Advances and Applications in Mathematical Sciences
Volume 21, Issue 7, May 2022, Pages 4183-4196
© 2022 Mili Publications, India

STATIC MALWARE DETECTION AND ANALYSIS USING MACHINE LEARNING METHODS

KARTIK MALIK[1], MANISH KUMAR[1], MEHUL KUMAR SONY[1], RADHA MUKHRAIYA[1], PALAK GIRDHAR[2*] and BHARTI SHARMA[2*]

Bhagwan Parshuram Institute of Technology
Delhi, India

Maharaja Surajmal Institute of Technology
Delhi, India
E-mail: palak.batra1985@gmail.com
bhartisharma@msit.in

Abstract

Detecting malware by a computer program is usually eliminated with the help of an anti-virus program that considers each and every program in the structure of known malware. One more method we can segregate malware is with the help of using machine learning algorithms. Known features of malware programs can be maneuverer to train the model in order to determine if a given program is a malware program. With this being stated, this paper makes use of PE file format along with machine learning statistics to determine whether a particular program is malicious or not.

I. Introduction

A cyber-attack is an unsolicited attempt to pilfer, expose, manipulate, incapacitate or damage information through uncertified access to computer systems. A major section of crime-inspired attackers looks for monetary profits through illicit activities like money laundering, data leaks, or business disruption [16]. With the constant boom in the IT sector, cyber security becomes a crucial topic. Various firms that are working in this anti-malware industry have been suggesting solutions to prevent cyber-attacks. The speed, volume, and complexity of malware have created new challenges for the anti-malware community.

2020 Mathematics Subject Classification: 68T01.
Keywords: Malware Detection; Malware Analysis; PE file format.
Received September 30, 2021; Accepted November 25, 2021

The task of discernment of malware by traditional techniques becomes even more strenuous as all malware applications often have more than one polymorphic layer to escape detection or use specific methods to automatically evolve themselves into an advanced version at short intervals. There are basically two techniques of malware analysis and detection: Static Malware Analysis and Dynamic Malware Analysis [12]. In dynamic analysis, a given file is executed in a sandbox environment whereas static analysis involves inspecting the given malware sample without actually running it. Anti-Malware software takes the dynamic approach [17] [18]. However, static analysis is a more detailed approach and may also prove more cost-efficient. This is the reason why machine learning algorithms are being inculcated to facilitate this process of malware detection more accurately and efficiently [13]. The main task is to distinguish malware present statically utilizing AI calculations with the assistance of the compact executable (PE file format).

The paper is organized as follows: section II discusses the related work. Section III discusses the background needed to develop the methodology. Section IV and V presents the proposed methodology and experimental results. Section VI summarizes the results and concludes.

II. Related Work

The authors in [1] discussed about a way to find a hardware-assisted computer program to monitor and disassemble memory access patterns [19]. Instead of a single model that categorizes malicious and 'healthy' files, here it focuses on studying one model for each application that separates its malicious and legal execution. The designed framework achieved a 99.0% acquisition rate with less than 5% falsehoods and performs better than previous proposals present in the literature for the detection of a malware program. What stands out is the reliability of machine learning rather than human comprehension. Authors of [2] focused on the various types of malware that exist in programs and their acquisitions using computer malware detectors. This malware program detector receives two types of inputs: malicious behavior information and the system under test. If the malware detector has an idea about malicious behavior, comparison is made with the test code and declare the code malicious or not. There are various ways to detect malware of these three different types: (i) Based on signature (detector can detect malware in infected file due to small patterns or signature on the malware acquisition website). (ii) On the basis of abnormalities (deviations from normal behavior). (iii) Based on specifics (focusing on the need for a program in the first phase is the implementation of specific rules that specify the effective behavior of any system that can be displayed in a protected system). All in all, the paper measures the critical aspects of a malware program, as well as the malicious code in detail.

