IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 14 November 2024, accepted 26 December 2024, date of publication 7 January 2025, date of current version 23 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3526878

RESEARCH ARTICLE

An Efficient Malware Detection Approach Based on Machine Learning Feature Influence Techniques for Resource-Constrained Devices

SUBIR PANJA[1,2], SUBHASH MONDAL[1,3], (Member, IEEE), AMITAVA NAG[1], (Senior Member, IEEE), JYOTI PRAKASH SINGH[4], (Senior Member, IEEE), MANOB JYOTI SAIKIA[5,6], (Senior Member, IEEE), AND ANUP KUMAR BARMAN[1], (Member, IEEE)

[1]Department of Computer Science and Engineering, Central Institute of Technology Kokrajhar, Kokrajhar, Assam 783370, India
[2]Department of Computer Science and Engineering, Academy of Technology, Adisaptagram, West Bengal 712121, India
[3]Department of Computer Science and Engineering (AI & ML), Dayananda Sagar University, Bengaluru, Karnataka 562112, India
[4]Department of Computer Science and Engineering, National Institute of Technology Patna, Patna, Bihar 800005, India
[5]Electrical and Computer Engineering Department, University of Memphis, Memphis, TN 38152, USA
[6]Biomedical Sensors & Systems Laboratory, University of Memphis, Memphis, TN 38152, USA

Corresponding authors: Manob Jyoti Saikia (msaikia@memphis.edu) and Amitava Nag (amitava.nag@cit.ac.in)

ABSTRACT The growing use of computer resources in modern society makes it extremely vulnerable to several cyber-attacks, including unauthorized access to equipment and computer systems' manipulation or utter breakdown. Malicious attacks in the form of malware cause significant harm to systems with limited resources. Hence, detecting these attacks and promptly implementing a computationally efficient technique is imperative. Utilizing a machine learning (ML) based model is a superior option for promptly identifying malware. This study develops fourteen machine learning models using a five-fold cross-validation technique on the dataset it obtained for research. We compute the execution time and memory used for each of the fourteen ML model developments, considering both all features and the reduced features after applying the data preprocessing methods. We utilized the Extra Tree classifier (ETC) to identify the top ten significant contributing features based on Gini impurity scores, which led to improved accuracy and reduced processing time. After that, we compared the experimental results and found that the Random Forest (RF) classification model on the reduced features set had a prediction accuracy of 99.39% and ROC-AUC values of 0.99. The ETC model prediction yields comparable results, confirming the viability of the suggested model. The proposed model is very resilient, exhibiting an extremely small standard deviation. It is also highly responsive, with reduced execution time and memory utilization.

INDEX TERMS Malware detection, random forest classifier, feature selection, extra tree classifier, resource-constrained device, execution time.

I. INTRODUCTION

Malware is malicious software intended to harm or destroy computers, networks, and other computer system resources. The term ''malware'' is an abbreviation for ''malicious software,'' which includes various sorts and portions that harm computer systems on numerous levels. Malware is distinguished by its malicious purpose, which goes against the functions of a system and excludes software bugs that cause inadvertent harm [1].

Malware is often categorized into a variety of categories. Viruses, worms, trojans, spyware, adware, ransomware, and fireless malware are some forms of malware that are severe threats to today's internet security. Regarding the selection of

The associate editor coordinating the review of this manuscript and approving it for publication was Tiago Cruz.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

these six types, the criteria were based on their prevalence and the unique challenges they pose in resource-constrained IoT environments. These types are among the most commonly referenced in the literature and frequently target IoT systems, where they exploit specific vulnerabilities. We chose to focus on these categories to ensure that the proposed model can address a broad range of relevant threats and adapt to the most pressing security risks in the IoT. The selected categories represent prevalent and impactful threats, particularly in IoT and edge environments, where these types of malware continue to challenge cybersecurity defenses.

1) **Computer Virus:**A computer virus is harmful software that may reproduce by altering legal software on the system and inserting its instructions. Viruses are dangerous because they may infect other files and turn them into bearers of the virus [2].
2) **Trojan Horse:** A Trojan horse is a form of malware that camouflages itself as a genuine item or application to deceive users into installing it. After being installed, it is capable of performing malicious operations such as data theft or providing illegal entry to the adversary [3].
3) **Spyware:** Spyware can pilfer sensitive information once it gets installed on a device. Spyware collects sensitive data that could be shared with businesses, third parties, or advertisements [4].
4) **Adware:** Adware is a type of malware that presents annoying advertisements to a computer user. Advertising served by adware shows infrequently, akin to a sporadic occurrence via pop-up windows [1].
5) **Ransomware:** Ransomware uses advanced methods of encryption to encode vital information on a specific computer and requests money in exchange for decrypting the information [5].
6) **Fireless malware:** Fireless malware operates in memory, infecting normal applications without leaving a trace or attempting to save itself in documents. The virus bypasses traditional detection methods by targeting the RAM and exploiting PowerShell and other automated methods often used by system administrators. A system reboot often removes the infection from the system since it does not remain on the storage medium [6].

Denial of service attacks and e-mail spam malware compromise the computer networks and create botnets of the networks [7].

A. PROBLEM STATEMENT

Malware undoubtedly infiltrates different network systems; therefore, a must-have defense system that gives high visibility and detection of breaches is necessary. To eradicate malware, there should be specific software that can swiftly detect harmful files. This necessitates continuous network inspection. Once the issue has been recognized, the malware must be removed from the network.

B. TRADITIONAL MALWARE DETECTION METHODS

Different software uses different algorithmic models to detect malware-affected files and find and remove them from the system. The three main traditional methods, behavioral, signature, and heuristic-based, are used to detect malware. These traditional three methods are described in brief in the following.

- Behavioral method: Behavioral-based malware detection is a technique employed by security applications to detect malware by analyzing its behaviors rather than just relying on established characteristics or trends. An automated analysis can be conducted on the behavior of each malware in imitative circumstances, resulting in the generation of behavioral reports [8].
- Signature-based method: Signature detection relies on the process of scanning input files for pre-established viral signatures. A popular signature is an individual set of bytes that is exclusive to the virus and does not appear in regular files. A signature of this sort is often recovered by a virus specialist following its investigation [9]. During the first stages, the method of detecting malware known as signature-based detection was commonly employed. Nevertheless, this technique is limited in its ability to identify unfamiliar and emerging infections [10].
- Heuristic-based method: Behavioral malware detection was implemented to address the shortcomings of the signature-based technique. However, to overcome certain limitations, heuristic approaches were introduced [11]. The heuristic-based malware detection strategy is a sophisticated way of detecting malware that relies on past experiences and employs several strategies, including rules and machine learning algorithms [12]. While it exhibits a high level of accuracy in identifying zero-day malware to some extent, it is incapable of detecting sophisticated malware.

C. OBJECTIVES

Due to the algorithmic high overhead complexity of resource-constrained devices, an accurate prediction model for detecting malware files is critical in any antivirus software system. Malware can be identified based on three system environments: windows-based Portable Executable (PE) files, android-based APK files, and IoT-based Executable and Linkable Format (ELF) files. Various static feature-based IoT malware detection techniques are discussed in [13]. More than 25% of cyber-attacks target IoT devices but above 80% of attacks target Windows-based systems [14]. So, providing a prediction model for timely and quickly detecting malware and protecting the Windows system is more concerning. Machine learning (ML) based prediction model nowadays is the best alternative approach to detect malware [15] in any form concerning a resource-constrained device within a

limited time and provides the decision to secure the computer system resources and the networks.

D. CONTRIBUTIONS

The dataset used in this research study was obtained from the open-source provided by the Meraz'18 [16] security partner of malware. The dataset was for the binary classification of Malware and Legitimate files and was separated into two categories for the result attributes. To divide the instances into malware and non-malware groups, fourteen different ML models were deployed, namely LGBM (Light Gradient Boosting Machine), HGB (Histogram-Gradient Boosting), XGB (Extreme Gradient Boosting), AB (Ada Boost), CB (Cat Boost), GB (Gradient Boost), RF (Random Forest), DT (Decision Tree), SVM (Support Vector Machine), LR (Logistic Regression), GNB (Gaussian Naive Bayes), BNB (Bernoulli Naïve Bayes), ETC (Extra Tree Classifier), and KNN (K-Nearest Neighbors). The models were trained with instances of 70% of the dataset, while the remaining 30% was utilized to test the prediction model to classify the malware as legitimate. The performance of the models was assessed using different performance metrics like k-fold cross-validation mean accuracy (MA), accuracy (A), recall (R), precision (P), F1-score (F), ROC-AUC score (RA), Cohen-kappa score (CK), and standard deviation (SD) were considered to address the overfitting of the model.

A comparative investigation of the deployed model was accessed, showing that eight out of fourteen models, LGBM, HGB, XGB, GB, CB, RF, DT, and SVM, obtain more than 99% k-fold mean accuracy. Minimal to no difference in accuracy and cross-validation accuracy indicates that there was no over-fitting of the model. To address the overfitting and the underfitting issues of the machine learning models, we have used the k-fold cross-validation technique with five-fold. This is a technique to train the model with a certain number of folds and to test the performance of the model using the rest of the folds, so every time model training and testing different unseen folds is used to control the biased prediction towards a particular class. Also, the standard deviation is used to check and verify the models' overfitting scenarios through their outcomes. The experimental results demonstrated that the difference between the mean accuracy of five-fold cross-validation and the standard accuracy after splitting the dataset into the ratio of 0.70:0.30 was almost zero or minimal. Also, the standard deviation was almost zero, so we can conclude that the models were trained and tested without overfitting or underfitting. The negligible disparity between accuracy and cross-validation accuracy indicates minimal overfitting, as observed. This consistency confirms that the model generalizes effectively to novel data, consistent with our objective to prevent overfitting. Other performance metrics of the best models produced highly encouraging findings with minimal variance and a high accuracy score. Significant low to almost zero false negative values added to the above model's brilliance. The proposed malware detection model in this study can be of enormous benefit in classifying files into legitimate and non-legitimate. The proposed model is one of a kind in its segment, wherein it uses all fourteen popular ML models to detect malware. The main research contribution of this study focuses on a few factors that are stated below:

1) This study presents a comprehensive application of fourteen machine learning models with a five-fold cross-validation technique, aiming to mitigate the issues of overfitting.
2) This article explores an efficient approach for detecting malware specifically designed for resource-constrained devices to reduce response times while maintaining high accuracy.
3) This leverages a critical feature selection technique to identify a subset of significant features essential for accurate prediction. This approach enhances model performance, achieving a prediction accuracy of 99.39% and an impressive ROC-AUC score of 0.999, providing valuable insights for advancing malware detection systems.

