Journal of Cyber Security
DOI: 10.32604/jcs.2022.031889

Tech Science Press

Article

check for updates

An Adaptive-Feature Centric XGBoost Ensemble Classifier Model for Improved Malware Detection and Classification

J. Pavithra* **and S. Selvakumarasamy**

SRM Institute of Science and Technology, Kattankulathur, 603203, India
*Corresponding Author: J. Pavithra. Email: pj1089@srmist.edu.in

Received: 01 September 2022; Accepted: 02 September 2022

Abstract: Machine learning (ML) is often used to solve the problem of malware detection and classification, and various machine learning approaches are adapted to the problem of malware classification; still acquiring poor performance by the way of feature selection, and classification. To address the problem, an efficient novel algorithm for adaptive feature-centered XG Boost Ensemble Learner Classifier "AFC-XG Boost" is presented in this paper. The proposed model has been designed to handle varying data sets of malware detection obtained from Kaggle data set. The model turns the XG Boost classifier in several stages to optimize performance. At preprocessing stage, the data set given has been noise removed, normalized and tamper removed using Feature Base Optimizer "FBO" algorithm. The FBO would normalize the data points, as well as perform noise removal according to the feature values and their base information. Similarly, the performance of standard XG Boost has been optimized by adapting the selection using Class Based Principle Component Analysis "CBPCA" algorithm, which performs the selection according to the fitness of any feature for different classes. Based on the selected features, the method generates a regression tree for each feature considered. Based on the generated trees, the method performs classification by computing the tree-level ensemble similarity 'TLES' and the class-level ensemble similarity 'CLES'. Using both methods calculates the value of the class match similarity 'CMS' based on which the malware has been classified. The proposed approach achieves 97% accuracy in malware detection and classification with the less time complexity of 34 s for 75000 samples.

Keywords: Malware detection; machine learning; XGBoost; PCA; ensemble learner; CBPCA; CMS; AFC-XGBoost

1 Introduction

The recent development in the IT sector has encouraged society to depend on that for various purposes. They even perform their most daily activities with support of that. The most organizations depend on that to perform their processes and encourage their organizations to work over internet even though the units of their organization located distributed. This encourages users in accessing various services through the Web and approaches various web sites. Even they access different web services provided by different solutions or organizations. The issue rise here when the user access a service which is not part of their organization or when you access the service from external world or when you provide access to a third party.

On the other hand, any network which provides services to users faces variety of security threats. The threat may be generated not just by an external member, but also by the internal trusted user. So, the service provider has the responsibility of monitoring such requests from malicious user and stops them from accessing valuable resources. Malwares are the most dangerous software's which are spread by different malicious organization through various forms being send to the receiver end while accessing their services. The purpose of the malwares are not just to damage your process, but also to access your various files and data or computing resources in the sense of malfunctioning the system being used. Also they can intrude to the operating system software and could control your entire system. So it is necessary to monitor such malwares of any form and classify them related to malignant and benign classes.

The presence of any intrusion attack can be identified in several ways. There are several approaches available such as rule based approaches, which maintains set of rules in detecting the presence of any intrusion attack. In recent times, machine learning algorithms are used in different scientific problems. In this way, they can be used in malware classification. There are a number of machine learning approaches available like genetic algorithm, support vector machine, fuzzy rules, neural network, decision tree,, regression tree and so on. Snort is the popular tool, which is an IDS works based on various rule sets available. Similarly, genetic algorithm, k means classifier, fuzzy classifier, principle component analysis are most popular algorithms being used towards detecting the malwares [1]. In this way, the random forest approach, which is a bagging-based approach used for the classification of malwares and decision tree, has been used in the same. However, the performance of the methods is not up to the expected level. They suffer with higher MSE (mean square error) because of the selection of feature and how the loss functions have been designed. In case of random forest algorithms, the classification is performed in parallel with all the trees as the methods maintains number of trees and finally makes an average to get the decisions. This really affects the performance of classification by increasing the mean square error value. But for a classification problem, it is necessary to reduce the error rate and loss functions. Also, they generate trees with large siblings and the number of levels is also higher, which increases the time complexity as well. So, the design of tree must be effectively performed to produce better results.

By considering all this, an efficient adaptive feature centric XGBoost Ensemble Classifier (AFC-XGBoost) is presented in this article. The reason here is, the classifier should produce less time complexity and the error rate must be reduced. Unlike other algorithms, the boost algorithm generates number of trees with less hierarchy and the classification will be done in a rapid way. Also, it performs sequential generation so that the error rate has been sequentially reduced at each function. However, the proposed method uses the beauty of XGBoost algorithm by modifying the kernel functions in measuring the class weights. The ensemble learner approach has been modified to compute various measures to support the performance development. The proposed approach would measure the similarity of ensembles in feature, class levels to improve the performance. The detailed approach is presented in the next section.

2 Related Works

There exist several approaches towards the classification of malware with machine learning algorithms. This section details a set of methods related to the problem.

A behavioral-based machine learning approach is presented towards malware detection is presented in [2], which handles the malwares by maintaining a number of behavior patterns which are trained with machine learning algorithms in recognizing social malwares. Similarly, in detecting the family of malwares and classifying them, a hybrid approach is presented in [3], which combines support vector machine (SVM) and active learning by learning (ALBL) to handle unlabeled data to support malware detection.

