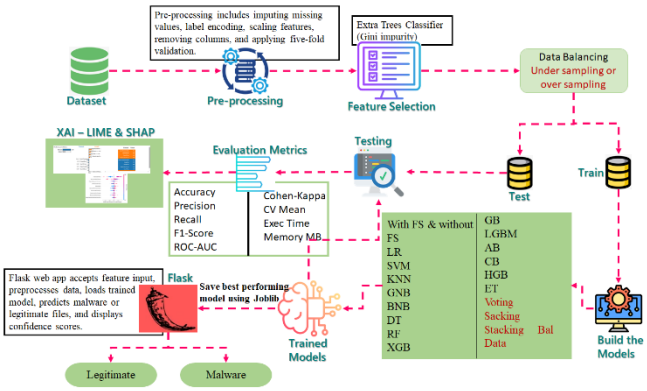


Fig. 1. System Architecture

Fig. 1 illustrates a comprehensive machine learning pipeline for malware detection. The technique begins with pre-processing of data and selecting features using Extra Trees Classifier and data balancing is the next step. After splitting into train and test sets, multiple algorithms—including ensemble methods like XGBoost and LightGBM—build the models. The most suitable model is evaluated and tested with the visualization of XAI (LIME/SHAP), and the model is deployed in the form of a Flask web application.

A) Data Set Collection

This collection contains 216352 executable files with 58 attributes, containing header data, data section attributes, imports, exports, resources, and version information. Structural and behavioral attributes are included (SizeOfCode, SectionsMeanEntropy and ResourcesMeanSize) and the files are classified as valid (benign) or malignant (valid column). It is a set of numerical and categorical data types with low rates of missing values. It provides a comprehensive description of executable features, which enables strict training and evaluation of malware detection models that are resource-constrained.

	ID	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion
0	1	b69acb3bb133974e48229627663f96d4	332	224	8450	8.0	0
1	2	1cbee4b3725629bd0aa6ac2ff500925f	332	224	258	9.0	0
2	3	b7027cf0cd31c820928950cbfe7e91ef	332	224	8450	8.0	0
3	4	156a0bb069f94d1e7c2508318805f2a4	332	224	8450	10.0	0
4	5	c72bf851fed5542abba904b1f3944cd5	332	224	8226	48.0	0

5 rows × 58 columns

Fig. 2. Malware Detection Dataset

B) Pre-Processing

The preparation pipeline ensures quality and consistency of data sets by applying cleaning, missing value imputation, scaling, feature selection and class balance provision, hence supplying data to be efficiently and quickly analyzed malware detection models.

Data Pre-Processing: Pre-processing involves purifying and manipulating the data-set to model with. Columns that were not considered necessary such as ID, md5, Machine and Unnamed: 57 have been removed. The median method is used to fill in the missing values to ensure that all the values are complete. Duplicates are identified and eliminated and the index of the data set is restored. MinMaxScaler is applied to numerical features to normalize them (between 0 and 1), improving machine learning models convergence and learning efficiency.

Feature Selection: In order to enhance the model efficacy, feature selection identifies and maintains the most significant properties. Techniques, such as the Extra Trees Classifier, are used to estimate the value of the features,

thereby reducing the dimensionality and the number of computations. Selecting the relevant features the system eliminates redundant or less relevant features thereby improving model generalization, interpretability and training performance. This ensures accurate malware detection and is also suitable in resource constrained environments as well as developing higher prediction effectiveness in the different executable samples.

Data Balancing: Data balancing corrects class imbalances in either pernicious or benign data to prevent bias training of a model. The use of techniques such as under-sampling and over-sampling is used to assure the equal representation of the two classes. Fair datasets enhance generalization of models, reduce false negatives and false positives and enhance robustness. The method reduces biased distributions, allowing machine learning classifiers to reliably detect the patterns of malware, particularly when they occur in the minority-class, with its computational efficiency while maintaining resource constraints in resource-constrained equipment.