The authors of [4] discuss the current analysis of the malware and methods of recovery. It is said that most of the techniques for finding a computer-friendly PE file system are contingent on machine learning algorithms. 56% of the mentioned methods (Clustering DBSCAN algorithm, Multilayer dependency chain, multiple kernel learning, machine learning, Jaccard similarity, K-means algorithm, Unsupervised-prototype-based clustering, Unsupervised: Clustering with Jaccard similarity, Regression techniques, Monte Carlo tree search, random sampling, SMV, classification algorithm, K-means algorithm, SVM classification) used supervised learning-based algorithms, 26% they use algorithms for unsupervised learning and 18% of lessons combined with both supervised and supervised learning methods. Speaking of features, with 97.1% acquisition accuracy, Opcodes are the most widely used features. Other important options are byte sequences and call acquisition strategies based on the API/system. Opcode-based acquisition methods offer high accuracy.

In [8], discussed how signature-based anti-malware tools are ineffectual when it comes to the detection of metamorphic malware. Here a machine learning approach has been talked about that focuses on the frequency of opcode occurrence. The approach calculates the frequency of opcode occurrence for all malware and benign programs to ameliorate the malware detection accuracy of unknown malware.

Authors in [9] introduces and discusses a framework for the detection and classification of various files (example: .exe, .pdf, .php) under the categories: malicious and benign, authors have used 2 level classifier, first is macro (for malware detection) and second is micro (for classification of malware type files such as Spyware, Trojan, adware etc.). It makes use of a cuckoo sandbox. Cuckoo sandbox model executes the sample files in a virtual environment and generates static and dynamic reports. A feature extraction module has been created based on the report provided by cuckoo model. Testing and Training sets contain 40% and 60% of malware samples respectively. We have observed that results for detection and classification problem using these models which developed for each of the machine learning algorithms: (decision tree, SMO and random forest)-Detection rates with an accuracy of 100%, 99%, and 97% were achieved using the decision tree, SMO, and random tree respectively. Accuracy of decision tree is 100% due to it's fact that the decision is made based on the feature cuckoo is malware type. Authors in [7] focused on reading MS Windows Portable Executable (PE) headers, instead of reading the whole PE file it just reads the PE header and ignores other details of the Executable. Using web spider and header-parser, using header-parser reads the headers of each file and compares them to find the most significant difference between benign and malicious files. Five major contributing features of PE header were found. "Size of Initialized Data" in most of the malicious executables was found to be 0. Other Important features were found to be Major image version, Checksum, DLL Characteristics, Section Name, Initialized Data Size. Though analyzing PE header is much faster processing than reading whole file data but this technique fails to detect some of the malware as some file details like max resource entropy and stack size entropy are not taken into consideration.

III. Background

3.1 PE File Format

The information required by the Windows OS Uploader to manage usable encrypted code is provided in PE format. These include powerful library links to links, API deployments, resource management data, import tables, and TLS data. The system gets to know about how much memory has to be set aside in order to map achievable memory through the column in the PE head. Non-mapped data is placed at the end of the file, past which sections will be mapped. Section tables, DOS Header, DOS Stub, Image optional header, PE File Header, DOS head, Data Dictionaries, and Section constitute PE file data structures.

