WORLD ORGANIZATION OF APPLIED SCIENCES JOURNALS

International Journal of Information Technology and Applied Sciences
ISSN (2709-2208)
Int. J. Inf. Tec. App. Sci. 3, No.4 (October-2021)
https://doi.org/10.52502/ijitas.v3i4.172

Malware Detection Using a Machine-Learning Based Approach

Safa Rkhouya[1], Khalid Chougdali[2]

[1] National School of Applied Sciences (ENSA) Ibn Tofail University, Kenitra, Morocco
[2] Engineering Science Laboratory, National School of Applied Sciences (ENSA) Ibn Tofail University, Kenitra, Morocco

Abstract: The purpose of this research work is to study the usage of machine learning in detecting malware. This paper presents a versatile framework, in which a dataset of more than 130000 files has been analyzed, to train and test four machine learning algorithms: Support Vector Machine, Decision Tree, Random Forest, and Gradient Boosting; The performance of each algorithm in malware classification, has been studied based on the: Accuracy, execution time, rate of false positives and false negatives, and area under the Receiver Operating Characteristic curve.

Keywords: Malware classification, PE files, SVM , Decision Tree, Random Forest, Gradient Boosting, Machine Learning.

1. Introduction

Computer viruses, or malware in general, have become one of the biggest problems for IT in recent years. This is evidenced by the millions of attacks per day on systems, especially in the era of Corona virus. Current antivirus products are unable to detect new viruses or variants of known viruses because they rely on the principle of signature-based methods. Although this method is very effective in detecting known viruses, it needs to be updated regularly to combat new viruses; this keeps virus programmers one step ahead of information security experts [1]. To solve this problem, machine learning methods and classifiers are used for computer virus detection. Since 2001, researchers and security experts have shown that this approach can indeed lead to successful malware detection [2].

In this research, a machine learning approach was used for malware detection. For this purpose, 124987 malware samples and 10400 legitimate samples were collected. From the analyzed samples, 54 features were extracted to create a large dataset; this dataset was later used to train and test four malware classification algorithms; the performance of these algorithms was analyzed using the following criteria: Execution time, accuracy, false positive rate, false negative rate, and area under the ROC curve.

A. *Research Question*

This research will attempt to answer the following questions:

- To what extent can machine learning be useful in malware detection?
- How do the different machine learning algorithms perform in malware detection?

B. *Limitations of the Research*

The limitations of this research are as follows:

- The files under study are Portable Executable (PE).
- The algorithms studied are: SVM, Decision Tree, Gradient Boosting and Random Forest.
- Since the algorithms used have many parameters, the default parameters were.

2. Data and Method

As mentioned earlier, 4 machine learning algorithms are used in this experiment. The method and framework used in this research are as follows:

a) Collection of test files.

b) Extracting many features from the binary files to get a good set of training data.

c) Identifying the relevant features for the classification.

d) Train and test each algorithm.

e) Test the efficiency of each algorithm and determine the rate of false positives and false negatives.

A. *Collecting of Test Files*

To create a good data set that will produce accurate results, it is important to have a large number of both legitimate and malicious files. The malicious files used in this work were obtained from the VirusShare website by contacting them. Legitimate files were downloaded from various websites on the Internet (standard Windows applications were also used).

- The number of malicious files used in this research is: 124987
- The number of legitimate files used in this research is: 10400

It is important to be careful when working with malicious files to avoid getting infected or causing a malware outbreak.

B. *Extracting features from files*

The first step in supervised machine learning is to create a dataset. The feature dataset is extracted from the previously collected files. The features that are extracted from the files are the PE headers. Before citing these features, take a look at the format of a PE file [3]; the following figure shows the general format of a PE file:

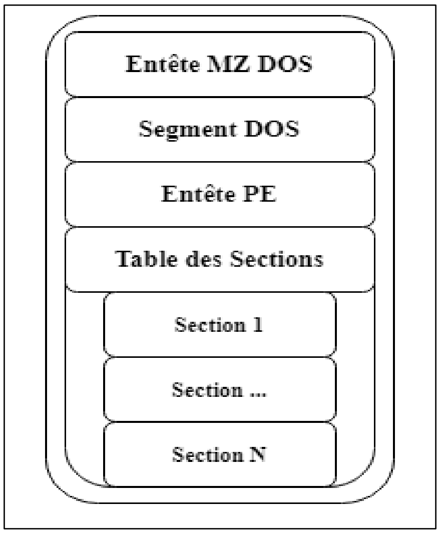

Figure 1. Format of a PE file

Machine learning uses integer or float data as recognition functionalities [4], most of the parameters of PE are already integer values (FieldSize, addresses, entropy...). The parameters of PE were extracted using the "pefile" library in Python. Upon extraction, relevant features were considered [5], below is the final list of extracted features:
AddressOfEntryPoint, BaseOfCode, BaseOfData, Characteristics, checksum, DllCharacteristics, ExportNb, FileAlignment, ImageBase, ImportsNb, ImportsNbDLL, ImportsNbOrdinal, LoadConfigurationSize, LoaderFlags,machine, MajorImageVersion, MajorLinkerVersion, MajorOperatingSystemVersion, MajorSubsystemVersion, md5, MinorImageVersion, MinorLinkerVersion, MinorOperatingSystemVersion, MinorSubsystemVersion, Name, NumberOfRvaAndSizes, ResourcesMaxEntropy, ResourcesMaxSize, ResourcesMeanEntropy, ResourcesMeanSize, ResourcesMinEntropy, ResourcesMinSize, ResourcesNb, SectionAlignment, SectionMaxRawsize, SectionMaxVirtualsize, SectionsMaxEntropy, SectionsMeanEntropy, SectionsMeanRawsize, SectionsMeanVirtualsize, SectionsMinEntropy, SectionsMinRawsize, SectionsMinVirtualsize, SectionsNb, SizeOfCode, SizeOfHeaders, SizeOfHeapCommit, SizeOfHeapReserve, SizeOfImage, SizeOfInitializedData, SizeOfOptionalHeader, SizeOfStackCommit, SizeOfStackReserve, SizeOfUninitializedData, Subsystem, VersionInformation Size.

C. *Selecting the Relevant features*

The idea of relevant feature selection is to reduce the 54 extracted features to a smaller set of features that are pertinent to distinguish legitimate files from malware files. The step of selecting relevant features is done by checking the difference between the values of features extracted from malicious and legitimate files. To process the data, Pandas [6], Scikit and Numpy libraries were used [7].

From the 54 features, 9 relevant features were selected and sorted according to their importance using the Extra Trees Classifier. The Trees classifier is based on an extraction of observations from the dataset and an extraction of attributes. For better interpretation, the attributes selected at the top of the tree are usually more important than the attributes selected at the end nodes of the tree because, in general, the higher subdivisions lead to greater information gain. The following figure (Figure 2) shows the selected relevant attributes and their importance:

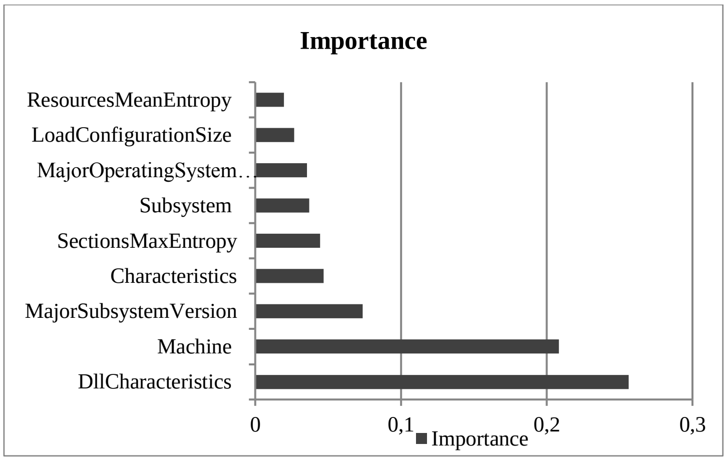

Figure 2. Relevant features and their importance

D. *Classifying*

After selecting the relevant features, the next step is classification; in this research, four algorithms were used:

Decision Tree [8], Gradient Boosting [9], SVM [10] and Random Forest [8], as these algorithms require many parameters; this experience was done with default parameters. The dataset is split into two arrays: an array named "X" that contains all values for each row except the "legitimate" column, and an array named "y" that contains only the "legitimate" value for that row. The classification algorithms compare the attribute values of "X" with the corresponding values of "y" to detect patterns in how different attribute values tend to affect the legitimacy of a file. Finally, the "X" and "y" arrays are split into two parts, a training set and a test set. The training set is provided to the classification algorithm to form a trained model. Once the model is built, it is used to classify the test set to test the accuracy of the model.

