SN Computer Science (2024) 5:162
https://doi.org/10.1007/s42979-023-02516-3

ORIGINAL RESEARCH

Classification of Malware from the Network Traffic Using Hybrid and Deep Learning Based Approach

Praful R. Pardhi[1,3] · Jitendra Kumar Rout[2] · Niranjan Kumar Ray[1] · Santosh Kumar Sahu[4]

Received: 15 May 2023 / Accepted: 21 November 2023 / Published online: 8 January 2024

Abstract

Mobile connectivity and smart devices are spreading worldwide. As a result, the use of mobile devices and applications is rising exponentially. Therefore, nowadays hackers target such smart devices to steal information and misuse it for malicious purposes. It becomes absolutely essential to protect sensitive information such as app. permissions, login credentials, browse history, media contents etc. from intruders. Security can be breached easily if smart techniques are not devised to safeguard mobile data. In this article, an attempt is made to classify the different types of malware and to protect the sensitive information on Android devices that significantly reduce network congestion and improve network throughput by increasing data transmission. The proposed hybrid approach consists of AdaBoost, random forest and deep learning methods jointly classify the sophisticated malware. The empirical results indicate that this achieves better classification and detection accuracy and is capable of identifying the potential threat more efficiently.

Keywords Malware classification · Mobile computing · Threats · Network security · Machine learning

Introduction

The usage of mobile computing is growing with the advent of innovative and smart technologies. The users can perform many tasks on mobile devices and may offload the tasks or applications to the cloud for faster execution of the tasks on the remote resources with more computational power [1]. The increased usage of mobile and handheld devices has also given rise to security threats. It is not easy to escape from these security threats. Mobile devices are also at risk, just like desktop machines [2].

There is a need for newer techniques that are dynamic in nature to handle these security threats. Most of the time, malware attacks the data. Mobile computing is evolving just like aligned technologies such as cloud computing, web computing, and fog computing [3]. The mobile devices undergo vertical and horizontal handoff while switching from one region to another; during the handoff, the network is also changed, which also gives rise to security threats [4]. Mobile communications require three things: software support, service provider support, and hardware support. The front end completely relies on the software and network technologies [5]. If the intruders get an entry to the user's mobile by breaching the network security in terms of malware, then the entire data of

Jitendra Kumar Rout, Niranjan Kumar Ray and Santosh Kumar Sahu contributed equally to this work.

✉ Jitendra Kumar Rout
jitu2rout@gmail.com

Praful R. Pardhi
pardhipr@rknec.edu

Niranjan Kumar Ray
rayniranjan@gmail.com

Santosh Kumar Sahu
santoshsahu@hotmail.co.in

1 School of Computer Engineering, KIIT Deemed to be University, Bhubaneshwar, Odisha, India

2 Department of Computer Science and Engineering, National Institute of Technology, Raipur, Chhattisgarh, India

3 Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, Maharashtra, India

4 GEOPIC, Oil Natural Gas Corporation Ltd., Dehradun, Uttarakhand, India

the user can be misused and the complete device can be hacked [6]. The facilities provided by mobile devices are appreciable and adaptable, but so much risk is involved in using mobile applications where user data is shared with the app providers [7]. The intruder may breach the security of the mobile device by getting entry into your device through malware, viruses, unsecured network connections, and malicious data [8].

There have been many solutions proposed by researchers in the past to deal with the problems of security threats to mobile users, but there is still a necessity to explore more innovative techniques to save the sensitive data of the mobile users as well as to save the mobile users from the intruders [9]. Mobile security mainly relies on the protection of mobile devices from the security threats associated with the exchange of data, wireless connections, and change of service provider during handoff. There are numerous malicious applications available that pose security risks to mobile users [10].

The Android-based mobile devices can adopt Machine Learning (ML) techniques for the classification of data and malware to protect the device from unwanted intruders [11–13]. The method for detecting the malware should be capable enough to detect the malware correctly or with high accuracy, and secondly, to protect the mobile device or user from the unseen attack by the intruder [14]. The Android-based malware detection techniques can be divided into two parts, i.e., dynamic detection approaches and static malware detection approaches [15, 16]. The static malware detection approaches make use of static or historical data, and the model is built on the basis of existing data [17]. The model will remain stringent and will not change with the change in mobile traffic or with the change in user behavior. The dynamic approaches can pose greater potential to learn from the dynamic environment and to detect malware quickly, irrespective of the volume of the traffic [18, 19].

The mobile devices are highly portable and can cover any geographical area while the user travels from one place to another. The main motto of the service provider is to provide a seamless service to the mobile user, irrespective of the geographical area [20]. The smart technologies have altered the traditional ways of connectivity of devices, and now the usage of high-tech smart technologies is made to provide uninterrupted services to mobile users [21, 22]. This seamless connectivity creates opportunities for hackers and application crackers to breach the mobile and/or network security to steal the information or data of the mobile users or to get complete control over the mobile device through a Trojan horse attack through malwares [23]. It is certainly very challenging to identify the threat and inhibit the technology's exploitation for saving mobile data. The growing utilization of varied communication technologies is producing serious threats such as bottlenecking of mobile traffic, information theft, and complete control of the user's mobile through remote access [24].

There are many advanced approaches for identifying security threats by using smart technologies in mobile and hand-held devices, but accuracy is still a factor of concern [25, 26]. The next section provides insights into the existing literature. This paper uses ML-based dynamic techniques to classify mobile traffic and, especially, identify network threats in terms of malware.

The research contributions of this manuscript are enlisted below:

- The data is collected in a real environment to generate a dataset. The dataset is scaled up to prepare a balanced dataset out of the collected data so that the dataset can be used as an input to the ML-based classifiers.
- Data visualization was done to determine the data distribution as well as identify whether the data is balanced or not.
- The balanced dataset is utilized to train the ML-based models. The classifiers have four target values to classify the mobile traffic into benign (benign traffic) and three abnormal classes that comprise malware (ransomware, SMSmalware, and scareware).
- Apart from ML models, Hybrid and Deep Learning (DL) based models were also applied and performance was evaluated.
- The proposed model permits the exclusion of normal mobile traffic generated by mobile devices from malicious and malware-oriented traffic. It also maximizes the throughput of the mobile network, minimizes the jamming of connections and maximizing the data transmission rate.

The next part of this article is devoted to elaborating on the state-of-the-art works. Section 3 explained the proposed mechanism to identify malware and segregate the malware-based data from the normal data over mobile channels. The results are presented in Sect. 4. The article is summarized in Sect. 5.

Related Work

There are many advanced approaches for identifying the security threats posed on mobile devices, including malicious traffic, malware, and spyware. The smart technologies in mobile and handheld devices are used for the identification of threats, but accuracy is still a factor of concern. We are proposing a novel mechanism to identify the threats to Android-based mobile platforms and address the gaps in the existing literature. This section explains the existing literature in the area of our research.

Malicious software detection plays a crucial role in the computer security and mobile security fields. A malware-based detection scheme is used for detection. Methods like metamorphism and polymorphism are used by malware creators to bypass these detection methods and breach security. Baghirov [11] has tried to pitch for controlling this security threat by presenting a detailed study on malware detection schemes. Apart from this, the research has provided detailed information on challenges for malware detection, tools to detect the malware, and existing remedies to control the security threats posed by the malware. Jahromi et al. [14], have discussed the two-hidden-layered-based Extreme Learning method. The method considers the local dependencies of similar features and determines the global dependencies of the extreme model in the hidden layer. Bagging is also used as an ensembling method to achieve more accurate results for detecting the malware. Jeon and Moon [15] have presented a novel malware detection with DL and opcode sequences. In this paper, malware detection using opcode sequences is discussed. The proposed method comprises a convolutional auto-encoder that converts long opcodes to short opcode sequences, which leads to an accuracy of 96% for detecting malware.

Wadkar et al. [17] proposed malware evolution detection using linear Support vector machine (SVM). Malware in a single set can be created from various code modifications. Changes within malware families can be identified by applying feature ranking based on linear SVM. Souri and Hosseini [18] have described a detailed survey on malware detection methods. The changing nature of malware makes the malware identifier's job harder to recognize the patches of malware in the normal codes. The survey also includes a detailed discussion about malware detection challenges, systematic approaches for identifying the malware, the structure of the malware codes, and malware classification factors. Kumar et al. [20] have presented the identification of malware using a neural network approach. It uses various neural network techniques like Convolution Neural Network (CNN), Recurrent Neural Network (RNN), and deep networks for the classification of malware. Raghavan et al. [21] have used the Hidden Markov Model (HMM) for the identification of malware, which is based on the hill climbing approach. The normal HMM is replaced by the boosted HMM, which yields more accurate results for malware detection. Darabian et al. [23] have described the detection of frequent opcode sequences for Internet of Things (IoT)-based apps using mining techniques. The maximally frequent patterns of opcodes are utilized to differentiate malicious applications from benign IoT applications. Different ML-based classifiers such as SVM, AdaBoost, multi-layer perceptron, decision tree, and random forest are used to detect security threats. Ren et al. [27] have proposed visualization methods for malware analysis. The authors have utilized two different visualization methods and applied these methods to Microsoft malware samples. These methods obtain good accuracy for malware detection: 98.36 percent with the former method and 99.08 percent with the lateral method. In the same way, Surendran et al. [28], have proposed Tree Augmented Naive Bayes (TANB) to detect malware on Android platforms. The dependencies between stagnant and dynamic characteristics are analyzed using TAN, and then the hybrid malware detection method is employed for the detection of the malware at the application level. The proposed method identifies whether the application is malicious or benign. The resultant method exhibits 97 percent accuracy. Table 1 shows the comparison of the existing methods using parameters like author names, publication year, methodology used, dataset utilized, types of samples used for analysis, accuracy achieved, attack detection, network traffic classification, and the limitations of the existing approaches.

