A New Malware Detection Model using Emerging Machine Learning Algorithms

Anik Dewanje[1] and Kakelli Anil Kumar[2]
(Corresponding author: Anik Dewanje)
Department of Computer Science and Engineering, Vellore Institute of Technology[1]
Vellore-632014, Tamil Nadu, India
Department of Analytics, School of Computer Science and Engineering, Vellore Institute of Technology[2]
Vellore-632014, Tamil Nadu, India
(Email: anikdewanje@gmail.com)
(Received Nov. 11, 2020; Revised and Accepted Dec. 19, 2020; First Online Dec. 19, 2020)

Abstract

Machine learning is a set of procedures that make the computer capable to learn without being expressly programmed. In other words, the machine-learning algorithm detects and validates the principles that support the information. With this information, the algorithm can 'consult' the properties of previously unrecognized samples. At the detection of malware, a previously unrecognized sample may be a new file to the program. Its subtle assets can be malware or benign. The set of customized terms in terms of data is called a model. Machine learning has many broad approaches that are needed in solving one approach. These methods have different capabilities and different appropriate functions. In this work, we have used advanced machine learning algorithms and use the most accurate algorithm in the proposed model to detect a file is malware or not.

Keywords: Cyber Security; Data Set; Malware Detection; Machine Learning; Prediction Model

1 Introduction

Being an emerging field of computer science, machine learning aims to enable computers to learn from data instead of explicit programming. To use PB-level data to make selections and perform what is impossible or best carried out in a certain situation, where the task for us humans is complex and time-consuming [15, 18]. Malware is an imminent threat faced by companies and users every day. Whether it's phishing emails or exploits in the entire browser, coupled with multiple evasion methods and other security vulnerabilities, today's defence systems can no longer compete with them [7, 9].

Using of Internet has increased dramatically over the past few years as today's community has been relying on worldwide communication excessively. Simultaneously, it is being utilized extensively by hackers and others with criminal intent, and with the emergence of the black market where they can buy or use malicious resources. This gives robust motivation for attackers to modify and escalate the complexity of the malicious program to reduce the possibility of success of antivirus programs which leads to the new execution of similar malware that can spread out of control, again and again [16]. According to AV-TEST report, roughly 350,000 new malicious samples are being recorded daily. This makes several problems for processing a big volume of unframed data acquired from the malware test

which creates difficulties for anti-virus companies to identify every attack and upgrade the software at the right time to restrain the penetration and spread [14].

Analyzing malware can provide incorrect data in many cases [25]. When obfuscating methods are used for analyzing the difficulty is increased further, so it will be difficult to see what kind of malware it is. The alternative for this is carrying out a rigorous analysis of malicious software's behaviour that can be a daunting job when we have to examine the increasing and growing figure of the new malicious sample. Because of that issue, it is, consequently, necessary to enhance an extensive framework where many malicious samples can be examined vigorously [22, 26].

New malware which appears every day is believed to be the most altered versions of the previous one using enlightened replication method. An enormous number of samples have been used. Having a large number of data sets contributes to the prophetic ability and dependability of the built model that gives a presentable outcome. For this work, we will develop an ml-based prediction model, which can be used as a preliminary filter system for all types of malware that may be classified from the unusual malicious sample. This provides the chance to pass over the static analysis on previously identified samples and concentrate particularly on testing the unusual sample that can increase the detection rate of anti-virus software [1].

2 Background of the Work

Users implementing machine learning can make product decisions independently. The quality of machine learning models affects performance and the state of user systems. Therefore, there are specific descriptions of machine learning-based malware detection [2,23]. The data-driven nature of this method should be emphasized. The model created relies largely on the data that it has during the training phase to understand if the feature is related to the predicted correct statistics [10].

Let's see why creating a representative data set is so important. Imagine that we collect training sets, and we ignore the fact that sometimes all files are bigger than 10 MB. They are all malware files and not for training and benign, the model will use this feature of the dataset and learn about any file that is more than 10 MB is a malware. The model will use this feature to detect. When it will apply to real-world data, this model will generate many false results. To overcome this issue, we need to add large-sized benign files to the training data set. Again, the model will not depend on incorrect data set properties. In short, we need to be able to correctly express the conditions under which the model will operate in the real world [4, 19]. It serves to collect representative data sets for the success of machine learning [8, 15].