The rest of this article is organized as follows: Section II presents a literature review. Section III describes the proposed methodology including the algorithm's datasets, data pre-processing operations, model training, and feature selection method. The experimental results are discussed in Section IV and Section V concludes the article.

II. LITERATURE REVIEW

This section describes the research study related to malware detection using classical ML techniques in different system environments and also considers articles that focus on feature selection-based detection models.

In article [17], researchers introduce an innovative method in cybersecurity that concentrates on identifying and examining disguised malware using a sophisticated ML framework. The research thoroughly evaluated the model using the Obfuscated-MalMem2022 dataset and showed its outstanding capability to reliably classify various forms of malware. Utilizing approaches like SMOTE to tackle class imbalance significantly improves the effectiveness of the model, resulting in close-to-perfect accuracy in binary and multi-class situations with 4 and 16 classes, respectively. The model attains an accuracy of over 99% in three separate situations. The researchers in [18] analyze malware behavior through dynamic analysis by executing samples in a controlled environment. This approach reveals real-time actions like network activity and system manipulation, providing insights to improve malware detection and defense strategies. In [19] the study enhances intrusion detection by implementing a hybrid hierarchical IDS using the updated KDD-99 dataset, containing 4,898,431 instances split equally for training and testing. It classifies normal and attack instances in multiple stages. The first classifier distinguishes between normal and attack types, while subsequent classifiers categorize attacks into DoS, R2L, U2R, and probing classes in a

hierarchical manner. The optimized classifier arrangement improves detection accuracy at each level. The SSVM-based IDS achieves a high detection rate of 97.91% and an Overall Detection Accuracy (ODA) of 98.79%, outperforming other classifiers such as SVM, PNN, DT, NFC, and kNN in this structured approach. In [20] researchers address the critical need for data security in modern internet-connected networks by implementing an intrusion detection system (IDS) that can distinguish between normal and attack data packets. Using the KDD-99 dataset, which includes 32,640 samples (12,440 normal and 20,200 attack samples), the data is divided equally into training and testing sets. Supervised learning models, specifically SVM and kNN, are applied with Principal Component Analysis (PCA) for dimensionality reduction. Results indicate that the PCA-kNN model achieved a maximum accuracy of 90.07% using cosine distance, demonstrating its effectiveness in identifying network intrusions.

In the study [21], the authors divided the paper into two major parts, one amongst them being feature extraction and the other being the incorporation of machine learning classifiers. The authors used the API calling feature in the sandbox to execute the sample data and extract them. The dataset used in this paper had 42797 malware instances and 1079 benign instances. The under-sampling method was used to balance the dataset, reducing the number of malware samples to 1079 instances and then differential splitting of train and test datasets in the ratio of 60:40, 70:30, and 80:20 percent. Five machine-learning classifiers were used to train the dataset: KNN, GNB, MNB, DT, and RF. Amongst all these classifiers, RF performed best with an accuracy of 90.38% on balanced and 98.91% on the imbalanced dataset with a percent 70:30 split ratio. The researchers in [22] introduce a specialized, efficient deep CNN designed specifically for classifying malware images. The proposed approach employs a CNN model with an Adam optimizer to obtain the distinctive properties of a given malware specimen. By inputting an image into the model, we use the obtained features to categorize the malware into its respective malware families. This provides valuable data for analysts who lack prior expertise and may be effectively implemented in IoT applications to assist in the development of safeguards against malware. The model has been verified by several trials, proving its ability to reach an accuracy of 96.64%. In [23] researchers introduce a new approach for extracting characteristics of behavior. It encodes absolute time data in characteristic sequences and uses a construction approach that considers covariance to make the system adaptable to individuals. Researchers chose a stacked bidirectional LSTM and a forward neural network to construct a deep learning detection of insider threats model called Behavior Rhythm Insider Threat Detection (BRITD). It can handle a wide range of internal threat situations and does a great job of finding insider threats, with an AUC of 0.9730 and an accuracy of 0.8072 with the CMU CERT dataset, beating all comparison benchmarks. Also, the authors in [24], the main goal of the researchers is to suggest using a long short-term memory (LSTM) network to find problems and a decision engine that can effectively find any intrusions. The detection process achieves this by leveraging several contextual aspects. Researchers have conducted rigorous testing of anomaly detection algorithms and decision engines utilizing authentic data obtained from actual vehicles. They thus provide the outcomes of the studies and thoroughly examine their discoveries. In [25], the authors used a Brazilian malware classification dataset with 121000 instances and 57 different attributes of a PE file. They used label encoding and one-hot encoding in their pre-processing phase to convert categorical data to machine-readable form, followed by extensive feature selection using ETC, wherein the attributes were reduced to only the 15 most important features. Then they used seven different ML models, namely DT, RF, GB, SVM, LR, XGB, and AdaBoost, to train the dataset. Amongst these, DT and RF were given at a staggering 99.7% accuracy.

In addition to the above, [26] studied including a proactive feature to anticipate malware attacks before they happen and respond appropriately to them in the cloud. They used a dataset provided by Microsoft, which had 4.04 GB of training and 3.5GB of testing data with approximately 9 million instances. They used GBDT algorithms, namely LGBM and XGB, to classify the instances. Some features with higher importance scores were used to predict the model's outcomes. The output returned two features containing the user's prediction and location ID. The observations were that LGBM displayed an accuracy of 73.89% with five folds for 500 iterations, whereas XGB displayed an accuracy of 70.49% with 1-fold for 14,000 iterations. The duration of implementation is where they diverged noticeably. LGBM completes the task in around 6 hours, 16.7 times quicker than XGB. The authors in [27] provide a feature selection framework and incorporate multiple machine learning methods to categorize the security level for ransomware detection and prevention in this study. The dataset utilized in this study comprised 138,047 samples with 54 characteristics. Feature selection methods, including variance threshold and variance inflation factor, were used to eliminate low variation and highly linked attributes from the data. Finally, different ML techniques, including DT, RF, NB, LR, and NN-based classifiers, were applied to a subset of 15 characteristics for ransomware classification. The experimental findings show that RF classifiers beat other approaches regarding accuracy, precision, and recall. RF gave a staggering 99% accuracy with 10-fold cross-validation. In [28], the authors extended the existing work on their dataset, containing 1156 instances consisting of 984 malicious and 172 benign instances. They added RF classifiers for feature selection, wherein they removed the least important features. They also incorporated 15-fold cross-validation to split the dataset to reduce overfitting and improve the model's accuracy

score and overall precision acumen. They entailed different machine learning methods, including KNN, SVM, BNB, DT, RF, LR, and HV. In binary classification, DT, RF, and HV (hard voting) models acquired the best accuracy at 98%, 97.8%, and 97%, respectively. Regarding multi-classification, RF, DT, and HV models attained the maximum accuracy at 95.8%, 92%, and 92%, respectively. In this article [29], the authors utilized a dataset including malicious and benign data from the classification of Malware with PE headers (Clamp) dataset provided by GitHub for their study. There were 2722 malware incidents and 2488 benign cases in the dataset. There are 69 characteristics in total in the dataset. The authors employed Principal Component Analysis (PCA) to conduct feature reduction. A total of 40 significant aspects were chosen. Classic machine learning algorithms like KNN, SVM, DT, RF, NN, AB, ETC, LDA, LR, PC (Passive Classifier), RC (Ridge Classifier), and SGD were used to classify these files. 10-fold cross-validation was also used to calculate the mean accuracy of the models. Amongst these ETC gave the best accuracy of 98.8%. After that, different deep models were utilized to increase the prediction accuracy further, namely MLP and CNN. Dense-1 and Dense-2 neural networks with varying neurons were used; 1D-CNN with 35 neurons gave the best accuracy of 98.5%. In a different approach, they also incorporated an ensemble approach wherein they combined dense-1, dense-2, and 1D-CNN. In the case of meta-learners, various classical ML algorithms were used. The result showed that the ensemble approach using ETC as the meta-learner gave the best accuracy of 99.9% with the above deep learning model.