Before starting the training and testing, a timer was set using the time library in Python to track the execution time of each algorithm. The following flowchart shows the idea used in classification:

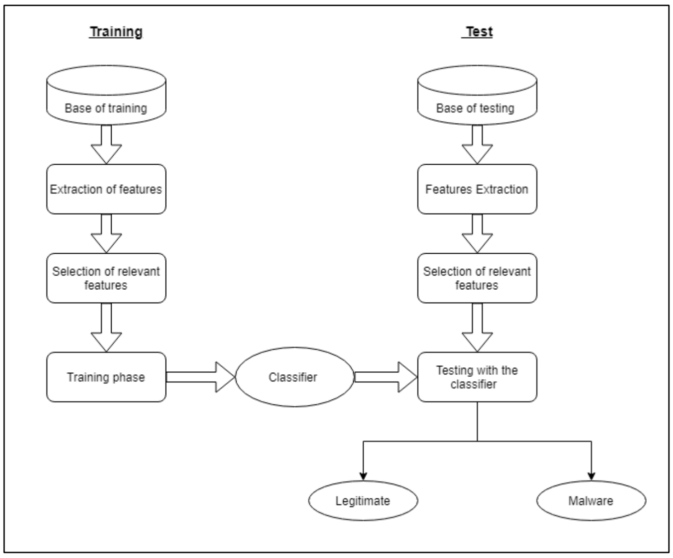

Figure 3. Training and testing flowchart

E. Studying the Efficiency of Algorithms

As mentioned earlier, the performance of each algorithm is measured by the following criteria: The running time, the accuracy, the rates of false positives and false negatives, and the AUC.

The accuracy of the model is determined by the score function from the Scikit library in Python.

The rates of false positives and false negatives are calculated using the confusion matrix [11]. Confusion matrix, also called error matrix, is a table that shows different predictions and test results and compares them with the actual values. These matrices are used in statistics, data mining, machine learning models and other artificial intelligence applications.

TABLE I. THE GENERAL FORMAT OF THE CONFUSION MATRIX

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	TP	FN
Predicted Negative (0)	FP	TN

TP stands for True Positives; TN stands for True Negatives, FP stands for False Positives, while FN refers to False Negatives. The scikit library also provides a function that returns the confusion matrix. The values used to measure efficiency are calculated as follows:

- False Positive rate (also called Fall out) is proportion of the incorrectly positive classified samples to all negative samples, it is calculated using the equation (1):

$$FPR = \frac{FP}{FP + TN} \tag{1}$$

- False Negative rate is the proportion of positives which yield negative test outcomes with the test, it is calculated using the equation (2):

$$FNR = \frac{FN}{FN + TP} \tag{2}$$

- The accuracy (also called score) is the proportion of the correctly classified samples to all samples, it is described in the equation (3):

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

In this experience, the running time is calculated from the start of the training to the calculation of the confusion matrix.

Area under ROC curve [12]: ROC curve is generally used in binary classification; It is constructed by plotting the TPR (True Positive Rate) on the Y-axis and the FRR (False Positive Rate) on the X-axis. The performance of each classifier is measured by the area under the ROC curve, where 0.5 represents random prediction and 1 represents perfect prediction.

3. Findings

A. *Algorithms Performance*

The following table shows the results of the performance criteria values for each of the tested algorithms:

TABLE II. EXPRIMENT RESULTS

Algorithm	Accuracy (%)	Running Time (in sec)	False positive Rate (%)	False Negative Rate (%)
Decision Tree	99.449412	0.723649	0.208100	4.676953
Gradient Boosting	99.323775	9.116513	0.188090	6.557377
Random Forest	99.648954	11.185557	0.108052	3.278689
SVM	99.434632	3009.639216	0.168081	5.351977

B. *Area Under Receiver Characteristic Operator Curve*

To get more insight on performance of all the algorithms in detecting the malware files, area under the ROC curve for all algorithms has been computed and plotted. The results are shown in Table 3 and ROC curves are presented in the Figure 4.

TABLE III. RESULTS OF AREA UNDER ROC CURVE FOR EACH ALGORITHM

Algorithm	**AUC**
Decision Tree	0.9896
Random Forest	0.9933
Gradient Boosting	0.9950
SVM	0.9910

