Performance of Malware Detection Classifier Using Genetic Programming in Feature Selection

Heba Al-Harahsheh, Mohammad Al-Shraideh and Saleh Al-Sharaeh.
King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
E-mail: Heba.moh.h@gmail.com, mshridah@ju.edu.jo and ssharaeh@ju.edu.jo

Keywords: malware detection, machine learning, feature selection, classifier, genetic programming

Received: September 11, 2021

The term "malicious software," which is commonly referred to as malware, describes malicious software that affects or harms computers, servers, or networks. While the numbers and complexity of malware have rapidly increased, developing a malware detection system is required to detect malware in the world of cybersecurity and test the behavior of its new features. While traditional techniques provide less efficiency in detecting new malware, machine learning techniques are used to achieve rapid malware detection in an intelligent way to improve detection performance, as malware and its application in the industry are constantly increasing. In this study, we developed a malware detection model by detecting malware using machine learning classifiers, after passing a new feature selection technique using genetic programming. We also compared the performance of all classifiers using the most recent feature selection techniques. Results show that Random Forest, Random Forest (4), and Random Tree give the best value in all experiments, while Hoeffding Tree and Decision Stump give lower values for F1-score and accuracy in all experiments. The feature selection method that proposed GPMP gives a better value than Filter-based with little differences. The accuracy and F1-score have the values of 0.881066 and 0.867546 for GPMP, and the values of 0.877624 and 0.862894 for Filter-based, respectively. The experimental results reveal that GPMP used fewer features than Filter-based, and this affected the computation and complexity of the model.

Povzetek: Analizirane so bile metode strojnega učenja za povečanje uspešnosti odkrivanja zlonamerne programske opreme.

1 Introduction

Nowadays, the problem of cybersecurity is growing due to the fact that all electronic devices are connected to the Internet. In addition, cybersecurity affects our daily life and the infrastructure of all fields because of the high connectivity between millions of hosts over the Internet.

Malware is considered eligible to modify the target device or application in order to gain full control of the unauthorized access, and the device can have access to other vulnerable devices to steal data.

The main reason of cyber-attack is malware. Accordingly, a malware detection technology must be developed to improve the legacy technology of the industrial security software used for detection. According to Kaspersky's research done in 2020, detecting new malicious files is increased by a rate of 5.2% every day [1].

Therefore, distinguishing between benign and malicious files is the most cybersecurity challenging task, which is used to detect suspicious files with higher accuracy and less time and cost. There are no highly efficient detection methods applied in the traditional methods because malware spreads very quickly on the network. Accordingly, most researchers try to use machine learning to get the best detection accuracy and reflect it in the new technologies or tools designed for malware detection and network Intrusion [2] [3] [4].

In this paper, we propose a new model using feature selection method and genetic programming that are used in a set of parallel classifiers for a more accurate model to detect malware at the lowest cost. The model is run using five methods of selecting features across ten classifiers, then they will be compared to show the best result at the lowest cost.

2 Related studies

Recently, much attention has been given to finding and developing new methods of malware detection, compared to existing methods, to cover the gap of malware detection challenges that arise by the increase of malware over time [5].

Malware detection and analysis help the analyst learning the type, category, and target of malware. Malware detection can be classified into two categories, mainly: static and dynamic analysis. Static analysis is the primary category that analyzes malware and collects data from a file without executing it. Dynamic analysis is the opposite as it executes the suspicious file in an isolated and controlled environment [6].

There are many research papers done to develop malware detection methods. In [7], for example, a detection system using effective low-dimensional features has been proposed. This system used ensemble algorithms for analysis to get better performance. The model applies detection technology to a large number of malwares with faster detection time.

Another research [8] studies two categories of classification in one model. Alotaibi proposed Multi-Level Malware detection using Triad Scale (MLMTS) model that work in multi stages. The first two levels of his proposed method perform static analysis and the third level performs dynamic analysis. The linear regression in machine learning was used in this model as an input of each level. Using MLMTS method in research experiments increases the accuracy and decreases false alarming, compared to other recent models.

The study done by [9] focuses on improving an effective and efficient approach for malware detection by using the behavior of malware families. The authors proposed this methodology because they knew that the attacker could modify API call features with no change in overall behavior. So, they worked on three steps: studying API calls to object operation by analyzing the malware, generating a dependency graph based on the information of these operations, and finally defining the family dependency graph for each malware. The evaluation results of the proposed approach showed that the approach can help some anti-virus companies to detect malware from a zero-day attack.