Also in this study [30], the author tried to simulate a real-world setup to predict malware online. OpenStack's cloud setup was used on the sandbox and was supplied with benign files for a while, and then malware was injected. The malware files were taken from VirusTotal and contained 40680 samples. Various ML algorithms were deployed to predict the malware: CNN, SVC, RF, KNN, GB, and GNB. Amongst these CNN DenseNet-121 model gave the best accuracy of 92.9%. In [31], the authors used the CIC dataset containing 42 malware variants. In total, it had 10854 instances and 88 features, out of which 4354 were malware and 6500 benign. Based on the correlation matrix, 77 important attributes were selected. Label encoding was used because of the presence of non-numerical data to convert them into machine-readable data. 70% of the dataset was used for training, and the rest 30% for testing. ML models like MLP, KNN, and RF were used to classify the files. After that, RF performed better with 62.2% accuracy; further tuning the model with parameters like max-depth and no of layers, the accuracy increased to 95.6%. The authors in [32] used a Debrian-215 dataset containing a total of 15036 cases in the dataset, including 5,560 malware samples and 9,476 benign samples. It had a total of 215 attributes. They calculated feature relevance using a tree-based classifier and chose ten essential characteristics. Algorithms include LR, KNN, NB, DT, RF, SVM, XGB, AD, and GB. The algorithms were trained on 70% of the dataset and tested on the remaining 30%. A k-fold cross-validation procedure with a value of 10 is used to measure the predictive performance of machine learning models. The result indicated that the XGB algorithm provided the highest accuracy of 98.72%. In [33], the authors used an open-source dataset contributed by Max Secure Systems, the malware partner of the Meraz-18 fest. The dataset had a total of 138047 instances of labeled training data and 88258 unlabelled test data, both having 55 features. Various ML classifiers were utilized, such as GNB, QDA, RF, LR, AB, KNN, DT, SVM, MLP with single layer training, MLP with multiple training layers and smaller epoch value, and MLP with multiple training layers and more considerable epoch value. The performance acumen of the models was tested using 10-fold cross-validation to reduce the overfitting problem. It was observed that standardized data takes less time to get trained. RF performed best with 99.51% mean accuracy, followed by DT with 99.24% accuracy over all 55 features. MLP with 400 neurons also gave 99.25% accuracy but had a higher computational cost. The authors observed that models with greater epochs and deeper or multiple layers performed better. The computational cost and memory requirement was very high with all the models deployed.

Moreover, in [34], several machine learning approaches were used in this study to dynamically analyze the registry, APIs, DLLs, and summary data. Amongst these, gradient boosting performed best with 94.64% accuracy. More than 92 static features are retrieved from executable files, and machine learning is then applied to the static features. From these, again, gradient classifiers gave the best result with 99.36% accuracy. The authors of the future would like to elaborate on this work by including deep learning models and an even larger dataset to make the model more robust. In [35], the authors utilized ZeVigilante, a framework to detect the activity of Zero-Day malware by incorporating several ML techniques. The dataset was from IEEE DataPort, wherein ZeVigilante's cuckoo-based correlated dataset was extracted. A balance filter was used to balance the dataset. The API calls and PE imports were used to analyze the malware files. The final dataset was split into 70% for training and the rest 30% for testing purposes Several ML algorithms were used for training and testing purposes, namely KNN, RF, NN, DT, SVM, and NB. Cross-validation with 10 folds was used to test the mean accuracy of each of these models. RF gave the best results with dynamic and static accuracies of 98.89% and 98.17%, respectively. The authors in [36], utilized the frequency of opcode occurrences to consider it as an indicator to differentiate malware files from benign files. Different executable malware files from different websites were taken and executed in the cuckoo sandbox. For training purposes, 100 benign files and 150 malware files, all collected from various websites, were considered. For testing purposes,

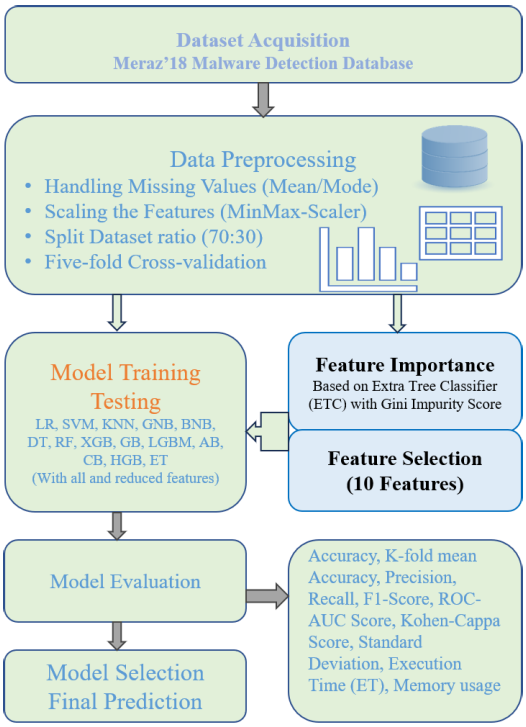

FIGURE 1. Proposed model workflow diagram for Malware Detection.

a total of 100 files were chosen for model testing. Three classification algorithms were used, namely RF, NB, and DT. Performance metrics like precision, recall, and f1-score were also evaluated to check the prediction acumen. It was observed that RF gave the highest accuracy of 97.97% accuracy.

Some of the Android-based malware prediction models were presented in different literature. A few of them are mentioned here to address the issues, but this study is not focused on an Android-based prediction model based on ML techniques. The authors in [37] worked on a larger dataset with 215 features to classify malware on Android devices. The dataset used was highly imbalanced, so to avoid ambiguity, up-sampling was done with the minority class. Feature reduction methods such as RFE and RFECV reduce features from 215 and vary them from 15 to 100. Eight different ML algorithms were used: LR, SVM, GNB, DT, RF, GB, LGBM, and XGB. Amongst these, LGBM gave the highest accuracy (train-test split) and cross-validation accuracy (10 folds) of 99.29% and 99.50%, respectively, using only 100 features. On the other hand, RF gave an accuracy of 99.1% with just 55 features. Also, [38], [39], and [40] proposed an android malware prediction model on a reduced feature set based on machine learning approaches. In [41], the authors proposed a hybrid ConvLSTM-CNN-driven android-based malware detection framework. They used an Android malware dataset and deployed a model with ten-fold cross-validation with an accuracy of 98.97%. In [42], the authors proposed text-based malware detection to control adversarial attacks using image visualization achieving an accuracy of 99.48%. We summarize the related literature in Table 10, highlighting each study's proposed model, dataset, key findings, strengths, and weaknesses for a clear comparative overview.

All the above-mentioned studies for malware detection have addressed either higher detection accuracy of using all features or the reduced number of features. However, the experimental results of the research with a reduced number of features are not satisfactory, as they led to a noticeable drop in key performance metrics like accuracy and F1-score. For example, accuracy declined when fewer features were used, showing that these features are crucial for effective malware detection. This decrease suggests that reducing features significantly weakens the model's ability to classify malware accurately. Considering the limitations of existing malware detection frameworks, we propose an efficient malware detection approach based on machine learning feature influence techniques for resource-constrained devices including low-power IoT devices, such as microcontrollers and sensors, as well as Android-based smartphones and tablets commonly used in edge computing contexts. These devices, often limited in processing power, memory, and battery life, necessitated the development of lightweight and efficient models.

III. PROPOSED METHODOLOGY

This section has provided a detailed discussion of the proposed methodology for the experimental setup. The experiment has required a dataset for deploying the ML model. In the first subsection, we have presented details of the collected dataset and described its features. In the second subsection, we have offered an in-depth explanation of the data preprocessing techniques employed to convert the raw dataset into a usable, processed version. We have then deployed the model using various machine learning algorithms to train and test the classifiers, followed by a comprehensive analysis. The experimental results obtained from the model have been described using several performance metrics. Figure 1 represents the workflow diagram of the proposed model.

A. DATASET COLLECTION AND DESCRIPTION

We utilized an open-source malware classification dataset from Kaggle [16] that was taken from Max Secure Software, a security partner of Meraz'18. It was a binary classification dataset, malware and genuine files were present in the raw data. This dataset is extracted from the statistical features of the portable executable (PE) files for the Windows operating system environment and also each cell values indicate the entropy of the different subsections of the PE files. Software that is intended to disrupt, harm, or gain unauthorized access to a computer system is known as malware. Legitimate files are programs that are safe for users to utilize and do not behave maliciously. The raw data had a total of 138047 instances with 55 independent features and one target attribute class. The legitimate class indeed outnumbered

the malware class in the dataset, with 96,724 instances of legitimate samples (class "0") and 41,323 instances of malware samples (class "1"). This severe imbalance has been noted as it influences model training and evaluation. The 'ID' column value is observed as a serial number that why it is omitted from the dataset as a feature. The feature description and their corresponding data types are represented in Table 1.

B. DATA PRE-PROCESSING

Data Pre-processing is the act of gathering raw data and putting it into a format that a computer or machine can interpret. Data cleansing, transformation, and balancing are the three primary phases in data pre-processing. In general, imbalanced data, missing or null or implausible zero values, a mix of categorical and numerical features, and unscaled data can be found in raw datasets collected from any open source or repository. When ML algorithms are used for these datasets, the results might be deceptive. We applied several pre-processing techniques on raw datasets to tackle these difficulties. The approaches utilized in the dataset pre-processing are discussed below. During the data analysis phase of the actual dataset, a total of fifty-five independent features are renamed with the abbreviation "H" and numbered serially as per the features that appear in the original dataset to simplify the entire execution process. The Pearson correlation coefficient matrix was found in every feature with the target class. The name of every feature with their short name along with correlation values is represented in Table 2. A Pearson correlation between variables X as features and Y as target class is calculated in Equation 1

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

1) MISSING OR IMPLAUSIBLE ZERO VALUES

There were no missing or null values or implausible zero values in the collected open-source dataset. so, there is no need for any imputation techniques to handle these situations.

2) DATA STANDARDIZATION/SCALING

Scaling is the process of minimizing the distance between data points to normalize the range of attributes in a dataset. Scaling the data allows a model to learn and grasp the problem more easily. As is customary, the values in the dataset were in a variety of units and scales. A few data points had unusually high and low values. We utilized the Min-Max Scaler function in this study to scale the data in a specified range of 0 to 1. Equation 2 represents the process of Min-Max scaling, which involves transforming the attributes data to a predetermined range, often from 0 to 1.