Although many methods are discussed by the researchers for detecting the malware, gaps are still found in terms of applying these techniques on the Android platform for detecting malware accurately and safeguarding the data of mobile users by segregating these malwares from the network traffic and only allowing benign applications. Hence, this paper proposes newer methods that analyze the traffic features for identifying the malware and segregating them to optimize the network bandwidth and its performance.

Performance Measures

The following Performance metrics are used to assess model accuracy & effectiveness, guiding improvements through quantitative evaluation.

- *Confusion Matrix:* This provides the values regarding the correctly identified records from the dataset with respect to each class.
- *Accuracy:* This parameter specifies the strength of the classifier by depicting the correctly made predictions by machine learning-based approaches.
- *Recall Score:* This parameter specifies the true positive cases that are identified in a correct manner by the classifier.
- *Specificity/Precision:* This parameter specifies the true negative cases that are correctly identified by the classifier.
- *F1-Score:* This is the mean of the specificity and recall values obtained by the classifier.

Table 1 Comparison of different approaches for Malware detection

Authors	Year	Methodology	Dataset used	Sample types	ACC	TC	Limitation
Baghirov [11]	2021	static, dynamic, Hybrid analysis & Machine learning methods discussed	VX Heaven, Kaggle, VirusShare, Ember, UCI Repository, Malheur	viruses, botnets, rootkits, backdoors, ransomware, spyware, or keyloggers	–	No	metamorphism and polymorphism methods bypass the signature-based techniques
Jahromi et al. [14]	2020	Two hidden layer-based extreme learning using LSTM & CNN Model	VXHeaven, Kaggle, Windows ransomware, IoT malware	Ransomware, IoT and a mix of different malware samples	99.65%	No	model considers only local dependencies rather than global dependencies between input features for malware and botnet detection
Jeon et al. [15]	2020	Convolutional Recurrent Neural Network Using Opcode Sequences from the executable file	own dataset comprising 1000 benign files and 1000 malware for the experiment	ransomware, spyware, adware, trojans, and backdoors	96%	No	Only Sequential opcode sequence is considered for experimentation
Wadkar et al. [17]	2020	linear support vector machine (SVM) based machine learning	Malicia dataset, VirusShare	trojan, worm, adware, backdoor	–	No	Proposed approach focused on static features only extracted from PE files
Souri et al. [18]	2018	signature- and behavior-based data mining techniques	ClamAV, VirusTotal, Google play store, Windows sandbox Malware, VirusTotal	–	95%	No	Failing to detect the polymorphic malware. Space and Time complexity for behavioral patterns
Kumar et al. [20]	2020	Neural network approach	Avron virus, Kaggle, Androguard	spyware, adware, trojans, and backdoors	96%	No	This approach will not work for all kinds of malwares. Not effective for unseen attacks
Raghavan et al. [21]	2019	Hidden Markov models	Malicia dataset	trojans, backdoors, and spyware	91.4%	No	Accuracy of malware detection depends on the boosting classifiers
Darabian et al. [23]	2019	Sequential pattern mining technique to detect frequent opcode sequences in IoT apps	Virus Total	IoT malware, goodware, and polymorphic malware	99%	No	More features can be extracted using stream online data mining for more accurate results
Ren et al. [27]	2020	Deep convolution neural networks	Kaggle	trojans, and backdoors	99.08 %	No	Only Grayscale images are used by the model for malware pattern detection
Surendran et al. [28]	2020	Tree Augmented naive Bayes	Drebin, AMD, AndroZoo and Github	adware, trojans, Trojan SMS, backdoors and Ransomeware	97%	No	Some malware applications may not be detected because of their adversarial methods
Proposed approach		ML & Deep Learning	Kaggle	Benign, Ransomware, SMSmalware and Scareware	97%	Yes	More accurate results can be obtained by processing the data faster way

Proposed Methodology

Android malware detection consists of the process of detecting the malicious process or program on a mobile device. This ensures the safety of mobile devices from harmful viruses and malicious attacks. A mobile-based malware dataset with 20,000 records is utilized to train the ML-based intelligent model, where Android platform-based data is used from Kaggle (supplied by Google for the experimental study). The proposed malware detection scheme uses the data in an 8:2 ratio, where training is performed on 80 percent of the data available in the dataset and testing is conducted on the remaining 20 percent. The proposed approach starts with data preparation.

Data Preparation

In this study, there are two datasets are used to training, validation and testing process. The details about these datasets are discussed as follows:

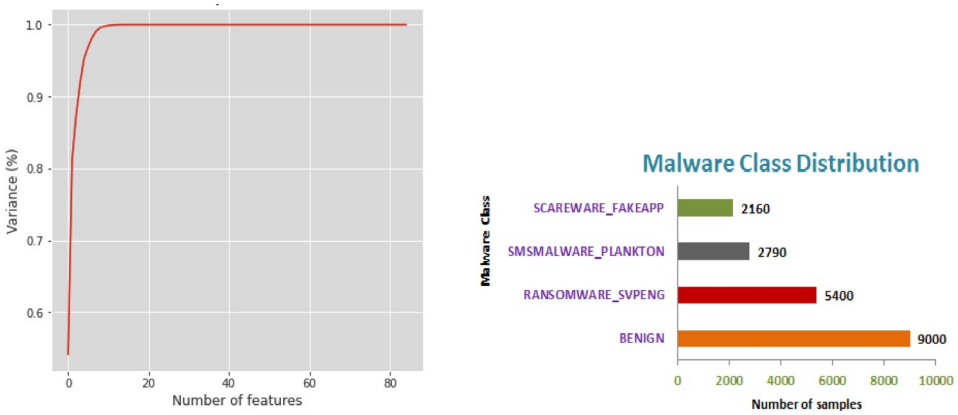
Android Malware Dataset

There are a total of 86 features in the input dataset. Hence, we need to apply feature engineering to the same. Feature engineering helps to build a complex but interpretable model by using all the features of the raw dataset. There are two steps in feature engineering:

- The selection of important features is based on a few statistical functions or domain knowledge, and the least important features are removed.
- Dimensionality reduction is performed using algorithms like PCA or T-SNE to reduce the number of features or merge the existing features into a smaller number of features.

Both of the above feature engineering methods will help increase the speed of the ML algorithm, improve accuracy, and avoid over-fitting. The most relevant features based on the variance that covers almost whole dataset as shown in Fig. 1a and the no. of features are listed in a heat map.

Table 2 shows the sample feature values of the first few records from the dataset considered for the evaluation of four types of malware. Table 3 gives the basic information about the dataset. We have calculated statistical measures like mean, median, mode, and standard deviation to correlate the different features and identify the relationships between the different features using the standard deviation, which helps us determine how the flow of information varies



(a) Feature reduction using PCA (b) Malware Class distribution

Fig. 1 **a** Feature reduction using PCA and **b** Malware Class distribution of the Android Dataset

Table 2 Sample feature values of the Malware dataset

Record No	Flow duration	Total fwd packets	Total backward packets	Received bytes	Flood status	Label
0	3	0.822038	0.822038	0.001838	0.23455	BENIGN
1	9	0.275513	0.275513	0.002236	0.460725	SCAREWARE_FAKEAPP
2	3	0.923707	0.923707	0.001751	0.00000	SMSMALWARE_PLANKTON
3	9	0.368775	0.368775	0.001776	0.439255	SCAREWARE_FAKEAPP
5	3	0.905217	0.905217	0.001767	0.00000	SMSMALWARE_PLANKTON

Table 3 Basic information of the used dataset

Description	Information
Number of rows	19350
Number of columns	86
Number of categorical features	2
Categorical features	label, node_status
Number of float features	56
Float Features	*act_data_pkt$_f$wd*, *active_max*, *active_mean*, *active_min*, *average_packet$_s$ize*, *bwd_packet_length_max*, *etc*.
Number of integer features	28
Integer features	ack_flag_count, avg_bwd_segment_size, bwd_avg_bytes_bulk, fwd_iat_mean, fin_flag_count, flow_bytes, flow_duration etc.

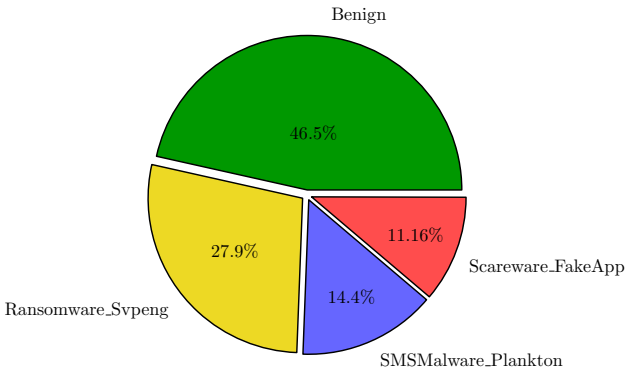


Fig. 2 Class distribution in percentage on Android Malware Dataset

and is useful for detecting four classes of malware from the available dataset.

From Fig. 1b, it can be concluded that the values are properly distributed and there is no data imbalance. The ML algorithms are applied to the balanced and clean data to retrieve the results. A pie chart in Fig. 2 shows the malware class distribution in percentage.

The statistical relationship between two variables is referred to as their correlation. A correlation could be positive, meaning both variables move in the same direction, or negative, meaning that when one variable's value increases, the other variables' values decrease. Correlation can also be neutral or zero, meaning that the variables are unrelated. Fig. 4 shows the heat map of selected features after feature reduction. Fig. 5 shows the boxplot of the features used for the outlier calculation, and a histogram of the distribution of data values is shown in Fig. 6 Examples: Positive Correlated Variables=*fwd_iat_total* and *total_fwd_packets*, Negative Correlated Variables=*fwd_iat_total* and *total_length_of_fwd_packets* and Not Correlated = *fwd_iat_total* and *total_length_of_bwd_packets*. Fig. 7 shows the outliers used in the heat map calculation.