Most of the currently used model families (such as deep neural networks) are called black-box models. The input of the black-box model is X, it will generate Y through a complex sequence of operations that is virtually undisturbed by humans. This can cause problems in real life. For instance, when a wrong alarm come up, we try to understand why this happened, we asked whether it was the training dataset or the model itself. The interpretation of the model determines the difficulty of the model. A false occurrence occurs when the algorithm accidentally replaces a benign file with a malicious libel. This work goal is to keep the level of false positivity as low or equal to zero as possible. It is not specific to machine learning applications. This is crucial because even a single false one among a million files can cause serious consequences for its users. It is a complex fact that there are many clean files in the world and they remain inhabited. To solve this issue, it is important to apply high requirements on both the training model and the matrix that will be developed during training, with a specific focus on models with low false positive (FPR) values. This is still not enough, as new benign files that previously remain invisible can sometimes be detected incorrectly. We consider this and apply the flexible design of a model that allows us to fix false positives (FPR), without completely backtracking on the model [6].

3 Problem Description

Malicious software, which is also called malware is one of the most serious problems in the cyber world today. Malicious software created by hackers is mainly polymorphic or metamorphic that can alter its code as it spreads [12, 21, 27]. Besides, the diversity and volume of their differentness undeniably undermine the efficiency of traditional techniques that often use signature-based strategies and are unable to identify previously unknown malware. Diversity of families with malware shares common patterns of behaviour based on their origin and purpose. Behavioural or acquired behavioural patterns can be utilized to identify the undefined malware sample from their identified family's behaviour using machine learning algorithms [5, 11, 24].

4 Working Model

In this work, we have used 5 different types of algorithms for predicting and comparing their results (See Figure 1). The used machine learning algorithms are Random Forest [17], Decision Tree [28], Adaboost, Gaussian Naive Bayes (GNB), Gradient Boosting [29]. To test the file of the model, the characteristics of the given file need to be extracted. The ML model is used to predict the class of a given file based on the trained model. The total dataset size is 50000 which has 50 independent variables and 1 dependable variable. In the dataset, 80% is the training set and 20% is the testing dataset.

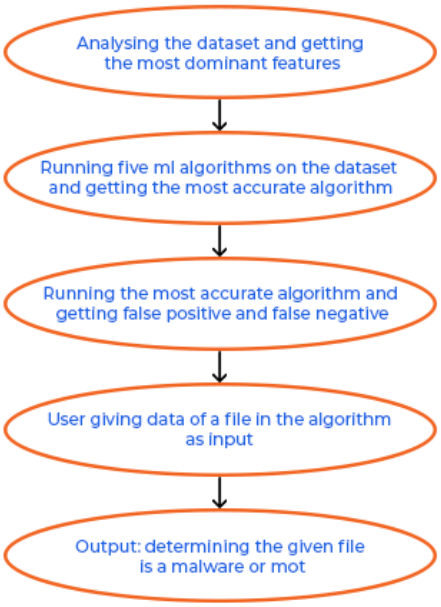

Figure 1: The designed model

5 ML Algorithms

5.1 Random Forest

Random Forest [13] is a well-known supervised learning algorithm. It builds multiple decision trees on different subsets of a given dataset and counts the common result to enhance the prediction accuracy. The relationship between the number of trees in the forest and its outcome: if the number of trees is larger, the prediction will be more accurate. For both Classification and Regression machine learning problems, Random forest can be used. It is an ensemble model which means, it combines various classifiers to resolve the complex issue and enhance the model performance. Random forest algorithm counts the prediction of every tree instead of depending on a single tree and depend on the highest predicted one, it provides the final prediction.

5.2 AdaBoost

It is an ensemble model that works particularly well with decision trees. The model key learns from past mistakes like misclassified data pinots. The learning process of Adaboost is, adding the weight to the misclassification data points as shown in (Algorithm 1).

AdaBoost algorithm predicts the new prediction by adding weight and multiplying a prediction of every tree. The final prediction made by the forest as a whole is decided by the group of decision trees with the greatest total.

Algorithm 1 Algorithm for the AdaBoost

1: Generate data points instruments. If there are 100 data points in the training set, then the weight of each starting point should be $1/100 = 0.01$.
2: Create a decision tree for every feature and analyze the outcome.
3: Evaluate the weighted error rate (e).
4: Then evaluate the weight of the decision tree, decision tree's weight = learning ratio $\times \log((1-e)/e)$.
5: Update the weight of the decision tree, so the next tree can learn from the previous decision tree's mistakes.
6: Create a new dataset.
7: Repeat Steps 2 to 6 up to the number of trees we set for training.
8: Make the final prediction using the forest of decision trees.