Multiple anti-virus scanners detection systems were proposed for enhancement selection performance in the work done by [10]. They proposed multiple anti-virus scanners that attempt to check if increasing the number of scanners affect detection results and how these scanners are able to maximize the accuracy. The experiment shows that there is a small effect of the number of scanners on accuracy, and if the number was increasing, the overall accuracy will be lowered rather than improved. Moreover, the final ranking of the scanners depends on the accuracy and gives the best chance to select the best combination of scanners.

The malware detection model in this study uses a specific feature selection method that is used in several classifiers to compare the scores in order to show the effect of contemporary feature selection on reducing the cost of training time in balanced and unbalanced datasets. The experimental results were obtained by comparing Precision, Recall, Accuracy, and F1-scare in all classifiers and by comparing the commuting time as well.

The following Table 1 provides a summary of the related work done on this field of study.

Table 1: Summary of the Related Work.

Paper	Classifiers Algorithms	Features	Feature Selection Method	Result	Objective	Limitations
[6]	Chi-square	APIs/System calls	-	Detecting accuracy up to 96.56%	Proposing a model for recognizing and detecting the malware from benign.	The limitations of this model are related to malware that have an evasion detection technique, and it was used to detect 5 classes of malware only.
[11]	Evolutionary Algorithm	Malware OpCodes	-	Detecting accuracy for all datasets between 85.80% and 87.67%	Using Evolutionary Algorithm to generate graph and compare the similar graph to detect the suspicious files. It was used for categorizing malware and detecting it.	The study shows that the detection approach was used to categorize the malware and detect it, but it does not show if it can detect and cover all classes of malware.
[12]	Hidden Markov Model (HMM), Support Vector Machine (SVM), Decision Tree (J48), and Random Forest (RF)	API-call, operations, and usage system library	Used term and inverse term frequency (TF-IDF) Logarithm for feature extraction	Random Forest classifier gives the best results, while HMM has the lowest performance	Evaluating classification approaches in terms of distinctive dynamic features and finding the best dynamic features.	Malware detection approaches were used to obtain the family classification and malware detection.
[7]	AdaBoost, random forest, XGBoost, rotation trees, and extra trees.	2-gram, 2-gramM, API-DLL, API, and WEM	frequency analysis and Expert knowledge to select a relevant feature	XGBoost reaches the highest rank in AUC-PRC and accuracy	Developing a novel technique to reduce feature dimensionality.	The study does not represent the time used to extract features by frequency analysis and expert knowledge.

[8]	Proposed a model with multi-level linear regression (MLAPAM and MDMLA)	Call sequences, fallouts, and arguments	MLMTS method used to generate a feature set	The proposed method (MLMTS) gives the maximal accuracy and minimum false positive, compared to other methods	Building a model in a Multi-Level for Malware detection using Triad Scale (MLMTS) based on a regression coefficient.	The experiment study was performed using one benchmark malware dataset.
[9]	Comparing the object operation of feature dependency graph and family dependency graph	API call	-	The proposed model gives highly efficient and effective results.	Building a malware detection system based on behavior of the malware family.	The justification of using the behavior-based features and the graphs is time consuming.
[10]	Comparing a multi-scanner as a black box	Features extracted from the malware were not considered. Only the rates from the scanners were	-	Combining multi anti-virus scanners with achieving high accuracy, and the result is having the best combination of scanners	Proposing three models to achieve the best accuracy of multi-scanner detection system and minimize the scanning cost.	The internal mechanism is not clear, and it needs more details about the features and classifiers used in all scanners.
[13]	Gradient Boosting Algorithm	Malware OpCodes	Deep learning-based feature extraction method, word2vec	Detecting accuracy up to 96%.	Developing a model to represent malware that mainly uses the malware opcodes.	The work conducted was on a short range of malware classes. The paper covered 8 different malware classes.

3 Datasets information

This section presents all datasets used in our experiments conducted for this study. Our approach needed several datasets to study how they affect malware detection performance. All datasets used are available online.

We used two types of balanced and imbalanced datasets for malware detection domains. They were also categorized into two groups: malicious or benign software, each with a different number of instances and features.

Table 2 shows in detail all information regarding each dataset used in this study in terms of the number of features, the number of classes, the number of instances, characteristics of data, and the type of distribution datasets whether they were balanced or imbalanced.

3.1 PE section headers

The "PE-section" header is a balanced dataset that was developed by Angelo Oliveira to extract dataset features from the "PE-section" portion of a group of PE malware and PE goodware files that appeared in Cuckoo Sandbox reports. This dataset was created for malware detection and classification purposes [14].

3.2 Malware analysis datasets top1000 PE imports

Angelo Oliveira generated "TOP-1000 PE Imports" which is imbalanced dataset that was created from 'pe_imports' part of Cuckoo Sandbox reports for a group of PE malware and PE goodware files [15].