Malware Android Dataset [29]

Utilizing machine learning for Android malware detection is an approach aimed at discerning and categorizing potentially harmful apps designed for Android devices. An accurate method for gauging application suspiciousness involves monitoring the network to which the Android device is connected. Machine learning, a subset of artificial intelligence, concentrates on creating computer programs with the ability to access and learn from data autonomously. This technology enables the construction of models that assess incoming data to make predictions and pinpoint anomalies. This capability can be harnessed for identifying malicious Android apps by constructing a model that seeks out patterns indicative of malicious behavior. The model can incorporate various attributes like requested permissions, API calls, network interactions, and more. Following training, the model can be deployed to classify new applications as either malicious or benign. The class distribution of this dataset is shown in Fig. 3.

The dataset comprises four distinct labels: Android_Adware, Android_Scareware, Android_SMS_Malware, and Benign. In total, the dataset encompasses 355,630 entries (rows) across 85 columns. The dataset was sourced from the CIC repository. Currently, the data exhibits the subsequent distribution of labels as given in Fig. 3.

Malware Classification

Android-based mobile data classification is a multiple classification problem where the classification of mobile data is made on the basis of four classes: benign traffic, ransomware-based data, SMS malware-based data, and scareware-based data. The normal traffic comprises benign traffic, and the remaining three are a part of the malicious category, or malware [30]. The architecture diagram of malware classification is described in Fig. 8.

Classification of the mobile-generated traffic with the aid of ML techniques

Ensembling-based techniques such as AdaBoost and Random Forest (RF) have been employed in our problem statement for detecting malware on Android platforms. The malware are classified into four classes as: benign data, ransomware-based malware, SMS malware, and scareware.

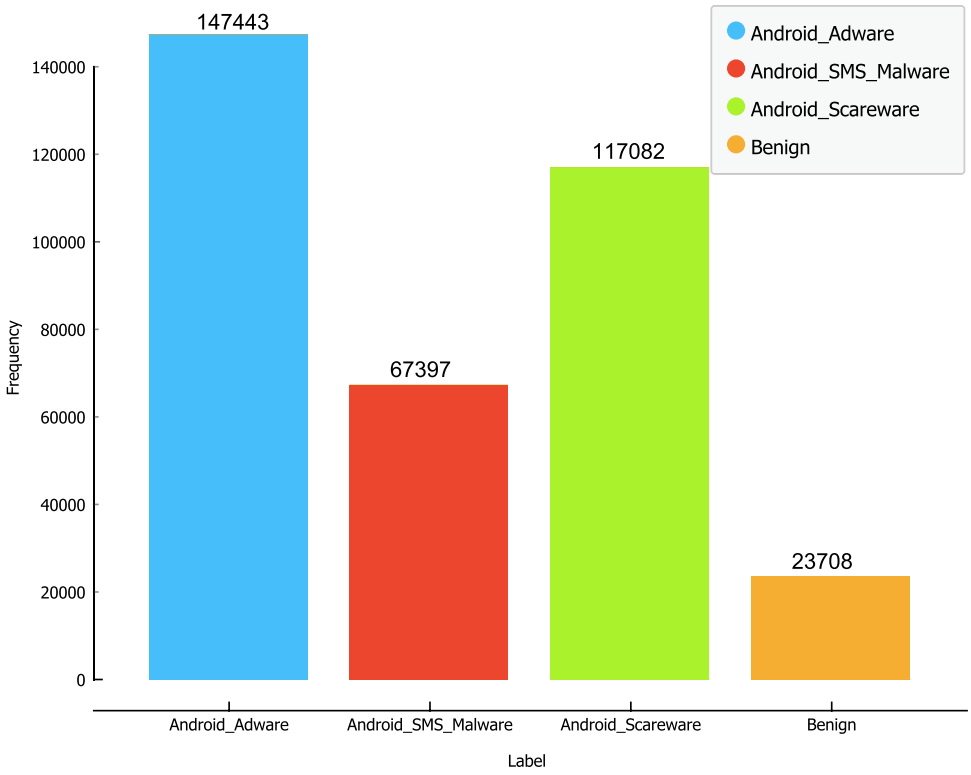


Fig. 3 Class distribution of Android Malware Dataset [29]

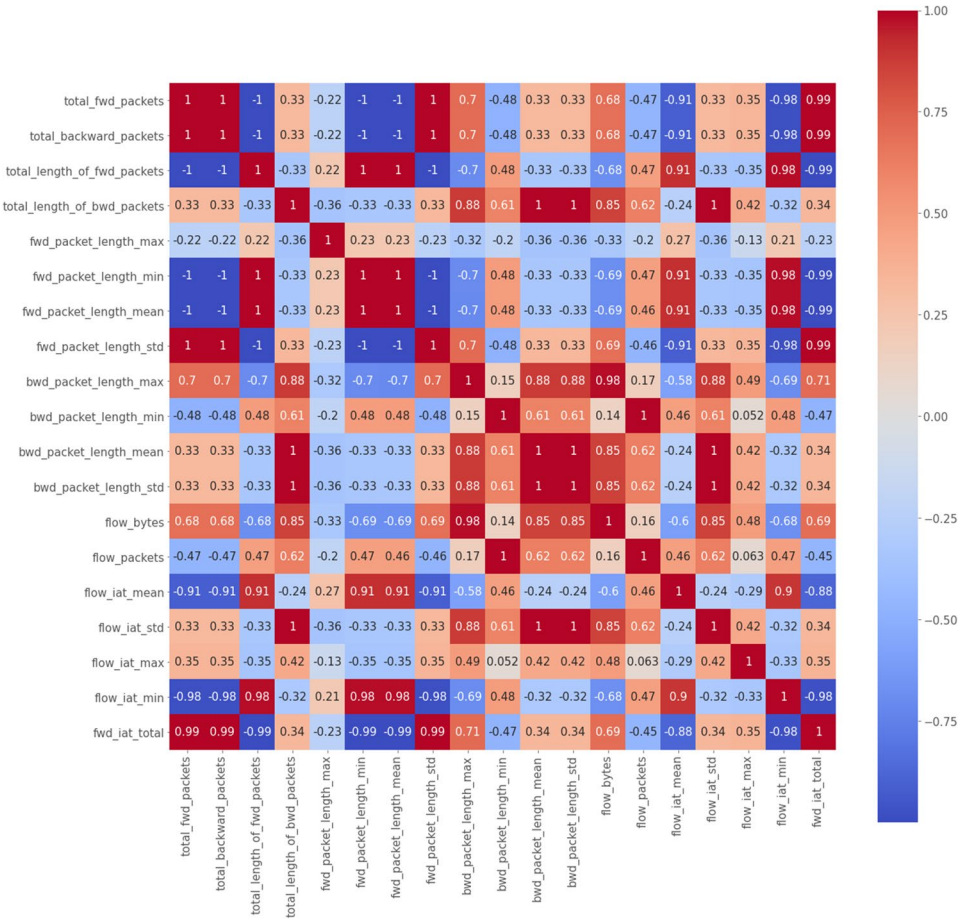


Fig. 4 Heat map of selected feature representing the correlation

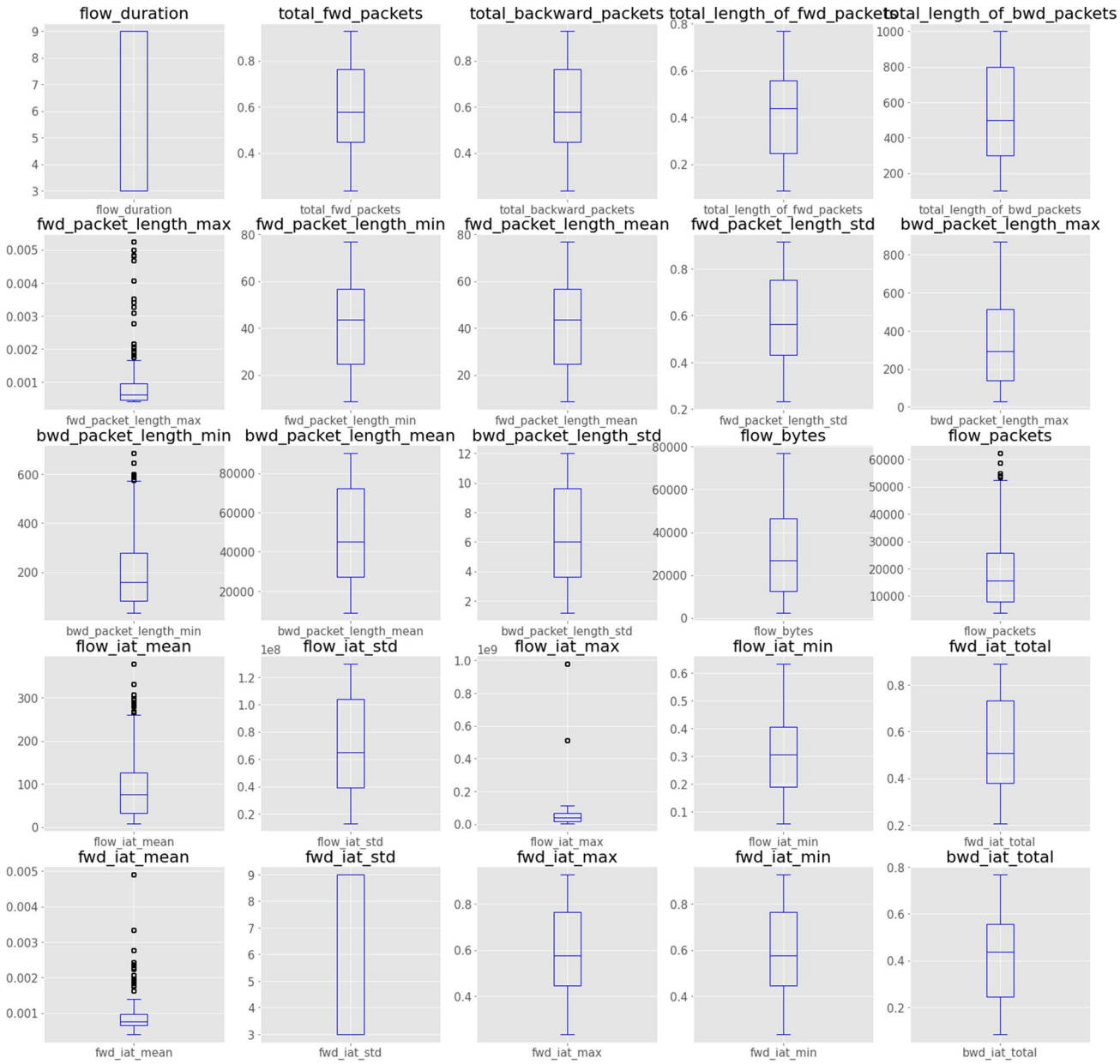


Fig. 5 Boxplot of the features of the dataset for outlier analysis

Our aim is to provide safety to mobile and IoT devices by capturing the malware and deflecting the malicious traffic generated by these malwares.