A genetic algorithm-based malware detection scheme for android devices is presented in [4], which applies genetic algorithm in feature selection and uses machine learning classifiers in detecting malwares. The problem of malware detection and classification is approached with a machine learning algorithm in [5], which uses the Cuckoo sandbox to analyze the effect of malware in isolated environment. The method extracts various features from the reports, and subsets of features are selected to maximize the accuracy.

A semantic behavior based recognition scheme is approached with a deep learning method in [6], which finds the spatial correlations with NLP tools and finds the semantic behaviors in classifying the malwares. An Opcode-based Android malware analysis approach is discussed in [7], which clubs different classifiers of machine learning in classifying the malwares. Similarly, a pixel based feature for malware classification is presented in [8], which uses pixel level features in identifying the family of malware. The model has been tested with different approaches like KNN, SVM, NB, Decision Tree, and Random Forest.

A machine learning technique is presented towards optimizing the feature to be used in analyzing the malwares in windows [9]. The method uses genetic algorithm in feature selection where the behavior features are used in classification with Cuckoo search. The system calls and operational codes in byte codes are used in analyzing malware with machine learning in [10]. The presence of Botnet framed by a set of IoT devices has been classified with a novel Scikit machine learning approach in [11], which groups the data using Weka and by applying the Scikit approach, the classification is performed. A behavioral frequency-based malware detection model is presented in [12], which finds malware according to the call invoked on system function calls. By monitoring the frequency of system calls, the machine learning model classifies the malwares and genuine tools. An Electromagnetic Emission Based Malware Analysis model is presented in [13], which uses Discrete Wavelet Transform (DWT) in extracting the features from spectrograms traces. Extracted features are used in generating fine grained patterns to identify the malware family.

A probability based classification model is presented in [14], which analyze the genuine of tools by computing probability and based on the threshold. By computing Malscore for static and dynamic analysis, the method performs classification. A permission-based machine learning model Significant Permission Identification (SigPID) for malware detection in android devices is presented in [15], which analyzes the usage of tool according to the permission given. The method works on three levels over permission data in classifying the tools. A domain generation-based malware classification algorithm is presented in [16], which classifies domains and cluster them to find such malicious domains. The method uses Hidden Markov model in predicting the features which are coming in to perform classification.

A malware defensive model for IoT environment is presented in [17], which uses two different approaches to select adversarial samples and by computing distance from cluster centers and probability values, the method performs classification. To identify the abuse in crypto mining generated by malware, an efficient approach is presented in [18]. The method finds a set of network flow which are relevant and able to classify a set of flow generated by crypto mining. The method works according to the network flow and finds malicious flow generated.

A Trend micro locality Hashing (TMLH) based approach is presented in [19], to support cloud environment. The method uses a cuckoo sandbox in analyzing the reports of tools in an isolated environment. Essential features are selected using principle component analysis and uses different classification algorithms. A malware detection scheme based on machine learning and visualization is presented in [20], which generates gray scale images and generates GIST descriptor from the images to perform classification with machine learning algorithms. Three different classification algorithms are used to classify malwares.

Presence of botnet has been detected with a multilayer framework in [21], which analyze the behavior of nodes in [22] network to find the botnet command and find the controller using a filtering technique. Similarly, the presence of cyber-attack in an IoT environment is handled with a machine learning model and feature engineering. The method uses the UNSW-NB15 data set to classify cyber attacks. Also, K-Medoid sampling and scatter search-based feature engineering techniques are used to obtain a representative dataset with optimal feature subsets.

A Cryptomining Detection based on machine learning in cloud is presented in [23], which monitor the system call of Linux kernel and detect the presence of pod using different machine learning algorithms. A Sequencing based ransom ware detection model is presented in [24], where DNA act-Ran uses machine learning algorithm in searching the specific sequence and uses frequency vectors in classifying the tool. The classification of malware and family has been approached with a prototype-based machine learning algorithm in [25], which extracts low-dimensional features by computing histogram entropy and uses the prototype selection algorithm with hyper rectangles. The method splits the input into different sub spaces according to hyper-rectangles and uses cover optimization algorithm is employed to find a small number of prototypes. The same has been used to perform classification.

The methods discussed above suffer to achieve higher performance in malware classification. This analysis and evaluation motivates the design of a novel efficient malware classification model to improve classification performance.

3 Adaptive Feature Centric XGBoost Ensemble Malware Classifier Model

The proposed model reads the data set and applies preprocessing using FBO-Preprocessor which removes noise records and normalizes the features by computing feature mean and other factors. Furthermore, the preprocessed data set has been applied with feature selection using CBPCA which computes class-based fitness score for each feature and selects a set of features using the histogram value of various features. In addition, the method generates the trees and is used for classification. At the test phase, the method applies preprocessing and extracts the features and computes different measures like Tree level ensemble similarity, Class level ensemble similarity to compute class match measure. Based on the value of the class match measure, the method performs malware classification. A detailed approach is presented in this section.

The functional architecture of proposed model has been pictorially represented in Fig. 1, where the functional components such as are discussed briefly in forthcoming section.

