Performance Analysis of Machine Learning Classifiers for Detecting PE Malware

ABM.Adnan Azmee[1], Pranto Protim Choudhury[2], Md. Aosaful Alam[3], Orko Dutta[4], Muhammad Iqbal Hossain[5]
Computer Science and Engineering, BRAC University
Dhaka, Bangladesh

***Abstract*—In this modern era of technology, securing and protecting one's data has been a major concern and needs to be focused on. Malware is a program that is designed to cause harm and malware analysis is one of the paramount focused points under the sight of cyber forensic professionals and network administrations. The degree of the harm brought about by malignant programming varies to a great extent. If this happens at home to a random person then that may lead to some loss of irrelevant or unimportant information but for a corporate network, it can lead to loss of valuable business data. The existing research does focus on some few machine learning algorithms to detect malware and very few of them worked with Portable Executables (PE) files. In this paper, we mainly focused on top classification algorithms and compare their accuracy to find out which one is giving the best result according to the dataset and also compare among these algorithms. Top machine learning classification algorithms were used alongside neural networks such as Artificial Neural Network, XGBoost, Support Vector Machine, Extra Tree Classifier, etc. The experimental result shows that XGBoost achieved the highest accuracy of 98.62 percent when compared with other approaches. Thus, to provide a better solution for this kind of anomalies, we have been interested in researching malware detection and want to contribute to building strong and protective cybersecurity.**

Keywords—Malware detection; machine learning; data protection; XGBoost; support vector machine; extra tree classifiers; artificial neural network

I. Introduction

Malware or malicious software, a dangerous computer code, intended to disturb, cripple or take control of the computer system without the approval of the user. It takes advantage of the technical faults or vulnerabilities in the operating system, hardware, and software. Malware is frequently used to take assets from the PC or attempting to take some significant data, records or cash from an individual. Malware has been there on the internet for quite a long amount of time and "Brain" is considered to be the first virus in the history on the personal computer (pc). It was originated by certain Pakistani young boys in their late teens for their motivations and purposes. Amjad Farooq Alvi was the mind behind the coding section of Brain [1]. In this 21st century, both malware and its spreading rate are growing rapidly along with the advancement of technology. Malware and viruses are also used in cyber warfare between countries. Iran and Saudi Arabia have been in cyber warfare for more than 10 years. However, every malware is not disastrous but it can cause certain limits of distress like it can cause the laptop or computer to slow down or run at a slow pace and can also cause irritating conduct like creating a series of pop-up advertisements.

Antivirus is often unaware of any new virus or malware that is being spread through the internet and by the time a solution comes, users have already been affected by the new virus. In addition, this can lead to the loss of useful information and money. Saving personal data is everyone's concern, but in this era of technology, it seems quite impossible. This malicious malware is the reason for losing billions of dollars as well as data. Cybercriminals will take an expected thirty-three billion records by 2023 as indicated by a recent report from Juniper Exploration [2]. Every year billions of data are being stolen from various sites and some of them are used even for bad purposes. Almost sixty million Americans have been influenced by data fraud as per a 2018 online overview by The Harris Survey. A similar overview demonstrates about fifteen million purchasers experienced data fraud in 2017 [3]. These attacks are also causing great financial loss. In the Accenture report, they say that a company on average costs us dollars due to malware attacks [4].

Much research has been conducted on making new techniques and strategies to assemble, study and ease noxious code as malware is spreading at an alarming rate on the web. Unfortunately, current host-based detection methods undergo weak detection models. These models focus on the highlights of a particular malware occasion and are regularly effectively evadable by obfuscation or polymorphism. Additionally, finders that check for the nearness of a grouping of framework calls displayed by a malware example are frequently evadable by framework call reordering. To address the inadequacies of weak models, a few powerful discovery approaches have been suggested that expect to recognize the conduct shown by a malware family. Albeit promising, these methodologies are tragically too hindered to even think about being utilized as constant finders on the end host, and they regularly require lumbering virtual machine innovation. In our thesis, we carry out a comparative study on different machine learning classification algorithms in detecting malware and benign PE (Portable Executable) files. We used a dataset from Kaggle, which was built using python library (PE files) and contains malicious and benign data of PE files. At first from the given dataset we perform data pre-processing and then implement them on the machine learning algorithms we select to work with. In our research, we used several machine learning algorithms to assemble several classifiers and then we used those classifiers to detect the PE malware and after that, we compared the performance of classifiers in accurately identifying malware. We compared different machine algorithms and tried to figure out which algorithm has the highest accuracy to detect the malware.

II. Literature Review

In [5], the authors worked on portable executables and tried to figure out which of the files are malware and created an integrated feature set based on raw and derived inputs. Moreover, they had used only six algorithms to compare between integrated and raw feature set. Furthermore, they used the 10 fold cross-validation technique. They created two datasets with the help of virus-share, Windows XP and Windows 7. One of them had two thousand seven hundred twenty-two malware and two thousand four hundred eighty-eight benign data and another one had one hundred twenty-nine malware data and thirty benign data. Lastly, they changed the feature numbers several times. In our paper, we use more algorithms than them, and our dataset is more ethical because we collected it from Kaggle. And, we have more data than their dataset.