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2}$$

where:-

- x is an individual data point
- x_{scaled} is the scaled value of x in the specified range between 0 and 1.
- $\min(x)$ is the minimum value of x in the dataset.
- $\max(x)$ is the maximum value of x in the dataset.

3) FEATURE TYPE

The dataset contained no categorical variables, all 55 features, and the outcome attribute had numerical values. Since all features, including the target feature, were already provided as labeled numerical values in the dataset, there was no need to apply any encoding techniques. Categorical data did not require conversion, so methods like one-hot encoding or label encoding were unnecessary.

4) DATA BALANCING

Upon initial inspection, the dataset appeared to be severely unbalanced, with the legitimate class significantly outnumbering the malware class. However, further analysis of model performance indicated that the high volume of instances in both classes provided sufficient representation for each category. This abundance of data allowed the model to learn effectively without the need for additional data balancing techniques, such as under-sampling or over-sampling, which could risk information loss or introduce synthetic data that might not accurately reflect real-world instances. Consequently, we decided to proceed without applying balancing techniques, as the model demonstrated stable and satisfactory performance with the existing class distribution. The target variable data distribution is depicted in Figure 2

The feature correlation heat map in Figure 3 utilizes the Pearson correlation coefficient to capture linear relationships between features, which is particularly useful for identifying redundancies and influential features. Dark green cells indicate strong positive correlations, while lighter greens reflect weaker or no correlations. We applied a significance threshold to filter out weakly correlated pairs, ensuring that only meaningful relationships are visualized. This approach allows us to reduce computational demands by focusing on a subset of important features, which is crucial for enhancing efficiency in resource-constrained environments.

C. MODEL TRAINING

The fourteen different ML classifiers with default parameters were implemented on the dataset to train the models for binary classification of malware detection namely LGBM, HGB, XGB, AB, CB, GB, RF, DT, SVM, LR, GNB, BNB, ETC, and KNN. To deploy the prediction model for the detection of malware, 70% of the total data instances of the dataset were used for training, and the remaining 30% were used for testing the model performance. Also, the k-fold cross-validation of five-fold techniques [43], [44] was applied to access the mean accuracy and also to eliminate the overfitting issue of the deployed model. For performance analysis, we employed measures such as

TABLE 1. The description of all the features of the considered dataset.

No.	Feature Name	Feature Type	No.	Feature Name	Feature Type
1	Machine	Categorical	28	SizeOfHeapReserve	Integer
2	SizeOfOptionalHeader	Integer	29	SizeOfHeapCommit	Integer
3	Characteristics	Categorical	30	LoaderFlags	Integer
4	MajorLinkerVersion	Float	31	NumberOfRvaAndSizes	Integer
5	MinorLinkerVersion	Float	32	SectionsNb	Integer
6	SizeOfCode	Integer	33	SectionsMeanEntropy	Float
7	SizeOfInitializedData	Integer	34	SectionsMinEntropy	Float
8	SizeOfUninitializedData	Integer	35	SectionsMaxEntropy	Float
9	AddressOfEntryPoint	Integer	36	SectionsMeanRawsize	Integer
10	BaseOfCode	Integer	37	SectionsMinRawsize	Integer
11	BaseOfData	Integer	38	SectionMaxRawsize	Integer
12	ImageBase	Integer	39	SectionsMeanVirtualsize	Integer
13	SectionAlignment	Integer	40	SectionsMinVirtualsize	Integer
14	FileAlignment	Integer	41	SectionMaxVirtualsize	Integer
15	MajorOperatingSystemVersion	Float	42	ImportsNbDLL	Integer
16	MinorOperatingSystemVersion	Float	43	ImportsNb	Integer
17	MajorImageVersion	Float	44	ImportsNbOrdinal	Integer
18	MinorImageVersion	Float	45	ExportNb	Integer
19	MajorSubsystemVersion	Float	46	ResourcesNb	Integer
20	MinorSubsystemVersion	Float	47	ResourcesMeanEntropy	Float
21	SizeOfImage	Integer	48	ResourcesMinEntropy	Float
22	SizeOfHeaders	Integer	49	ResourcesMaxEntropy	Float
23	CheckSum	Integer	50	ResourcesMeanSize	Integer
24	Subsystem	Categorical	51	ResourcesMinSize	Integer
25	DllCharacteristics	Categorical	52	ResourcesMaxSize	Integer
26	SizeOfStackReserve	Integer	53	LoadConfigurationSize	Integer
27	SizeOfStackCommit	Integer	54	legitimate	Boolean

k-fold mean accuracy, accuracy, recall, precision, F1-score, Cohen-kappa score, and ROC-AUC score, respectively. The Execution Time (ET) and Memory Requirement (MR) for each deployed model were also considered, keeping in mind the resource-constrained device to evaluate the model efficiency. We conducted a comparative examination of the efficacy of all ML models and observed that DT, RF, XGB, GB, AB, and HGB provided an accuracy of 100%. On further cross-validation with five-fold, it was observed that LGBM and CB along with the six of the above-mentioned models all gave k-fold mean accuracies of 100%. Other performance metrics produced excellent outcomes as well. As the above models already performed with excellent efficacy overall, so there is no requirement for hyperparameter tuning. The experimental results of all the prediction models are depicted in Table 3 and Table 4 indicates the total execution time and peak memory usage required to evaluate the model performance.

D. FEATURE IMPORTANCE MODEL

To find out the feature importance among all the fifty-five features of the above-mentioned dataset, the Embedded Extra-Tree Classifier (ETC) [45] was applied using the criterion 'Gini' impurity score [46]. The Gini impurity is calculated using the Equation 3. During the data analysis phase of the entire features of the dataset, it is observed

TABLE 2. Feature name and correlation coefficient values of each feature concerning target class.

#	Column/Feature Name	Pearson Correlation Coefficient values with a target label	#	Column/Feature Name	Pearson Correlation Coefficient values with a target label
H1	ID	-0.793226	H29	SizeOfHeapReserve	-0.156260
H2	Machine	0.548835	H30	SizeOfHeapCommit	-0.002506
H3	SizeOfOptionalHeader	0.547498	H31	LoaderFlags	-0.002649
H4	Characteristics	0.221956	H32	NumberOfRvaAndSizes	-0.003523
H5	MajorLinkerVersion	0.017320	H33	SectionsNb	-0.207782
H6	MinorLinkerVersion	-0.146652	H34	SectionsMeanEntropy	-0.343933
H7	SizeOfCode	0.017476	H35	SectionsMinEntropy	-0.152840
H8	SizeOfInitializedData	-0.004958	H36	SectionsMaxEntropy	-0.624229
H9	SizeOfUninitializedData	-0.003997	H37	SectionsMeanRawsize	0.001175
H10	AddressOfEntryPoint	-0.000134	H38	SectionsMinRawsize	0.059346
H11	BaseOfCode	-0.006232	H39	SectionMaxRawsize	-0.000790
H12	BaseOfData	-0.001136	H40	SectionsMeanVirtualsize	0.001734
H13	ImageBase	0.008245	H41	SectionsMinVirtualsize	0.056466
H14	SectionAlignment	-0.002429	H42	SectionMaxVirtualsize	-0.001332
H15	FileAlignment	0.125169	H43	ImportsNbDLL	0.038395
H16	MajorOperatingSystemVersion	0.002402	H44	ImportsNb	0.116415
H17	MinorOperatingSystemVersion	-0.001702	H45	ImportsNbOrdinal	0.128112
H18	MajorImageVersion	0.084410	H46	ExportNb	0.134408
H19	MinorImageVersion	0.083220	H47	ResourcesNb	0.090405
H20	MajorSubsystemVersion	0.380393	H48	ResourcesMeanEntropy	-0.202432
H21	MinorSubsystemVersion	-0.001213	H49	ResourcesMinEntropy	0.299112
H22	SizeOfImage	-0.002603	H50	ResourcesMaxEntropy	-0.392855
H23	SizeOfHeaders	0.010125	H51	ResourcesMeanSize	-0.003824
H24	CheckSum	-0.195329	H52	ResourcesMinSize	-0.001774
H25	Subsystem	0.514352	H53	ResourcesMaxSize	-0.005529
H26	DllCharacteristics	-0.630177	H54	LoadConfigurationSize	-0.011666
H27	SizeOfStackReserve	-0.521642	H55	VersionInformationSize	0.379646
H28	SizeOfStackCommit	-0.003226			

that the three columns named "ID: H1", "Machine: H2", and "SizeOfOptionalHeader: H3" and their corresponding cell values are constant except for "ID" numbers. There is no significance of "ID" as a feature and due to the constant values of the other two columns, it does not have any significance as a feature in this case of the feature importance method. So, three columns were dropped from the total columns and finally, fifty-two columns were selected as features. ETC in general implements a meta-estimator that employs averaging to increase prediction accuracy and reduce overfitting. The meta-estimator fits a number of randomized decision trees (also known as extra trees) on different sub-samples of the dataset.

Selecting features based on the feature's importance using one of the criteria like 'Gini' can be highly rewarding. Reduction of the number of features reduces the amount of data processed and hence reduces execution time. Since feature selection has the ability to greatly improve the

TABLE 3. Experimental results of all the deployed models considering all features.