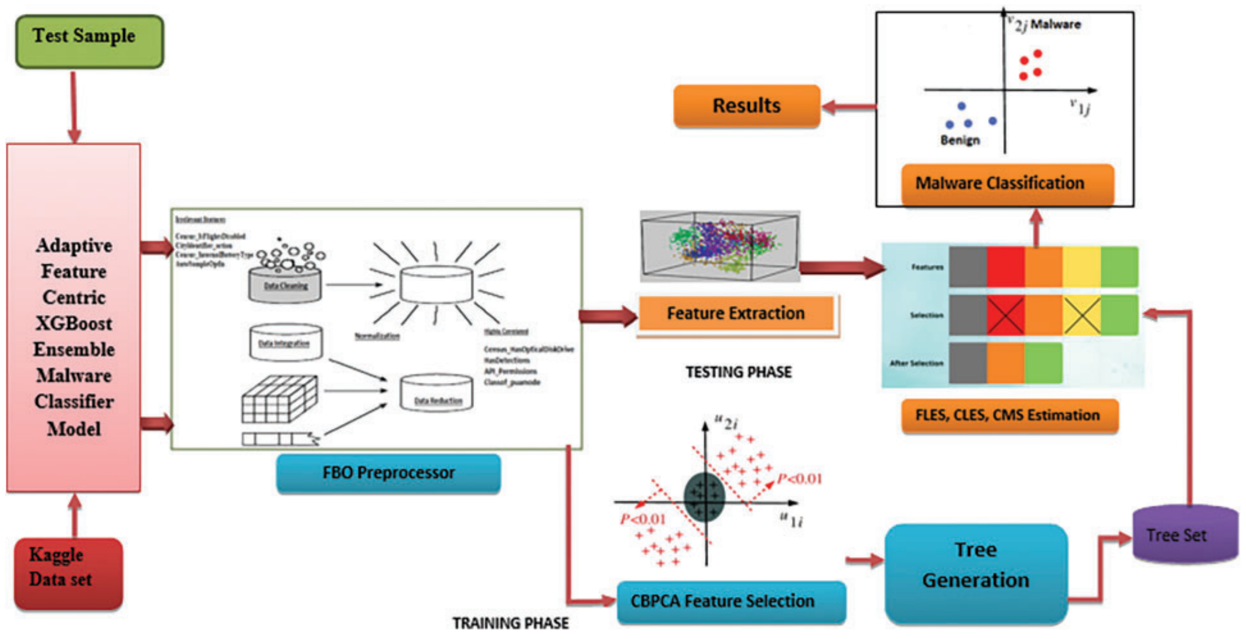


Figure 1: Architecture of proposed model

3.1 FBO Preprocessor

The data set considered would have number of features. The Kaggle data set obtained from open-source platform contains 52 features in total which contains data, platform, software, operating system and etc. The data set has a number of features which also include incomplete records. The feature base Optimizer algorithm works on the data set to not just eliminate the noisy records but also improve the quality of the data set. To do this, the features present in the data set have been initially identified. Further, each record has been traced and identified for the presence of all features. The features have been classified into three classes like numeric, binary, and alphanumeric. With this, the method computes class based mean value (CBMV) for the features of numeric class which is being measured by counting the class samples. Similarly, for the binary class features, the method computes the frequency measures on each class. For example, for a feature with binary value class, the method identifies a list of samples with the value 1 and a list of samples with the value 0. Using both of them, and with the total samples of the class, the method computes the value of feature frequency measure (FFM). Finally, an FFM with maximum class is identified and assigned. For categorical feature like name, year, platform of the malware, the method computes multi-functional relative score (MFRS) which is being measured according to the distance with the numeric features of various classes. The classes with maximum MFRS are selected and the feature value with more frequency has been identified and assigned. This improves the quality of the data to be used in classification.

FBO Algorithm:

```
Given: Data set Kds
Obtain: Preprocessed data set Pkds
Start
        Read data set kds.
```

(Continued)

FBO Algorithm: Continued

```
        Find the feature list Fel = $Fel \cup \left(\overset{size(Kds)}{\underset{i=1}{(Features \in Kds(i))}} \rightarrow (Feature \ni Fel)\right)$
        Find class of traces Tcs = $Tcs \cup \left(\sum_{i=1}^{size(Kds)} kds(i).Class \ni Tcs\right)$ **#80 features has been listed**
        For each feature f
                Categorical set CS = $\sum_{i=1}^{size(Tcs)} (Tcs(i).f == Category) \ni Cs$
                If Numeric then
                        For each class c of Trace
                                Trace set Ts = $Ts \cup \left(\sum_{i=1}^{size(Kds)} kds(i).class == c\right)$
                                Compute Class-Based Mean Value CBMV.
                                CBMV = $\frac{\sum_{i=1}^{size(Ts)} Ts(f).value}{size(Ts)}$
                                Add to the Fms mean set Fms.
                        End
                Else-if Binary then
                        Compute Feature Frequency Value FFV.
                FFV = $Max\left(Count\left(\sum_{i=1}^{size(Ts)} Ts(i)(f) == 1\right), Count\left(\sum_{i=1}^{size(Ts)} Ts(i)(f) == 0\right)\right)$
                Else
                        Compute multi-feature relative score MFRS.
                        MFRS = $\frac{\sum_{i=1}^{size(Tcs)} Tcs(i).f == Cs(i)}{size(Tcs)}$
                End
        End
        For each trace f
                        If f is numeric &&Tcs(i).f == Null then
                                Tcs(i)(f) = Fms(f)
                        Elseif f is binary &&Tcs(i).f == Null then
                                Tcs(i)(f) = FFV(f)
                        Else if f is categorical &&Tcs(i).f == Null then
                                Tcs(i)(f) = choose categorical value with maximum MFRS.
                        End
        End
        If clears all then
                        Add to preprocessed data set Pkds.
        Else
                        Remove from data set.
        End
Stop
```

Fig. 2 defines the process which is performing on the preprocessing algorithm. The data cleaning removes the noise, normalizations perform the selection of important features, and the highly correlated features are given as an output to perform the next stage that is feature extraction which leads to classification. The method computes the value of class-based mean value (CBMV) for numeric data, estimates feature frequency value (FFV) for binary values and computes multi-feature relative score (MFRS) for other features. For each trace, the method estimates the above mentioned values and if any trace clears all the above, then it has been considered for further processes of feature selection. Otherwise, it has been removed from the data set. The preprocessed data set has been used to perform feature selection.