C) Training and Testing:

The data is split into the training ratios of 70:30 and the even-numbered testing ratio. Naked features are preprocessed, then balanced, and models are evaluated based on accuracy, precision, recall, F1-score, ROC-AUC, Cohen-Kappa, execution and memory usage. Cross-validation ensures stability and reduced over-fitting, which provide a valid measure of malware detection.

D) Algorithms

Logistic Regression (LR): Preprocessed information was then processed to give an approximation of what the probability of malware being either dangerous or benign is, which is an easy to interpret and compute, and computationally efficient method of classification suitable on resource constrained devices, and provides a reasonable level of detection, and a significant level of false positives [24].

$$\hat{y}_i = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \quad (1)$$

Support Vector Machine (SVM): Learns harmful and innocent examples, discovers optimal decision boundary in high dimensional feature space, non-linear interactions, and provides accurate and robust classification to devices with limited memory and processing capacities.

$$minimize\ \frac{1}{2}\ ||W||^2 + C\sum_{i=1}^{n} \quad \xi_i \quad (2)$$

K-Nearest Neighbors (KNN): Categorizes samples based on their proximity to closest neighbors in feature space offering flexible and simple malware pattern identifiers with minimum preprocessing and high efficiency on a wide range of feature spaces.

$$distance(x, X_i) = \sqrt{\sum_{j=1}^{d} \quad \left(x_j - X_{i_j}\right)^2} \quad (3)$$

Gaussian Naive Bayes (GNB): Assesses the likelihood of malware or benign classification by the use of feature statistics, which is a fast and highly efficient, and interpretable probabilistic classifier suitable where processing resources are limited.

Bernoulli Naive Bayes (BNB): Analyzes types of binary features to label samples as hazardous or harmless, offers rapid prediction, and minimal memory usage, [28] can be used on the devices which require efficient but lightweight detectors.

Decision Tree (DT): Builds hierarchical judgments of feature values to categorize malware, which offers interpretable and computationally efficient and effective in capturing the complex relationships, at the same time being acceptable to devices with constrained resources.

$$I\ (i) = 1 - \sum_{i=1}^{k} \quad p_i^2 \quad (4)$$

Random Forest (RF): Brings some decision trees together to classify and make the data more accurate and reduce overfitting, therefore, provides a robust and high performance method that can work well even in resource-limited environments.

$$Gini = 1 - \sum_{i=1}^{C} \quad (Pi)^2 \quad (5)$$

Extreme Gradient Boosting (XGB): Builds sequential trees to improve malware vs benign sample predictions, get more accurate, less overfitting, and good interactions of the features in resource-limited settings.

Gradient Boosting (GB): Progressively corrects weak learners to improve classification accuracy and better understanding between feature linkages and efficiency in limited devices computing.

$$\hat{y} = f\left(W^L f\left(W^{L-1} \cdots f(W^1 X + b^1) + b^{(L-1)}\right) + b^L\right) \quad (6)$$

LightGBM (LGBM): Scales tree based learning to feature sets of large size, reducing treatment of resources and accelerating training, therefore providing fast scaled and accurate detection on resource restricted systems.

AdaBoost (AB): Combines numerous weak classifiers to focus on hard data, improving the overall detection accuracy, as well as provides a robust, computationally low enough solution that can be used in resource-limited environments.

$$H(x) = sign\left(\sum_{t=1}^{T} \quad \propto_t h_t(x)\right) \quad (7)$$

CatBoost (CB): Streamlines numeric and categorical data processing based on ordered boosting and symmetric tree construction to accurately classify malware, with minimum memory consumption and time.

$$F(x) = F_0(x) + \sum_{m=1}^{M} \quad \sum_{i=1}^{N} \quad f_m(x_i) \quad (8)$$

Histogram-based Gradient Boosting (HGB): Maps features to bins to make them easier to train and build sequential trees as well as trade off prediction accuracy against computational economy under limited contexts.

Extra Trees (ET): establishes a number of randomized decision trees and unites predictions, enhancing resistance, precision and efficiency in handling high-dimensional data groups in a memory limited setting.