In [6], Singhal made a model that was intended to help in the network security of enterprises. He did it because they felt that the signature-based antivirus system could be enough for household purposes but might be a threatening issue for networks of enterprises. However, he used only three algorithms to compare with their model and worked on only five thousand data. His model gave ninety-eight percent accuracy but only worked well for enterprise networks, not with the personal computers of home. In addition, he did not mention any processing techniques like PCA, LDA but we used some of those techniques in our work. We are also ahead of him in comparison to the highest accuracy rate of algorithms and the number of data in the dataset.

A PE malware detection system is created by the authors of the paper [7]. Actually, they worked in three steps, these are feature extraction, selection, and classification. For feature extraction, pefile, which is a python module had been used. However, they only used five hundred fifty-two data, which is much less than us. Moreover, they used the chi-square test, which found the relation between features with statistics and eventually works well with the small amount of data. If there are much data, it can lead to erroneous conclusions whereas our model did well with a large amount of data. We use a variety of algorithms like decision trees, boosting, and neural networking. Lastly, their accuracy rate was 97.25%, we found more accuracy in our work even with a large amount of data.

Another real-time malware detection work was done on [8] by authors. They extracted only 35 features, again less than our number of features. They used only four algorithms; CNN, MLP, SVM and random forest and netmate technique for feature extraction. It is another paper using fewer algorithms and got less accuracy rate than us. Each of their algorithms gave an accuracy rate of more than eighty-five percent. Whereas in our work, we got a more than eighty-five percent accuracy rate in several algorithms. And they did not use any feature reduction technique.

In [9], Merabet *et al.* research had been done on machine learning malware detection. In the early stage of our work, we got different ideas from this paper. According to them, the heuristic machine learning is better than the signature method which cannot stop new malware and dynamic method which has time and space complexity They made the comparison on several steps which are necessary for machine learning malware detection. Firstly, they compare among different feature extraction techniques like signature-based, dll function call, binary sequence, assembly sequence, pe file header, machine activity matrix, entropy signal. Secondly, they showed their survey on feature selection methods like information gain, redundant feature removal, principal component analysis, random forest, self-organizing feature map, wavelet transform. Lastly, they discussed three algorithms, these are supported machine vector, random forest, and artificial neural network and they compare accuracy rate based on feature selection and algorithms.

A hybrid machine learning technique is used for malware detection in [10] by the authors. They had the target of tracing and categorizing malware and had a better accuracy rate. For gaining that, they collected 5.05 GB benign along with 1.28 GB malware data and extract of n-gram and pe type of data. They only used three supervised classifiers (J48, Random Forest, Naive Bayes) and one unsupervised classifier (Self-organizing feature map). They used the information gain technique for feature selection.

And, in [11], Anderson *et al.* created a big dataset and opened it for all for research purposes. Their dataset contained three hundred thousand malicious data, three hundred thousand benign data, and three hundred thousand data which were not labeled. For test purposes, they reduced it to one hundred thousand malicious data and three hundred thousand benign data and make another dataset. However, they used eight types of features, only used LightGBM and proved that they gave a better performance than another deep learning model, malcov.

III. Proposed Approach

The workflow of our approach is given below Fig 1 to distinguish obscure malware by using machine learning. Our approach includes several steps to make it more accurate, efficient and effective. Our approach includes preprocessing of the dataset, feature selection, Training/Testing data in machine learning classification algorithm to detect malware or benign. After applying the classification algorithm we analyzed and visualized the result.

The dataset makes the machine learning training feasible. The dataset is used to train the model for performing various actions, to work automatically. The training dataset is a dataset in which the machine learning algorithm has trained and the dataset we use to validate the accuracy of our model is called testing dataset. Datasets can be built, downloaded from the website. Our dataset was collected from Microsoft Kaggle which contains nineteen thousand six hundred eleven samples. The specialty of our dataset is too many features. Our dataset contains seventy-nine features.