Model	A (%)	MA (%)	SD	RA	P	R	F	CK
RF	100.0000	100.00000	0.000000	1.000	1.000	1.000	1.000	1.000
DT	100.0000	99.997930	0.000025	1.000	1.000	1.000	1.000	1.000
XGB	100.0000	99.998965	0.000021	1.000	1.000	1.000	1.000	1.000
AB	100.0000	99.997930	0.000025	1.000	1.000	1.000	1.000	1.000
GB	100.0000	99.997930	0.000025	1.000	1.000	1.000	1.000	1.000
HGB	100.0000	99.998965	0.000021	1.000	1.000	1.000	1.000	1.000
CB	99.9952	99.997930	0.000025	1.000	1.000	1.000	1.000	1.000
LGBM	99.9903	99.997930	0.000041	1.000	1.000	1.000	1.000	1.000
SVM	99.9710	99.963780	0.000236	1.000	0.999	1.000	1.000	0.999
ETC	99.9590	99.946187	0.000159	1.000	0.999	1.000	0.999	0.999
KNN	99.9300	99.889270	0.000198	1.000	0.998	1.000	0.999	0.998
LR	99.9203	99.898584	0.000308	1.000	0.998	0.999	0.999	0.998
BNB	94.8014	94.797787	0.001134	0.941	0.985	0.837	0.905	0.870
GNB	59.3167	62.093238	0.055072	0.927	0.421	0.997	0.592	0.301

TABLE 4. Execution time & memory requirement of all the considered models.

Model	ET (Seconds)	Peak Memory (MiB)	Increment (MiB)
RF	72.866	3012.05	4.18
DT	2.019	3006.62	1.36
XGB	61.483	3021.84	4.82
AB	2.180	3030.91	0.07
GB	153.355	3036.53	5.04
HGB	21.266	3050.68	12.75
CB	32.805	3142.21	63.99
LGBM	33.614	3073.45	21.15
SVM	433.654	992.76	47.36
ETC	61.484	2404.1	0.05
KNN	257.597	1160.09	0.02
LR	9.223	1072.23	0.45
BNB	1.697	2978.03	11.66
GNB	1.904	2915.42	53.73

models' performance, it may be a crucial step in the machine learning prediction process. The quantity of data needed to train a model increases exponentially with the number of features used to train the model. Even though it may seem logical to provide the model with all the information possessed the idea is that the more data supplied, the better it can learn and generalize. This is commonly referred to as the curse of dimension.

$$\text{Gini}(p) = 1 - \sum_{i=0}^{1} p_i^2 \tag{3}$$

where p_i is the percentage of class i observations in the dataset or subset.

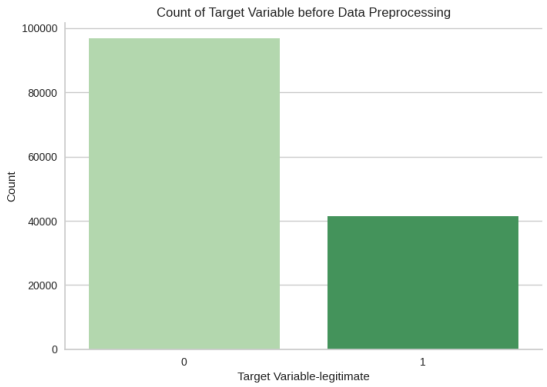

FIGURE 2. The legitimate target column data distribution.

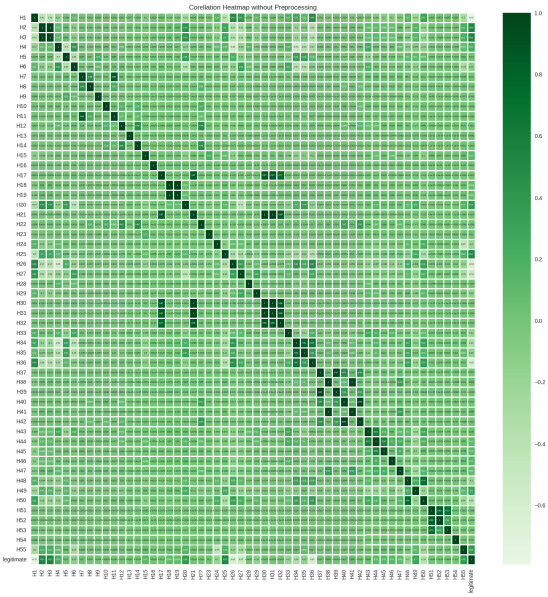

FIGURE 3. The feature correlation heat map.

This study intends to apply ETC to determine the significance of each feature in terms of how it affects the malware prediction model's accuracy. It was discovered that some characteristics had importance index scores that were substantially greater than those of other features. The execution time of the model may be significantly decreased by training it with fewer features in a large dataset, and this can also assist in achieving the same or even greater accuracy with fewer data.

After applying the feature importance technique, the embedded ETC calculates the Gini impurity score of each feature. Table 5 indicates the features' rank in decreasing order and plots the rank accordingly in the same order in Figure 4 to visualize the actual importance of each feature.

According to the feature importance score, the top 10 features were selected to deploy the final prediction model for malware detection. The corresponding correlation coefficient values of each feature indicate the linear relationship with the target variable ranging from -1 to $+1$, either perfect higher negative or positive correlation. It is observed that the highest negative and positive correlation values are selected as the highest importance of features in the embedded ETC method. The top ten features with their importance score and correlation values are represented in Table 6.

In our Feature Importance Model, the Extra Trees Classifier (ETC) assigns feature importance scores based on the frequency and quality of splits that each feature provides within its decision trees. Unlike simple correlation-based selection, this model considers how frequently a feature contributes to the decision boundaries and the quality of these contributions in enhancing model accuracy. Thus, features are ranked not solely by their correlation values but by their overall utility to the classification task as evaluated by the ETC. While features with extreme positive and negative correlation values are initially considered for selection due to their potential significance, correlation alone does not determine feature importance. The ETC method evaluates each feature's actual impact on predictive accuracy, so even if a feature shows high correlation, it may be excluded if it contributes minimally to the model's performance during training. This approach ensures that only features with a tangible impact on accuracy are prioritized.

Similar data pre-processing approaches and the same type of ML models were considered for the malware prediction model. The reduced number of features with five-fold cross-validation techniques was used to compute the model performance using different metrics. It is observed that the RF model performed well with accuracy and mean accuracy of 100%, the standard deviation is null and all other parameters reached maximum gain. But in terms of execution time and memory requirement, the DT model performed better results with less ET and less memory space needed. Concerning resource-constrained devices with less processing, smaller memory units, and limited power backup requirements, the DT and AB models demonstrated robust performance in this study. AdaBoost (AB), as a boosting-based ensemble classifier, minimizes error iteratively, adapting effectively to complex relationships within the data. While Naive Bayes variants, such as GNB, initially showed high accuracy due to the dataset's high multicollinearity and absence of outliers, they were less adaptable than AB under these conditions. This is primarily because Naive Bayes relies on feature independence and conditional probability, which limits its robustness in the presence of high inter-feature correlations. The reduced features model performances are shown in Table 7 and Table 8.

Table 8 illustrates the differences in computation time and memory usage across various models. These differences are primarily due to the models' complexity and design. Deep learning models, with their large parameter sets and multi-layered structures, inherently require more memory and processing time to handle extensive computations. Conversely, simpler models, such as decision trees and linear classifiers, are more lightweight and efficient, making

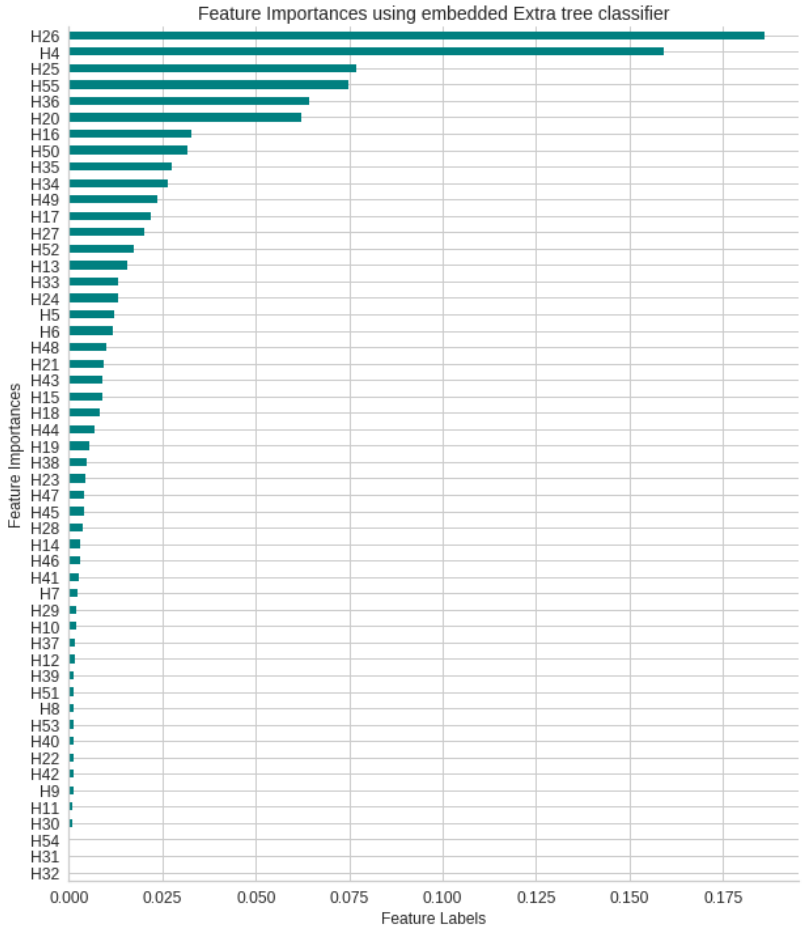

FIGURE 4. Features of importance label according to Gini Impurity values.