AdaBoost Classifier AdaBoost is an ML algorithm that is utilized for classification and prediction as well. In our problem statement, the malware has intelligently become part of the benign data, and it is difficult to segregate the malware from the benign traffic.

- AdaBoost uses a chain of algorithms to make the right decisions. It uses minimal memory and the time taken by AdaBoost is also optimal for generating the results.
- Other ML-based approaches rely on one algorithm only, where the outcome is compromised by the single decision maker, but in AdaBoost, multiple algorithms work in parallel to provide accurate classification.
- AdaBoost handles missing values in the supplied input data. It supports parallel execution of decision trees and the aggregation is performed to give accurate results.

Random Forest (RF) Classifier: RF is a supervised ML-based algorithm that also belongs to the class of cascaded algorithms that make use of multiple decision trees to produce an aggregated outcome; in other words, we can say that

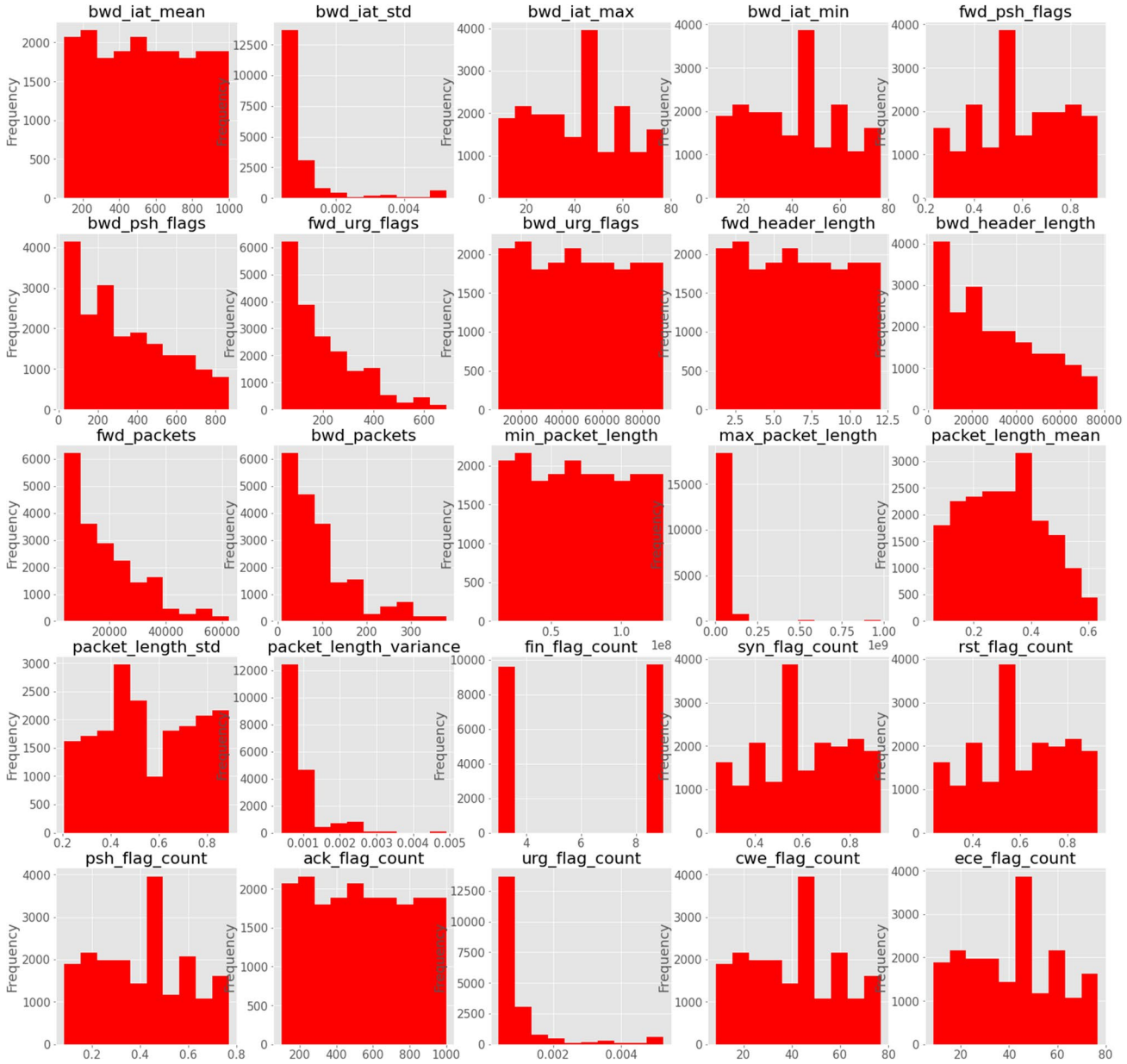


Fig. 6 Histogram of distribution of data features

it can use decision trees which assist in selecting the classes in a step-by-step approach. In this paper, an RF classifier is implemented for a comparative study. RF is known for providing quite accurate results, whether there is a binary or multivariate classification. The cascaded approach allows improving the performance of the RF classifier by taking input from multiple classifiers or trees and generating the output by assembling the inputs.

Steps for an RF Classifier:

- *Step 1:* The data points from the dataset are selected at random.
- *Step 2:* The identified data points are supplied to construct the decision trees.
- *Step 3:* The input for deciding the total decision trees is taken from the user as N.
- *Step 4:* The steps from 1 to 3 are repeated for 2 (N-1) times.

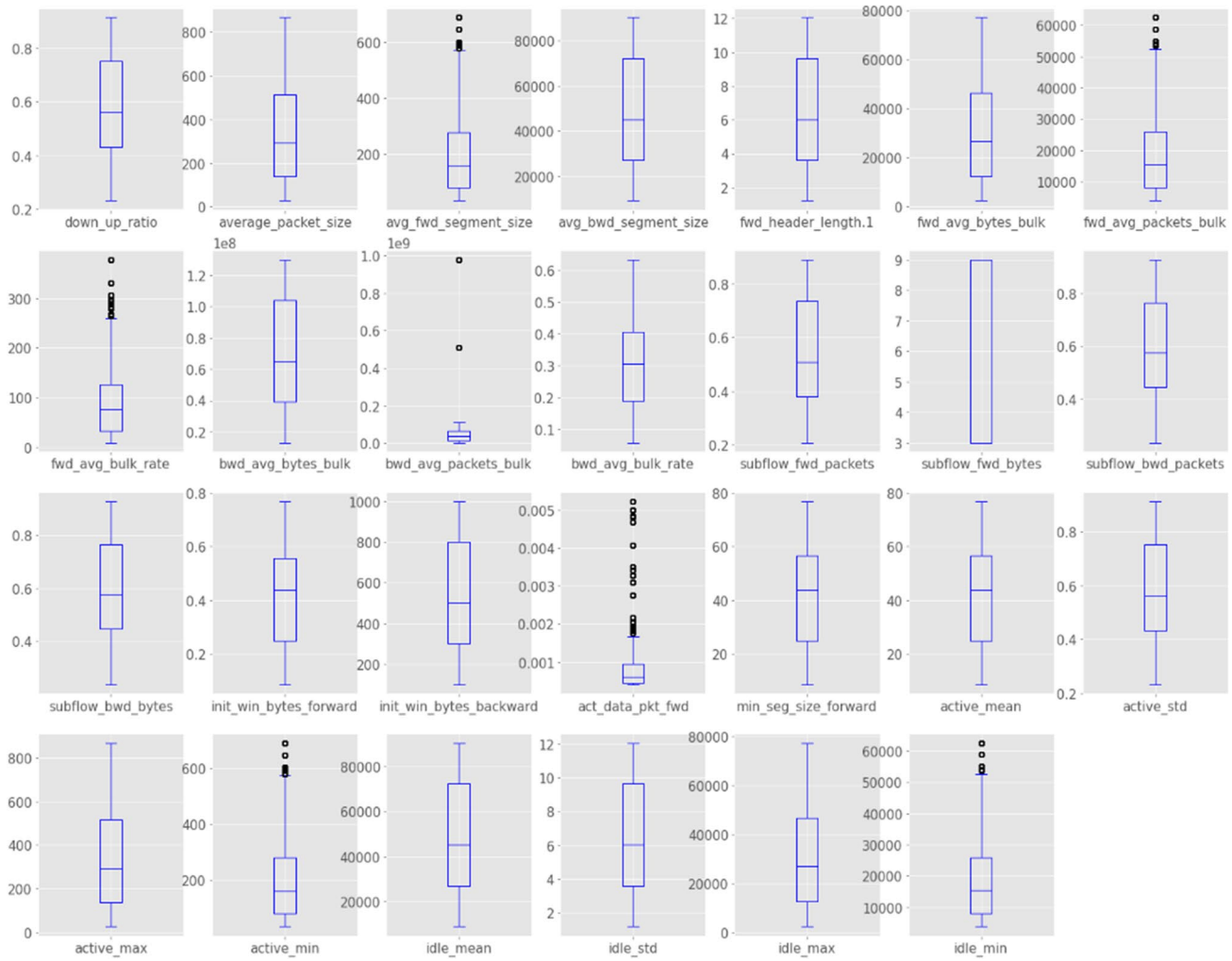


Fig. 7 Outliers of the dataset representing the correlation

- *Step 5:* The output from each tree is recorded and the new data points are selected again.
- *Step 6:* The output of the overall model provides a complete classification.