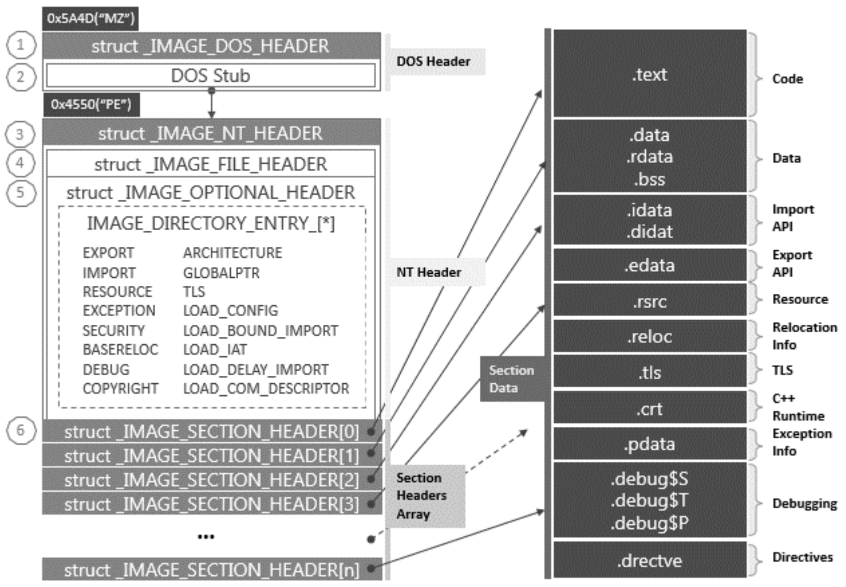


Figure 1. PE File Format.

3.2 Dataset Used

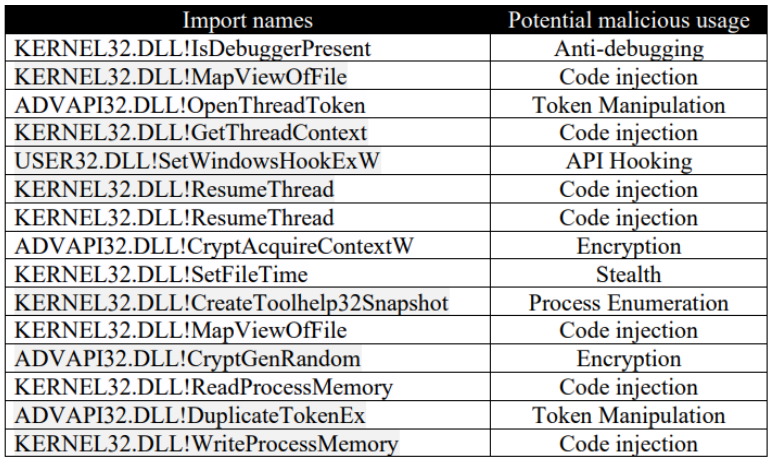
The used dataset is a hybrid of the Ember dataset [3] and the unprocessed data that was acquired from the malware security partner of Meraz'18-Annual Techno Cultural festival of IIT Bhilai. This unprocessed data comprises malicious and legitimate files. The Ember Database is an open-source data set used to train static PE machine learning Modes. The dataset consists of highlights separated from 1.1M double records: 200K test tests (100K kind, 100K noxious) and 900K preparing tests (300K favorable, 300K malevolent, 300K unlabeled).

3.3 Checksum

A checksum is a series of letters and numerals used to examine whether a data or file has been changed during storage or imparting. Checksum is often accompanied by software downloaded from the network to ensure that the file or files were not compromised during movement. [4]

3.4 PE Imports

Portable executable can bring in code from other Pes. For this to happen, the PE file name and functions are specified. It is important to inspect the imports to get a consistent image of what the PE is doing. Few imported functions are representative of potential malicious operations such as crypto API is used for unpacking the encryption and APIs used for anti-debugging. Few examples of potential infected imports are:



Import names	Potential malicious usage
KERNEL32.DLL!IsDebuggerPresent	Anti-debugging
KERNEL32.DLL!MapViewOfFile	Code injection
ADVAPI32.DLL!OpenThreadToken	Token Manipulation
KERNEL32.DLL!GetThreadContext	Code injection
USER32.DLL!SetWindowsHookExW	API Hooking
KERNEL32.DLL!ResumeThread	Code injection
KERNEL32.DLL!ResumeThread	Code injection
ADVAPI32.DLL!CryptAcquireContextW	Encryption
KERNEL32.DLL!SetFileTime	Stealth
KERNEL32.DLL!CreateToolhelp32Snapshot	Process Enumeration
KERNEL32.DLL!MapViewOfFile	Code injection
ADVAPI32.DLL!CryptGenRandom	Encryption
KERNEL32.DLL!ReadProcessMemory	Code injection
ADVAPI32.DLL!DuplicateTokenEx	Token Manipulation
KERNEL32.DLL!WriteProcessMemory	Code injection

Figure 2. Potential Malware Imports.