Voting Classifier: Majority voting (which combines predictions made by a large number of models) can help improve the classification strength and accuracy, as well as, minimize the processing needs in small contexts.

$$\hat{y}\ = argmax_c\left(\sum_{i=1}^{n} \quad II\ (\hat{y}_i = c)\right) \quad (9)$$

Stacking Classifier: Combines the output of basic models with a meta-classifier to enhance accuracy and reliability by identifying the relationship between predictions hence making possible to effectively detect malware in resource-constrained environments.

$$y^{\wedge} = g(Y_{base}) = g\big(f_1(x), f_2(x), \dots, f_m(x)\big) \quad (10)$$

Stacking Classifier Balanced Data: Utilizes equal datasets and simplistic classes and a meta classifier to reduce bias, and increases the detection of both majority and minority classes without compromising computing efficiency in small scope situations.

E) Integration of XAI and Flask Framework:

The system is based on Explainable Artificial Intelligence (XAI) practices, including LIME and SHAP, to provide clear and interpretable data on malware detection predictions. LIME offers more localized explanations through focusing on feature contributions to the instance, whereas SHAP offers a broader level of interpretability globally through measuring feature relevance on the many samples and illustrates how their contribution provides changes to the model outputs, thereby augmenting confidence and reliability.

An interface based on Flask will be used to easily deploy the model and explainable artificial intelligence capabilities. Users can feed in feature values, make real-time predictions and have interpretations offered by LIME and SHAP. This incorporation ensures actual practicality and deficiency strained resources such as devices are capable of achieving interpretable, efficient and actionable malicious detectives.

4. Experimental Results

Accuracy: Accuracy of a test is the ability of a test to distinguish correctly between sick and healthy cases. In order to evaluate the validity of a test, it is necessary to calculate the true positives and true negatives proportion in all the tests conducted. This could be mathematically stated as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} (11)$$

Precision: Precision is a measure of the proportion of the correctly identified cases out of the total positive cases. As such, with this formula of calculating the accuracy:

$$Precision = \frac{True\ Positive}{True\ Positive\ + False\ Positive} (12)$$

Recall: The recall of machine learning is a statistic that determines the ability of a model to identify all the relevant examples of a particular type. The ratio of correct predictions of positive observations as compared to the total actual positives, it provides an idea about the effectiveness of a model identifying occasionalities of a particular class.

$$Recall = \frac{TP}{TP\ +\ FN} (13)$$

F1-Score: The F1 score is a statistic of assessing machine learning model accuracy. It combines precision and recall of a model. The measure of accuracy was a way of measuring the number of correct predictions produced by a model on the entire dataset.

$$F1\ Score = 2 * \frac{Recall\ X\ Precision}{Recall + Precision} * 100 (14)$$

AUC-ROC Curve: AUC-ROC Curve is a dimensional measurement of the performance of classification at varied levels of threshold. ROC plots the False Positive vs True Positive. The AUC is used to evaluate the average ability of the model to discriminate between classes and a larger AUC indicates a better model.

$$AUC = \sum_{i=1}^{n-1} (FPR_{i+1} - FPR_i) \cdot \frac{TPR_{i+1} + TPR_i}{2} \quad (15)$$

Cohen Kappa: The Kappa (κ) provided by Cohen is a statistical measure used to evaluate the extent of agreement between two raters (or judges, observers, and so on), who classify things into separate sets. It is especially beneficial when decisions are subjective, and the classes are not ordered in nature (are not nominal).