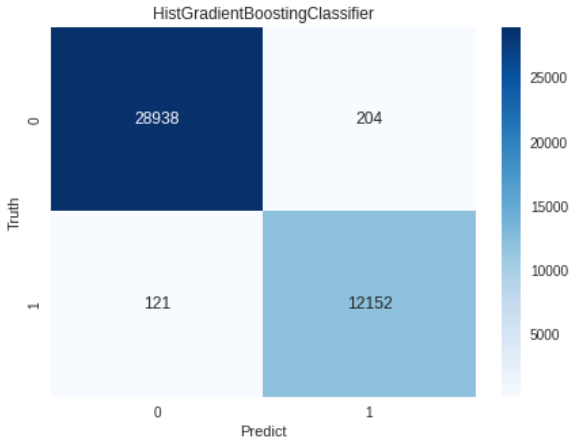

FIGURE 5. The confusion matrix of the HGB model.

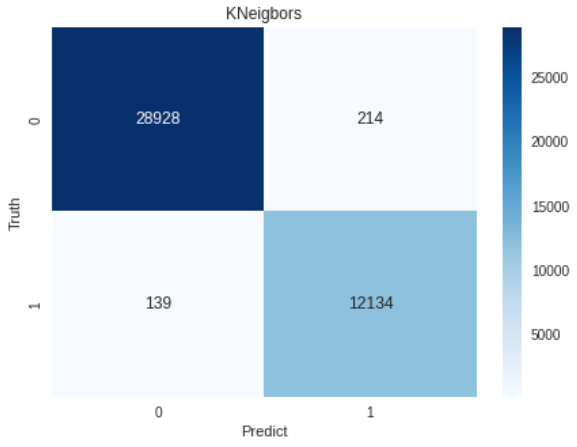

FIGURE 6. The confusion matrix of the KNN model.

them suitable for environments with limited computational resources. Furthermore, models that incorporate feature selection or dimensionality reduction steps can achieve reduced computational demands by focusing on a smaller, relevant subset of features. These factors collectively explain the observed variations in resource usage across different models.

TABLE 5. Features importance score concerning Gini Impurity values of all features of the dataset.

Features Name	Gini Impurity score	Features Name	Gini Impurity score	Features Name	Gini Impurity score
H26	0.186086	H6	0.011644	H10	0.001956
H4	0.159214	H48	0.010127	H37	0.001690
H25	0.076922	H21	0.009403	H12	0.001628
H55	0.074759	H43	0.008993	H39	0.001385
H36	0.064260	H15	0.008947	H51	0.001365
H20	0.062274	H18	0.008371	H8	0.001332
H16	0.032815	H44	0.006896	H53	0.001308
H50	0.031606	H19	0.005606	H40	0.001308
H35	0.027557	H38	0.004586	H22	0.001303
H34	0.026430	H23	0.004376	H42	0.001234
H49	0.023793	H47	0.004152	H9	0.001189
H17	0.021893	H45	0.004087	H11	0.001017
H27	0.020098	H28	0.003535	H30	0.000823
H52	0.017204	H14	0.003100	H54	0.000089
H13	0.015701	H46	0.002895	H31	0.000006
H33	0.013094	H41	0.002496	H32	0.000005
H24	0.013086	H7	0.002256		
H5	0.012038	H29	0.002064		

TABLE 6. Top-Ten features importance score with Pearson's Correlation values.

Feature #	Feature Name	Feature Importance Value	Correlation values
H26	DllCharacteristics	0.186086	-0.630177
H4	Characteristics	0.159214	0.221956
H25	Subsystem	0.076922	0.514352
H55	VersionInformationSize	0.074759	0.379646
H36	SectionsMaxEntropy	0.064260	-0.624229
H20	MajorSubsystemVersion	0.062274	0.380393
H16	MajorOperatingSystemVersion	0.032815	0.002402
H50	ResourcesMaxEntropy	0.031606	-0.392855
H35	SectionsMinEntropy	0.027557	-0.152840
H34	SectionsMeanEntropy	0.026430	-0.343933

The confusion matrix of the top seven models whose false negative (FN) values are almost zero percent in the test model is shown in Figure 5 to Figure 11, and the ROC-AUC curve of all the deployed models is depicted in Figure 12 concerning to reduced features model.

IV. COMPARATIVE ANALYSIS WITH THE SOTA MODELS

The comparative analysis of experimental results of the related literature with the proposed research has been presented by considering the different performance metrics. Only the ML-related malware prediction and classification

TABLE 7. Experimental outcomes of the reduced features models.

Model	A	MA	SD	RA	P	R	F	CK
RF	99.3891	99.2528	0.000470	0.999	0.988	0.992	0.990	0.985
ETC	99.3867	99.2694	0.000536	0.999	0.988	0.992	0.990	0.985
LGBM	99.2998	99.2466	0.000636	1.000	0.985	0.991	0.988	0.983
HGB	99.2153	99.1525	0.000540	1.000	0.983	0.990	0.987	0.981
CB	99.2080	99.0748	0.000384	0.999	0.984	0.990	0.987	0.981
KNN	99.1477	99.0293	0.000454	0.996	0.983	0.989	0.986	0.980
DT	99.0414	98.9279	0.000405	0.989	0.983	0.985	0.984	0.977
GB	98.9303	98.8813	0.000144	0.999	0.982	0.982	0.982	0.974
XGB	98.9086	98.8782	0.000389	0.999	0.981	0.982	0.982	0.974
AB	98.6816	98.5885	0.000412	0.998	0.982	0.974	0.978	0.968
GNB	93.1209	93.6398	0.012393	0.984	0.830	0.966	0.893	0.843
LR	89.4555	93.1565	0.035792	0.963	0.815	0.834	0.824	0.749
BNB	70.3803	69.9541	0.000110	0.713	0.875	0.001	0.001	0.001

TABLE 8. Execution time & memory requirements of all the deployed model.

Model	ET (Seconds)	Peak Memory (MiB)	Increment (MiB)
RF	37.660163	1464.17	0.00
ETC	18.113758	1470.13	0.00
LGBM	11.479233	1464.17	0.00
HGB	7.950551	1464.17	0.00
CB	5.215514	1467.95	3.78
KNN	33.331011	1464.14	0.00
DT	1.950927	1464.14	0.00
GB	48.564586	1464.17	0.00
XGB	20.964785	1464.17	0.00
AB	15.836037	1464.17	0.00
GNB	0.536815	1464.14	0.00
LR	7.780586	1464.13	0.00
BNB	0.667455	1464.14	0.00

model for static PE files for the Windows system is considered to compare with this study. The main contributing factor to concluding the novelty of this work is the parameters like accuracy (A), mean accuracy (MA), precision (P), recall (R), and ROC-AUC score (RA). The execution time and memory usage were also calculated to conclude from a resource-constrained device perspective.

To compare the proposed experimental outcomes with the article [25] demonstrates that the authors conducted their experiment after applying the preprocessing steps one-hot encoding and label encoding to convert the categorical features into numerical, but our collected dataset does not have any categorical features and no need to apply any encoding techniques. Due to the dummy variable trap, there might be an increase in the execution time and memory usage that is not suitable and advisable in resource-constrained IoT devices. Also, due to the high number of feature selections (15 features) in the article [25] somehow achieved a slightly high accuracy compared to our research only selection of 10 features, the experimental results achieved the highest with

TABLE 9. Result analysis of the proposed model with SOTA literature.

#Ref	Model(s) used	A	P	R	F	RA	CK
[21]	RF, DT, KNN,GNB, MNB	98.94					
[25]	DT, RF, GB, SVM, LR, XGB, AB	99.70					
[26]	LGBM, XGB	73.89					
[27]	DT, RF, NB, LR, NN	99.00	0.99	0.97	0.97		
[28]	KNN, SVM, BNB, DT, RF, LR, and HV	97.80	0.96	0.87	0.91		
[29]	KNN, SVM, DT, RF, NN, AB, ETC, LDA, LR, PC RC, SGD	98.80	0.98	0.98	0.98	0.99	0.98
[30]	CNN, SVC, RF, KNN, GB, and GNB	92.90	1.00	0.85	0.92		
[31]	MLP, KNN, RF	95.60					
[32]	LR, KNN, NB, DT, RF, SVM, XGB, AD, GB	98.72				0.99	
[33]	GNB, QDA, RF, LR, AB, KNN, DT, SVM, MLP (55 Features)	99.51					
[34]	AB, GB, LR, RF, DT, Bagging, Linear Ridge	99.36				0.99	
[35]	KNN, RF, NN, DT, SVM, NB.	98.89	0.99	0.99	0.99	0.98	
[36]	RF, DT, NB	97.97	0.97	0.98	0.97		
Proposed Model	RF, based on a 10-feature selection	99.39	0.99	0.99	0.99	0.99	0.99

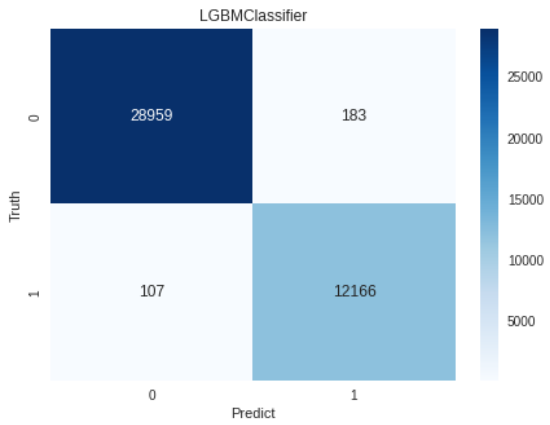

FIGURE 7. The confusion matrix of the LGBM model.