3.5 Feature Selection

In order to minimize the amplitude of our dataset and revamp the performance of the machine learning algorithm, we made use of the FCBF (Fast Correlation-based Feature Selection method). The FCBF feature selection method starts with a full set of adjectives with equal uncertainty in order to analyze the correlation between features and remove unwanted features [20]. The FCBF has two components:

Select the set of Machine Learning features Cyber Security *F* features associated with the target class. Remove obsolete features and save highlights only in the Cyber Security *F*.

The FCBF method stops if there are no unnecessary components left to be removed. FCBF operates, generally, quicker than other subset selection methods, and is measurable and stand-alone of the learning algorithm. As a result, selecting a feature needs to be done only once. After applying the FCBF method, each PE file is represented by 55 relevant, non-confusing features.

(a) Selected Features

- feature Subsystem (0.129329)
- feature Size of Optional Header (0.123128)
- feature ID (0.106538)
- feature Sections Min Entropy (0.066363)
- feature Load Configuration Size (0.062270)
- feature Check Sum (0.061694)
- feature Minor Image Version (0.048122)
- feature Base of Data (0.047889)
- feature Resources Mean Entropy (0.041872)
- feature Resources Min Entropy (0.041053)
- feature Machine (0.037433)
- feature File Alignment (0.030600)
- feature Sections Mean Entropy (0.021315)

3.6 Performance Measures Used

(a) False Positive

When a positive value is predicted by the model for the input but actual value is found to be positive [9].

False positive rate = (False Positive Value) / (True Negative + false positive values)

(b) False Negative

When a negative value is predicted by the model but actual value is found to be positive [7].

False negative rate = (False negative Value) / (True Positive + false negative values)

3.7 Machine Learning Algorithms Used for Analysis

(a) Decision Trees

Decision Tree is most widely used algorithm for regression and classification tasks. A decision tree is a tree like graph where at each node we ask a question at each attribute [10]. Edges of decision tree represents answer to each node and leaf node represents actual label of the node class. This process is recursively repeated at each subtree at the new nodes.

(b) Random Forest

Random forest generates number of decision trees on various subsets of datasets and takes in account mean to improve predict accuracy of that dataset [8].

(c) Guassian Naïve Bayes

The Gaussian Naive bayes algorithm learns the possibilities of an object with specific group or category characteristics. The naïve bayes classifier is an algorithm that classifies objects based on the Baye's perspective. In mindless split bars we assume that features are independent of a particular category. Although Independence of features is often a misnomer, but in practice the Naive Bayes do well with complex class dividers [6]. We know that naive bayes work well with almost any functionality and feature dependence and achieve their best performance in two opposite situations: completely independent features and performance-dependent features.

Issues in Gaussian Naive Bayes it will not be reliable if there are slightly differences in the attribute distributions compared to the training dataset [14].

(d) Ada Boost

Ada Boosting a technique in machine learning that is based on the ideology to create a extremely accurate prediction rule by combining so many relatively weak and irregular rules. Our approach uses AdaBoost to describe or understand it as a learning process, by comparing both the strengths and weaknesses of different methods.

Issues in AdaBoost is that it needs a quality dataset. So, it is important that Noisy Data and outliers have to be avoided before adopting an AdaBoost Algorithm [13] [21].

(e) Gradient Boosting

Gradient boosting algorithm is used when we have to reduce bias error [5]. A gradient boosting classifier is a cluster of machine learning algorithms that clubs several weak learning models to generate a strong predictive model. We mostly use Decision Tree for Gradient Boosting [11]. We use Gradient Boosting Algorithm for Regression and Classification Problem. This method is heavily dependent on outliers.

Issues-One of the disadvantages of boosting is that it is sensitive to outliers and also gradient boosting is almost impossible to scale up [9].