$$Kappa(k) = \frac{P_o - P_e}{1 - P_e} \quad (16)$$

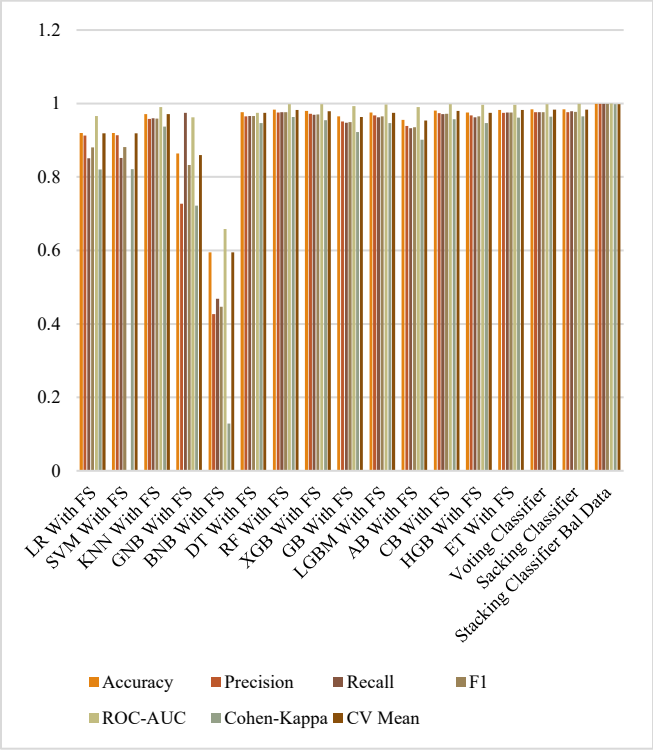
Table.1 Performance Evaluation – With FS

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC	Cohen-Kappa
LR With FS	0.919514	0.912591	0.850868	0.880649	0.965814	0.820046
SVM With FS	0.920007	0.913387	0.851530	0.881375	NaN	0.821145
KNN With FS	0.971235	0.957999	0.959649	0.958823	0.989887	0.936721
GNB With FS	0.863680	0.727554	0.974173	0.832994	0.962115	0.721861
BNB With FS	0.594953	0.426860	0.468809	0.446852	0.658044	0.128453
DT With FS	0.975950	0.965029	0.966094	0.965561	0.974350	0.947085
RF With FS	0.983253	0.975732	0.976292	0.976012	0.998002	0.963148
XGB With FS	0.979524	0.971559	0.969714	0.970636	0.997722	0.954918
GB With FS	0.964687	0.950959	0.947684	0.949319	0.992612	0.922223
LGBM With FS	0.975734	0.967649	0.962651	0.965143	0.996837	0.946532
AB With FS	0.955228	0.938797	0.932497	0.935637	0.990076	0.901313
CB With FS	0.980556	0.973480	0.970730	0.972103	0.997729	0.957181
HGB With FS	0.975626	0.967307	0.962695	0.964995	0.996766	0.946300
ET With FS	0.982667	0.974475	0.975895	0.975185	0.996755	0.961868
Voting Classifier	0.983761	0.976734	0.976734	0.976734	0.998550	0.964262
Sacking Classifier	0.984162	0.976131	0.978544	0.977336	0.998642	0.965164

Stacking Classifier Bal Data	0.999026	0.998770	0.999286	0.999028	0.999989	0.998053

Table 1 evaluates models using accuracy, precision, recall, F1-score, ROC-AUC, Cohen-Kappa, cross-validation mean, execution time, and memory usage. The Stacking Classifier with balanced data uniquely achieves superior performance across all metrics.

Fig. 3. Comparison Graph– With FS



As can be clearly seen in Figure 3, Stacking Classifier using balanced data achieves the highest accuracy, precision, recall, and F1-score which is better than any other model, measured against the performance metrics evaluated.

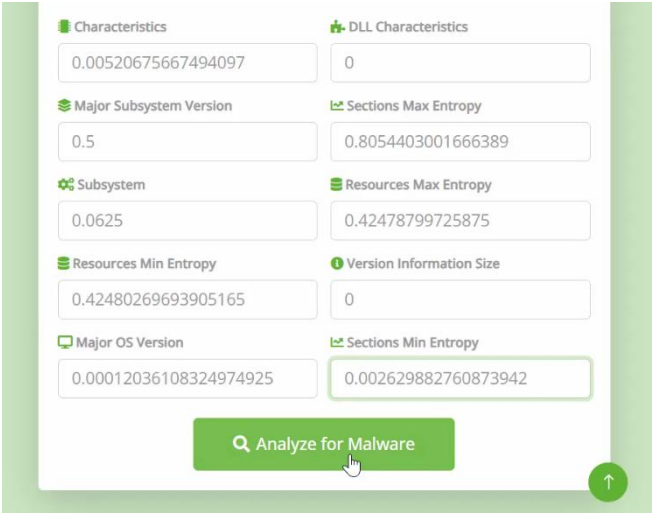


Fig. 4. Enter Input Data

In Fig. 4, the input screenshot displays a user interface where malware feature values are entered for model-based classification and analysis.