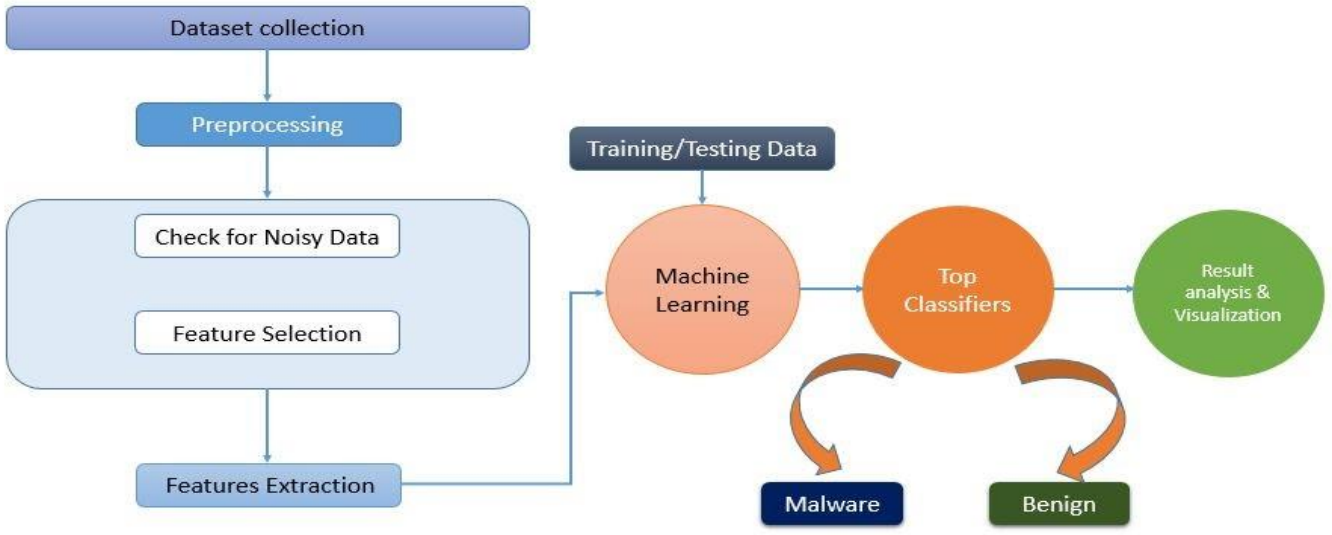


Fig. 1. The Workflow of our Approach.

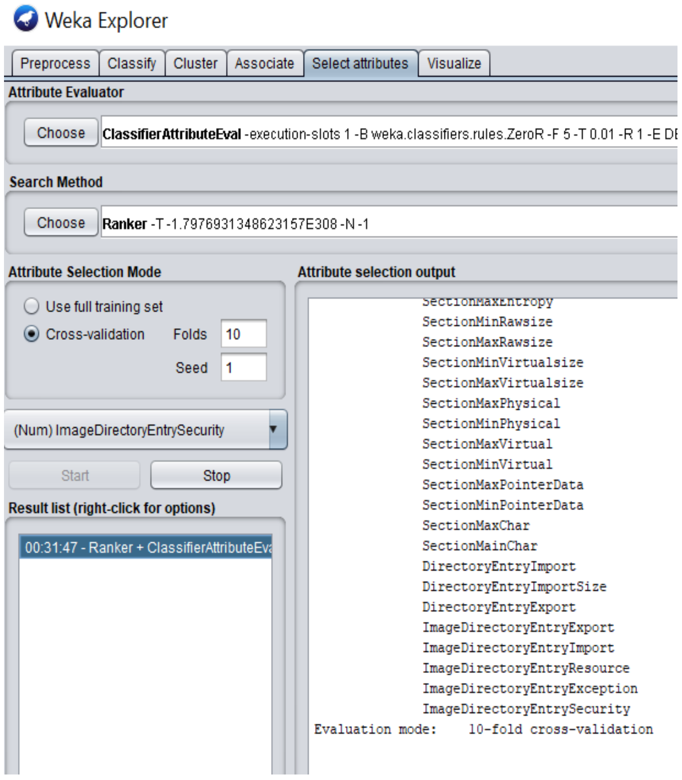
Data preprocessing is an information mining strategy that is used to change the unrefined information in an accommodating and appreciable format. One of the data preprocessing steps is PCA (Principal Component Analysis) which we used in our approach. PCA is a dimension reduction technique to avoid overfitting. It takes a dataset and "rotates" it, taking the original axes characterized by the original factors, and making new axes that are linear combinations of the old data. The linear combination exact is picked with the end goal that each progressive segment amplifies fluctuation along that new dimensions. It is a method of summarizing the data. Another preprocessing step we used in the dataset is that we used standard scalar. Standardization of a dataset is a typical necessity for some, machine learning estimators. They may carry on severely if the individual features do not basically look like standard usually scattered information, For example, numerous elements utilized in the target capacity of a learning algorithm accept that all features are based on zero and fluctuation in a similar request It will transfer data that its distribution will have a standard deviation of one and mean value zero.

Noisy data is pointless data. Any information that has been received, stored, or changed in such a way, that it can't be perused or utilized by the program that initially made it tends to be depicted as noisy. We checked for any noisy data in our dataset and we found nothing.

Feature Selection is where you consequently or physically select those Features which contribute most to your prediction variable or yield in which you are keen on.

Having unimportant Features in data can minimize the accuracy of the models and cause the model to learn reliant on immaterial Features. Feature selection can reduce overfitting, training time [12]. In our dataset, we have seventy-nine features. Among them, we worked with seventy-seven features. We dropped the target column and another column “Name”. We used Weka for feature selection as shown in Fig. 2.

Weka which stands for Waikato Environment for Knowledge Analysis is a collection of information analysis and visualization tools, exhibited by the University of Wakito. In weka in the attribute evaluator, we used ClassifierAttributeEval and for the searching method, we used Ranker. Ranker ranks each attribute and returns a score. For attribute selection mode we used 10 fold cross-validation. However, we used all the features of our work.