5.3 Gradient Boosting

Gradient Boosting is comparable to AdaBoost because they both utilize the combination of decision trees to provide the final prediction. Like the other boosting model, this model's key also learns from past mistakes. Gradient Boosting learns from mistakes (residual error), rather than updating the data point weights shown in (Algorithm 2) [20].

5.4 Decision Trees

The decision tree algorithm belongs to the supervised learning algorithm's family. But the difference between this algorithm and other supervised algorithms is, this algorithm can also solve regression and classification problems. The main purpose of using this algorithm is to develop a prediction model that can predict the label or amount of object variance by reading fundamental decision regulations derived

Algorithm 2 Algorithm of Gradient Boosting

1: Evaluate the target label means.
2: Count the remainders.
3: Create a decision tree.
4: Guess the target label using all the trees included.
5: Now calculates the residual of the decision tree and save residual errors.
6: Redo the steps 3 to 5 up to the number of trees we have set for training.
7: Once you have been trained, make the final prediction using every tree's target value.

from the previous training dataset. In this algorithm, we start at the root of the tree for predicting a record's class label. Based on the comparison between the values of the root attribute and the record attribute, we follow the branch depend on that value and proceed to the following node.

5.5 Gaussian Naive Bayes (GNB)

The Naïve Bayes algorithm is a Bayes Theorem's application-based, collection of simple probabilistic classifiers. This algorithm considers every feature as an independent variable. In supervised learning, this classifier can be trained well and can use in complex real-world problems. The main superiority of this classifier is, it can use without having a big amount of training dataset. It can make classification with a small dataset [3].

The Gaussian naive Bayes (GNB) is a classifier of the Naïve Bayes family in the sense that it uses Gaussian distribution for estimating the data distribution. For instance, assume that ith attribute is continuous and the mean of the ith attribute is $\mu_{c,i}$ and variance are represented by $\sigma^2_{c,i}$, c is the given class label. Therefore, the probability of the input value, x_i can be calculated by Equation (1) given below also called the normal distribution.

$$p\left(x_i \mid c\right) = \frac{1}{\sqrt{2\pi\sigma^2_{c,i}}} e^{\left(-\frac{\left(x_i-\mu_{c,i}\right)^2}{2\sigma^2_{c,i}}\right)} \tag{1}$$

6 Implementation

6.1 Data Pre-processing

Data that is collected from the real-world is often noisy, missing values, and remains in an unusable format that cannot be used in the machine learning model directly. It is a necessary task to refine the data and make it useable for the machine learning model, which in turn enhances the precision and efficiency of the machine learning model. It involves the following steps shown in (Algorithm 3).

At last finding out the important features form the sample to perform the workflow. Best algorithm with max accuracy is Random Forest and Gaussian Naive Bayes (GNB) has the least accuracy. The main cause for Naive Bayes' has the least accuracy rate is that the notion of conditional independence among the attributes in this classifier is weak and is rarely suited to real problems except in cases where attributes are extracted from independent systems. Implicitly, Naive Bayes considers that all attributes are completely independent. It's almost impossible for us to find a set of completely independent predictors in real life. Besides, if a component has not been identified in the training data set and which is available on a phase variable, the algorithm will assume a probability of 0 (zero) and cannot make a prediction, which is commonly known as zero frequency. Even for Gaussian equations, Naive

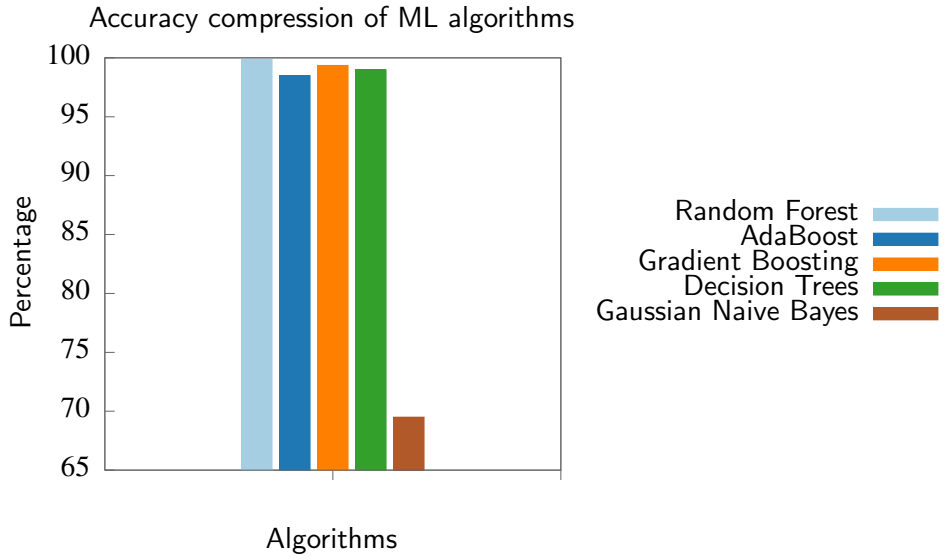

Figure 2: Accuracy compression of ML algorithms

Algorithm 3 Algorithm for data Pre-processing

1: Obtaining the dataset
2: Importing libraries
3: Import data sets
4: Find missing data
5: Coding of categorical data
6: Division of training data set and test set
7: Feature scaling