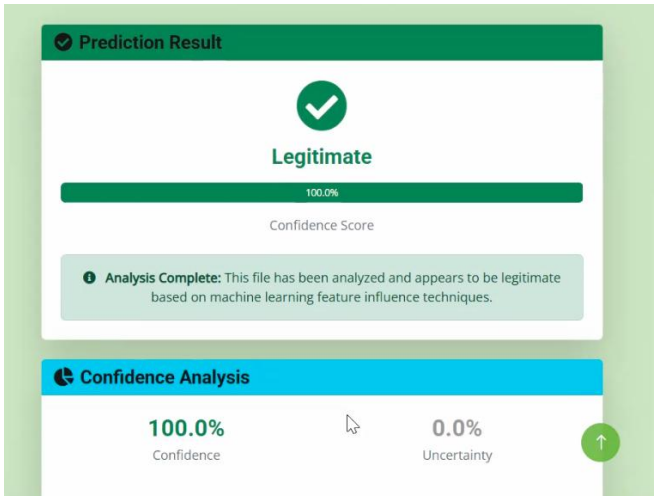


Fig. 5. Predicted Result

A snapshot of the output, as shown in Figure 5, shows that the result to be expected is the identity of the legitimate file, which is represented in this case as legitimate.

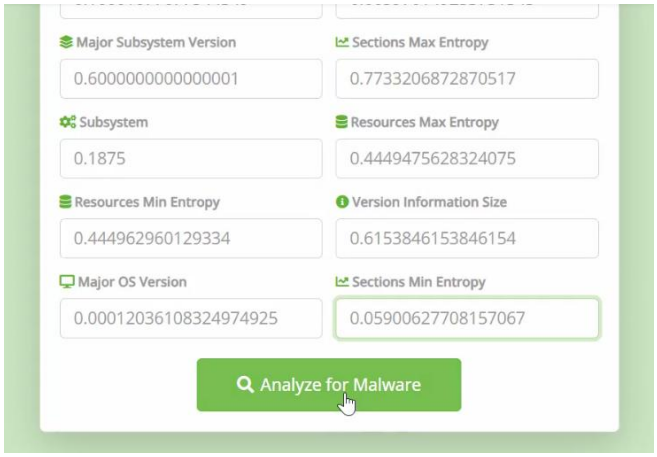


Fig. 6. Enter Input Data

In Fig. 6, the input screenshot illustrates a user interface where users enter malware feature parameters for prediction and classification purposes.

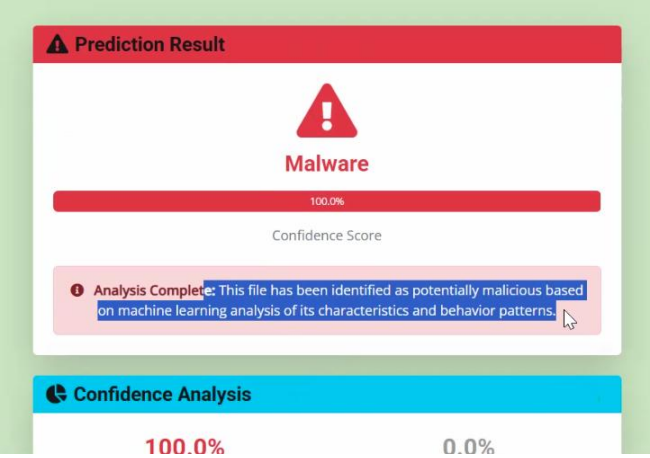


Fig. 7. Predicted Result

The screenshot in figure 7 depicts an output that shows a projected output of malware which means the appearance of the harmful activity given the input data.