Attribute Evaluator

ClassifierAttributeEval -execution-slots 1 -B weka.classifiers.rules.ZeroR -F 5 -T 0.01 -R 1 -E DE

Search Method

Ranker -T -1.7976931348623157E308 -N -1

Attribute Selection Mode

Use full training set [✓ Cross-validation] Folds 10 Seed 1

(Num) ImageDirectoryEntrySecurity

Result list (right-click for options)

00:31:47 - Ranker + ClassifierAttributeEva

Attribute selection output

```
                 SectionMaxEntropy
                 SectionMinRawsize
                 SectionMaxRawsize
                 SectionMinVirtualsize
                 SectionMaxVirtualsize
                 SectionMaxPhysical
                 SectionMinPhysical
                 SectionMaxVirtual
                 SectionMinVirtual
                 SectionMaxPointerData
                 SectionMinPointerData
                 SectionMaxChar
                 SectionMainChar
                 DirectoryEntryImport
                 DirectoryEntryImportSize
                 DirectoryEntryExport
                 ImageDirectoryEntryExport
                 ImageDirectoryEntryImport
                 ImageDirectoryEntryResource
                 ImageDirectoryEntryException
                 ImageDirectoryEntrySecurity
Evaluation mode:    10-fold cross-validation
```

Fig. 2. Feature Selection by Weka.

After the feature selection process, the next step is training and testing of machine learning classifiers. We split our dataset into training and testing sets. Then we train and test the dataset in some classification algorithms. Training data are used to fit and tune the models. Test data are represented as unseen data to evaluate the models [13]. Test data is utilized to perceive how well the machine can foresee new answers dependent on training. In our dataset, we used eighty percent data as training data and twenty percent as testing data. After splitting the dataset we train our model. We have used nine classification algorithms for our approach. The nine classifiers are

1) Logistic Regression
2) K-Nearest Neighbor (KNN)
3) Random Forest
4) Adaboost
5) Support Vector Machine (SVM)
6) Decision Tree
7) XGBoost
8) Artificial Neural Network (ANN)
9) Extra Tree Classifier

These classification algorithms were used with particular laws to detect whether it is malware or benign.

After receiving the outcome, they were analyzed and visualized. As we used nine classification algorithm we analyzed their accuracy, Precision, Recall, True Positive Rate (TPR), False Positive Rate (FPR), f1-Score, Support, Confusion Matrix. For the visualization part, we have used Heatmap, Distribution Graph, Correlation Matrix, Pie plot, Counterplot.

IV. Dataset Description

Each executable document has a typical arrangement called Common Object File Format (COFF), an organization for executable, object code, shared library PC records utilized on Unix frameworks [14]. Also, PE (Portable Executables) design is one such COFF format accessible today for executable, object code, DLLs, FON font documents, and core dumps in thirty-two bit and sixty-four-bit versions of Windows operating systems. PE header actually holds the required data for the operating system to run the installation files like exe type data [15]. This comprises dynamic library references for linking, API export, and import tables, resource management data and TLS data. The Data Structures that are on the disk are the exact data structures that are being used in the memory. Instead of direct mapping of PE files into memory as a single memory-mapped file, the Win32 loader decides which portions of the PE files need to be mapped. Our dataset is from Kaggle and they used the python pefile module to generate information of pe header and sections of pe file. PE format is of two types. PE format is a well-known windows 32-bit format and PE+ is a well-known windows 64-bit format. The PE File structure Section symbolizes a portion of memory that contains either code or data and Section table contains numbers of sections. PE header holds the information that the operating system needs to know to run the executable, for example: (.exe) type files. Lastly, the DOS Mz header contains the offset of the PE header and Dos Stub prints whether the executables will run or not.

Here, Fig 3 represents the structure of a PE file which is given.

We have used seventy-seven columns as input because the column “name” for input has string values and the column “malware” is our target or the output column. In TABLE I, only a few important descriptions of columns are given.

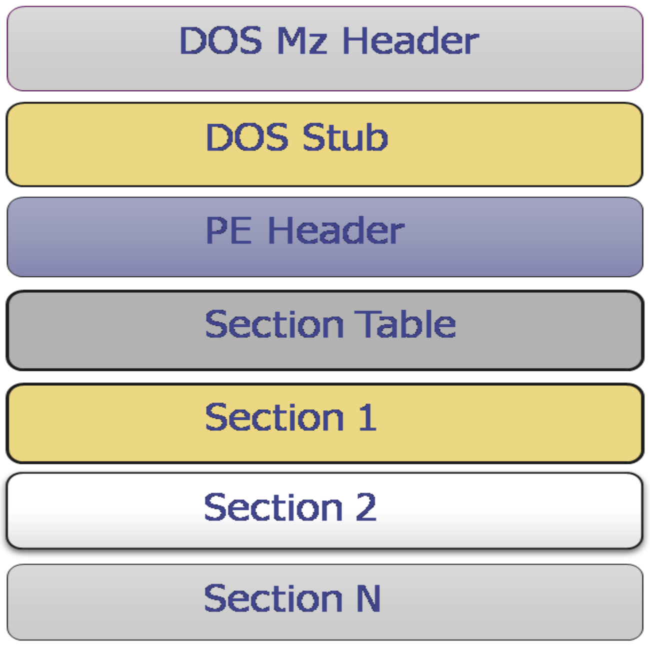


Fig. 3. Structure of PE Header.

TABLE. I. DATASET DESCRIPTION

	COLUMN NAME	DESCRIPTION
1	Magic	We need to know whether the executable image is of thirty-two bits or sixty-four bits and the magic field tells this accurately.
2	AddressOfEntryPoint	This holds the RVA (Relative Virtual Address) of the Entry Point (EP) of the module and is normally found in the text section.
3	BaseOfCode	BaseOfCode holds the RVAs of the beginning of the code.
4	BaseOfData	BaseOfData holds the data section of the beginning of the code.
5	ImageBase	Executable file needs to be memory-mapped to an exact location in memory and ImageBase is the address where it is done. 0x10000 is the default ImageBase for an executable in Windows NT and 0x400000 is the default ImageBase for DLL.
6	SectionAlignment	SectionAlignment designates to the alignment of the sections of PE in the memory.
7	FileAlignment	FileAlignment indicates the alignment of the sections of PE in the file.