3.3 API call sequence

The imbalanced "API Call Sequence" dataset contains 42,797 malware and 1,079 goodware of API call sequences gathered by the extracted "calls" part of Cuckoo Sandbox reports [16].

3.4 Malware detection data

This imbalanced dataset was created by Takbiri in June 2019 as a result of his study done on detecting malware using Low-level Architectural Features of malware [17].

3.5 BIG malware dataset from Microsoft

Microsoft team created a balanced dataset from their competition for Malware Classification Challenge which is called "BIG 2015" [18].

Table 2: List of Used Datasets.

Dataset	Alias Name	# of Feature	# of Instances Used	# of Classes	Features Characteristics	Dataset Class Distribution
PE Section Headers	BS1	5	43293	2	Integer, Float, Text	Balanced
TOP-1000 PE Imports	DS2	1001	47580	2	Integer, Float, Text	Imbalanced
API Call Sequence	DS3	101	43876	2	Integer, Float, Text	Imbalanced
Malware Detection Data	DS4	16	70	2	Integer, Float, Text	Imbalanced
BIG Malware Dataset from Microsoft	DS5	69	5210	2	Integer, Float, Text	Balanced
ClaMP (Classification of Malware with PE headers)	DS6	55	5184	2	Integer, Float, Text	Balanced
Malware Executable Detection	DS7	531	373	2	Integer, Text	Imbalanced
Windows Malware Detection (REWEMA)	DS8	631	6271	2	Integer, Text	Balanced
Malware Classification	DS9	56	216352	2	Integer, Text	Imbalanced
Malware Goodware Dataset	DS10	27	50210	2	Integer, Float, Text	Imbalanced

3.6 ClaMP (Classification of Malware with PE headers)

The CLaMP balanced dataset is built from portable, executable files in header field values and from a combination of malware and benign samples to be used in the detection system [19].

3.7 Malware executable detection

Rumao created a dataset containing a set of features extracted from malware and goodware for Windows executable files. It blends two features of Windows executables: binary hexadecimal system calls feature, and DLL calls as hybrid features, in order to create this dataset. This imbalanced dataset contains 301 malicious programs, while the goodware contains 72 cases [20].

3.8 Windows Malware Detection (REWEMA)

Windows Malware Detection Dataset (REWEMA), as a balanced dataset, contains 3136 malicious programs and 3135 benign executable files. Features were extracted from disassembling executable files and selecting a set of useful file attributes [21].

3.9 Malware classification

Malware classification dataset uploaded to Kaggle website by Paul. Which is Imbalanced dataset, it contains 75503 malware and 140849 goodware features [22].

3.10 Malware goodware dataset

This dataset was uploaded to Kaggle in February 2021. This Imbalanced dataset contains 50210 instances features for malware and goodware files [23].

4 Method

4.1 Methodology design

The malware datasets described in Section 3 were collected to test the proposed method for the detection system. All ten datasets were classified and categorized into two categories of malware and benign software. In addition, these datasets have been further categorized into two other types: balanced and imbalanced datasets, and this categorization is based on the disproportion of the malware and benign category in each dataset.

Five feature selection techniques, which are described below in Section 4.2, were used in this study, and passed through fourteen machine learning classifiers in parallel. This objective model computes detection performance at the lowest cost. In our approach, we divided the ten datasets into a training and test set with percentages of 70% and 30%, respectively.

In this work, the model is designed and evaluated by making the following main steps: [1] Data cleaning was performed for all datasets before they are split for training and testing to fix all problems in the datasets (missing value, removing outliers, and resolving discrepancies, among others), and [2] five feature selection methods were used (Chi-Square, Filter-based, Wrapper-based, GPM, and GPMP). Then, [3] The number of features was selected for each feature selection method to compare performance, then it was calculated based on the number of features used in each method to test the performance based on how this method extract relevant features that reflect the effect in the overall performance of the discovery model. After that, [4] excessive oversampling SMOTE technique was applied in imbalanced datasets. [5] The release of new datasets was then introduced after applying feature selection and SMOTE methods in the classification model (14 classifiers) to measure

$$X^2 = \sum_{i}^{n} \left(\frac{(O_i - E_i)^2}{E_i} \right) \quad , i = 1,2,..,n \tag{1}$$

$$GPM = \sum_{i}^{k} (W_{Fk})/k \quad , i = 1,2,\dots,k \tag{2}$$

$$GPMP = (\sum_{y}^{k} (W_{Fk})/k) - (\sum_{o}^{z} (W_{Fk-low})/z) \quad , y = 1,2,\dots k \mid o = 1,2,\dots,z \tag{3}$$

predictions. [6] The performance evaluation scale for this detection model was accuracy, F1, accuracy, and recall. [7] The rating scale was finally compared for all datasets in all feature selection methods and all classifiers as well.

