International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-9 Issue-6, March 2021

Towards Optimization of Malware Detection using Extra-Tree and Random Forest Feature Selections on Ensemble Classifiers

Fadare Oluwaseun Gbenga, Adetunmbi Adebayo Olusola, Oyinloye Oghenerukevwe Elohor

Abstract: ***The proliferation of Malware on computer communication systems posed great security challenges to confidential data stored and other valuable substances across the globe. There have been several attempts in curbing the menace using a signature-based approach and in recent times, machine learning techniques have been extensively explored. This paper proposes a framework combining the exploit of both feature selections based on extra tree and random forest and eight ensemble techniques on five base learners- KNN, Naive Bayes, SVM, Decision Trees, and Logistic Regression. K-Nearest Neighbors returns the highest accuracy of 96.48%, 96.40%, and 87.89% on extra-tree, random forest, and without feature selection (WFS) respectively. Random forest ensemble accuracy on both Feature Selections are the highest with 98.50% and 98.16% on random forest and extra-tree respectively. The Extreme Gradient Boosting Classifier is next on random-forest FS with an accuracy of 98.37% while Voting returns the least detection accuracy of 95.80%. On extra-tree FS, Bagging is next with a detection accuracy of 98.09% while Voting returns the least accuracy of 95.54%. Random Forest has the highest all in seven evaluative measures in both extra tree and random forest feature selection techniques. The study results uncover the tree-based ensemble model is proficient and successful for malware classification.***

Keywords: Extra-tree, random forest, K-Nearest Neighbors, Extreme Gradient Boosting Classifier, Random forest ensemble.

I. INTRODUCTION

Malware is generally the most popular as noxious software. Malware is keyloggers, adware, spyware, botnet, worm, ransomware, rootkits, backdoor, trojan, infection among others, and there is an expansive scope of their families that are dynamic and broadly developing on the web each day. They duplicate themselves into the frameworks in various manners; either through various media or through the famous methods of getting them transferred into the frameworks as the confirmed applications. The massive influx of web clients, communication infrastructure, and present-day PCs in PC security has represented an extraordinary security danger to private information put away.

Manuscript received on March 16, 2021.
Revised Manuscript received on March 25, 2021.
Manuscript published on March 30, 2021.

Fadare Oluwaseun Gbenga, Department of Computer Science, Ekiti State University, Nigeria.
Adetunmbi Adebayo Olusola, Professor, Department of Computer Science, Federal University of Technology, Akure.
Oyinloye Oghenerukevwe Elohor, Department of Computer Science, Ekiti State University, Nigeria.

Variants of malware are developed to gain unlawful access to the systems and to get profitable benefits by illegal means. Reproduction of malware is dangerous to the security of the internet, the privacy of users, commercial factories, and governmental information.

The recent influx in network usage and high-speed digital communication has motivated the creation of new harmful malicious-code for various unlawful and unethical purposes. The variant of malware is growing, anti-virus companies cannot meet the yearnings of security protection, therefore, resulting in millions of systems being infected. A singular malware attack is capable of resulting in the breach of data and great losses. The overview directed by AV-Test [1] revealed that they are 350,000 new noxious code every day and these undesirable codes have a negative effect in the PC world. The measure of noxious variations is expanding quickly. For instance, 69,277,289 examples of noxious substances are identified by Kaspersky [2]. The general measure of malware variations expanded by 22% over the most recent two years to 600 and seventy (670) million examples identified by McAfee [3]. This measure of malevolent examples is unnecessarily huge, requiring dire methods for detecting malicious samples. Fast, internet-enabled communication gadgets enable malware to transmit and attack hosts rapidly and therefore it is important to recognize and dispose of new malware at the youthful age of their life cycle. The detection of malware assumes a fundamental part of PC security. New investigates essentially use machine learning approaches to deal with distinguish noxious samples. Different methods are bound for classifying new malicious samples from the existing signature method of detecting them. The traditional means of detecting malware are ineffective to compete with the new variant of malware samples. Traditional signature methods are incapable to compete with the new malicious samples. Numerous antivirus organizations gracefully protection components against malicious samples however their guard is not delivering any certain outcome. Machine learning approaches can enrich the literature and provide better experimental results in classifying malware. A few changes and enhancements occur in machine learning approaches like applying different feature selection methods and machine learning techniques. The feature selection methods are a fundamental aspect of the classification of the malware and benign as it eliminates superfluous features, decreases data dimensionality, improves precision, and limits preparing time before classification [4].

The feature selection procedure empowers machine-learning algorithm to decrease over-fitting, makes it simpler to decipher, reduces the complexity of a model, and trains quicker. Ensemble techniques are incredibly viable and yield better results as far as accuracy since they simply combine the
strength of several weak learners to form strong learners. This paper proposes a framework combining the exploit of both feature selections based on extra tree and random forest and eight ensemble techniques on five base learners- KNN, Naive Bayes, SVM, Decision Trees, and Logistic Regression. This article is structured into five sections. The next section discusses related work. The third section discusses the methodology while the fourth section discusses the experimental setup and result discussion. The fifth section discusses the conclusion.

II. RELATED WORK

Over the most recent twenty years, researchers have made several attempts in detecting malware using machine learning approaches. A few of these attempts are discussed below: Harshalatha and Mohanasundaram [5] observed a drastic increase in malware in recent times and this becomes a noteworthy problem in the cybersecurity space. They proposed a hybrid strategy of feature selection techniques with ensemble machine learning methods to better improve the accuracy of malware detection. Feature selection methods: Percentile, KBest, and Extra Trees Classifiers were used to select the most valuable features, and four ensemble classifiers: Bagging, Random Forest, AdaBoost, Gradient Boosting, were chosen for the classification. The result showed that a random forest classifier with a hybrid model gave the best outcome of 91.50 % accuracy.