V. Algorithms

We are using supervised machine learning which refers to the events when we have all information are labeled or in another way, we can say we know how all information is classified. Classification and regression are types of supervised learning where we aware of the classes of classification and values in regression.

A logistic regression algorithm is used to predict discrete or categorical values. It is mainly a classification algorithm that is used in cases like fraud detection, email spam detection, etc. where we have to make decisions between yes and no. It predicts the likelihood of the event of an occasion by fitting information into the logit function. Generally, a logistic regression model calculates the class membership probability for one of the two categories in the data set [16]. However, the Logistic curve is not a straight curve like linear regression. It is known as the sigmoid curve and here probability.

$$p = 1/\ 1 - e\hat{\ } - z \tag{1}$$

where z= mx+c. This equation of probability ensures that the predictor will be between 0 and 1. In our work after preprocessing by standard scaler, we used the logistic regression classifier.

K-Nearest Neighbor (KNN) is the simplest machine learning algorithm that is used for both classification and regression models. Whenever the model is fed with testing data it finds the distance of that point with every other point in the training data. Then it finds the nearest k members for that point. The implementation of KNN can be done by following some steps which are given below:

- Load the information.
- Initialize the estimation of k.
- For getting the anticipated class, emphasize from 1 to add up to the number of training information points.
- Calculate the difference between test information and each row of training information. Here we will utilize Euclidean separation as our separation metric since it's the most prominent technique. For example, if we have two points P1(X1, Y1) and P2(X2, Y2) then

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

Random Forest, like its name induces, is various decision trees joined into a single model. How Random forest capacities are by at first building trees and afterward accumulating them. The more the data the better the outcome as we merge all the more learning. As all the decision trees are merged, the figures will be closer to the mark. When looking at is done with substitutions, it draws the training set for current trees. After it's fulfillment, around thirty-three percent of the cases are chosen not to take any other potentially detrimental action for the model. As trees are added to the forest, the running fair check of the classification can be dictated by using the out-of-sack data. Central purposes of using random forest include dealing with both course of action and backslide issues, working with a categorical and fast and tenacious variable, normally handles missing regards, not requiring scaling and is less influenced by noise.

Boosting, in general, is a sequential process where we create multiple weak classifiers into a big ensemble strong classifier. It helps to improve the learners by focusing on areas where the system is not performing well. One of the best algorithms in this sector is AdaBoost, also known as Adaptive Boosting. The algorithm mainly increases the weight of certain training data inputs by using ensemble learning. There are some steps on how AdaBoost performs:

- First of all, it initializes weights to all training points. At initialization, every point that is used to train the classifier will be assigned a weight which is equal to the reciprocal of the total number of points that are present in my training data set.
- Then it calculates the error rate for each weak classifier that is present by growing multiple decision stumps and pick the decision stumps with the lowest error rate.
- The voting power of the weak classifier is computed and then appended the classifier into the ensemble classifier.
- The next step is to update the weight of every training point based on whether it was classified correctly or incorrectly by the previous classifier. The weight goes up for a point if it was classified incorrectly and goes down if it was classified correctly.

Support Vector Machine is a supervised machine learning algorithm that can be used for both classification and regression. The goal of SVM is to find the hyperplane which divides the two classes of the data. Under the assumption that two classes are considered the training set D (input space) of N pairs (xi, yi), i = 1,..., N, which is used during the training process can be defined as follows [17]:

$$D = \{ (xi, yi) \big| xi \ \in \ R^n, yi \ \in \{-1\ , +1\ \}\} \tag{3}$$

Given a training dataset the SVM algorithm searches for a plane (a hyperplane for n > 3) in the input space that separates the positive samples from the negative ones. In the original SVM model, all hyperplane in Rn is parameterized by a vector named w and constant b.

$$w^T x + b = 0 \tag{4}$$

For hyperplane (w, b) defined in (2) that separates the data, we can formulate the classification rule.

$$h(x) = sign\ (w^T x + b) \tag{5}$$

Here h(x) will correctly classify the sample data.

The decision tree algorithm can be utilized to take care of both classification and regression issues. We can speak to any boolean capacity on discrete characteristics utilizing the Decision tree. There are a few presumptions we have to settle on while utilizing decision trees. At first, we need to consider the root hub as the training set. Feature values are liked to be straight out. On the off chance that the qualities are continuous, at that point, they are discretized preceding the structure of the model. Based on quality qualities records are conveyed recursively. We utilize factual strategies for requesting qualities as root or the inside hub. In Decision Tree, the significant test is to recognizable proof of the characteristic for the root hub at each level. This procedure is known as attribute selection. We have two famous property choice measures:

- Information Gain.
- Gini Index.