The advantage of the RF algorithm is that it increases the accuracy of the model and eliminates overfitting of the dataset.

Hybrid approach (Artificial Neural Network (ANN) with SVM): Neural networks (NN) attempt to design a computer model of the human brain. The basic objective of NN is to allow the system to compute the tasks faster as compared to other computational techniques when there is a lot of data. Artificial neural networks (ANN) provide solutions in a faster and more accurate manner where a large number of iterations and large computations are involved. On the other hand, SVM uses nonlinear matching to allow the linear classification of data. When, in a hybrid approach, the features of both the models, ANN and SVM are combined, the system gives accurate classification results in a faster manner. In our proposed model, ANN employs multi-layer connections and various activation functions to deal with nonlinear problems. SVM defines the number of layers needed to design the architecture of the neural networks. Basically, the hybrid model uses the best features of both the classifiers and comes up with a collaborative model that is a combination of ANN and SVM [31] and eventually it leads to promising results [32]. The results of the proposed hybrid model are compared with those of the RF classifier and the AdaBoost classifier by using statistical evaluators, as explained in the next section.

Deep Learning Classifier: A deep learning classifier is a type of machine learning algorithm that utilizes ANNs with many layers to identify patterns in data. Deep learning classifiers are particularly effective for tasks involving image recognition, natural language processing, and speech recognition [33]. In deep learning classifiers, the input data is passed through multiple layers of interconnected nodes, called neurons, in a neural network. Each neuron processes

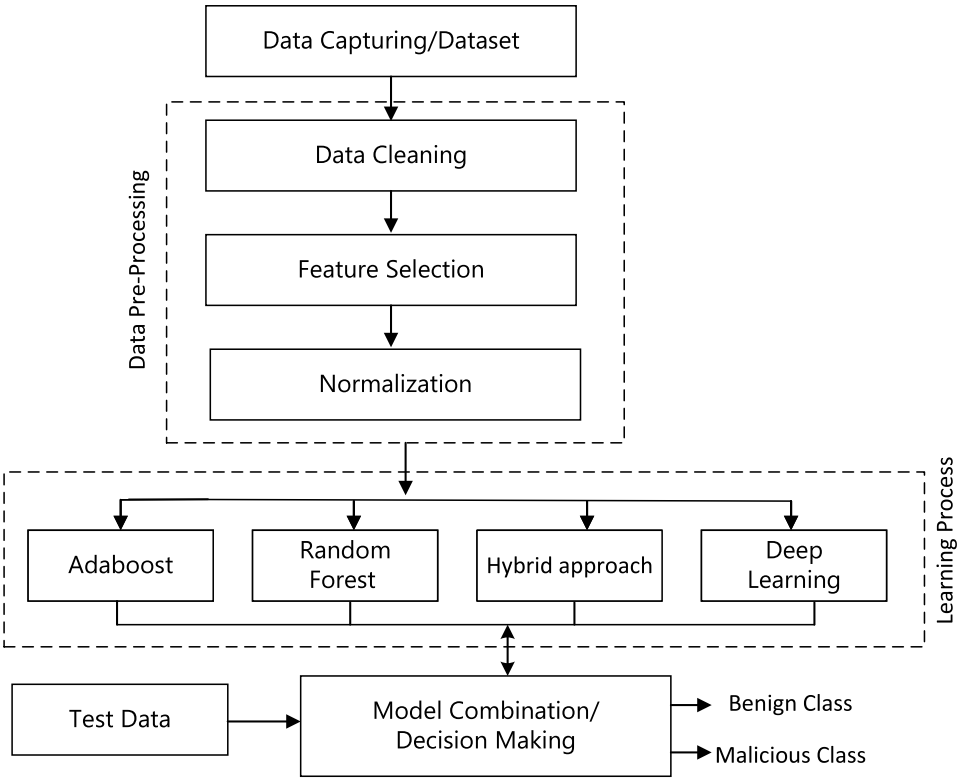


Fig. 8 Architecture diagram of Malware classification

the input data and passes the output to the next layer until the final output layer, which provides the classification result. The training process involves adjusting the weights and biases of the neurons to minimize the difference between the predicted output and the actual output [34].

The main advantage of deep learning classifiers is their ability to automatically learn and extract features from the data without requiring manual feature engineering. This makes them highly effective for complex tasks with large amounts of data. However, deep learning classifiers require a lot of computational resources and training data, and can be prone to overfitting if not properly regularized [35, 36]. Here are the steps for the deep learning classifier algorithm:

- *Step 1:* Load and preprocess the data: This involves reading in the data, splitting it into training and testing sets, and performing any necessary preprocessing, such as normalization or feature scaling.
- *Step 2:* Define the model architecture: This involves selecting the type and number of layers, the activation functions, and other hyperparameters, such as the learning rate and batch size.
- *Step 3:* Compile the model: This involves specifying the loss function, optimizer, and evaluation metric for the model.
- *Step 4:* Train the model: This involves using the fit() method to train the model on the training data. During training, the weights of the model are adjusted to minimize the loss function.
- *Step 5:* Evaluate the model: This involves using the evaluate() method to evaluate the model's performance on the test set. The evaluation metric can be used to compare the performance of different models.

These steps may be repeated multiple times to fine-tune the model and improve its performance. Additionally, other techniques such as regularization, early stopping, or data augmentation may be used to further improve the model's performance. The proposed algorithm for malware classification is given in Algorithm 1.

Algorithm 1. Proposed Algorithm for Malware Classification

Require: Dataset D comprising N instances with M features
Ensure: Predicted Class Label.

procedure PRE-PROCESS_DATA(D) ▷ Data Preprocessing
    Missing_Val_Imputation()
    Remove_Duplicate_Sample()
    Feature_Ranking()
    Standardize()
    Split_Data(X_train, Y_train, X_test, Y_test)
    return D'
end procedure
procedure ADABOOST(D') ▷ Level 1 Learner
    PC_AB=AB_Model(X_train, Y_train, X_Test, Y_Test)

    return $PCLevel1''$
end procedure
procedure RANDOM_FOREST(D') ▷ Level 2 Learner
    PC_RF=RF_Model(X_train, Y_train, X_Test, Y_Test)

    return $PCLevel2''$
end procedure
procedure HYBRID_MODEL(D') ▷ Level 3 Learner
    PC_HM=Hybrid_Model(X_train, Y_train, X_Test, Y_Test)
    return $PCLevel3''$
end procedure
procedure DNN_MODEL(D') ▷ Level 4 Learner
    PC_DNN=DNN_Model(X_train, Y_train, X_Test, Y_Test)
    return $PCLevel4''$
end procedure
procedure MODEL_EVALUATION ▷ Model Combination and Decision
    D'= PRE-PROCESS_DATA(D)
    $PCLevel1''$=ADABOOST(D')
    $PCLevel2''$=HYBRID_MODEL(D'')
    $PCLevel3''$=RANDOM_FOREST(D')
    $PCLevel4''$=DNN_MODEL(D'')
    PC=Decision($PCLevel1''$, $PCLevel2''$,$PCLevel1''$, $PCLevel2''$)
    Plot CONFUSION_MATRIX(PC, AC) ▷ Plot Confusion Matrix
    Plot ROC(PC, AC) ▷ Plot ROC Curve
    Model_Assessment(PC, AC) ▷ Performance Evaluation Matrices
end procedure

Time and Space Complexity Analysis

The time and space complexity of a combined model that includes AdaBoost, Random Forest, Deep Learning, and Hybrid approach using Artificial Neural Network components would depend on several factors, including the specific algorithms used, the size of the dataset, the complexity of the individual models, and the hardware resources available. Let's use mathematical notations to describe the time and space complexities of the combined model. Let

- N be the number of samples in the dataset.
- M be the number of features in the dataset.
- T be the number of iterations (boosting rounds) for AdaBoost.
- K be the number of trees (estimators) in the Random Forest.
- L be the number of layers in the neural networks.

1. **AdaBoost**

- Time Complexity: O(T * N)
- Space Complexity: Generally not very memory-intensive compared to other models.

2. **Random Forest:**

- Time Complexity: O(K * N * M * log(M))
- Space Complexity: Memory requirements depend on the number of trees and their associated data structures.

3. **Deep Learning (Neural Networks with backpropagation):**

- Time Complexity: O(I * N * L), where I is the number of iterations (epochs) through the training data.
- Space Complexity: Memory requirements depend on the network architecture and batch size.

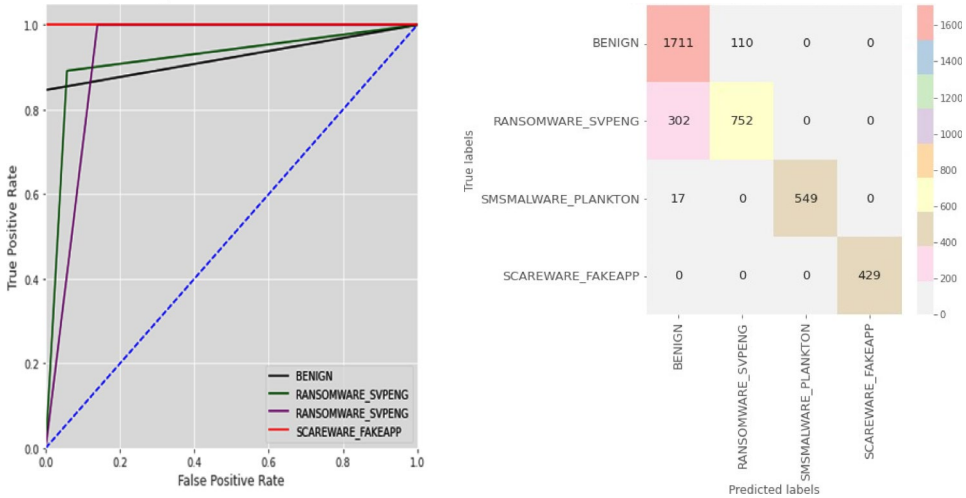
4. **Artificial Neural Network Model (in general):**

- Time Complexity: O(I * N * L), similar to deep learning.
- Space Complexity: Memory requirements depend on the network architecture.