IV. Methodology

The above-mentioned dataset of PE files was used and important features were extracted from the same using variance threshold. Out of 54 features, 13 features with high variance thresholds were selected. K fold validation was used to split the data in the ratio of 80:20. Numerous machine learning algorithms were implemented on the processed data. The algorithms used are gradient boost, Naive Bayes, Adaboost, random forest and decision tree. The accuracies, false-negative, and false-positive rates were calculated and contrasted. The model with the superlative accuracy was stored and used to run further detection.

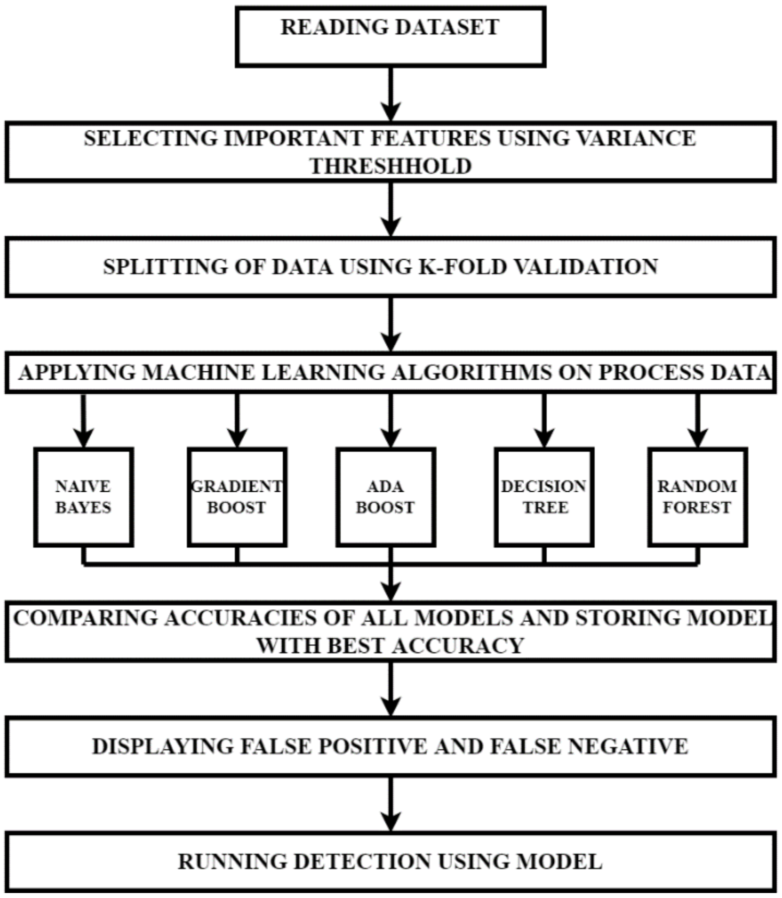

READING DATASET
SELECTING IMPORTANT FEATURES USING VARIANCE THRESHHOLD
SPLITTING OF DATA USING K-FOLD VALIDATION
APPLYING MACHINE LEARNING ALGORITHMS ON PROCESS DATA
NAIVE BAYES
GRADIENT BOOST
ADA BOOST
DECISION TREE
RANDOM FOREST
COMPARING ACCURACIES OF ALL MODELS AND STORING MODEL WITH BEST ACCURACY
DISPLAYING FALSE POSITIVE AND FALSE NEGATIVE
RUNNING DETECTION USING MODEL

Figure 3. Work Flow of Propsoed Methodology.

V. Result

We calculated the accuracies of all the applied algorithms, where random forest outperformed all the fellow algorithms. The accuracy was 99.970879%, with false positive rate of 0.020665% and false negative rate of 0.058162%. Random forest provided us best accuracy as it is capable of handling large datasets with high dimensionality and also prevents overfitting issues.