Sharma et al. [6] proposed another strategy dependent on neural organizations for the detection of network intrusion. The authors proposed that Extra-Tree was picked to choose relevant features. Prior to classification, the proposed intrusion detection system performed better than all-new machine learning-based intrusion detection systems in terms of accuracy of the detection. It was deduced that the framework accomplished achieved 98.24% and 99.76% accuracy respectively for multi-class classification on the KDDcup99 and UNSW datasets

The improvement in machine learning algorithms has made the performance of malware detection accuracy better over the most recent ten years [7]. Nevertheless, there are still some hindrances like high storage usage, low scalability in learning, learning high-dimensional vectors, pre-processing a large amount of malware. This paper talked about low-dimensional yet relevant features for a malware detection framework. Binary or disassembly files represented by extracted five types of malware features known as the novel WEM (Window Entropy Map) image were used with variable length, and the set of frequently used APIs to reduce the processing time. The overall outcome of the following tree-based ensemble models: rotation trees, extra trees, random forest, and XGBoost and examined. In terms of accuracy AUC-PRC measurement, XGBoost is the best.

The authors constructed an underlying list of capabilities of 197 features by extricating them from a few PE header fields [8]. Wrapper and Filter highlights features procedures were picked in dimensionally-reduction. The picked features were utilized to prepare tree Bagging, Random Forest, Decision Tree, Bagging, and Boosted choice tree. The most elevated consequence of these methodologies was 0.998 AUC, 1.4% FP, and 99.1% accuracy, having a place with the random forest. This paper proposes a framework combining the exploit of both feature selections based on extra tree and random forest and eight ensemble techniques on five base learners- KNN, Naive Bayes, SVM, Decision Trees, and Logistic Regression. Consequently, the requirement for this work.

Damodaran et al.[9] suggested the using of the frequency of Windows XP, API-calls to classify samples of malware. Using a Decision Table (DT) and Random Forest, they implemented several classification algorithms and achieved a 97 percent. To classify malware, the authors in [10], used a hybrid static/dynamic model. They rely on the Hidden Markov Model (HMM) with API calls to classify malware with a 98 percent accuracy score for malware families. An approach to classifying malware using ensemble learning algorithms was proposed by Fang et al. [11]. As a feature selection technique, they rely on Term Frequency-Inverse Document Frequency (TF-IDF). In various scales of datasets, TF-IDF selects the highly ranked features from highly discriminatory features. In order to build a lightweight malware classification system, Zhang et al. [12] used ensemble learning. Even if the training data is imbalanced, the resulting classifier has the capacity to classify malware into its corresponding family.

III. METHODOLOGY

The proposed Malware detection depicted in Figure 1 comprises of four phases. Data acquisition and preprocessing, feature selection, model building, and evaluations. Dataset was extracted using python programming language and it was pre-processed. Feature selection methods were applied to the dataset. Dataset was standardized with standard scaler using the python programming language. Dataset was split into a ratio of 70% for the training set and 30% for the testing set. The training set is a dataset that machine learning algorithms and ensemble algorithms must act upon. The dataset we use to test the accuracy of our model is called the testing dataset. Base learners (KNN, Naive Bayes, SVM, Decision Trees and Logistic Regression) and ensembles (Bagging, AdaBoost, Gradient Boosting, Extreme Gradient Boosting Classifier, Light Gradient Boosting Classifier, Voting, Extra tree, and random forest) were applied on the training set and testing set. The feature selection techniques chosen for this study are random forest and extra tree process. We train and test the dataset with five classification algorithms. Training data are used to fit and fine-tune the models. Then we train and test the dataset with eight ensemble classifiers. The model was evaluated with the seven evaluative measures. Figure 2 reveals the proposed malware detection algorithm

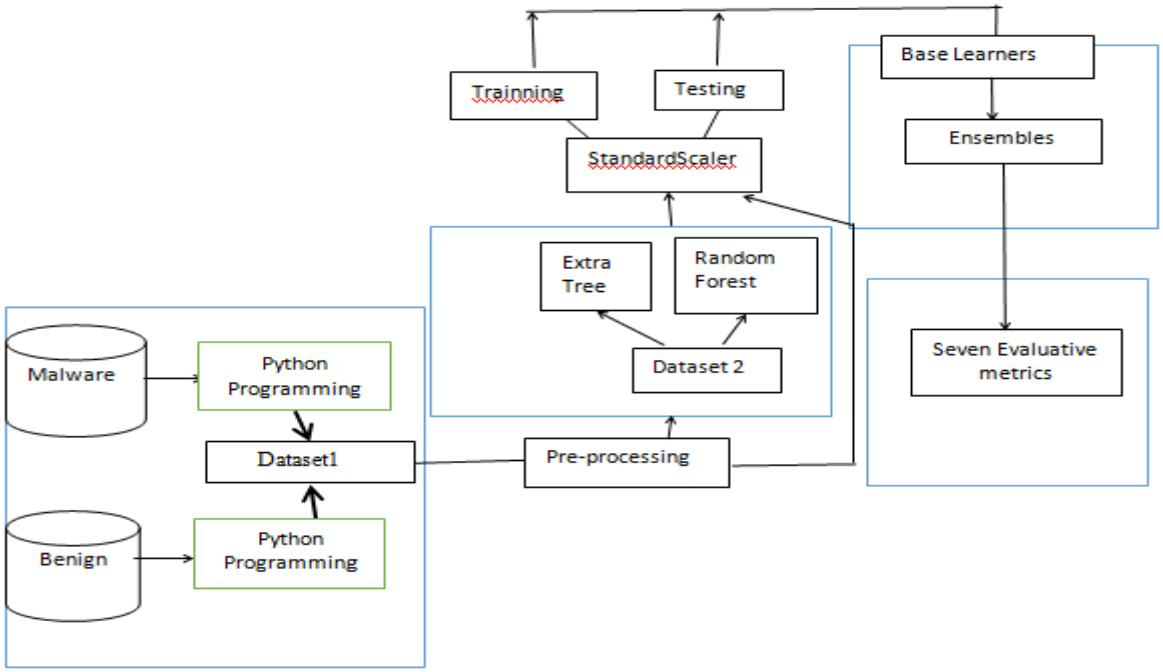

Fig 1. An Overview of proposed Malware Detection Model

Downloading of Benign and Malware Data

1. Feature Extraction by Python using PEFILE
2. Removal of Non-numeric types and zero-column fields from the Dataset
3. Feature Selection by Extra Tree and Random Forest
4. Training of Dataset by Base learners
5. Training of Dataset by Ensembles
6. Evaluative measures by seven evaluative metrics

Fig2. Algorithm for The Proposed Malware detection System

A. Data description

Portable Executables of 9,428 of 49.3 G of benign files are gotten from clean window applications 32 bits and 64 bits Windows 7 system, Windows XP, and downloaded from various benign websites repositories: Ninite [13], Downloads [14], and Softpedia [15]. Totalvirus [16] is used to scan all the downloaded files to ascertain if the files are truly benign or malware. Totalvirus [16] contains almost 80 AV engines, only downloaded files with zero detected rate from all the 80 AV engines were selected. These are benign files.