5. **Proposed Model:** Assuming we run these models sequentially (one after another) and denote the combined time and space complexities as follows:

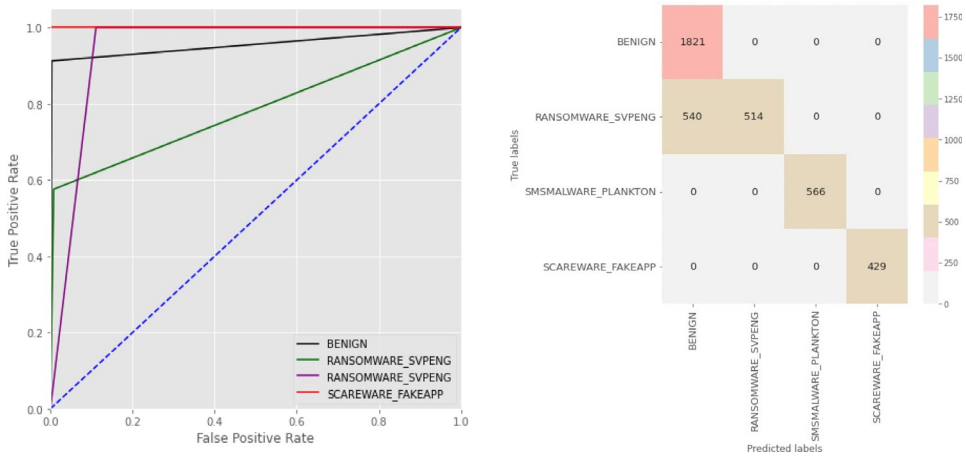
- Combined Time Complexity: O(T * N) + O(K * N * M * log(M)) + O(I * N * L) + O(I * N * L)
- Combined Space Complexity: The space complexity would depend on the storage requirements of all models and associated data structures.

*Note that this is a simplified representation, and in practice, there might be additional factors to consider, such as data preprocessing time, model ensemble overhead, hardware



(a) ROC (b) Confusion Matrix

Fig. 9 **a** ROC curve and **b** confusion matrix using Adaboost Approach



(a) ROC (b) Confusion Matrix

Fig. 10 **a** ROC curve and **b** confusion matrix using Random Forest

parallelism, and library optimizations. The actual time and space complexities could also differ based on implementation details and specific hyperparameters chosen for each model.

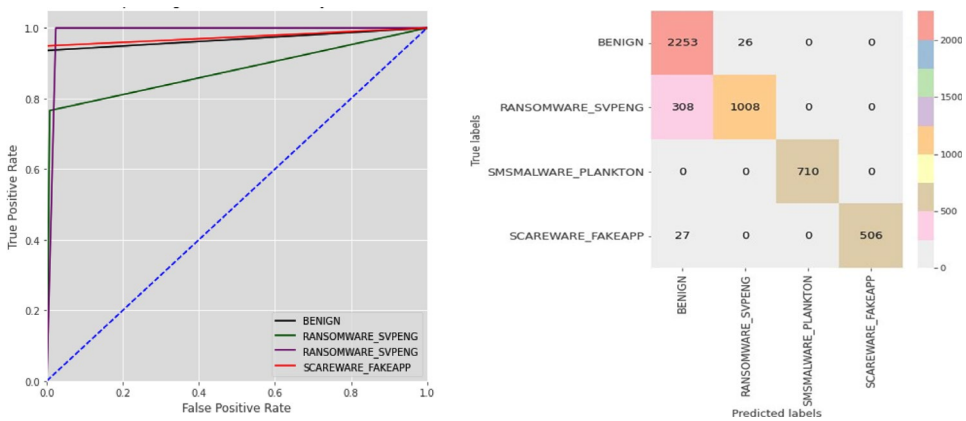
Results and Analysis

The performance of the ML and hybrid model, which is the combination of ML and deep learning approaches, is measured using statistical evaluators such as confusion matrix, F1-score, recall, and precision. The metrics vary from problem to problem because the problem statement decides how an evaluation can be made of the applied techniques to check the viability of the proposed methods.

We have designed three models for the problem statement of classifying mobile data, where two ML approaches are considered: a hybrid approach, which uses a combination of artificial neural network and SVM method, and a pure DL classifier. Now, to get feedback on the performance, we have utilized statistical standard performance metrics that can check the performance on the basis of true positive, true negative, false positive, and false negative classifier values. It is always crucial to analyze the accuracy of the models prior to producing results. The evaluation matrices of AdaBoost as shown in Fig. 9 utilized on the problem statement are given below:

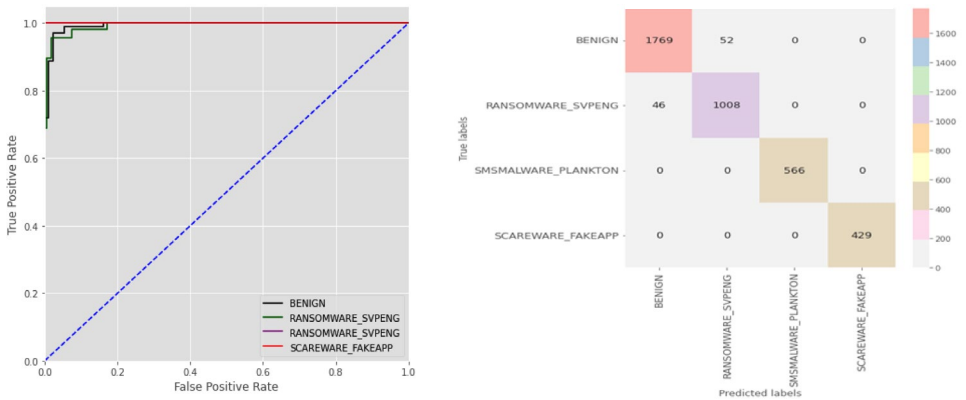
The interpretation of the confusion matrix & RoC curve of Random Forest as shown in Fig. 10 for the testing dataset:

- Out of 429 records of "Scareware_fakeapp"; 429 records are correctly predicted.
- Out of 566 records with the target variable as "Smsmalware_plankton". 566 records are identified correctly.
- Out of 1054 records with the target variable as "Ransomware_svpeng". 514 records are identified correctly, and 540 are misinterpreted.
- Out of a total of 1821 records with the target variable as "Benign". The 1821 records are identified accurately.



(a) ROC (b) Confusion Matrix

Fig. 11 **a** ROC curve and **b** confusion matrix using ANN based Hybrid model



(a) ROC (b) Confusion Matrix

Fig. 12 **a** ROC curve and **b** confusion matrix obtained using Deep Learning Model

The interpretation of the confusion matrix & Roc curve of the hybrid approach as shown in Fig. 11 for testing dataset is:

- Out of 533 records of “Scareware_fakeapp"; 506 records are correctly predicted and 27 records are misclassified.
- Out of 710 records with the target variable as “Smsmalware_plankton". 710 records are identified correctly.
- Out of 1316 records with the target variable as “Ransomware_svpeng". 1008 records are identified correctly and 308 are misinterpreted.
- Out of a total of 2279 records with the target variable as “Benign". 2253 records are identified accurately, and 26 are misinterpreted.

Following are the observations for the above confusion matrix and the ROC curve of the Deep learning classifier shown in Fig. 12:

- In the testing set, there are a total of 429 records with the target variable as “Scareware_fakeapp". 429 records are correctly predicted.

Table 4 Training and testing accuracy score of all models

Classifier	Training accuracy	Testing accuracy	Precision score	Recall score	F1 score
AdaBoost Classifier	83.27%	83.20%	0.90	0.89	0.89
Random Forest Classifier	80.31%	81.09%	0.88	0.93	0.89
Hybrid Classifier	89.66%	90.08%	0.91	0.98	0.94
Deep Learning Classifier	97.47%	97.14%	0.97	0.97	0.97

Table 5 Comparison of existing techniques with the proposed approach

Author	Existing techniques	Training accuracy	Testing accuracy
Jeon & Moon [15]	Convolutional Recurrent Neural Network Using Opcode Sequences from the executable file	96 %	–
Souri & Hosseini [18]	signature- and behavior-based data mining techniques	95.00 %	–
Kumar et al. [20]	Neural network approach	96.00 %	–
Raghavan et al. [21]	Hidden Markov models	91.40 %	–
Proposed Model	AdaBoost Classifier	83.27%	83.20%
	Random Forest Classifier	80.31%	81.09%
	Hybrid Classifier	89.66%	90.08%
	Deep Learning Classifier	97.47%	97.14%