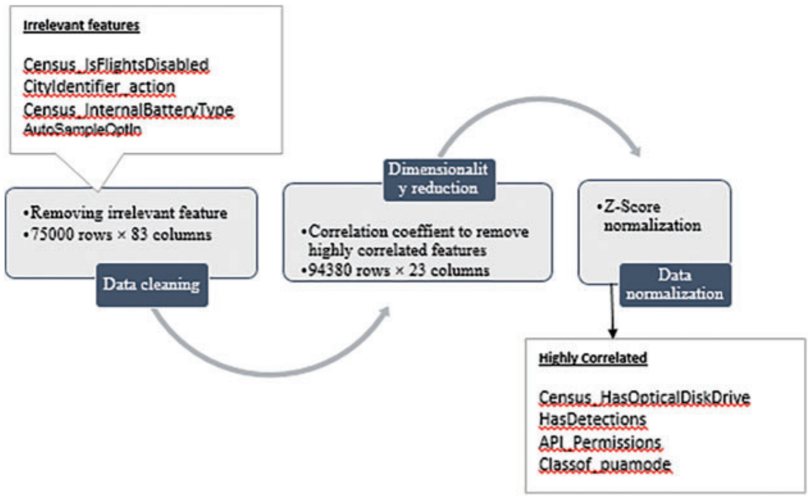


Figure 2: FBO preprocessor

3.2 CBPCA Feature Selection

The preprocessed data set has been applied with feature selection using the Class Based Principal Component Analysis (CBPCA) approach. The method first identifies the set of features present in the data set given. The features are classified according to their type as numerical, binary and categorical. Now, for each feature, the method computes feature histogram values, which returns a set of unique values of any feature and their histogram like image processing. According to the number of samples, and the type of the feature, the method computes the Class Orient Fitness Score (COFS), which is measured based on the variance of feature. According to the COFS value, the method selects a subset of features to be used for tree generation and classification.

Algorithm:

```
Given: Preprocessed data set Pds.
Obtain: Feature setFs, Histogram Set Hgs.
Start
    Find the feature list Fel = $\overset{size(Kds)}{\underset{i=1}{Fel \cup ((Features \in Kds(i)) \rightarrow (Feature \ni Fel))}}$
    Find Feature Types FType = $\sum_{i=1}^{size(Fel)} Fel(i).Type$
    Initialize Feature set Fs.
    For each feature f
            Generate histogramFhist = Histogram (Pds(Fel(f))
            If Ftype == categorical, then
```

(Continued)

Algorithm: Continued

```
            Compute Class orient categorical Fitness measure (COCFM).
            $COCFM = size\ (Fhist) > \left(\frac{1}{8} \times size\ (Pds)?\ 1{:}0\right)$
                If COCFM>0
                    Fs = FS F
                    hgs = Hgs∪F
                End
            Else if Ftype == Numerical then
                Compute Class orient numerical fitness measure CONFM.
                $CONFM = size(Fhist) > \frac{1}{3} \times size\ (Pds)\ ?1{:}0$
                If CONFM > Th then
                    Add to the set Fs = FS ∪F
                    Hgs = Hgs∪F
                End
            Else
                    Add to the set Fs = FS ∪F
            End
End
Stop
```

The above discussed algorithm visualized in Fig. 3, it computes Class orient categorical Fitness measure (COCFM) for the categorical data and computes class orient numerical fitness measure (CONFM) for the numeric values. According to the value of COCFM, CONFM values, and threshold, the method performs feature selection. The feature selection algorithm performs feature selection by computing class based fitness measure class orient numerical and class orient categorical fitness measures to identify the suitable features. An identified feature has been added to the feature set which has been used to perform malware classificiation.

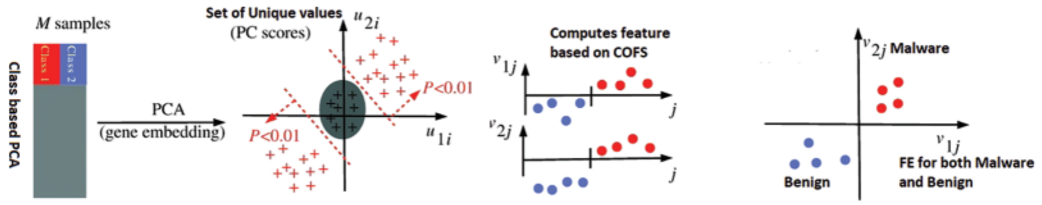


Figure 3: CBPCA feature extraction

3.3 Tree Generation

The features of the data set being extracted are used to generate a regression tree here. The number of trees has been decided on the histogram size. When the size of histogram is identified as higher, than the value, then the variance this needs to be considered greatly. Similarly, when the size of histogram is less or null, then it can contain low-variance feature values. So, according to the histogram size of any feature, the method computes minimum feature value and maximum feature value and based on that the number of trees to be generated for a feature has been measured. Similarly, for each feature, the method computes the minimum and maximum with the number of histogram values. Using these values, the method computes the tree fitness score, TFS for several features. Finally, a subset of features with different fitness scores has been selected and according to the value of histogram the number of trees for each feature has been measured. Based on the number of small trees was generated to reduce the loss ratio and error rate.