89G of 14,247 instances of malware was downloaded from Virushare [17] and Virussign [18], online repositories. Table I spells the distribution of the ten malware types in the dataset. The combination of malware and benign resulted in a total of 23,675 instances used for experimental study. The widespread of Windows affords it a captivating environment for virus creators to write malicious codes. In this paper, we center on the portable executable format for 64-bit Windows operating systems. In this paper, we simply consider non-packed programs. Packing is a technique that is used legally by software developers to build their programs from reverse engineering and malware writers use it to conceal the malicious program from being detected by AV engine [19]. The packed binary portable executables are the only executable that was separated from our data. Eighty-three (83) features are extracted using standardized PE File format from the dataset by using a python program. Thirteen features were dropped and they are target column, a column named "Name of file", non-numeric data type, and zero-column fields remaining seventy (70) features.

Table I-: Malware Data Type

No	Malware Type	Counts	No	Counts
1	Trojan	2493	6	983
2	Trojan-Dropper	1353	7	1350
3	Trojan-Spy	755	8	996
4	Virus	2165	9	1089
5	Worm	1601	10	1462

B. Feature Selection

Feature selection methods are helpful for some purposes like removing irrelevant features, improve accuracy, reduce the processing time, dimensionality reduction and improve accuracy [4][20]

C. Feature Selection by Extra Tree

Extra Trees is one of the tree-based ensemble classifiers that have similar behavior to the Random Forest ensemble

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{S} Entopy(S_v) \quad (1)$$

$$Entropy(S) = \sum_{i=1}^{c} - P_i \log_2(pi) \quad (2)$$

Where,

S: a training set , P_i: the proportion of rows with output label is i, c: Number of Unique Class labels

Table II-: Feature Selection by Extra Tree

	Data fields	Feature Names	Score
1	30	SectionAlignment	0.113655
2	13	ImportDllCount	0.078167
3	26	NumberOfSections	0.065881
4	2	Characteristics	0.064402
5	40	SectionNb	0.058152
6	37	SectionMinEntropy	0.054215
7	20	MajorSubsystemVersion	0.041195
8	59	e_lfanew	0.038691
9	12	ImportDllAllCount	0.036036
10	6	DllCharacteristics	0.029869
11	34	SectionMeanEntropy	0.026577
12	19	MajorOperatingSystemVersion	0.026253
13	31	SectionMaxEntropy	0.024546
14	18	MajorLinkerVersion	0.022189
15	17	MajorImageVersion	0.021728

classifier. It combines the group of decision trees known as "forest" and produce the classification outcome. Every decision tree is produced with the training class. Every tree produces some k sample of features from the training sample. The decision is reached by forming the formulas of Information Gain and Entropy. The (1) and (2) represent the formulas for Information Gain and Entropy respectively.

D. Feature Selection Using Random Forest

A random forest is a kind of tree-based learning algorithm that is formed from a combination of decision trees to perform classification. Decision trees have multiple of several nodes, that test some features based on certain conditions. This test results in a split of the data. Both groups of data continue to the next node, where they will be tested and split again until they reach the terminal node and we obtain the output. When more decision trees are rejoined to form a random forest model, it becomes impracticable to examine all trees and guess what features in the forest are the most relevant for making the predictions. The feature importance method for random forests is capable of marking conclusions about what features contribute most to the decision making in the model and help the users to better understand the model.

Table III-: Feature Selection by Random Forest

No	Data fields	Feature Names	Scores
1	12	ImportDllAllCount	0.125741
2	1	BaseOfCode	0.097676
3	26	NumberOfSections	0.069765
4	59	e_lfanew	0.055370
5	13	ImportDllCount	0.048440
6	21	MajorImageVersion	0.048385
7	3	CheckSum	0.047604
8	40	SectionNb	0.035139
9	30	SectionAlignment	0.027267
10	2	Characteristics	0.025205
11	37	SectionMinEntropy	0.023696
12	0	AddressOfEntryPoint	0.022844
13	7	ExportRVA	0.022204
14	18	MajorLinkerVersion	0.021916
15	31	SectionMaxEntropy	0.021396
16	38	SectionMinRawSize	0.021308
17	19	MajorOperatingSystemVersion	0.020584
18	29	ResSize	0.018338
19	34	SectionMeanEntropy	0.018167

E. Base Learners

Here, some of Base Learners calculations utilized in this exploratory examination are quickly inspected as along these lines:

F. Naïve Bayes

Naïve Bayes is a proportion of the probabilistic issue with the obscure testing events, it proposes (3) to create the back likelihood of each gathering and afterward the gathering with the most noteworthy worth is its forecast [21]

$$C = arg_{C_i} max P(C_i) \prod_J P(F_J/C_i) \quad (3)$$

Where *P(Ci)* is the probability of group *i*, *P(Fj|Ci)* is the conditional probability of each feature value given the group *i*.

G. Support Vector Machine

Support Vector Machine is centered on the structural risk minimization generalization from the applied mathematics learning theory [22].

It is especially suited for finding binary classification problems. SVM can find an optimal separating hyper-plane between two groups in a higher dimensional feature space.

The optimal hyperplane is subject to the constraint as:

$$min = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{l} \xi_i \quad (4)$$

s.t $y_i(w.x_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0 \quad i = 1, \ldots..i$

Where w is normal to the hyper-plane, ξ_i is an error for the i-th instance.

C is a sample to constitute the tradeoff between increasing the margin and decreasing the training error.

H. K-Nearest Neighbor

K-Nearest Neighbor(KNN) is the least complex AI calculation that is utilized for both characterization and relapse models. At whatever point the model is tried from testing information it finds the separation of that point with each other point, in the preparation information. At that point it finds the closest k individuals for that point.