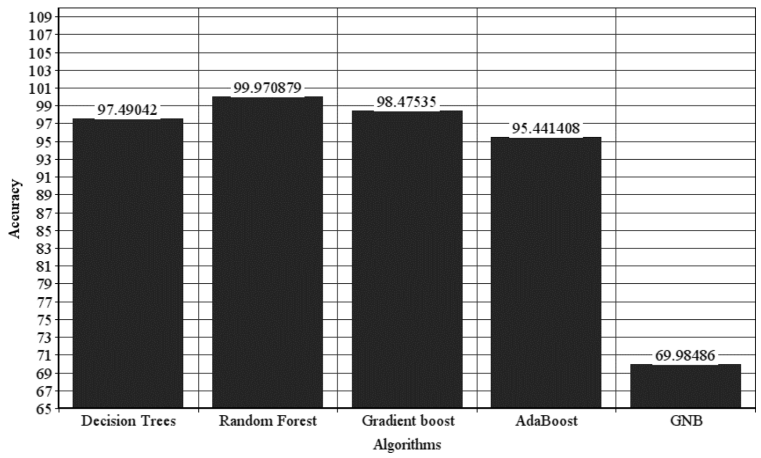


Figure 4. Comparing Accuracies of various algorithms

Table I. Other measuring parameters.

Algorithm	False Positive Rate	False Negative Rate
Decision Trees	1.530022%	6.104094%
Random Forest	0.020665%	0.058162%
Gradient boost	0.917912%	3.254330%
AdaBoost	1.225704%	12.145145%
GNB	0.000000%	99.987866%

VI. Conclusion

In this work, we have presented a methodology for comparing the performance and robustness of various machine learning algorithms for malware detection. Machine learning algorithms have found better approach to the existing anti-virus software in terms of detection. Anti-virus software's are built to detect only known malicious files, but unable to generalize or detect on new threats. Machine Learning algorithms gives a more generalize solution towards malware detection. The performance of various machine learning algorithms is analyzed in this work. Different algorithms work well in different situations. From the analysis and results, it is observed that random forest is the best model for our cause.

The above-mentioned results portray that maximum entropy of all the PE section entropies would be the most compliant method to discern benign PE files and malicious PE files. This aligns with our presupposition that says that with PE files, high entropy isn't really very common [2]. A significant role is played by the signature status of the PE file. To be precise, if the PE file doesn't have a signature assigned to it or if the assigned signature isn't authentic, there are bright chances that the given PE file is infected.

Talking about the paramount attributes, the next in line would be section names and permissions. A malware program often uses compression processes to move without the detection of viral marks. This results in irregular section names and illegal write approvals.

We also recognize that cynical import stages have contributed to the accuracy of the model. In these features, heterogeneous API functions were classified into certain categories (evasion, encryption, remote allocation) and were grouped according to that. In all these respective groups, there may be a few restrictions on different DLLs. This has allowed us to understand infected activities and avoided overfitting at the same time.

References

[1] P. Agrawal and B. Trivedi, Analysis of android malware scanning tools, International Journal of Computer Sciences and Engineering 7(3) (2014), 807-810.

[2] A. Al-Marghilani, Comprehensive Analysis of IoT Malware Evasion Techniques. Engineering, Technology and Applied Science Research 11(4) (2021), 7495-7500.

[3] H. Anderson and P. Roth, EMBER: An Open Dataset for Training Static PE Malware Machine Learning 2022 Models, [online] arXiv.org. Available at: <https://arxiv.org/abs/1804.046371

[4] Bearden Ruth and Dan Chai-Tien Lo, Automated Microsoft office macro malware detection using machine learning, IEEE International Conference on Big Data (Big Data), IEEE, 2017.

[5] C. Bentéjac, A. Csörgő and G. Martínez-Muñoz, A comparative analysis of gradient boosting algorithms, Artificial Intelligence Review 54(3) (2021), 1937-1967.

[6] Z. J. Bi, Y. Q. Han, C. Q. Huang and M. Wang, (2019, July), Gaussian naive Bayesian data classification model based on clustering algorithm, In 2019 International Conference on Modeling, Analysis, Simulation Technologies and Applications (MASTA 2019) 396-400, Atlantis Press.