Algorithm:

```
Given: Feature Set Fs, Histogram Set Hgs
Obtain: Tree sets of Tree Set Ts
Start
    Read the feature set Fs and the histogram set Hgs.
    For each feature f
        Compute the minimum value of histogram Min-Hist = $\overset{size(Hgs(f))}{\underset{i=1}{Min}}(Hgs(f(i))$
        Compute the maximum value of histogram Max-Hist = $\overset{size(Hgs(f))}{\underset{i=1}{Max}}(Hgs(f(i))$
        Compute the number of histograms Nhist = size(Hgs(f))
        Generate range valuesFrange = split(minhist, maxhist, nhist)
        Compute Tree Fitness Score FTS = $\frac{\text{Nhist}}{\text{Frange}}$
    End
    For each feature with FTS>3
            Generate Tree FTree.
            Add other features as leaf.
            For each other feature Of
                add to tree Ftree;
                If leaf level == 3, then
                    Break.
                End
            End
            Add trees to tree set Ts.
    End
Stop
```

The algorithm discussed above represents how the tree generation is performed. The method estimates the tree fitness score for various features and their feature sets by computing the minimum and maximum values of histogram and total number of histograms available. Using the TFS value, the method selects the feature for tree generation. Similarly, a subset of trees are generated for each feature and added to the tree set. The generated tree set has been used to perform the classification later.

3.4 Malware Classification

The proposed malware classification algorithm works based on the ensemble learning classifier with the tree generated. The modified XGBoost algorithm has been designed to measure the tree-level ensemble similarity (TLES) and Class level Ensemble Similarity (CLES) measured with all the trees available with the tree set. As the tree has been generated and available for both malignant and benign class, there will be limited number of trees for each class of the test samples. Now, the method takes the test sample and extracts the features of the test sample. With the features extracted, the method identifies only the features that are identified from the feature selection phase. With the features identified, the method visits each tree and match the ensemble features to compute Tree level ensemble similarity which is computed based on the number of feature conditions gets through and the total number of feature conditions exist in the tree. Similarly, the method computes the class level ensemble similarity (CLES) based on the values of TLES. Using both values, the method computes the value of Class Match Similarity CMS. Based on the value of CMS, the method identifies the class of the sample whether malignant or benign.

Algorithm:

```
Given: Test sample T, Tree sets of tree Ts
Obtain: Class C
Start
    Read T and Ts.
    For each class C
        For each tree Ti
            Compute Tree-Level Ensemble Similarity (TLES).
            $\mathrm{TLES} = \frac{\text{Number of Levels or Conditions or Feature of T clears}}{\text{Total Conditions or features contains}}$
        End
            Compute Class-Level ensemble similarity (CLES).
            $\mathrm{CLES} = \frac{\sum_{i=1}^{\mathrm{size(Ts(C))}} \mathrm{Ts\,(C)\,(i)\,.TLES} \geq 1}{\mathrm{size\,(Ts\,(C))}}$
            Compute class match similarity $\mathrm{CMS} = \mathrm{CLES} \times \frac{\sum_{i=1}^{\mathrm{size(Ts)}} \mathrm{Ts\,(C)\,.TLES}}{\mathrm{Size\,(Ts\,(c))}}$
    End
        Choose the class C with maximum $\mathrm{CMS} = \underset{i=1}{\overset{Size\,(Ts)}{Max}}\,(Class\,(C)\,.CLES)$
Stop
```

The algorithm discussed above measures the class match similarity based on the value of tree level and class level similarity measures for a sample given with all the classes of trees. Now, based on the CMS value, a single class has been selected as the class of the given sample. This is how the method identifies the malware and classifies the given sample.

4 Results and Discussion

The proposed Adaptive feature centric XGBoost ensemble learning model has been implemented and evaluated with Kaggle data set of malware detection. The method has been implemented using Python, and the performance produced by the proposed model has been presented here and compared with the result of other approaches. Finally, the method has higher accuracy in malware classification and has been designed to fit the K feature data set and can be automatically adapted for N number of solutions.

The details of the data set used for the performance evaluation of proposed algorithm have been presented in the Table 1. According to this, the performance of the various methods are measured and presented in this section. The performances of the approaches are measured on the following parameters.

Table 1: Details of the simulation

Parameter	Values
Hyper parameters	Epochs, learning rate
Data set	From kaggle belongs to microsoft
Number of instances	More than 75000
Number of features	56
Types of features	Numeric, binary, alpha numeric
Tool Used	Python

4.1 Malware Classification Accuracy

The malware classification accuracy represents the performance of any approach in identifying the malware exactly for a given number of samples. The value of malware classification accuracy has been measured based on the number of true positive and true negative classes with the total samples. It has been measured as follows:

$$\text{MCA} = \frac{\text{Number of True Positive} + \text{Number of True Negative}}{\text{Total Samples}} \times 100 \tag{1}$$

From the Eq. (1), the addition of true positive and true negative from the total samples helps to calculate the malware classification accuracy. The values of MCA for the 75000 samples are listed.

With reference from the Table 2, the performance of malware classification accuracy is measured by varying the number of samples in the data set. In each case, the performance of the malware classification has been measured and compared with the results of other approaches. The results show that the proposed AFC-XGBoost approach produced higher classification accuracy than any other method considered.