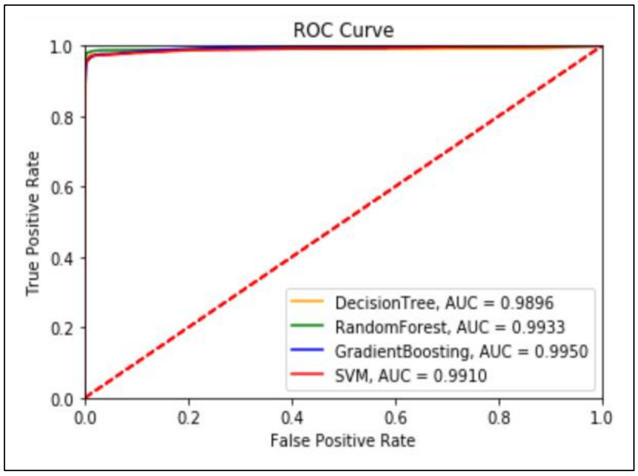

Figure 4. ROC Curve

4. Discussion

Although 54 features were used to detect malware files, not all were equally important in distinguishing malware files from legitimate files. Only 9 features proved to be relevant. This can be explained by looking at the nature of the malware binaries and the functionalities they are programmed to perform. For example, DLLCharacteristics is a relevant feature with an importance of over 0.25. This PE header was found to be relevant in identifying malware. Jinrong Bai (2014) interpreted this by explaining that most types of malware are executable images, while most parts of benign software are dynamic link libraries. therefore, the mean values of characteristics, ImageBase, and Dllcharacteristics show significant differences between malware and benign software [13].

From the experimental results, it can be inferred that all the algorithms used perform well in the accuracy measurement and achieve above 99%. While the Random Forest algorithm performed the best in terms of score and false negative rate, the Decision Tree algorithm performed the best in terms of execution time. On the other hand, the execution time of the SVM algorithm is strikingly high compared to the other algorithms, although the execution time depends on the CPU and does not give much indication of the performance in such experiments, but it still gives an idea of the computational operations per algorithm; the Gradient Boosting algorithm achieved a high false negative rate but a low false positive rate.

Overall, Random Forest was found to be the best and most accurate algorithm used in this research. These results are in line with other research on the same topic, where the 9 PE headers were also found to be relevant for malware classification. The only difference is in the importance of each feature and the accuracy of the algorithms. In many other studies, the algorithms did not achieve high accuracy while in this study the accuracy is higher. This can be explained by the fact that the dataset used in this study is much larger (more than 130000 files), which makes this study outstanding.

5. Conclusion

Malware attacks are becoming one of the biggest threats to national information security, especially in these pandemic times when cyber attacks are increasing tremendously. Malware programmers are developing new techniques every day to avoid detection. To counter these attacks, information security professionals must use new technologies to detect this malware.

In this paper, we presented a method to classify malware using PE headers. More than 130000 files were used to create the dataset for training four machine learning algorithms. The efficiency of these algorithms was studied in terms of accuracy, execution time and UAC. Random Forest showed high performance compared to the other 3 algorithms with accuracy above 99.64 percent and 0.9933 area under the ROC curve.

From the experimental results on UAC, accuracy, false positive rate, and false negative rate, it can be concluded that the proposed method can successfully detect malware with high accuracy.

REFERENCES

[1] Ashwini Mujumdar, Gayatri Masiwal, and Dr B Meshram. "Analysis of signature-based and behavior-based anti-malware approaches". In: International Journal of Advanced Research in Computer Engineering and Technology (IJARCET) 2.6 (2013), pp. 2037–2039.

[2] Matthew G Schultz et al. "Data mining methods for detection of new malicious executables". In: Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001. IEEE. 2000, pp. 38–49.

[3] Visual Studio. "Microsoft Portable Executable and Common Object File Format Specification". In: Saatavilla: https://download. microsoft. com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/pecoff_v83. docx. Haettu 5 (2021).

[4] Ethem Alpaydin. Introduction to machine learning. MIT press, 2020.

[5] Zane Markel and Michael Bilzor. "Building a machine learning classifier for malware detection". In: 2014 second workshop on anti-malware testing research (WATeR). IEEE. 2014, pp. 1–4.

[6] Fu Song and Tayssir Touili. "PoMMaDe: pushdown model-checking for malware detection". In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. 2013, pp. 607–610.

[7] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: the Journal of machine Learning research 12 (2011), pp. 2825–2830.

[8] J. Ross Quinlan. "Induction of decision trees". In: Machine learning 1.1 (1986), pp. 81–106.

[9] Hassan Chouaib. "Sélection de caractéristiques: méthodes et applications". In: Paris Descartes University: Paris, France (2011).

[10] Olivier Bousquet. "Introduction au Support Vector Machines (SVM)". In: Center mathematics applied, polytechnique school of Palaiseau (2001).

[11] Ronny Merkel et al. "Statistical detection of malicious pe-executables for fast offline analysis". In: IFIP International Conference on Communications and Multimedia Security. Springer. 2010, pp. 93–105.

[12] Kevin Markham. "ROC curves and Area Under the Curve explained". In: data school 19 (2014).

[13] Jinrong Bai, Junfeng Wang, and Guozhong Zou. "A malware detection schemebased on mining format information". In:The Scientific World Journal2014 (2014).