XGBoost is an algorithm that has as of late been commanding applied machine learning and Kaggle competitions for organized or unthinkable information. XGBoost is a usage of gradient boosting decision trees intended for speed and execution.XGBoost modifies the boosting algorithm based on GBDT(Gradient Boosting Decision Trees). This algorithm is made out of numerous regression trees, and the last outcome is an added substance mix of the decision results of all subtrees. In any case, the glaring exclusion of GBDT is the need to use the leftover mistake of the n-1th tree when training the nth tree, which makes GBDT hard to be conveyed on the disseminated framework. regular item is added to the loss function to avoid overfitting which is given below:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j{}^2 \qquad (6)$$

Artificial Neural Network Algorithm (ANN) is developed from the idea of how the neurons of human brains are connected, and how the nervous system performs its work. Artificial Neural Network has many uses and classification is one of them. It can perform well for large datasets; instead of taking the entire dataset it takes data samples. Basically, the artificial neural network has three different layers, the first one is the input layer, where data are given as input, the second one is the hidden layer and the final one is the output layer. Depending on the number of hidden layers varies. ANN displaying begins with randomly appointed weight coefficients. At that point, a lot of information designs are nourished forward over and again, and loads of the neurons are changed until the yield coordinates intimately with the real values. In our model, we used two hidden layers. We used Rectified Linear Unit (ReLU) and Sigmoid function as the activation function in our model. After the model is defined we compiled it, we used cross-entropy as the loss function. For optimization, we used Adam optimization algorithm that is the extension of a greatly used optimization algorithm stochastic gradient descent.

The Extra-Tree technique (representing extremely randomized trees) was proposed with the primary goal of further randomizing trees working with regards to numerical information features, where the decision of the ideal cut-point is answerable for an enormous extent of the fluctuation of the incited tree. The Extra-Trees algorithm constructs a gathering of unpruned choice or relapse trees as per the traditional top-down strategy. Its two principal contrasts with other tree-based outfit techniques are that it parts nodes by picking cut-focuses completely indiscriminately and that it utilizes the entire learning test (as opposed to a bootstrap copy) to develop the trees. The Extra-Tree has two parameters: K, the number of properties arbitrarily chose at every node and nmin, the least example size for splitting a node. It is utilized a few times with the (full) unique learning test to create a troupe model (we mean by M the number of trees of this group). The forecasts of the trees are collected to yield the last expectation, by the greater part vote in characterization issues and number juggling normal in relapse issues. From the predisposition change perspective, the justification behind the Extra-Trees strategy is that the unequivocal randomization of the cut-point and property joined with gathering averaging ought to have the option to diminish change more emphatically than the more fragile randomization plans utilized by different strategies. The use of the full unique learning test as opposed to bootstrap copies is persuaded to limit predisposition.

VI. Result Analysis and Visualization

A. Data Visualization

In order to work with any dataset, it is very important to visualize it, python has a bunch of libraries with the help of which we can easily visualize the data. We used the python visualization library to plot different plots. Fig 4 represents the Heatmap and Fig 5 shows the Countplot of the dataset.

We used the python seaborn library to plot the heatmap. Heatmap is actually a colored tabular matrix where the color of each cell depends on the data contained in it. It shows the correlation. We also used count plot. A count plot is used to visualize the number of items present in each category in the form of a rectangular bar. We used the seaborn visualization libraries count plot to visualize our data. In our data 0 stands for benign and 1 stand for malware.

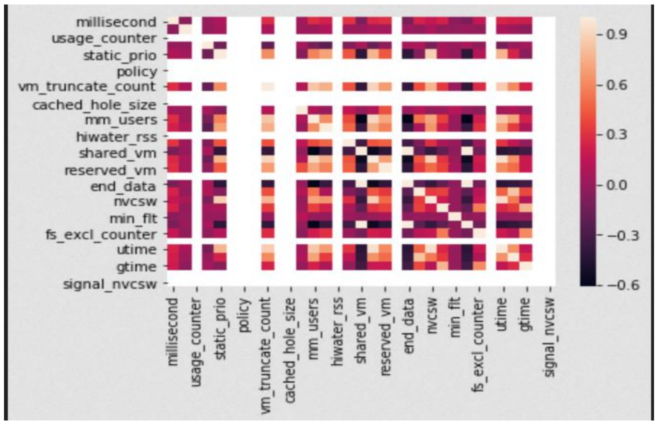


Fig. 4. Heatmap of the Input Data.

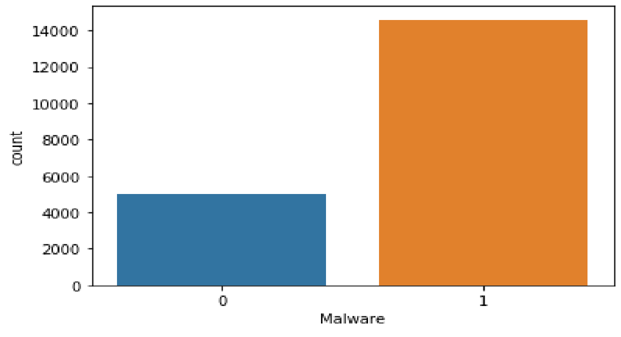


Fig. 5. Count Plot of our Dataset.

B. Result Analysis

In this part, we are going to analyze the result with the help of different evaluation metrics such as classification report, confusion matrix, roc curve, accuracy score, etc.