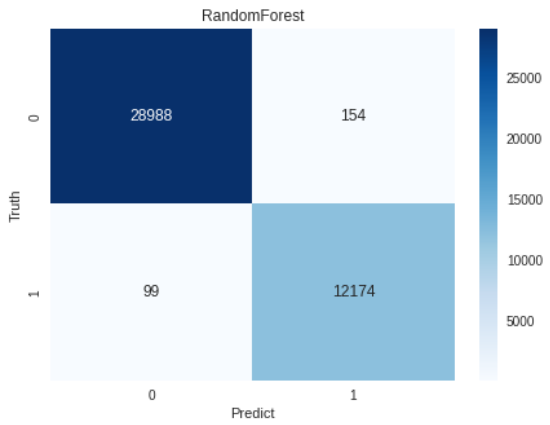

FIGURE 8. The confusion matrix of the RF model.

less execution time and are suitable in real-time resource-constrained IoT devices. Moreover, our research brilliantly handled the different ranges of data instances using the MinMax normalization preprocessing approach.

To evaluate performance under different computational loads, we experimented with two feature sets: a reduced set of the top ten most relevant features and the full set of fifty-five features. The reduced feature set ultimately provided the optimal balance between accuracy and computational efficiency, making it the final configuration selected for deployment. The comparison with SOTA models is based on the performance of this final ETC model using the top ten features. This approach not only demonstrates competitive accuracy but also ensures that the model remains efficient

TABLE 10. Analysis and comparison of the proposed model with the SOTA literature model.

#Ref	Dataset	Feature Selection	Best Model & Accuracy	Strengths	Weaknesses
[21]	42797 malware, 1079 benign	Under-sampling	RF (98.91%) on imbalanced dataset	High accuracy on imbalanced data, diverse ML models used	Imbalanced dataset, limited data preprocessing
[25]	Brazilian dataset with 121000 instances	ETC for top 15 features	DT & RF (99.7%)	High accuracy, extensive ML model comparison	Limited feature diversity, model scalability concerns
[26]	Microsoft dataset with ∼9 million instances	Not specified	LGBM (73.89%)	Proactive attack anticipation, fast execution	Lower accuracy, high resource requirements
[27]	138,047 samples	Variance threshold, VIF	RF (99%)	Excellent performance metrics, reliable feature selection	Limited dataset diversity, no proactive feature
[28]	1156 instances	RF	DT, RF, HV (97.80%)	High accuracy with cross-validation, low overfitting	Small dataset, potential overfitting in cross-validation
[29]	Clamp dataset, 2722 malware, 2488 benign	PCA to reduce to 40 features	ETC (98.8%)	High accuracy, deep model integration	High computational cost, complex model stacking
[30]	VirusTotal samples, 40680 instances	Not specified	CNN (92.9%)	Real-world simulation setup, CNN integration	Lower accuracy, absence of feature selection
[31]	CIC dataset with 10854 instances	Correlation matrix for 77 features	RF (95.6%)	Improved accuracy after tuning	Initial low accuracy, parameter tuning overhead
[32]	Debrian-215 with 15036 cases	Tree-based feature selection	XGB (98.72%)	Balanced dataset, robust feature relevance calculation	Dataset-specific results, lack of deep learning
[33]	Max Secure Systems, 138047 labeled, 88258 unlabeled	Not specified	RF (99.51%)	High accuracy, variety of models	High computational cost, memory-intensive
[34]	Executable files	Static feature retrieval	Gradient Boosting (94.64%)	High accuracy in static features	Lower accuracy in dynamic analysis
[35]	ZeVigilante's IEEE DataPort	Balance filter	RF (98.89%)	Zero-day detection, balanced data	Limited dataset diversity
[36]	250 malware, 100 benign	Opcode occurrences	RF (97.97%)	Unique feature selection, high accuracy	Small dataset, limited model diversity
Study	Custom dataset with feature reduction	ETC for top 10 features	RF (99.39%) with reduced features	High accuracy, low memory and execution time, robust & resilient model	Potential scope for further deep learning integration

for real-time malware detection in IoT devices. The added comparative analysis highlights the strengths and weaknesses of our model in terms of accuracy, processing time, and memory usage relative to other SOTA techniques, underlining the practical advantages of our approach in IoT settings. The comparison with this proposed work and the result analysis is represented in Table 9. The blank cell indicates the corresponding literature has not observed any results to the corresponding parameters. Also in Table 10, we provide a comprehensive summary of the related literature, detailing each study's proposed model, dataset, claimed outcomes, strengths, and limitations. This table serves to give a clear and structured comparative overview of various approaches, helping to contextualize our study within the existing research landscape and illustrating the advantages and challenges associated with each method. By including information on

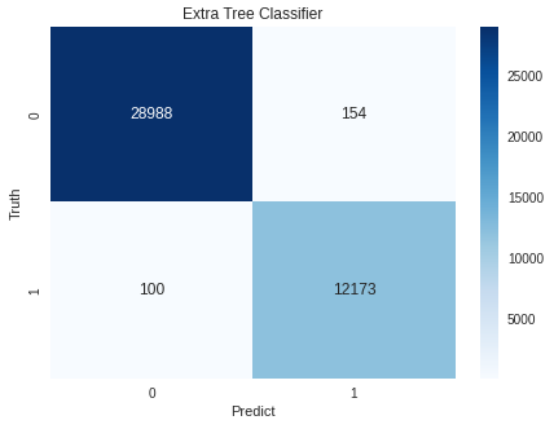

FIGURE 9. The confusion matrix of the ETC model.

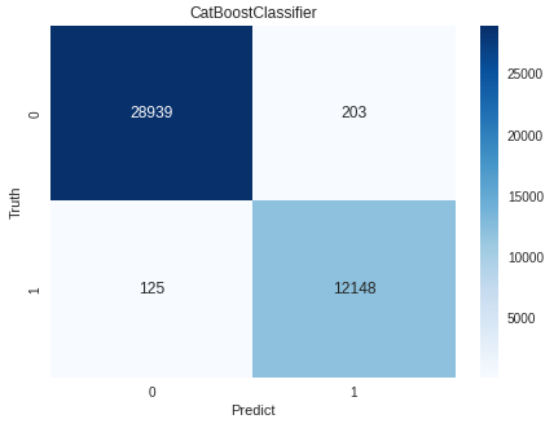

FIGURE 10. The confusion matrix of the CB model.

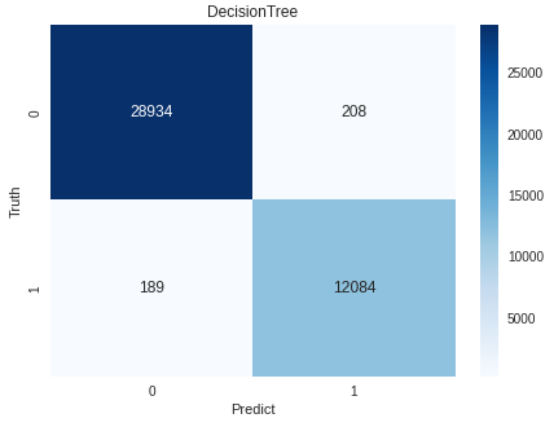

FIGURE 11. The confusion matrix of the DT model.

key findings, strengths, and weaknesses, we aim to offer insights into how different models and datasets have been leveraged in prior studies, thereby informing our approach and emphasizing the novelty and relevance of our proposed model.

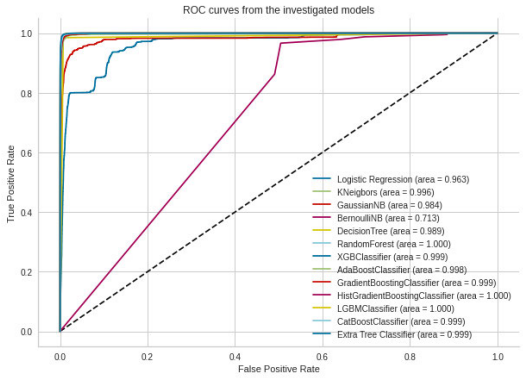

FIGURE 12. The RoC-AuC probabilistic curve of all the deployed models.

V. CONCLUSION AND FUTURE SCOPE

This work establishes a new standard for predicting and detecting malware using traditional ML methods. It takes into account all relevant criteria and reduces the number of important data points associated with a PE file label for each instance, whether it is genuine or malicious. Initially, research is done by deploying fourteen ML models with all the features on an acquired dataset. The findings demonstrate great promise, as the RF and DT models achieved a mean accuracy of 100% with little standard deviation. However, it is worth noting that both models require significant execution time and memory uses, which may be impractical for resource-constrained devices. To reduce the execution time of the proposed model, the ETC is selected to determine the most important features based on their impurity values. The top ten features are selected according to their correlation values and important scores. The model is tested with five-fold cross-validation techniques at reduced features on all the deployed models as before. The RF model performed much better with an accuracy of 99.39%, a ROC-AUC score of 0.99, and less standard deviation, and mean accuracy indicating a non-overfitted model. Also, the other five models LGBM, HGB, CB, KNN, and DT performed better with an accuracy of 99% and above. The execution time and memory usage are very less reduced to half on the reduced feature model of RF. The presented model is robust, stable, and responsive concerning resource-constrained devices. In the future, the model will be tested and incorporated with the flavor of artificial neural networks and combined with other diverse datasets.

A. AVAILABILITY OF DATA AND MATERIAL

The unprocessed data was acquired from the malware security partner of Meraz'18, the annual techno-cultural event of IIT Bhilai. The initial data set consisted of both malware and genuine files. It can be accessed at https://www.kaggle.com/competitions/malware-detection/data?select=data.csv. The deployment code is with the custody of the equal contributing author, Subhash Mondal (ph22cse1001@cit.ac.in), and upon request it will be shared.

B. FUNDING

There is no financial or non-financial funding from any source. It is not applicable.