The usage of KNN should be possible by following a few stages which are given beneath:

- Burden the data.
- Introduce the assessment of k.
- Forgetting the foreseen class, underline from 1 to Sindicate the quantity of preparing data focuses.
- Figure the distinction between test data and each column of preparing data. Here we will use Euclidean division as our partition metric since it's the most unmistakable strategy. For instance, in the event that we have two focuses P1(X1, Y1) and P2(X2, Y2) at that point

$$d(P1,P2) = \sqrt{(Y2-Y1)^2 + (X2-X1)^2} \quad (5)$$

I. Logistic regression algorithm

A logistic regression algorithm is utilized to foresee discrete or straight out qualities. It is basically an order calculation that is utilized in cases like misrepresentation identification, email spam location among others, where we need to settle on choices among yes and no.

It predicts the probability of the occasion of an event by fitting data into the rationale work. Typically, a strategic relapse model computes the class enrollment likelihood for one of the two classifications in the informational index, [23]. Notwithstanding, the Logistic bend is definitely not a straight bend like a direct relapse. It is known as the sigmoid bend and here likelihood.

$$p = 1/1 - e^{\wedge} - z \quad (6)$$

where z= mx+c. This equation of probability ensures that the predictor will be between 0 and 1

J. Decision Trees

C4.5 is the model that has a place with regulated machine learning type. It attempts to part the dataset into littler subsets by testing for one component at every hub. The calculation continues parting the dataset into littler lumps until all the perceptions in a single subset have a place with one class, for our situation 'malware' or 'generous'. The test directed on each split is finished by utilizing Information Gain or Gini record. C4.5, an improvement of ID3, utilizes an augmentation to data increase known as the addition proportion.

Increase proportion handles the issue of predisposition by normalizing the data gain utilizing Split Info. Java execution of the C4.5 calculation is known as J48, which is accessible in WEKA information mining apparatus.

K. Ensembles

Ensemble learning is a compelling part of AI that is utilized to improve the accuracy and performance of traditional machine learning classifiers [23]. It works by making a center gathering of students and joining their yields for ultimate choice creation. Machine learning exploits integral data of various classifiers to improve the presentation and precision of the choice. The ensemble method techniques examined included Gradient Boosting Classifier, AdaBoost, Bagging, Extreme inclination Boosting Classifier, Light Gradient Boosting Classifier, Random Forests, and Extra Trees Classifier. Furthermore, their concise descriptions are given below:.

L. Adaptive boosting

Adaptive boosting (AdaBoost) at first appoints equivalent loads to each preparation perception. It utilizes numerous feeble models and allocates higher loads to those perceptions for which misclassification was watched. As it utilizes different powerless models, joining the consequences of the choice limits accomplished during numerous cycles, the exactness of the misclassified perceptions is improved, and consequently the precision of the general emphases is additionally improved.

The weak models are evaluated using the error rate as given in (7):

$$\varepsilon_t = Pr_{\sim D_t}[h_t(X_i) \neq Yi] = \sum_{i:h_{t(X_i)} \neq Y_i} D_t \quad (7)$$

where ε_t is the weighted mistake gauge, $Pr_{\sim D_t}$ is the likelihood of the arbitrary model *i* to the distribution D*t*, h_t are the hypotheses of the weak learner, $X_i$is the training observation, Y_i is the target variable, *t* is the cycle number. The prediction error is one if the classification is wrong and 0 if the classification is correct.

M. Extra Trees Classifier

Extra Trees Classifier is an ensemble learning technique on a very basic level dependent on choice trees.

The Extra Trees calculation works by making an enormous number of unpruned choice trees from the preparation dataset. Like the random forest, the Extra Trees calculation will haphazardly test the highlights at each split purpose of a choice tree. Additional Trees resembles Random Forest, in that it manufactures numerous trees and parts hubs utilizing arbitrary subsets of highlights. In Extra Trees, irregularity doesn't originate from bootstrapping of information but instead originates from the arbitrary parts everything being equal. Extra-Trees work by diminishing change while simultaneously expanding predisposition. Additional Trees calculation chooses a split point indiscriminately. The quality of Extra Trees when contrasted with standard choice trees are their computational proficiency and lower fluctuation.

M. Random Forest

Random Forest uses expansion to the Bagging approach [24]. In Bagging, every classifier is constructed independently by working with a bootstrap test of the information. In an ordinary choice tree classifier, a choice at a hub split is made dependent on all the elements ascribes. In any case, in Random Forest, the best boundary at every hub in a choice tree is produced using a haphazardly chosen number of highlights. This random selection of features helps random forest models to not just scale well when there exist numerous highlights per include vector, yet additionally to diminish the relationship (connection) between the component credits. Random forest uses information content of a node as the splitting criterion

The information content is defined as:

$$I(N) = |S|H(S) - |S_L|H(S_L) - |S_R|H(S_R) \quad (8)$$

Where |S| = input size,

$|S_{I,R}|$ = Size of left. Right Subclasses of S.

$$H(S) = Shannon\ entropy\ of\ S = \sum_{i=+1}^{m} (p_i) log_2(p_2) \quad (9)$$

N. Gradient Boosting

Gradient Boosting or GBM is another ensemble machine learning algorithm that works for both relapse and characterization issues. GBM utilizes the boosting procedure, consolidating various powerless learners to shape a solid learner. It is a voracious calculation and can overfit a preparing dataset rapidly. It can benefit from regularization techniques that punish different pieces of the calculation and by and large improve the exhibition of the calculation by reducing overfitting.

O. Extreme Gradient Boosting

Extreme gradient boosting is the improvement of gradient boosting which depends on the boosting algorithm and considered as an ensemble classifier. It streamlined the inclination gradient boosting usage to works quicker and added regularization boundaries to evade overfitting. This experiment sets the learning rate for each boosting round to 0.01 and set the quantity of helped trees to 200. XGBoost is an advanced implementation of the gradient boosting algorithm. XGBoost has ended up being an exceptionally viable ML algorithm, broadly utilized in AI rivalries. XGBoost has a high prescient force and is very nearly multiple times quicker than the other gradient boosting techniques.