5. Conclusion

The study confirms that machine learning methods can perform very high levels of detection and still be resource efficient when faced with resource constraints. The careful preprocessing, the Extra Trees-based feature selection, and the effective data balancing were used to enable strict evaluation of numerous classifiers. Random Forest model was able to achieve excellent results with no feature selection with 98.97, 98.61, 98.43 and 98.52 accuracies, precisions, recalls and F1 scores, respectively. After the feature selection was implemented, the balanced Stacking Classifier experienced a significant improvement in terms of the accuracy 99.90%, the precision 99.88%, the recall 99.92 and an F1 score of 99.90, as support of the benefits of lower dimensionality ensemble learning. Explainable AI algorithms in the form of LIME and SHAP further increased transparency, namely with prominent features clearly demonstrated and useful decision-making. The developed trained models were deployed in a Flask web application, which ensures a safe interaction, pre-processing of real-time inputs, and the prediction was displayed to make use of it practically. The mechanism gives specific system identifications as true or malware, enabling the speedy, understandable, and scalable malware discovery suited to restricted computer systems and requiring rapid response to threats and small operating overhead throughout the world.

Future studies can focus on the lightweight deep learning models, including CNNs and recurrent networks which can be utilized on the embedded systems. Online and incremental learning assists immediate response to new viruses. Forming bigger datasets of various types of malware and zero-day exploits will enhance the resilience. By improving hybrid ensemble frameworks and integrating models that include edge computing and IoT systems, it may be easy to identify malware fast and accurately with a minimum of computation and memory requirements.

REFERENCES

1. Basak, M., Kim, D. W., Han, M. M., & Shin, G. Y. (2024). Attention-Based Malware Detection Model by Visualizing Latent Features Through Dynamic Residual Kernel Network. Sensors, 24(24), 7953.
2. Gutierrez, R., Villegas-Ch, W., Godoy, L. N., Mera-Navarrete, A., & Luján-Mora, S. (2024). Application of deep learning models for real-time automatic malware detection. IEEE Access, 12, 107742-107756.
3. Torres, M., Álvarez, R., & Cazorla, M. (2023). A malware detection approach based on feature engineering and behavior analysis. IEEE Access, 11, 105355-105367.
4. Gorment, N. Z., Selamat, A., Cheng, L. K., & Krejcar, O. (2023). Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions. IEEE Access, 11, 141045-141089.
5. Euh, S., Lee, H., Kim, D., & Hwang, D. (2020). Comparative analysis of low-dimensional features and tree-based ensembles for malware detection systems. IEEE Access, 8, 76796-76808.
6. H. Gill, ‘‘Malware: Types, analysis and classifications,’’ Univ. Bradford, U.K., Tech. Rep., 2022.