Bayes carries the weakness of the conditional independence of the attributes. On the other hand, the random forest is a better algorithm to produce a forecasting model, for both classification and regression problems. The default hyperparameters feature of this model gives excellent results and the system works well to prevent overfitting. Furthermore, a good indicator of the importance is that it provides facilities.

6.2 Algorithm Evaluation

Evaluating the random forest algorithm as shown in (Algorithm 4).

6.3 Final Test

We take two files. VirusPro.exe and AdobePDF.exe and put the values of their features (Machine, SizeOfOptionalHeader, Characteristics, ImageBase, SectionsMeanEntropy, SectionsMaxEntropy, MajorSubsystemVersion, Subsystem, DllCharacteristics). After analyzing the features our machine learning model classifies the file as malware or not malware and the performance of these ML algorithms have shown in Figure 2.

Algorithm 4 Random forest algorithm for evaluation

1: Choose a random sample "K" from the given "m" features, where k ¡¡ m.
2: Calculate the node "d" from the selected data features.
3: According to the best partition point, divide the node into sub-nodes.
4: For creating the forest, repeat the first three steps for "n" times, where "n" is the number of trees.
5: The pseudocode for prediction by using the decision tree algorithm is given below:
6: Count the prediction of every randomly created decision tree.
7: The highest predicted object will be the final prediction.

7 Conclusions

In this work, our prime focus was to develop a machine learning model which can identify malicious samples as truthfully as possible, with having a zero false-positive rate. To achieve it, we have preferred different machine learning algorithms and got the highest accuracy rate for the Random forest algorithm. After that, we used the random forest algorithm for malware detection and our proposed model has successfully detected the malware sample. We were almost near to our target, even though not having a zero false-positive rate still. Our model false-positive rate was 0.48992% and the false-negative rate was 1.008782%. So, to be a segment of a competitive business outcome, various alternative strategies must be added to this model. According to our observation, machine learning-based malware detection model is also the best technology along with the conventional detection techniques like anti-virus. It is very important to note that to enhance the malware detection rate, the machine learning models are highly essential and also result in better accuracy.

References

[1] Mouhammd Al-Kasassbeh, Safaa Mohammed, Mohammad Alauthman, and Ammar Almomani, "Feature selection using a machine learning to classify a malware," in *Handbook of Computer Networks and Cyber Security*, pp. 889–904, Springer, 2020.

[2] Abdullah A. Al-khatib and Waleed A. Hammood, "Mobile malware and defending systems: Comparison study," *International Journal of Electronics and Information Engineering*, vol. 6, no. 2, pp. 116–123, 2017.

[3] K. Dharmarajan, A. S. Arunachalam, S. Vaishnavi Sree, "Malware detection and classification using random forest and adaboost algorithms," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, pp. 2863–2868, 2019.

[4] Ahmed Bensaoud, Nawaf Abudawaood, and Jugal Kalita, "Classifying malware images with convolutional neural network models," *arXiv preprint arXiv:2010.16108*, 2020.

[5] William Blanc, Lina G Hashem, Karim O Elish, and MJ Hussain Almohri, "Identifying android malware families using android-oriented metrics," in *2019 IEEE International Conference on Big Data (Big Data'19)*, pp. 4708–4713, IEEE, 2019.

[6] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," *Computers & Security*, vol. 73, pp. 326–344, 2018.

[7] Tushaar Gangavarapu, C. D. Jaidhar, and Bhabesh Chanduka, "Applicability of machine learning in spam and phishing email filtering: Review and approaches," *Artificial Intelligence Review*, pp. 1–63, 2020.

[8] Daniel Gibert, Carles Mateu, and Jordi Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, p. 102526, 2020.