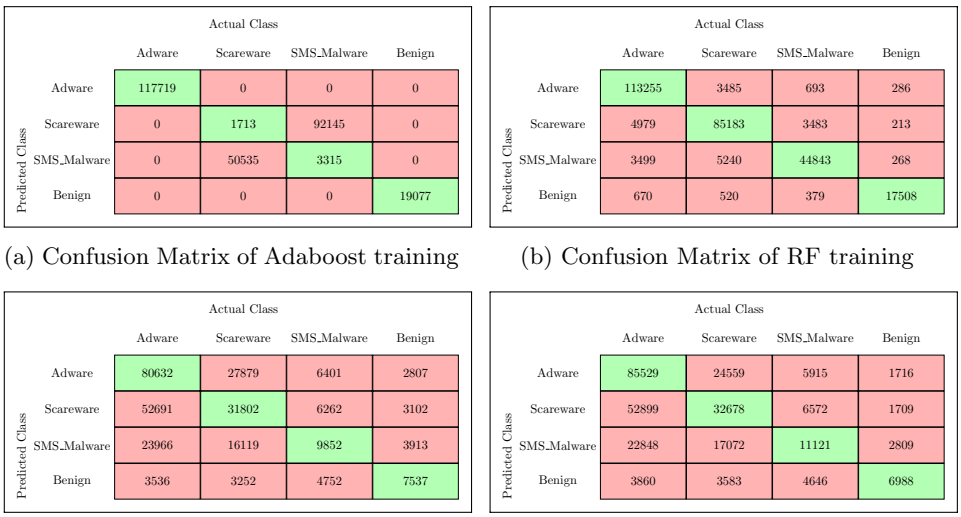


Fig. 13 Confusion matrix during Training process

(a) Confusion Matrix of Adaboost training

(b) Confusion Matrix of RF training

(c) Confusion Matrix of ANN training

(d) Confusion Matrix of DL training

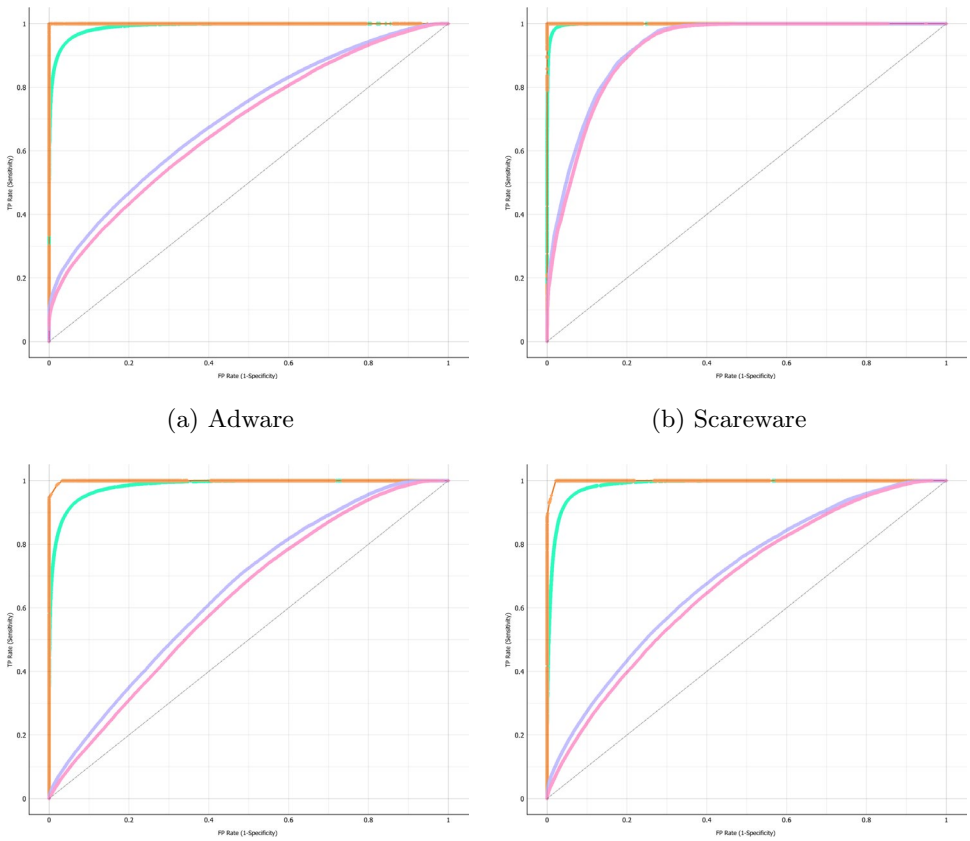
- In the testing set, there are a total of 566 records with the target variable as “Smsmalware_plankton". 566 records are correctly predicted .
- In the testing set, there are a total of 1054 records with the target variable as “Ransomware_svpeng". 1008 records are correctly predicted, and 46 are misclassified.
- In the testing set, there are a total of 1821 records with the target variable as “Benign". 1769 records are correctly predicted, and 52 records are misclassified.

It can be inferred from the results shown in Table 4, that the DL-based model performs the best with respect to the performance evaluation parameters. The next best performance is shown by an ANN-based hybrid model, followed by AdaBoost and then Random Forest. All the machine- and DL-based algorithms are able to classify malware and normal traffic with good accuracy, but our proposed model outperforms the other ML approaches.

The comparison of the proposed approach with existing malware detection approaches is shown in Table 5, The proposed approach is compared with other approaches using the Kaggle dataset. The earlier approaches that are implemented only on this dataset are considered for comparison, and the proposed approach performs well in the classification of malware.

Result Analysis using Malware Android Dataset [29]

The proposed model is also trained, validated and tested with Malware Android dataset [29] by partitioning the data



(a) Adware (b) Scareware

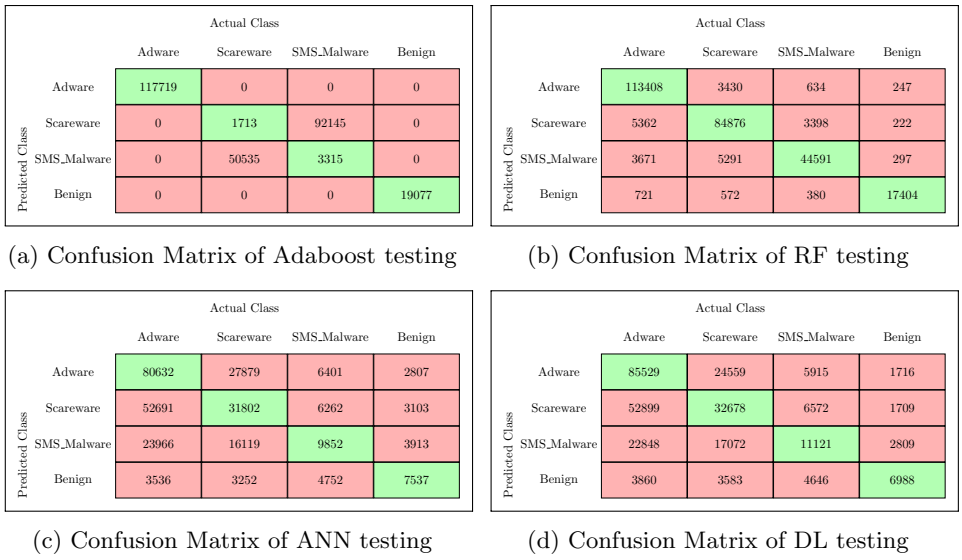
(c) SMS Malware (d) Benign

Fig. 14 Training ROC curve for **a** Adware, **b** Scareware, **c** SMS Malware and **d** Benign classes using Adaboost, Random forest, ANN & DL methods respectively

Table 6 Training Accuracy score of all models on Dataset 2

Classifier	AUC	CA	Precision score	Recall score	F1 score	MCC
AdaBoost classifier	1.0000	0.9823	0.9823	0.9823	0.9822	0.9740
Random forest classifier	0.9865	0.9166	0.9165	0.9166	0.9160	0.8769
Hybrid classifier	0.5032	0.4563	0.4376	0.4563	0.4346	0.1713
Deep learning classifier	0.5388	.4791	0.4626	0.4791	0.4560	0.2025

Fig. 15 Confusion matrix during Testing Process



Actual Class

Predicted Class	Adware	Scareware	SMS_Malware	Benign
Adware	117719	0	0	0
Scareware	0	1713	92145	0
SMS_Malware	0	50535	3315	0
Benign	0	0	0	19077

(a) Confusion Matrix of Adaboost testing

Actual Class

Predicted Class	Adware	Scareware	SMS_Malware	Benign
Adware	113408	3430	634	247
Scareware	5362	84876	3398	222
SMS_Malware	3671	5291	44591	297
Benign	721	572	380	17404

(b) Confusion Matrix of RF testing

Actual Class

Predicted Class	Adware	Scareware	SMS_Malware	Benign
Adware	80632	27879	6401	2807
Scareware	52691	31802	6262	3103
SMS_Malware	23966	16119	9852	3913
Benign	3536	3252	4752	7537

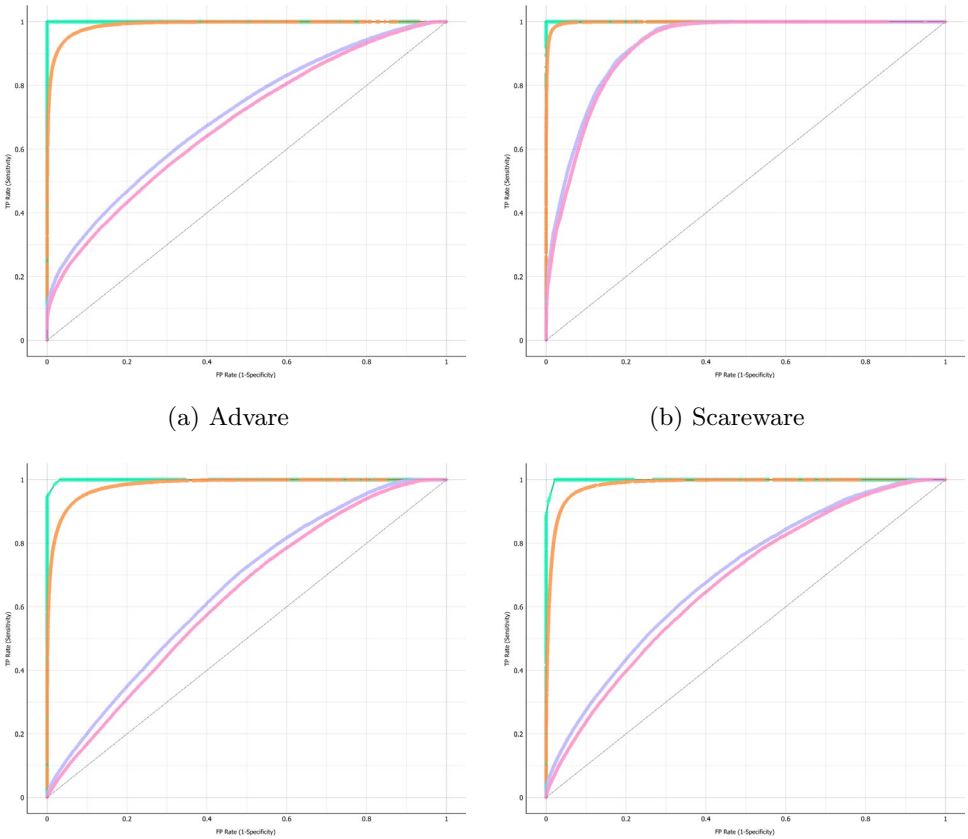
(c) Confusion Matrix of ANN testing

Actual Class

Predicted Class	Adware	Scareware	SMS_Malware	Benign
Adware	85529	24559	5915	1716
Scareware	52899	32678	6572	1709
SMS_Malware	22848	17072	11121	2809
Benign	3860	3583	4646	6988