We used nine classifiers to classify the malware and benign data, they are Logistic Regression, K-Nearest Neighbor (k-NN), Random Forest, AdaBoost, Support Vector Machines (SVM), Decision Trees, XGBoost, Artificial Neural Network (ANN) and Extra Tree Classifier. After preprocessing we split the data into training and test set. We trained the classifier with the training set and then we tested them using the test data. We evaluated the performance of the classifiers with the help of different metrics. Different metrics used by us for evaluation are discussed as follows:

C. Confusion Matrix

One of the most widely used methods of evaluating the machine learning algorithm is the confusion matrix. TABLE II shows the different combinations of Confusion Matrix.

Here, True Negative (T.N) means that the algorithm predicted negative and the result is actually negative, False Positive (F.P) means that the algorithm predicted positive but the result is actually negative, False Negative (F.N) means that the algorithm predicted negative but the result is actually positive and True Positive (T.P) means that the algorithm predicted positive and the result is actually positive.

True Positive Rate (TPR): TPR is also known as recall. It shows the ability of our classifier to detect malware

TABLE. II. CONFUSION MATRIX

Confusion Matrix	Negative (Predicted)	Positive (Predicted)
Negative (actual)	True Negative (TN)	False Positive (FP)
Positive (actual)	False Negative (FN)	True Positive (TP)

TPR = True Positive / (True Positive + False Negative) (7)

False Positive Rate (FPR): FPR shows that the possibility of benign files wrongly classified as malware.

FPR = False Positive/ (False Positive + False Negative) (8)

D. Classification Report

The classification report is another evaluation metrics for the evaluation machine learning algorithm. We used the python sklearn libraries classification report to show the classification report of different algorithms. With the help of the classification report, we calculate the parameters like Precision, Recall, F1 score, Support, etc.

Precision is the ratio of correctly predicted positive to the total predicted positive sample. The recall is the ability of an algorithm to find all positive samples. F1 Score is the weighted average of Precision and Recall. support is the number of samples of the true sample that lie in that class i.e. number of occurrences of each class.

E. ROC Curve

Receiver Operating Characteristic curve (ROC Curve) is used to plot the true positive rate against the false-positive rate. With the help of ROC Curve, we can find the capability of a classifier to differentiate between different classes. The Area Under Curve (AUC) is the area under the ROC Curve, with the help of AUC score we get the idea of how well the model is performing.

F. Accuracy Score

An accuracy score is the most common metric for evaluating a model. We used sklearn Accuracy score metrics to find the accuracy score for different algorithms.

Accuracy score= (TP+TN)/ (TP+TN+FP+FN) (9)

TABLE III represents the result of the experiment, which is given.

TABLE. III. COMPARISON BETWEEN DIFFERENT ALGORITHMS EVALUATION METRICS

Applied Algorithms	Accuracy		Precision	Recall	f1-score	support	TPR	FPR
Logistic Regression	96.3%	0	0.92	0.93	0.93	990	0.974	0.075
		1	0.98	0.97	0.98	2933		
KNN	95.9%	0	0.92	0.92	0.92	986	0.974	0.083
		1	0.97	0.97	0.97	2937		
Random Forest	98.1%	0	0.94	0.98	0.96	953	0.994	0.057
		1	0.99	0.98	0.99	2970		
AdaBoost	97.2%	0	0.94	0.95	0.94	976	0.984	0.064
		1	0.98	0.98	0.98	2947		
SVM	96.3%	0	0.91	0.94	0.93	968	0.979	0.077
		1	0.98	0.97	0.98	2955		
Decision Tree	97.2%	0	0.95	0.95	0.95	996	0.980	0.051
		1	0.98	0.98	0.98	2927		
XGBoost	98.6%	0	0.96	0.98	0.97	975	0.994	0.037
		1	0.99	0.99	0.99	2948		
Artificial Neural Network	98.0%	0	0.98	0.94	0.96	995	0.993	0.057
		1	0.98	0.99	0.99	2928		
Extra Tree Classifier	95.9%	0	0.93	0.91	0.92	1014	0.969	0.069
		1	0.97	0.98	0.97	2909		

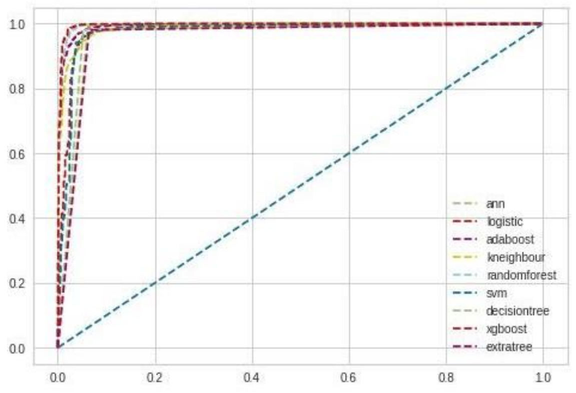


Fig. 6. Combined ROC Curve of All Algorithms.

At first, we used classification algorithms like logistic regression, K-nearest neighbor and SVM (Support Vector Machine) to classify our data. For small to medium size, tabular data most of the time decision tree-based algorithm performs best. Since we are using tabular data, we used many decision tree-based algorithms like decision tree, then random forest which basically is an ensemble method that consists of multiple decision trees. We used an extra tree classifier that works like a random forest but is much faster. Boosting algorithm like AdaBoost and XGBoost is used. We also used Artificial Neural Network to classify our data. Among the algorithms, XGBoost performs best due to its features like regularization, cross-validation, tree pruning, etc. Fig 6 represents the combined ROC curve of all algorithms.