7. M. Al-Fawa'Reh, J. Abu-Khalaf, P. Szewczyk, and J. J. Kang, ''MalBoT DRL: Malware botnet detection using deep reinforcement learning in IoT networks,'' IEEE Internet Things J., vol. 11, no. 6, pp. 9610–9629, Mar. 2024.
8. I. Firdausi, C. Lim, A. Erwin, and A. S. Nugroho, ''Analysis of machine learning techniques used in behavior-based malware detection,'' in Proc. 2nd Int. Conf. Adv. Comput., Control, Telecommun. Technol., Dec. 2010, pp. 201–203.
9. D. Venugopal and G. Hu, ''Efficient signature based malware detection on mobile devices,'' Mobile Inf. Syst., vol. 4, no. 1, pp. 33–49, Jan. 2008.
10. Ö. A. Aslan and R. Samet, ''A comprehensive review on malware detection approaches,'' IEEE Access, vol. 8, pp. 6249–6271, 2020.
11. Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, ''A survey on heuristic malware detection techniques,'' in Proc. 5th Conf. Inf. Knowl. Technol., May 2013, pp. 113–120.
12. K. Alzarooni, ''Malware variant detection,'' Ph.D. thesis, Dept. Comput. Sci., Univ. College London, London, U.K., Mar. 2012.
13. Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, ''A survey of IoT malware and detection methods based on static features,'' ICT Exp., vol. 6, no. 4, pp. 280–286, Dec. 2020.
14. T. Sasi, A. H. Lashkari, R. Lu, P. Xiong, and S. Iqbal, ''A comprehensive survey on IoT attacks: Taxonomy, detection mechanisms and challenges,'' J. Inf. Intell., vol. 2, no. 6, pp. 455–513, Nov. 2024.
15. M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, ''Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches,'' Heliyon, vol. 10, no. 1, Jan. 2024, Art. no. e23574.
16. (2018). Meraz'18 Dataset. Accessed: Aug. 15, 2024. [Online]. Available: https://www.kaggle.com/competitions/malware detection/dataselect=data.csv
17. M. A. Hossain and M. S. Islam, ''Enhanced detection of obfuscated malware in memory dumps: A machine learning approach for advanced cybersecurity,'' Cybersecurity, vol. 7, no. 1, p. 16, Jan. 2024.
18. A. M. Bhatia, I. Kumar, and N. Mohd, ''Dynamic analysis of a malware sample: Recognizing its behavior using forensic application,'' in Proc. 4th IEEE Global Conf. Advancement Technol. (GCAT), Oct. 2023, pp. 1–6. 12664
19. N. Mohd, A. Singh, and H. S. Bhadauria, ''Intrusion detection system based on hybrid hierarchical classifiers,'' Wireless Pers. Commun., vol. 121, no. 1, pp. 659–686, Nov. 2021.
20. I. Kumar, N. Mohd, C. Bhatt, and S. K. Sharma, ''Development of IDS using supervised machine learning,'' in Soft Computing: Theories and Applications: Proceedings of SoCTA 2019. Cham, Switzerland: Springer, 2020, pp. 565–577.
21. M. Goyal and R. Kumar, ''Machine learning for malware detection on balanced and imbalanced datasets,'' in Proc.Int. Conf.Decis.AidSci.Appl. (DASA), Nov. 2020, pp. 867–871.
22. A. Hota, S. Panja, and A. Nag, ''Lightweight CNN-based malware image classification for resource-constrained applications,'' Innov. Syst. Softw. Eng., pp. 1–14, Jul. 2022.
23. S. Song, N. Gao, Y. Zhang, and C. Ma, ''BRITD: Behavior rhythm insider threat detection with time awareness and user adaptation,'' Cybersecurity, vol. 7, no. 1, p. 2, Jan. 2024.
24. V. Tanksale, ''Intrusion detection system for controller area network,'' Cybersecurity, vol. 7, no. 1, p. 4, Feb. 2024.
25. A. Kumar, K. Abhishek, K. Shah, D. Patel, Y. Jain, H. Chheda, and P. Nerurkar, ''Malware detection using machine learning,'' in Proc. 1st Indo-Amer. Conf., 2nd Ibero-American Conf. Knowl. Graphs Semantic Web(KGSWC),Mérida,Mexico.Cham,Switzerland:Springer, Nov. 2020, pp. 61–71.
26. V. Patel, S. Choe, and T. Halabi, ''Predicting future malware attacks on cloud systems using machine learning,'' in Proc. IEEE 6th Intl Conf. Big Data Secur. Cloud (BigDataSecurity), IEEE Intl Conf. High Perform. Smart Comput., (HPSC), IEEE Intl Conf. Intell. Data Secur. (IDS), May 2020, pp. 151–156.
27. M. Masum, M. J. Hossain Faruk, H. Shahriar, K. Qian, D. Lo, and M. I. Adnan, ''Ransomware classification and detection with machine learning algorithms,'' in Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC), Jan. 2022, pp. 0316–0322.
28. I. Shhadat, B. Bataineh, A. Hayajneh, and Z. A. Al-Sharif, ''The use of machine learning techniques to advance the detection and classification of unknown malware,'' Proc. Comput. Sci., vol. 170, pp. 917–922, Jan. 2020.
29. R. Damaševičius, A. Venčkauskas, J. Toldinas, and Š. Grigaliunas, ''Ensemble-based classification using neural networks and machine learning models for windows PE malware detection,'' Electronics, vol. 10, no. 4, p. 485, Feb. 2021.
30. J. C. Kimmell, M. Abdelsalam, and M. Gupta, ''Analyzing machine learning approaches for online malware detection in cloud,'' in Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP), Aug. 2021, pp. 189–196.