P. LightGBM Classifier

LightGBM Classifier is one of the machine learning algorithms utilized for arrangement and relapse. It joins models from various calculations to create a new iterative one. Gradient boosting is one of the most fiercely utilized AI calculations because of its exactness and proficiency [25,26]. Light GBM beats the various calculations when the dataset is amazingly huge. Contrasted with different calculations, Light GBM sets aside lesser effort to run on a colossal dataset. LightGBM utilizes tree-based calculations and follows leaf-wise methodology while different calculations work in a level-wise methodology design.

Q. Voting

Voting is the most intuitional method to combine heterogonous classifiers. Each base-level classifier casts a vote for its prediction and then combines classifiers according to the plurality vote.

A voting classifier may be a reasonable alternative if a single strategy is unable to meet the desired accuracy threshold. In the short voting classifier, instead, it enables the combining of various classifiers adopting a majority vote to determine which class should be considered the winning class during the prediction.

R. Bagging

Bagging (Bootstrap Aggregation) is essentially used to decrease the difference of a choice tree classifier. His objective is to make a few subsets of information from preparing test picked arbitrarily with substitution. Every assortment of subset information is utilized to prepare their choice trees. As an impact, we get a group of various models. Normal of the apparent multitude of expectations from various trees are utilized which is more vigorous than a solitary choice tree classifier.

IV. PERFORMANCE METRICS

For appraisal reason, we used the Overall Accuracy (OA), False Positive, False Negative, True Positive, True Negative, Recall, Precision, F1-Score, False Positive Rate, ROC, Cohen Kappa, and AUC. False Negatives (FN): the number of malicious samples classified as benign. True Negatives (TN): the number of benign samples classified as benign. False Positive (F.P) implies wrongly classifier favorable as malware. True Negative (T.N) means correctly classify benign as benign. The recall is the capacity of a calculation to locate every single positive example. A recall is equivalent to a True Positive rate. Precision is the proportion of accurately anticipated positive to all out the anticipated positive example. F1-Score is the weighted normal of Precision and Recall. As the accuracy and recall in characterization measure is a couple of contradictory measures, the utilization of F1-Score can adequately adjust the precision and review, and the more like 1 of F1-Score mathematical worth methods better classifier execution.
False Positive Rate is otherwise known as the probability of false alarm. Accuracy score= (TP+TN)/(TP+TN+FP+FN).

Precision=TP/(TP+FP), Recall=TP/(TP+FN). The ROC (AUC), Area of a classifier is the likelihood of the classifier positioning an arbitrarily picked positive occasion higher than a haphazardly picked negative occurrence.

The model is surveyed through a couple of introduction evaluation measures, precision, recall, review, f1-score, true positive rate, roc., cohen kappa, and AUC.

V. EXPERIMENTAL SETUP AND RESULTS DISCUSSION

The dataset was split in the ratio of 70:30 for the training and testing. Seventy percent (70%) of each member type of malware consulting 9,973 of malware and 6,600 of benign were used for the training while testing data comprising of 4,274 of malware and 2,828 of benign were used for the experimental setup. Benign was set to 1 and malware was set to 0, so experimental was based on a binary-class experiment. Feature selection on the training set is extra trees and random forest. Extra Trees selects 15 best features while random forest selects 19 best features among 70 features based on set the threshold. Table II and Table III show the feature selection by extra tree and feature selection by random forest respectively. For without feature selection, we used the resulted seventy (70) features extracted from the dataset. The results of base learners with feature selection techniques both extra tree and random forest of the dataset and without feature selection is shown in Table IV.

Table IV: Confusion Matrix and Accuracy for the base learners with and without feature selection

Base Learners	Feature Selections	TP	TN	FP	FN	Accuracy
Logistic Regression	Extra Tree	3978	2631	296	197	93.06
	Random Forest	3977	2632	297	196	93.06
	Without Feature Selection	3438	2260	836	568	80.23
Decision Tree	Extra Tree	4067	2733	207	95	95.75
	Random Forest	4097	2749	177	79	96.40
	Without Feature Selection	3754	2463	520	365	87.54
Support Vector Machine	Extra Tree	4007	2644	267	184	93.65
	Random Forest	4014	2649	260	179	93.82
	Without Feature Selection	3438	2260	836	568	80.23
Naive Bayes	Extra Tree	3970	2626	304	202	92.86
	Random Forest	3923	2825	351	243	91.64
	Without Feature Selection	3252	2162	1022	666	76.23
K-nearest neighbors	Extra Tree	4100	2752	174	76	96.48
	Random Forest	4097	2749	177	79	96.40
	Without Feature Selection	3768	2474	506	354	87.89

In terms of accuracy, Table IV shows that the KNN classifier returned the highest accuracy of 96.48% with the extra tree as feature selection technique while the decision tree classifier and KNN returned the highest accuracy of 96.40% with the random forest as a feature selection technique. Naïve Bayes returned the lowest accuracy of 92.86% and 91.64% with extra tree and random forest as feature selection techniques respectively. Without feature selection technique, KNN returned the highest accuracy of 87.89%, while, Naïve Bayes returned the lowest accuracy of 76.23%. Also, for logistic regression, extra tree and a random forest returned 93.06% which is greater than 80.23% for the without feature selection. In Decision Tree Classifier, random forest returned the highest of 96.40% while without feature selection returned the lowest of 87.54%. In Support Vector Machine, random forest returned the highest of 93.82% while without feature selection returned the lowest of 80.23%. In Naïve Bayes, an extra tree returned the highest of 92.86% while the without feature selection returned the lowest of 76.23%.

In K-nearest neighbors, an extra tree returned the highest of 96.48% while the without feature selection returned the lowest of 87.89%. This shows that feature selection techniques extra tree and random forest have a greater influence on detecting the accuracy of the classification.