Table 2: Analysis on malware classification accuracy

Malware classification accuracy			
Algorithms	25000 Samples	50000 Samples	75000 Samples
DWT	73	77	81
SigPID	75	78	84
HMM	87	89	91
AFC-XGBoost	89	92	96

The performance methods in classifying the malware have been measured and presented in Fig. 4. The performance of various methods are measured on varying number of samples and compared in Fig. 4. In all the cases, the proposed approach has produced higher performance than any other approach. The changes in hyper parameters such as epoch and the learning rate provide the appropriate results in classification. The main advantage in the proposed model has taken the lesser number of epochs and learning rate. Here, 0.01 is the learning rate and you get the best results in 150 epochs.

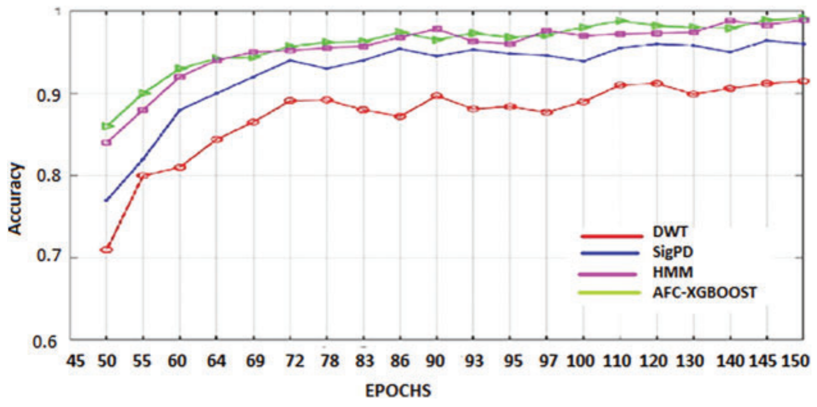


Figure 4: Performance on malware classification accuracy

In Table 3, the validation accuracy for the detection of malware is 0.974 achieved in epoch 150. The gradual increase of epochs reflects in the detection accuracy. The range of values varies from run 0 to run 3 where depends on the time. It has taken less epochs when compare to the existing algorithms like SigPID, HMM and DWT. The proposed algorithm helps in enhancing the detection method and the variation in max-depth and the fold weight.

Table 3: Validation accuracy for both malware and benign

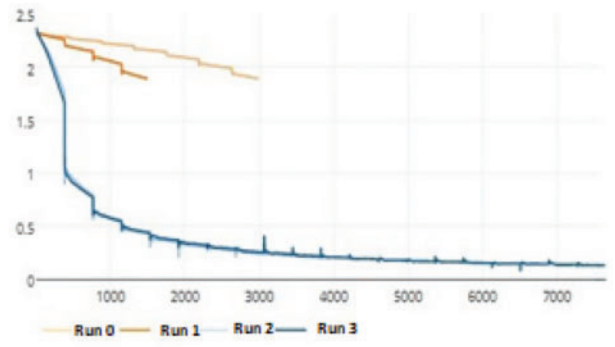
Validation accuracy		
Epochs	Benign	Malware
50	0.71	0.705
70	0.82	0.829
90	0.88	0.859
110	0.91	0.94
135	0.92	0.958
150	**0.97**	**0.974**

In Figs. 5a and 5b, represents the training loss and training accuracy. In Figs. 5c and 5d, represents the validation loss and accuracy. There is the gradual increase from run 0 to run 3 which can be visualized in the graph.

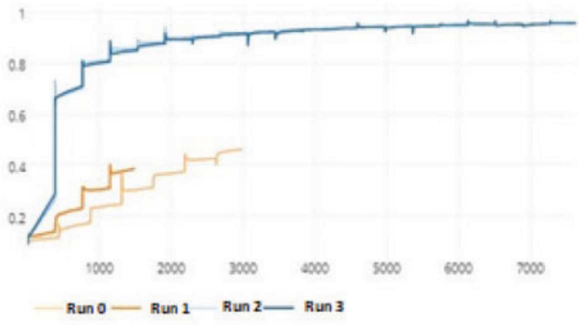
4.2 False Classification Ratio

The false classification ratio is the measure which represents the defect classification made by the algorithm. It has been measured based on the number of false positive and false negative classification produced by various approaches. It has been measured as follows:

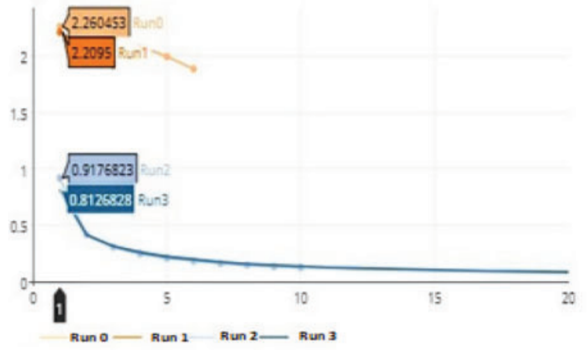
$$\text{FCR} = \frac{\text{Number of False Positive} + \text{ Number of false negative classification}}{\text{Total samples}} \times 100 \tag{2}$$



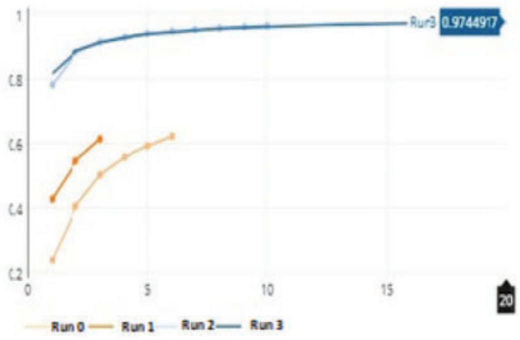
(a) Training Loss



(b) Training Validation



(c) Validation Loss



(d) Validation Accuracy

Figure 5: (a) Training loss (b) Training validation (c) Validation loss (d) Validation accuracy

With reference to Eq. (2), the calculation of false classification ratio for the total samples has been calculated and the values were listed in Table 4.