C. ACKNOWLEDGEMENTS

The authors express their heartfelt appreciation to anonymous peer reviewers who generously dedicated their time and expertise to review and provide constructive feedback on this research paper. Also, the extent to acknowledge the Dayananda Sagar University, Bengaluru, India, and Central Institute of Technology Kokrajhar, Assam, India for providing us the research support.

REFERENCES

[1] I. Ideses and A. Neuberger, "Adware detection and privacy control in mobile devices," in *Proc. IEEE 28th Conv. Electr. Electron. Eng. Isr. (IEEEI)*, Dec. 2014, pp. 1–5.

[2] A. Iqbal, M. N. Aman, R. Rejendran, and B. Sikdar, "Unveiling the connection between malware and pirated software in Southeast Asian countries: A case study," *IEEE Open J. Comput. Soc.*, vol. 5, pp. 62–72, 2024.

[3] D. Cevallos-Salas, F. Grijalva, J. Estrada-Jiménez, D. Benítez, and R. Andrade, "Obfuscated privacy malware classifiers based on memory dumping analysis," *IEEE Access*, vol. 12, pp. 17481–17498, 2024.

[4] S. Poornima and R. Mahalakshmi, "Automated malware detection using machine learning and deep learning approaches for Android applications," *Meas. Sensors*, vol. 32, Apr. 2024, Art. no. 100955.

[5] M. Aljabri, F. Alhaidari, A. Albuainain, S. Alrashidi, J. Alansari, W. Alqahtani, and J. Alshaya, "Ransomware detection based on machine learning using memory features," *Egyptian Informat. J.*, vol. 25, Mar. 2024, Art. no. 100445.

[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/data?select=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.

[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. Aid Sci. 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. Grigaliūnas, "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.

[31] D. Chandrakala, A. Sait, J. Kiruthika, and R. Nivetha, "Detection and classification of malware," in *Proc. Int. Conf. Advancements Electr., Electron., Commun., Comput. Autom. (ICAECA)*, Oct. 2021, pp. 1–3.

[32] U. V. Nikam and V. M. Deshmuh, "Performance evaluation of machine learning classifiers in malware detection," in *Proc. IEEE Int. Conf. Distrib. Comput. Electr. Circuits Electron. (ICDCECE)*, Apr. 2022, pp. 1–5.

[33] F. Abri, S. Siami-Namini, M. A. Khanghah, F. M. Soltani, and A. S. Namin, "Can machine/deep learning classifiers detect zero-day malware with high accuracy?" in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 3252–3259.

[34] M. Ijaz, M. H. Durad, and M. Ismail, "Static and dynamic malware analysis using machine learning," in *Proc. 16th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST)*, Jan. 2019, pp. 687–691.

[35] F. Alhaidari, N. A. Shaib, M. Alsafi, H. Alharbi, M. Alawami, R. Aljindan, A.-U. Rahman, and R. Zagrouba, "ZeVigilante: Detecting zero-day malware using machine learning and sandboxing analysis techniques," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–15, May 2022.

[36] P. Mohandas, S. K. S. Kumar, S. P. Kulyadi, M. S. Raman, V. Vasan, and B. Venkataswami, "Detection of malware using machine learning based on operation code frequency," in *Proc. IEEE Int. Conf. Ind. 4.0, Artif. Intell., Commun. Technol. (IAICT)*, Jul. 2021, pp. 214–220.

[37] N. A. Sarah, F. Y. Rifat, M. S. Hossain, and H. S. Narman, "An efficient Android malware prediction using ensemble machine learning algorithms," *Proc. Comput. Sci.*, vol. 191, pp. 184–191, Jan. 2021.

[38] J. Lee, H. Jang, S. Ha, and Y. Yoon, "Android malware detection using machine learning with feature selection based on the genetic algorithm," *Mathematics*, vol. 9, no. 21, p. 2813, Nov. 2021.

[39] A. Sangal and H. K. Verma, "A static feature selection-based Android malware detection using machine learning techniques," in *Proc. Int. Conf. Smart Electron. Commun. (ICOSEC)*, Sep. 2020, pp. 48–51.

[40] A. Mahindru and A. L. Sangal, "FSDroid:- a feature selection technique to detect malware from Android using machine learning techniques: FSDroid," *Multimedia Tools Appl.*, vol. 80, no. 9, pp. 13271–13323, Apr. 2021.
[41] I. U. Haq, T. A. Khan, A. Akhunzada, and X. Liu, "MalDroid: Secure DL-enabled intelligent malware detection framework," *IET Commun.*, vol. 16, no. 10, pp. 1160–1171, Jun. 2022.
[42] F. A. Demmese, A. Neupane, S. Khorsandroo, M. Wang, K. Roy, and Y. Fu, "Machine learning based fileless malware traffic classification using image visualization," *Cybersecurity*, vol. 6, no. 1, p. 32, Dec. 2023.
[43] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *J. Roy. Stat. Soc. Ser. B, Stat. Methodol.*, vol. 36, no. 2, pp. 111–133, Jan. 1974.
[44] S. Geisser, "The predictive sample reuse method with applications," *J. Amer. Stat. Assoc.*, vol. 70, no. 350, p. 320, Jun. 1975.
[45] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.
[46] B. Li, J. Friedman, R. Olshen, and C. Stone, "Classification and regression trees (cart)," *Biometrics*, vol. 40, no. 3, pp. 358–361, 1984.

SUBIR PANJA received the Bachelor of Engineering degree from The Burdwan University, the Master of Engineering degree from Jadavpur University, and the Ph.D. degree in IoT security from the Central Institute of Technology Kokrajhar, Assam. He is currently an Associate Professor with the Academy of Technology, Adisaptagram, Hooghly, West Bengal. He has over 17 years of teaching expertise. His research interests include cyber security, the IoT applications, image classification, resource optimization using AI and ML, and resource optimization.

SUBHASH MONDAL (Member, IEEE) received the B.Tech. and M.Tech. degrees in computer science and engineering from the University of Calcutta, Kolkata, India, in 2005 and 2007, respectively. He is currently pursuing the Ph.D. degree in CSE with the Central Institute of Technology Kokrajhar, Kokrajhar, Assam, India.

He is currently an Assistant Professor in CSE (AI & ML) with Dayananda Sagar University, Bengaluru, Karnataka, India. He has more than 18 years of teaching experience. His research interests include advanced machine learning, deep learning, computer vision, bio-informatics, federated learning, and the IoT allied to artificial intelligence. He has published more than 34 research publications in various journals, book chapters, and conferences. He is a Faculty Adviser of IEEE RAS and CIS, SBC, and DSU.

AMITAVA NAG (Senior Member, IEEE) is currently a Professor in computer science and engineering with the Central Institute of Technology Kokrajhar, Assam, India. He has more than 50 research publications in various international journals and conference proceedings. His research interests include the IoT, information security, advanced machine learning, deep learning, and computer vision. He is a fellow of IEI.

JYOTI PRAKASH SINGH (Senior Member, IEEE) is currently an Associate Professor with the Department of Computer Science and Engineering, National Institute of Technology Patna, India. He was involved as an investigator in the MeitY-sponsored project to develop algorithms for spam calls/fake calls in a telephonic conversation. He has co-authored seven textbooks and one edited book. He has published over 65 international journal publications in SCIE-indexed journals of reputed publications, such as IEEE, ACM, Springer, Elsevier, and Taylor & Francis. He has also published more than 60 international conference proceedings. His research interests include social media mining, malware analysis, information security, and text analytics. He is a Senior Member of ACM. He is a Life Member of the Computer Society of India (CSI), Indian Society for Technical Education (ISTE), and the Institution of Engineers (IE). He received the S4DS Data Scientist (Academia) Award by the Society for Data Science, in 2020. He is an Associate Editor of the *International Journal of Electronic Government Research*. He has been recognized as world ranking of the top 2% scientists in the area of artificial intelligence, in 2021, according to a survey given by Stanford University, USA.

MANOB JYOTI SAIKIA (Senior Member, IEEE) received the B.E. degree in electronics and communication engineering from Visvesvaraya Technological University, Belgaum, India, in 2009, the M.Tech. degree in bioelectronics from Tezpur University, Tezpur, Assam, India, in 2013, and the Ph.D. degree in electrical engineering from the University of Rhode Island, Kingston, RI, USA, in 2019.

He was an Assistant Professor with the Electrical Engineering Department, University of North Florida. He was a Research Associate with the School of Engineering, Tufts University, from September 2019 to August 2022. He was also a Senior Research Associate with the Department of Engineering, Boston College, from May 2022 to August 2022. From January 2016 to July 2019, he was a Research Assistant in a project funded by the National Science Foundation, USA. He is currently the Director of the Biomedical Sensors & Systems Laboratory and an Assistant Professor with the Electrical and Computer Engineering Department, University of Memphis. From 2012 to 2015, he was awarded a Senior Research Fellowship from the Ministry of Science and Technology, India, working with Indian Institute of Science, Bengaluru, India. His research interests include biomedical instrumentation, sensors, neuroimaging (fNIRS and EEG), signal processing, machine learning, and the Internet of Things.

ANUP KUMAR BARMAN (Member, IEEE) received the M.Sc., M.Tech., and Ph.D. degrees from Gauhati University, Assam. He is currently an Assistant Professor in computer science and engineering with the Central Institute of Technology Kokrajhar, Assam, India. He is actively engaged in a project called CLIA-Cross-Lingual Information Access executed in collaboration with various reputed institutes, such as Gauhati University, IIT Mumbai, IIT Hyderabad, and IIT Kharagpur. He is also engaged in various research activities, such as the development of the stemmer, word sense disambiguation module and parser, and so on for the Assamese language. He has more than 20 research publications in various international journals and conference proceedings. His research interests include natural language processing, machine learning, information retrieval, and information security.