Table V: Confusion Matrix and Accuracy for the Ensemble Methods with and without feature selection

Ensemble Methods	Feature selection Techniques	TP	TN	FP	FN	Accuracy
Bagging	Extra Tree	4165	2801	109	27	98.09
	Random Forest	4172	2807	102	21	98.27
	Without Feature Selection	3869	2539	405	289	90.23
AdaBoosting	Extra Tree	4132	2779	142	49	97.31
	Random Forest	4151	2790	123	38	97.73
	Without Feature Selection	3824	2490	450	338	88.90
Gradient Boosting	Extra Tree	4128	2777	146	51	97.23
	Random Forest	4160	2797	114	31	97.96
	Without Feature Selection	3835	2498	439	330	89.17
Extreme Gradient Boosting Classifier	Extra Tree	4156	2793	118	35	97.85
	Random Forest	4176	2810	98	18	98.37
	Without Feature Selection	3926	2588	348	240	91.72
Light Gradient Boosting Classifier	Extra Tree	4160	2796	114	32	97.94
	Random Forest	4169	2805	105	23	98.20
	Without Feature Selection	3879	2554	395	274	90.58
Majority Voting	Extra Tree	4059	2726	215	102	95.54
	Random Forest	4069	2735	205	93	95.80
	Without Feature Selection	3681	2447	593	381	86.29
Random Forest	Extra Tree	4167	2804	107	24	98.16
	Random Forest	4181	2813	93	15	98.50
	Without Feature Selection	3895	2566	379	262	90.97
Extra Tree	Extra Tree	4162	2799	112	29	98.01
	Random Forest	417269	2802	105	21	98.27
	Without Feature Selection	3867	2538	407	290	90.19

Table V is the table for ensemble methods with extra tree and random forest as feature selection techniques and without feature selection technique. Random forest classifier had the highest accuracy of 98.16% of all ensemble approaches with the feature selection technique of extra tree while random forest had the highest accuracy of 98.50% of all ensemble approaches with feature selection technique of random forest. Without the feature selection technique, the extreme gradient boosting classifier had the highest accuracy of 91.72%. Voting had the lowest accuracy of 95.54% of all ensemble approaches

with feature selection technique of extra tree while voting had the lowest accuracy of 95.80% of all ensemble approaches with the feature selection technique of random forest. Without the feature selection technique, voting had the lowest accuracy of 86.29%. In the Bagging ensemble method, the random forest had the highest accuracy of 98.27%, followed by extra tree an accuracy of 98.09% while the ensemble method without feature selection method had a lowest accuracy of 90.23%. In the AdaBoosting ensemble method, the random forest had the highest accuracy of 97.737%, followed by the extra tree with an accuracy of 97.31% while the ensemble method without feature selection method had the accuracy of 88.90%. In the Gradient Boosting ensemble method, random forest recorded the highest accuracy of 97.96%, followed by extra tree with the accuracy of 97.23% while the ensemble method without feature selection method had lowest accuracy of 89.17% . In the Extreme Gradient Boosting Classifier ensemble method, random forest recorded the highest accuracy of 98.37%, followed by the extra tree with an accuracy of 97.94% while ensemble method without feature selection method had lowest accuracy of 91.72%. Light Gradient Boosting Classifier, Majority Voting, Random Forest and Extra Tree follow the same pattern in terms of accuracies. This was inconsistent with [27] which proved that a combination of methods gives better classification with high accuracy of 100% with the Random Forest ensemble classifier. These show that feature selection techniques extra tree and random forest have a greater influence on detecting the accuracy of the ensembles. Experimentation from Table 4 and Table V demonstrated that the presentation of the group model is better than the individual non-ensemble classifiers in terms of performance and accuracies.

Table VI: Seven Evaluative Measures of All Ensembles for Feature Importance by Extra Tree

Feature Importance by Extra Tree	Recall	Precision	F1-Score	False positive rate	ROC	Cohen Kappa	AUC-
Bagging	0.9936	0.9745	0.9840	0.0375	0.9923	0.8870	0.9572
Ada Boosting	0.9882	0.9668	0.9774	0.0486	0.9735	0.8572	0.9274
Gradient Boosting	0.9878	0.9658	0.9767	0.0499	0.9893	0.8535	0.9323
Extreme Gradient	0.9916	0.9724	0.9819	0.0405	0.9914	0.8777	0.9442
Light Gradient	0.9924	0.9733	0.9825	0.0392	0.9913	0.8808	0.9510
Majority Voting	0.9755	0.9495	0.9623	0.0731	0.9641	0.8512	0.9189
Random Forest	0.9945	0.9750	0.9845	0.0368	0.9922	0.8879	0.9621
Extra Tree	0.9931	0.9740	0.9834	0.0385	0.9918	0.8879	0.9567

Table VI shows the seven parametric measures of the ensembles with the feature selection technique of the extra tree. In this study, Random Forest returning the highest of all except in ROC, 0.9945 in the recall, 0.9750 in precision, 0.9845 in f1-score, false-positive rate of 0.0368, cohen kappa of 0.8879, and AUC of 0.9621. Majority Voting returning the lowest of all; 0.9755 in the recall, 0.9495 in precision, 0.9623 in f1-score, false-positive rate of 0.0731, 0.9641 in roc, 0.8512 in cohen kappa, and 0.9189 of AUC.

Table VII: Seven Evaluative Measures of All Ensembles for Random Forest

Random Forest	Recall	Precision	F1-Score	False positive rate	ROC	Cohen	AUC-
Bagging	0.9950	0.9761	0.9855	0.0351	0.9934	0.8949	0.9688
Ada Boosting	0.9911	0.9712	0.9810	0.0422	0.9811	0.8726	0.9401
Gradient Boosting	0.9926	0.9733	0.9829	0.0392	0.9926	0.8811	0.9557
Extreme Gradient	0.9957	0.9771	0.9863	0.0337	0.9940	0.8967	0.9645
Light Gradient	0.9945	0.9754	0.9849	0.0361	0.9932	0.8909	0.9599
Majority Voting	0.9776	0.9520	0.9646	0.0697	0.9789	0.8681	0.9379
Random Forest	0.9964	0.9782	0.9872	0.0319	0.9949	0.9055	0.9713
Extra Tree	0.9950	0.9761	0.9855	0.0361	0.9945	0.9007	0.9648

Table 7 shows the seven parametric measures of the ensembles with the feature selection technique of random forest.