[7] R. Couronné, P. Probst and A. L. Boulesteix, Random forest versus logistic regression: a large-scale benchmark experiment, BMC bioinformatics 19(1) (2018), 1-14.

[8] Denisko, Danielle and Michael M. Hoffman, Classification and interaction in random forests, Proceedings of the National Academy of Sciences 115(8) (2018), 1690-1692.

[9] Laskari Naveen Kumar and Suresh Kumar Sanampudi, TWINA at SemEval-2017 Task 4: Twitter sentiment analysis with ensemble gradient boost tree classifier, Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017.

[10] Mas' ud, Mohd Zaki, et al., Analysis of features selection and machine learning classifier in android malware detection, International Conference on Information Science and Applications (ICISA), IEEE, 2014.

[11] J. Mateo, J. M. Rius-Peris, A. I. Marana-Perez, A. Valiente-Armero and A. M. Torres, Extreme gradient boosting machine learning method for predicting medical treatment in patients with acute bronchiolitis, Biocy bernetics and Biomedical Engineering 41(2) (2021), 792-801.

[12] S. Megira, A. R. Pangesti and F. W. Wibowo, Malware analysis and detection using reverse engineering technique, In Journal of Physics: Conference Series IOP Publishing 1140 012042 (1) (2018).

[13] Mokoena, Tebogo and Tranos Zuva, Malware analysis and detection in enterprise systems, IEEE International Symposium on Parallel and Distributed Processing with Applications and IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), IEEE, 2017.

[14] M. Ontivero-Ortega, A. Lage-Castellanos, G. Valente, R. Goebel and M. Valdes-Sosa, Fast Gaussian Naïve Bayes for searchlight classification analysis Neuroimage 163 (2017), 471-479.

[15] Patel H. Harsh and Purvi Prajapati, Study and analysis of decision tree based classification algorithms, International Journal of Computer Sciences and Engineering 6(10) (2018), 74-78.

[16] Sethi Kamalakanta, et al., A novel malware analysis framework for malware detection and classification using machine learning approach, Proceedings of the 19th International Conference on Distributed Computing and Networking, 2018.

[17] C. Gupta, G. Chawla, K. Rawlley, K. Bisht and M. Sharma, Senti_ALSTM: Sentiment Analysis of Movie Reviews Using Attention-Based-LSTM, In Proceedings of 3rd International Conference on Computing Informatics and Networks: ICCIN 2020 (p211), Springer Nature.

[18] Charu Gupta, Prateek Agrawal, Rohan Ahuja, Kunal Vats, Chirag Pahuja and Tanuj Ahuja, Pragmatic Analysis of Classification Techniques based on Hyperparameter Tuning for Sentiment Analysis, International Semantic Intelligence Conference (ISIC'21), Delhi 459 (2021), 453-459.

[19] K. Goel, C. Gupta, R. Rawal, P. Agrawal and V. Madaan, FaD-CODS Fake News Detection on COVID-19 Using Description Logics and Semantic Reasoning, International Journal of Information Technology and Web Engineering (IJITWE) 16(3) (2021), 1-20.

[20] Sewak, Mohit, Sanjay K. Sahay and Hemant Rathore, Comparison of deep learning and the classical machine learning algorithm for the malware detection, 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2018.

[21] Palak Girdhar and Deepali Virmani, An Analysis of Feature Selection Method in Mobile Malware Detection, International Journal of Engineering Applied Sciences and Technology ISSN No. 2455-2143, 3(3) (2018), 56-61.

[22] Xu Zhixing, et al., Malware detection using machine learning based analysis of virtual memory access patterns, Design, Automation and Test in Europe Conference and Exhibition (DATE), 2017, IEEE, 2017.

[23] Ye, Yanfang, et al., Combining file content and file relations for cloud based malware detection, Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. 2011.

[24] Y. Zhang, M. Ni, C. Zhang, S. Liang, S. Fang, R. Li and Z. Tan, Research and application of AdaBoost algorithm based on SVM, In 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC) (2019), 662-666.