Here the combined ROC curve shows the ratio of true positive rate and false positive rate of different algorithms. Here the AUC (Area Under Curve) for XGBoost is the highest that is 0.99.

VII. Conclusion and Future Works

In our work, we used multiple methods such as KNN, logistic regression, SVM, XGBoost, Decision Tree, etc. classifiers are used along with the artificial neural network. The dataset used in the proposed model had 19611 samples and the dataset was labeled. All the nine methods are compared that were used to detect the malware. After comparing all the three methods, we found that XGBoost got the highest accuracy of 98.6% whereas TPR is 0.99 and FPR is 0.037 with an AUC of 0.99. The second-best accuracy we got from Random Forest was 98.1%.KNN and Extra Tree Classifier got the lowest accuracy which was 95.9%. In any case, completely assess the reasonableness of our methodology, a lot more examination need to direct. While our underlying outcomes are promising, more works are needed to improve the technique and accuracy of our proposed model. For the future, we will consider a more advanced neural network alongside ANN. We are planning to build a hybrid model and an antivirus in the future, in which we are going to implement the hybrid machine learning algorithm. The antivirus would detect the PE file and predict whether they are malware or benign. We hope it would be able to tackle zero-day attacks, which the current antivirus is not able to.

References

[1] J. Harán and J. Harán, "Malware of the 1980s: Looking back at the Brain Virus and the Morris Worm | WeLiveSecurity", WeLiveSecurity, 2019. [Online]. Available: https://www.welivesecurity.com/2018/11/05/malware-1980s-brain-virus-morris-worm. [Accessed: 30- Dec-2019].

[2] “10 cyber security facts and statistics for 2018", Us.norton.com, 2019. [Online]. Available: https://us.norton.com/internetsecurity-emerging-threats-10-facts-about-todays-cybersecurity-landscape-that-you-should-know.html. [Accessed: 30- Dec- 2019].

[3] "LifeLock Official Site | Identity Theft Protection", Lifelock.com, 2019. [Online]. Available: https://www.lifelock.com/learn-identity-theft-resources-how-common-is-identity-theft.html. [Accessed: 30- Dec-2019].

[4] "Cybersecurity Consulting Services | Accenture", Accenture.com, 2019. [Online]. Available: https://www.accenture.com [Accessed: 30- Dec-2019].

[5] A. Kumar, K. Kuppusamy and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set", Journal of King Saud University - Computer and Information Sciences, vol. 31, no. 2, pp. 252-265, 2019. Available: 10.1016/j.jksuci.2017.01.003.

[6] P. Singhal, "Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks", International Journal of Network Security & Its Applications, vol. 4, no. 1, pp. 61-67, 2012. Available: 10.5121/ijnsa.2012.4106.

[7] "A Chi-Square-Based Decision for Real-Time Malware Detection Using PE-File Features", Journal of Information Processing Systems, 2016. Available: 10.3745/jips.03.0058.

[8] "CSDL | IEEE Computer Society", Computer.org, 2019. [Online]. Available: https://www.computer.org/csdl/proceedings-article/icoin/2018. [Accessed: 30- Dec- 2019].

[9] H. Merabet and A. Hajraoui, "A Survey of Malware Detection Techniques based on Machine Learning", International Journal of Advanced Computer Science and Applications, vol. 10, no. 1, 2019. Available: 10.14569/ijacsa.2019.0100148.

[10] R. Yang, V. Kang, S. Albouq and M. Zohdy, "Application of Hybrid Machine Learning to Detect and Remove Malware", Transactions on Machine Learning and Artificial Intelligence, vol. 3, no. 4, 2015. Available: 10.14738/tmlai.34.1436.

[11] Anderson, H. S., Roth, and Phil, “EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models,” arXiv.org, 16-Apr-2018. [Online]. Available: https://arxiv.org. [Accessed: 10-Dec-2019].

[12] "Feature Selection Techniques in Machine Learning with Python", Medium, 2019. [Online]. Available: https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e. [Accessed: 30- Dec- 2019].

[13] "Chapter 6: Model Training with Machine Learning - Data Science Primer", EliteDataScience, 2019. [Online]. Available: https://elitedatascience.com/model-training. [Accessed: 30- Dec- 2019].

[14] "PE Format - Win32 apps", Docs.microsoft.com, 2019. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/debug/pe-format. [Accessed: 30- Dec- 2019].

[15] "Exploit Development 02—PE File Format 1", Medium, 2019. [Online]. Available: https://medium.com/@MKahsari/exploit-developm ent-02-pe-file-format-1-998c252b5670. [Accessed: 25- Dec- 2019].

[16] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review", Journal of Biomedical Informatics, vol. 35, no. 5-6, pp. 352-359, 2002. Available: 10.1016/s1532-0464(03)00034-0.

[17] Cs.cornell.edu, 2019. [Online]. Available: http://www.cs.cornell.edu/courses/cs578/2003fa/slides_sigir03_tutorial-modified.v3.pdf. [Accessed: 30- Dec- 2019].