Random Forest recorded the highest of all the evaluative traits, 0.9964 in the recall, 0.9782 in precision, 0.9872 in f1-score, false-positive rate of 0.0319, roc of 0.9949, cohen kappa of 0.9055, and AUC of 0.9713. Majority Voting recorded the lowest of all the evaluative traits with the recall of 0.9776, the precision of 0.9520, f1-score of 0.9646, false-positive rate of 0.0697, roc of 0.9789, cohen kappa of 0.8681, and AUC of 0.9379.

Table VIII: Seven Evaluative Measures of All Ensembles for Without Feature Selection Methods

Without selection feature	Recall	Precision	F1_Score	False	ROC	Cohen	AUC-
Bagging	0.9304	0.9052	0.9176	0.1376	0.8392	0.7302	0.8308
Ada Boosting	0.9187	0.8947	0.9065	0.1531	0.8272	0.7242	0.8180
Gradient Boosting	0.9207	0.8973	0.9088	0.1495	0.8311	0.7107	0.8242
Extreme Gradient	0.9424	0.9186	0.9303	0.1185	0.8381	0.7281	0.8302
Light Gradient	0.9338	0.9076	0.9205	0.1339	0.8321	0.7267	0.8272
Majority Voting	0.9062	0.8613	0.8832	0.1951	0.8198	0.7089	0.8091
Random Forest	0.9370	0.9113	0.9240	0.1287	0.8418	0.7312	0.8314
Extra Tree	0.9302	0.9048	0.9173	0.1382	0.8371	0.7298	0.8293

Table VIII shows the seven parametric measures of the ensembles without the feature selection technique method. Extreme Gradient Boosting recorded the highest in four evaluative traits out of seven, recall of 0.9424, the precision of 0.9186, f1-score of 0.9303. and false positive rate of 0.1185 while the rest of the traits went to the random forest. Majority Voting returning the lowest with recall of the 0.9062, the precision of 0.8613, f1-score of 0.8832,

false-positive rate of 0.195, roc of 0.8198, Cohen Kappa of 0.7089, and AUC of 0.8091.

Table IX: Seven Evaluative Measures of All Ensembles with their Feature Selection Methods

Ensembles	Feature Selection	Recall	Precision	F1_Score	False	ROC	Cohen	AUC-
Bagging	Extra Tree	0.9936	0.9745	0.9840	0.0375	0.9923	0.8870	0.9572
	Random Forest	0.9950	0.9761	0.9855	0.0351	0.9934	0.8949	0.9688
	Without F.S	0.9304	0.9052	0.9176	0.1376	0.8392	0.7302	0.8308
Ada Boosting	Extra Tree	0.9882	0.9668	0.9774	0.0486	0.9735	0.8572	0.9274
	Random Forest	0.9911	0.9712	0.9810	0.0422	0.9811	0.8726	0.9401
	Without F.S	0.9187	0.8947	0.9065	0.1531	0.8272	0.7242	0.8180
Gradient Boosting	Extra Tree	0.9878	0.9658	0.9767	0.0499	0.9893	0.8535	0.9323
	Random Forest	0.9926	0.9733	0.9829	0.0392	0.9926	0.8811	0.9557
	Without F.S	0.9207	0.8973	0.9088	0.1495	0.8311	0.7107	0.8242
Extreme Gradient Boosting	Extra Tree	0.9916	0.9724	0.9819	0.0405	0.9914	0.8777	0.9442
	Random Forest	0.9957	0.9771	0.9863	0.0337	0.9940	0.8967	0.9645
	Without F.S	0.9424	0.9186	0.9303	0.1185	0.8381	0.7281	0.8302
Light Gradient Boosting	Extra Tree	0.9924	0.9733	0.9825	0.0392	0.9913	0.8808	0.9510
	Random Forest	0.9945	0.9754	0.9849	0.0361	0.9932	0.8909	0.9599
	Without F.S	0.9338	0.9076	0.9205	0.1339	0.8321	0.7267	0.8272
Majority Voting	Extra Tree	0.9755	0.9495	0.9623	0.0731	0.9641	0.8512	0.9189
	Random Forest	0.9776	0.9520	0.9646	0.0697	0.9789	0.8681	0.9379
	Without F.S	0.9062	0.8613	0.8832	0.1951	0.8198	0.7089	0.8091
Random Forest	Extra Tree	0.9945	0.9750	0.9845	0.0368	0.9922	0.8879	0.9621
	Random Forest	0.9964	0.9782	0.9872	0.0319	0.9949	0.9055	0.9713
	Without F.S	0.9370	0.9113	0.9240	0.1287	0.8418	0.7312	0.8314
Extra Tree	Extra Tree	0.9931	0.9740	0.9834	0.0385	0.9918	0.8879	0.9567
	Random Forest	0.9950	0.9761	0.9855	0.0361	0.9945	0.9007	0.9648
	Without F.S	0.9302	0.9048	0.9173	0.1382	0.8371	0.7298	0.8293

Table IX shows the seven parametric measures of the ensembles with their feature selection techniques. It can be seen that in all the ensemble methods, Random Forest returned the highest of all seven evaluative measures. This was inconsistent with [24] which indicated that even with less feature selection used, the Random Forest classifier with 0.992 performs comparatively better in malware classification, much better than the popular classification algorithms such as SVM with 0.956 accuracy.

VI. CONCLUSION

Feature Importance by an extra tree and random forest are selected for feature selection techniques that are trained on the eight ensembles. From table 4 to table 5, it could be seen that feature selection techniques have great influences on the classification results. The study demonstrates that the tree-based ensemble model is compelling and effective for malware classification. Besides, we investigate the adequacy of various ensemble learning methods to support the accuracy and execution of the malware identification. The ensemble model performs better than a single classifier model in terms of improving the detection accuracy. It plainly distinguishes that machine learning algorithms are extremely helpful for the arrangement and grouping of malware tests for little datasets and for enormous volumes of information. The preparation time for grouping is additionally diminished given eliminating features in the feature selection process for classification. In future, we will execute the proposed approach on a huge of dataset and will act in the profound examination for the classification of packed binaries malicious software, which is viewed as exceptionally undermining in current research.

LIMITATION OF THE STUDY

Care was taken in navigating through dangerous malware sites; virusshare.com and virussign.com and the packed binaries malicious software were excluded form the research. Only the binaries executables were used for the research.