Table 4: Analysis on false ratio on malware classification

False classification ratio %			
Algorithms	25000 Samples	50000 Samples	75000 Samples
DWT	27	23	19
SigPID	25	22	16
HMM	13	11	9
AFC-XGBoost	11	8	4

The ratio of false classification introduced by different methods at the presence of different number of samples in the data set are measured and compared in Table 3. In each case, the proposed AFC-XGBoost algorithm has produced less false classification ratio in all the test cases than other approaches.

The graphical representation for the performance of false ratio is visualized in Fig. 6. The performance of algorithm varies for the number of samples and the important thing is to get the less false ration in detection and noticed that our novel model AFC-XGBoost achieved the less false positive rate in classification of malware.

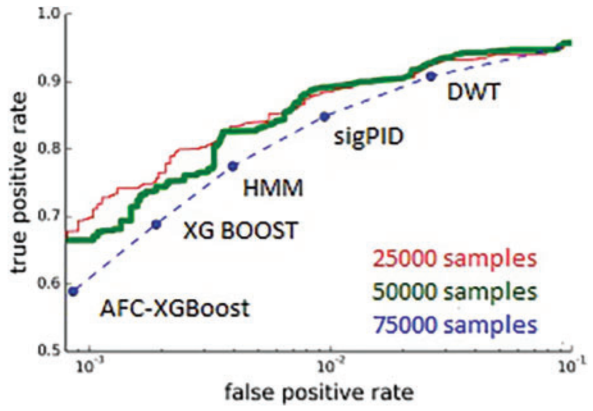


Figure 6: Performance on false ratio in malware classification

4.3 Time Complexity

Time complexity measures the value of time taken for classification of various algorithms based on the given samples. It has been measured as follows:

$$\text{Time Complexity} = \frac{\textit{Total Time Taken for Malware Classification}}{\textit{Total Number of Test samples submitted}} \tag{3}$$

With reference to the Eq. (3), the analysis is performed by considering different number of samples and features and the time taken for classification. In each case, it compares the results with the existing approaches. The proposed AFC-XGBoost has produced less time complexity compare to other methods.

From Table 5, it is observed and high lightened value is obtained based on the calculation of time complexity. Here, AFC-XGBoost occupied less time which is concluded based on comparison with various algorithms. In each case, the proposed approach has produced less time complexity in classification compared to other approaches.

Table 5: Analysis on time complexity on malware classification

Time complexity in seconds			
Algorithms	25000 Samples	50000 Samples	75000 Samples
DWT	67	83	97
SigPID	55	72	86
HMM	43	61	79
AFC-XGBoost	**21**	**25**	**34**

The performance of time complexity produced by different methods are measured and compared in Fig. 7, where the proposed AFC-XGBoost algorithm has produced less time complexity compare to other methods.

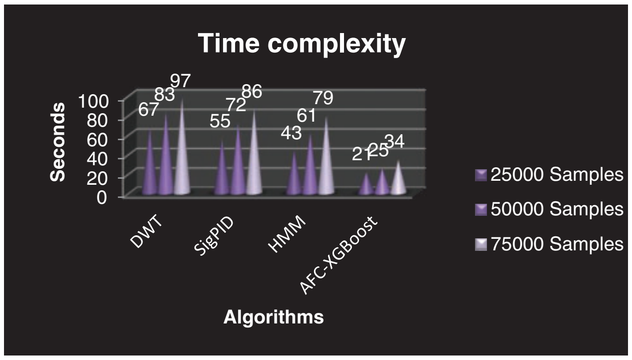


Figure 7: Performance on time complexity in malware classification

5 Conclusion

An efficient adaptive feature centric XGBoost Malware classification model has been presented and the model reads data set and performs preprocessing with Feature Base Optimizer algorithm which eliminates noisy records as well normalize the tuples by computing class based mean value (CBMV) for the features of numeric class and computes the frequency measures on each class toward binary values. Similarly, feature frequency measure (FFM) is computed for the categorical values. At the feature selection stage the method applies Class Based Principle Component Analysis (CBPCA) algorithm which is computes class orient fitness score. Further the method generates the regression trees according to histogram features of different features. At the test phase the method computes class match measure (CMS) being computed using tree level ensemble similarity (TLES) and class level ensemble similarity (CLES). Based on the value of CMS, the method performs classification of malwares. The proposed approach improves the performance of malware classification with the data set considered and reduces the false ratio and time complexity with the better accuracy of 97%.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

[1] N. Udayakumar, V. J. Saglani, A. V. Cupta and T. Subbulakshmi, "Malware classification using machine learning algorithms," in *2nd Int. Conf. on Trends in Electronics and Informatics (ICOEI)*, America, pp. 1–9, 2018.
[2] S. Choudhary and A. Sharma, "Malware detection and classification using machine learning," in *Int. Conf. on Emerging Trends in Communication, Control and Computing (ICONC3)*, America, pp. 1–4, 2020.
[3] C. Chen, C. Su, K. Lee and P. Bair, "Malware family classification using active learning by learning," in *22nd Int. Conf. on Advanced Communication Technology (ICACT)*, America, pp. 590–595, 2020.