(d) Confusion Matrix of DL testing

Fig. 16 Testing ROC curve for **a** Adware, **b** Scareware, **c** SMS Malware and **d** Benign classes using Adaboost, Random forest, ANN & DL methods respectively



(a) Advare

(b) Scareware

(c) SMS Malware

(d) Benign

into an 80:20 ratio. The results obtained during the training & testing process are shown in Tables 6 and 7. The detection accuracy using Adaboost and Random Forest is satisfactory whereas using ANN and Deep Learning methods is not acceptable. But, using the proposed ensemble approach the detection accuracy is drastically improved. The Confusion matrix for training & testing results of different methods is shown in Figs. 13 and 15 respectively. The combined ROC

Table 7 Testing Accuracy score of all models on Dataset [29]

Classifier	AUC	CA	Precision score	Recall score	F1 score	MCC
AdaBoost classifier	1	0.982	0.982	0.982	0.982	0.974
Random forest classifier	.987	0.915	0.915	0.915	0.914	0.874
Hybrid classifier	0.503	0.456	0.438	0.456	0.435	0.171
Deep learning classifier	0.539	.479	0.463	0.479	0.456	0.203

curves for training & testing of different approaches are shown in Figs. 14 and 16.

Conclusion

Effective classification of mobile data can certainly assist in speeding up the transmission rate of data and deflecting the unwanted data inserted by intruders on the mobile channels to increase the false traffic on the mobile channels and slow down the transmission speed of the data. To handle the problem of correct identification of the data and to separate the unwanted traffic, newer and smarter mechanisms are proposed in this study. The classification permits mobile data to cross the internet gateways without any hurdles and deflects unwanted or malicious data at the earliest possible time to prevent bottle-necking on the mobile channels. The classification is made in this experiment by using three different approaches to form one hybrid technique to achieve the research objective. The performance evaluators exhibit the results by using statistical parameters, and it is observed that the proposed hybrid approach outperforms the ML techniques with respect to accuracy in classification, sensitivity, and specificity scores. This study involves the utilization of two Android malware datasets to enhance the model's training by incorporating a greater number of malicious instances.

In the future, we will create a testbed mobile network using multiple devices and capture the real network traffic. We will process and classify the real-time network traffic using new approach and can also use the real-time data set of recent malware attacks. The performance can be tested & evaluated using different classification techniques of Deep learning to optimize the performance of the system.

Author contributions All authors contributed equally to this work.

Funding Not applicable.

Data availability We have used two datasets for android malware detection. The first one is available from the corresponding author on reasonable request, and the second one is available from Kaggle, and the link for it is given in Reference No. [29].

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This is to acknowledge that, this manuscript is original, has not been published before and is not currently being considered for publication elsewhere.

References

1. Liu X, Lin Y, Li H, Zhang J. A novel method for malware detection on ml-based visualization technique. Comp Secur. 2020;89:101682.
2. Agarkar S, Ghosh S. Malware detection & classification using machine learning. In: IEEE International Symposium on Sustainable Energy. Signal Processing and Cyber Security (iSSSC). 2020;2020:1–6.
3. Priyadarshan P, Sarangi P, Rath A, Panda G. Machine learning based improved malware detection schemes. In: 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2021; pp. 925–931.
4. Li S, Zhou Q, Zhou R, Lv Q. Intelligent malware detection based on graph convolutional network. J Supercomput. 2022;78(3):4182–98.
5. Huang X, Ma L, Yang W, Zhong Y. A method for windows malware detection based on deep learning. J Signal Process Syst. 2021;93:265–73.
6. Kumar AA, Anoosh G, Abhishek M, Shraddha C. An effective machine learning-based file malware detection-a survey. In: Proceedings of International Conference on Communication, Computing and Electronics Systems (ICCCES 2019), 2020; pp. 355–360.
7. Mahindru A, Sangal A. Mldroid-framework for android malware detection using machine learning techniques. Neural Comput Appl. 2021;33(10):5183–240.
8. Ünver HM, Bakour K. Android malware detection based on image-based features and machine learning techniques. SN Appl Sci. 2020;2:1–15.
9. Meijin L, Zhiyang F, Junfeng W, Luyu C, Qi Z, Tao Y, Yinwei W, Jiaxuan G. A systematic overview of android malware detection. Appl Artif Intell. 2022;36(1):2007327.
10. Mareschal B, Kaur M, Kharat V, Sakhare SS. Convergence of smart technologies for digital transformation. Tehnički glasnik 2021;15(1):II–IV.
11. Baghirov E. Techniques of malware detection: Research review. In: 2021 IEEE 15th International Conference on Application of Information and Communication Technologies (AICT), 2021; pp. 1–6.

12. Wen L, Yu H. An android malware detection system based on machine learning. AIP Confer Proc. 2017;1864(1):020136.
13. Sahu SK, Mohapatra DP, Rout JK, Sahoo KS, Luhach AK. An ensemble-based scalable approach for intrusion detection using big data framework. Big Data. 2021;9(4):303–21.
14. Jahromi AN, Hashemi S, Dehghantanha A, Choo KKR, Karimipour H, Newton DE, Parizi RM. An improved two-hidden-layer extreme learning machine for malware hunting. Comput Secur. 2020;89:101655.
15. Jeon S, Moon J. Malware-detection method with a convolutional recurrent neural network using opcode sequences. Inf Sci. 2020;535:1–15.
16. Agrawal P, Trivedi B. Machine learning classifiers for android malware detection. Data Manage Anal Innov Proc ICDMAI. 2020;1(2021):311–22.
17. Wadkar M, Troia FD, Stamp M. Detecting malware evolution using support vector machines. Expert Syst Appl. 2020;143:113022.
18. Souri A, Hosseini R. A state-of-the-art survey of malware detection approaches using data mining techniques. Human-centric Comput Inform Sci. 2018;8(1):1–22.
19. Damodaran A, Troia FD, Visaggio CA, Austin TH, Stamp M. A comparison of static, dynamic, and hybrid analysis for malware detection. J Comp Virol Hack Tech. 2017;13:1–12.
20. Kumar MVR, Kumar A, Bando A, Shah H, Reddy SC. A survey of deep learning techniques for malware analysis. Int J Adv Sci Technol. 2020;29(4):6031–42.
21. Raghavan A, Troia FD, Stamp M. Hidden markov models with random restarts versus boosting for malware detection. J Comput Virol Hack Tech. 2019;15:97–107.
22. Chandak AV, Ray NK. IOT data classifications for smart home deployment. SN Comput Sci. 2022;3(1):95.
23. Darabian H, Dehghantanha A, Hashemi S, Homayoun S, Choo KKR. An opcode-based technique for polymorphic internet of things malware detection. Concurr Comput Pract Exp. 2020;32(6):e5173.
24. Bakour K, Ünver HM, Ghanem R. The android malware detection systems between hope and reality. SN Appl Sci. 2019;1:1–42.
25. Abbadi MA, Al-Bustanji AM, Al-kasassbeh M. Robust intelligent malware detection using lightgbm algorithm. Int J Innov Technol Explo Eng. 2020;9(6):1253–60.
26. Meijin L, Zhiyang F, Junfeng W, Luyu C, Qi Z, Tao Y, Yinwei W, Jiaxuan G. A systematic overview of android malware detection. Appl Artif Intell. 2022;36(1):2007327. https://doi.org/10.1080/08839514.2021.2007327.
27. Ren Z, Chen G, Lu W. Malware visualization methods based on deep convolution neural networks. Multimedia Tools Appl 2020;79;10-975–10-993.
28. Surendran R, Thomas T, Emmanuel S. A tan based hybrid model for android malware detection. J Inform Secur Appl. 2020;54:102483.
29. Subhadeep C. Android malware detection. https://www.kaggle.com/dsv/4987461. 2023. Accessed 15 July 2023.
30. Aceto G, Ciuonzo D, Montieri A, Pescapé A. Multi-classification approaches for classifying mobile app traffic. J Netw Comput Appl. 2018;103:131–45.
31. Baldwin J, Dehghantanha A. Leveraging support vector machine for opcode density based detection of crypto-ransomware. Cyber Threat Intell 2018;107–136.
32. Pandey MK, Singh MK, Pal S, Tiwari BB. Prediction of phishing websites using stacked ensemble method and hybrid features selection method. SN Comp Sci. 2022;3(6):488.
33. Vinayakumar R, Soman K, Poornachandran P, Sachin Kumar S. Detecting android malware using long short-term memory (lstm). J Intell Fuzzy Syst. 2018;34(3):1277–88.
34. Akhtar N, Mian A. Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access 2018;6:14-410–14-430.
35. Alzaylaee MK, Yerima SY, Sezer S. Dl-droid: Deep learning based android malware detection using real devices. Comp Secur. 2020;89:101663.
36. Sahu SK, Mohapatra DP, Rout JK, Sahoo KS, Pham QV, Dao NN. A LSTM-FCNN based multi-class intrusion detection using scalable framework. Compu Electr Eng. 2022;99:107720.