REFERENCES

1. AV-TEST (2019), The Independent IT-Security Institute, https://www.av-test.org/en/statistics/malware/. Accessed 2 November 2019.
2. Kaspersky Security Bulletin (2016), Overall statistics, https://securelist.com/kaspersky-security-bulletin-2016-executive-summary/76858/. Accessed 12 May 2016.
3. McAfee Labs Threats Report (2017),https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-jun-2017.pdf. Accessed 2 June 2017.
4. Chandrashekar G. and Sahin F., "A survey on feature selection methods", Computers & Electrical Engineering., vol. 40(1), 2014, pp.16-28.
5. HarshaLatha P. and Mohanasundaram R, "A New Hybrid Strategy for Malware Detection Classification with Multiple Feature Selection Methods and Ensemble Learning Methods", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249–8958., vol. 9(2), 2019, pp. 4013-4019.
6. Sharma J, Charul G, Ole-Christoffer G., and G. Morten G., "Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation", EURASIP Journal on Information Security., vol. 15(1), 2019. pp. 15-27.
7. Euh S., Hyunjong L, Donghoon K., and Doosung H., "Comparative Analysis of Low-Dimensional Features and Tree-Based Ensembles for Malware Detection Systems", *IEEE Access*., vol. 8(2), 2020 pp. 76796-76808.
8. Tian R., Islam L, Batten L, and Versteeg S., "Differentiating malware from cleanware using behavioural analysis", 5th international conference on malicious and unwanted software, IEEE, 2010.

9. Damodaran A., Troia F.D., Visaggio A., Austin H, .Stamp A., Comparison of static, dynamic, and hybrid analysis for malware detection, Journal of Computer Virology and HackingTechniques., vol. 13(1), 2017, pp.1-12.
10. Fang Y., Yu B., Tang Y., Liu L., Lu Z., Wang Y, and Yang Q. "A new malware classification approach based on malware dynamic analysis", In Australasian onference on Information Security and Privacy, Springer, Cham, 2017, pp. 173–189.
11. Zhang Y., Huang Q., Ma X, Yang Z, and Jiang J. "Using multi-features and ensemble learning method for imbalanced malware classification", IEEE Trustcom/BigDataSE/ISPA, IEEE, 2016, pp. 965–973.
12. Bai, J. Wang, and G. Zou G., "A malware detection scheme based on mining format information, Sci. World Journal, vol. 12(4), 2014 pp. 36-48.
13. Ninite (2019). Benign data, www.ninite.com. Accessed in 29 November 2019.
14. Download (2018). Benign data, www.downloads.com. Accessed in 29 November 2019.
15. Softpedia (2019). Benign data, www.softpedia.com. Accessed in 29 November 2019.
16. Totalvirus (2019). Online file checker, www.totalvirus.com. Accessed in 29 November 2019.
17. Virushare (2019). Malware data, www.virushare.com. Accessed in 22 November 2019.
18. Virussign (2019). Malware data, www.virussign.com. Accessed in 22 November 2019.
19. Singh A, and Lakhotia A., "Game-theoretic design of an information exchange model for detecting packed malware in Malicious and Unwanted Software" (MALWARE), 6th International Conference, 2011, pp. 1–7 .
20. Li J., Cheng K., Wang S., Morstatter F., Trevino R.P., Tang J., and Liu H., "Feature selection: A data perspective", ACM Computing Surveys (CSUR)., vol. 50(6), 2018, pp. 94-100,
21. Schultz M.G., Eskin E, Zadok F., and Stolfo S.J., "Data mining methods for detection of new malicious executables" Proc. IEEE Symp. Security, Privacy, 2001.
22. Konstantinou E. and Wolthusen S. Metamorphic Virus. Analysis and Detection, Technical Report RHUL-MA-2008-02, Royal Holloway University of London, 2008.
23. Dreiseitl S.and Ohno-Machado L., "Logistic regression and artificial neural network classification models: a methodology review, Journal of Biomedical Informatics, vol. 35(6), 2002, pp. 352-359.
24. Breiman L., "Bagging predictors", Machine Learning, vol. 24, 2002, pp. 123-140.
25. Ke G., Meng Q., Finley T., Wang T., Chen W., Ma W, Q. Ye Q., and .Liu T.Y., " LightGBM: A Highly Efficient Gradient Boosting Decision Tree", 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 2017.
26. Ramnathv and Gdequeiroz (2017), "Gradient Boosting Machines", https://github.com/ledell/useR-machinelearning-tutorial/blob/master/gradient-boosting-machines. Accessed 25 Sepember (2017).
27. Dada E.G., Bassi J.S., Hurcha Y.J. and Alkali A.H. , "Performance Evaluation of Machine Learning Algorithms for Detection and Prevention of Malware Attacks", IOSR Journal of Computer Engineering (IOSR-JCE)., vol. 21(3),2019, pp. 18-27.
28. HarshaLatha P. and Mohanasundaram R. "Improving Malware Detection Classification Accuracy with Feature Selection Methods and Ensemble-based Machine Learning Methods", International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3071, vol. 9(2), 2019, pp. 2055-2059.

AUTHORS PROFILE

Fadare Oluwaseun Gbenga has a Bachelor of Science and Master of Science in Computer Science. He is currently pursuing PhD at the Ekiti State University, Nigeria. He is dynamically engaged in research in the area of cyber-security.

Adetunmbi Adebayo is a Professor in the Department of Computer Science, Federal University of Technology Akure where he obtained his PhD in Computer Science in 2008. He was a recipient of CAS-TWAS postgraduate fellowship at the Institute of Computing Technology, Beijing in 2006 and a Visiting Scholar to Massachusetts Institute of Technology in 2012 under MIT International Science and Technology Initiatives Empowering the Teachers Program. He has several publication in reputable peer-reviewed journals and has also served as reviewers to several peer reviewed journals. His research interests are machine learning, information security and computational linguistics. He is a member of IEEE computer society and Computer professional Council of Nigeria.

Oyinloye Oghenerukevwe Elohor has a B.Sc., M.Tech., Ph.D in Computer science, her area of research is information security