[4] A. Fatima, R. Maurya, M. K. Dutta, R. Burget and J. Masek, "Android malware detection using genetic algorithm based optimized feature selection and machine learning," in *42nd Int. Conf. on Telecommunications and Signal Processing (TSP)*, America, pp. 220–223, 2019.

[5] K. Sethi, R. Kumar, L. Sethi, P. Bera and P. K. Patra, "A novel machine learning based malware detection and classification framework," in *Int. Conf. on Cyber Security and Protection of Digital Services (Cyber Security)*, America, pp. 1–4, 2019.

[6] J. Zhang, "CLEMENT: Machine learning methods for malware recognition based on semantic behaviours," in *Int. Conf. on Computer Information and Big Data Applications (CIBDA)*, America, pp. 233–236, 2020.

[7] N. Tarar, S. Sharma and C. R. Krishna, "Analysis and classification of android malware using machine learning algorithms," in *3rd Int. Conf. on Inventive Computation Technologies (ICICT)*, America, pp. 738–743, 2018.

[8] M. Z. Osman, A. F. Z. Abidin, R. N. Romli and M. F. Darmawan, "Pixel-based feature for android malware family classification using machine learning algorithms," in *Int. Conf. on Software Engineering & Computer Systems and 4th Int. Conf. on Computational Science and Information Management (ICSECS-ICOCSIM)*, America, pp. 552–555, 2021.

[9] A. Irshad, R. Maurya, M. K. Dutta, R. Burget and V. Uher, "Feature optimization for run time analysis of malware in windows operating system using machine learning approach," in *42nd Int. Conf. on Telecommunications and Signal Processing (TSP)*, America, pp. 255–260, 2019.

[10] H. Naeem, F. Ullah, M. R. Naeem, S. Khalid, D. Vasan *et al.,* "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, pp. 102154, 2020.

[11] D. Susanto, M. A. S. Stiawan, M. Arifin, Y. Idris and R. Budiarto, "IoT botnet malware classification using weka tool and scikit-learn machine learning," in *7th Int. Conf. on Electrical Engineering, Computer Sciences and Informatics (EECSI)*, America, pp. 15–20, 2020.

[12] A. Walker and S. Sengupta, "Insights into malware detection via behavioral frequency analysis using machine learning," in *MILCOM IEEE Military Communications Conf. (MILCOM)*, America, pp. 1–6, 2019.

[13] N. Chawla, H. Kumar and S. Mukhopadhyay, "Machine learning in wavelet domain for electromagnetic emission based malware analysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3426–3441, 2021.

[14] D. Xue, J. Li, T. Lv, W. Wu and J. Wang, "Malware classification using probability scoring and machine learning," *IEEE Access*, vol. 7, pp. 91641–91656, 2019.

[15] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an *et al.,* "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[16] Y. Li, K. Xiong, T. Chin and C. Hu, "A machine learning framework for domain generation algorithm-based malware detection," *IEEE Access*, vol. 7, pp. 32765–32782, 2019.

[17] M. Khoda, T. Imam, J. Kamruzzaman, I. Gondal and A. Rahman, "Robust malware defense in industrial IoT applications using machine learning with selective adversarial samples," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4415–4424, 2020.

[18] A. Pastor, A. Mozo, S. Vakaruk, D. Canavese, D. R. López *et al., "*Detection of encrypted cryptomining malware connections with machine and deep learning," *IEEE Access*, vol. 8, pp. 158036–158055, 2020.

[19] R. Kumar, K. Sethi, N. Prajapati, R. R. Rout and P. Bera, "Machine learning based malware detection in cloud environment using clustering approach," in *11th Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT)*, America, pp. 1–7, 2020.

[20] B. A. Ouahab, L. Elaachak, Y. A. Alluhaidan and M. Bouhorma, "A new approach to detect next generation of malware based on machine learning," in *Int. Conf. on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, America, pp. 230–235, 2021.

[21] W. N. H. Ibrahim, S. Anuar, A. Selamat, O. Krejcar, R. G. Crespo *et al.,* "Multilayer framework for botnet detection using machine learning algorithms," *IEEE Access*, vol. 9, pp. 48753–48768, 2021.

[22] M. Panda, A. A. A. Mousa and A. E. Hassanien, "Developing an efficient feature engineering and machine learning model for detecting IoT-botnet cyber attacks," *IEEE Access*, vol. 9, pp. 91038–91052, 2021.

[23] R. R. Karn, P. Kudva, H. Huang, S. Suneja and I. M. Elfadel, "Cryptomining detection in container clouds using system calls and explainable machine learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 674–691, 2021.

[24] F. Khan, C. Ncube, L. K. Ramasamy, S. Kadry and Y. Nam, "A digital DNA sequencing engine for ransomware detection using machine learning," *IEEE Access*, vol. 8, pp. 119710–119719, 2020.

[25] B. Baek, S. Euh, D. Baek, D. Kim and D. Hwang, "Histogram entropy representation and prototype based machine learning approach for malware family classification," *IEEE Access*, vol. 9, pp. 152098–152114, 2021.