[9] Nikhil Govil, Kunal Agarwal, Ashi Bansal, and Astha Varshney, "A machine learning based spam detection mechanism," in *Fourth International Conference on Computing Methodologies and Communication (ICCMC'20)*, pp. 954–957, IEEE, 2020.
[10] Anand Handa, Ashu Sharma, and Sandeep K Shukla, "Machine learning in cybersecurity: A review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, pp. e1306, 2019.
[11] Manabu Hirano and Ryotaro Kobayashi, "Machine learning based ransomware detection using storage access patterns obtained from live-forensic hypervisor," in *Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS'19)*, pp. 1–6, IEEE, 2019.
[12] Li-Chin Huang, Chun-Hsien Chang, and Min-Shiang Hwang, "Research on malware detection and classification based on artificial intelligence," *International Journal of Network Security*, vol. 22, no. 5, pp. 717–727, 2020.
[13] Areeba Irshad, Ritesh Maurya, Malay Kishore Dutta, Radim Burget, and Vaclav Uher, "Feature optimization for run time analysis of malware in windows operating system using machine learning approach," in *42nd International Conference on Telecommunications and Signal Processing (TSP'19)*, pp. 255–260, IEEE, 2019.
[14] Nitesh Kumar, Subhasis Mukhopadhyay, Mugdha Gupta, Anand Handa, and Sandeep K Shukla, "Malware classification using early stage behavioral analysis," in *14th Asia Joint Conference on Information Security (AsiaJCIS'19)*, pp. 16–23, IEEE, 2019.
[15] Ori Or-Meir, Nir Nissim, Yuval Elovici, and Lior Rokach, "Dynamic malware analysis in the modern era state of the art survey," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–48, 2019.
[16] R Mohanasundaram P HarshaLatha, "Classification of malware detection using machine learning algorithms: A survey," *International Journal of Scientific & Technology Research*, vol. 9, no. 2, pp. 1796–1802, 2020.
[17] S Abijah Roseline and S Geetha, "Intelligent malware detection using oblique random forest paradigm," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI'18)*, pp. 330–336, IEEE, 2018.
[18] Asaf Shabtai, Yuval Fledel, and Yuval Elovici, "Automated static code analysis for classifying android applications using machine learning," in *International Conference on Computational Intelligence and Security*, pp. 329–333, IEEE, 2010.
[19] Sanjay Sharma, C Rama Krishna, and Sanjay K Sahay, "Detection of advanced malware by machine learning techniques," in *Soft Computing: Theories and Applications*, pp. 333–342, Springer, 2019.
[20] Latika Singh and Markus Hofmann, "Dynamic behavior analysis of android applications for malware detection," in *International Conference on Intelligent Communication and Computational Techniques (ICCT'17)*, pp. 1–7, IEEE, 2017.
[21] Ming-Yang Su, Jer-Yuan Chang, and Kek-Tung Fung, "Android malware detection approaches in combination with static and dynamic features.," *International Journal of Network Security*, vol. 21, no. 6, pp. 1031–1041, 2019.
[22] Zheng-Zhi Tang, Xuewen Zeng, Zhichuan Guo, and Mangu Song, "Malware traffic classification based on recurrence quantification analysis.," *International Journal of Network Security*, vol. 22, no. 3, pp. 449–459, 2020.
[23] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.
[24] Matúš Uchnár and Peter Fecil'ak, "Behavioral malware analysis algorithm comparison," in *IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI'19)*, pp. 397–400, IEEE, 2019.
[25] R Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, and Sitalakshmi Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019.

[26] Di Xue, Jingmei Li, Tu Lv, Weifei Wu, and Jiaxiang Wang, "Malware classification using probability scoring and machine learning," *IEEE Access*, vol. 7, pp. 91641–91656, 2019.
[27] Hanqing Zhang, Senlin Luo, Yifei Zhang, and Limin Pan, "An efficient android malware detection system based on method-level behavioral semantic analysis," *IEEE Access*, vol. 7, pp. 69246–69256, 2019.
[28] Boyou Zhou, Anmol Gupta, Rasoul Jahanshahi, Manuel Egele, and Ajay Joshi, "Hardware performance counters can detect malware: Myth or fact?," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 457–468, 2018.
[29] Qingguo Zhou, Fang Feng, Zebang Shen, Rui Zhou, Meng-Yen Hsieh, and Kuan-Ching Li, "A novel approach for mobile malware classification and detection in android systems," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 3529–3552, 2019.

Biography

Anik Dewanje is currently pursuing a bachelor's degree program in computer science and engineering in specialization with information security at Vellore Institute of Technology, Vellore, India.

Dr Kakelli Anil Kumar is currently working as an associate professor in the school of computer science and engineering at Vellore Institute of Technology, Vellore, India.