The result of the model focuses on the performance to obtain the results of balanced and imbalanced datasets. All these steps were performed for the ten datasets (whether balanced or unbalanced) to study whether our proposed approach will obtain good performance in all datasets with different characteristics.

4.2 Feature selection.

In this work, two main steps were applied in datasets before running the feature selection technique.

4.3 Data cleaning

In this study, we applied a data cleaning for all datasets. It is about preparing raw data to start working on feature selection by drop outliers, cleaning missing values, encoding (text, integer, date, and float, among others), and scaling data [24].

4.3.1 Using data augmentation technique

Synthetic Minority Over-sampling Technique (SMOTE) algorithm is one of the well-known augmentation techniques that are used in imbalanced datasets to solve minority class problems. In the imbalanced dataset, there are too few instances of minority classes that affect model decisions [25].

In this study, we used the SMOTE over-sampling technique to balance the number of classes in the datasets by adding new synthesized instances of the minority class. We also tested another SMOTE technology that is under-sampling by removing the random instances of the majority class, so that it is balanced against the minority class. However, the detection efficacy decreased because some datasets have too few minority classes which results in decreasing the dataset, and this will affect the training and testing phase. Therefore, the main augmentation technique that we used in this study for all imbalanced datasets is the SMOTE over-sampling technique [26].

4.3.2 Feature selection techniques

In this part of our study, we used five methods for feature selection, where three of them were commonly used in machine learning, and they are: Chi-Square, filter-based, and wrapper-based. The remaining methods are Genetic Programming Mean (GPM) and Genetic Programming Mean Plus (GPMP). They were developed in our study using genetic programming (GP) algorithm using the open-source frame-work HeuristicLab (Heuristic and Evolutionary Algorithms Laboratory) [27].

Table 3: Five Feature Selection Methods.

#	Feature Selection Method	Alias name
1	Chi-Square	Chi
2	Genetic programing Mean (GPM)	GPM
3	Genetic programing Mean Plus (GPMP)	GPMP
4	Filter-based	Filter
5	Wrapper-based	Wrapper

The GP method was used to create a weight for all features in hidden computations and to release the feature at relatively close values. We added two thresholds to the output result of the GP algorithm to find the most important and most relevant feature, in order to get more accuracy in perdition. In the first threshold used in GPM, the mean of all features values was computed, and all features were greater than the threshold.

In GPMP, we changed the threshold by adding a chance for the remaining features whose values are below the mean, and that was done by creating another interim threshold which was added to the original threshold value to add a change for the features where their values are near the original threshold. See equation (1) that defines Chi-Square, where O is the observed value and E is the expected value for all values.

Equation (2) represents the calculation of GPM method, WFk is the weight for the feature, and the integer number K represents all features y=1, 2, ..., K.

Equation (3) is similar to equation (2), but it subtracts the mean of all weights of features under the total mean as an interim threshold is used to increase the chance for other features that have a value less than the original threshold.

The main difference between these methods is that when we apply them in our approach, we find that a number of some specific features affect the computational cost and model detection performance. Each method evaluated feature values and compared them to the target value to find the strongest relationship between the target values depending on method statistical measures.

Table 3 shows the five feature selection methods used in this study and their alias used in the charts.

We found that each method has its own set of features that are identified to be used in the detection model. The difference in the number of features and the identified features themselves will be certainly reflected in the final

Table 4: Number of features used for all feature selection methods.

	Number of Features used						Percentage of Features used				
Dataset	**Chi-Square**	**GPM**	**GPMP**	**Filter-based**	**Wrapper-based**	**Total Feature NO**	**Chi-Square**	**GPM**	**GPMP**	**Filter-based**	**Wrapper-based**
DB1	3	2	4	3	3	5	60%	40%	80%	60%	60%
DB2	948	802	829	113	518	1001	95%	80%	83%	11%	52%
DB3	100	20	33	99	29	101	99%	20%	33%	98%	29%
DB4	15	7	15	14	15	16	94%	44%	94%	88%	94%
DB5	55	12	20	50	61	69	80%	17%	29%	72%	88%
DB6	43	13	16	29	37	55	78%	24%	29%	53%	67%
DB7	483	70	70	133	201	531	91%	13%	13%	25%	38%
DB8	151	59	48	563	611	631	24%	9%	8%	89%	97%
DB9	54	15	18	34	46	56	96%	27%	32%	61%	82%
DB10	19	7	9	20	25	27	70%	26%	33%	74%	93%
Total							**79%**	**30%**	**43%**	**63%**	**70%**

results of the detection model. Table 4 shows the differences between the number of features identified in each method.

4.4 Evaluation metrics

To evaluate our proposed detection model approach, we used the common evaluation metrics. These metrics are accuracy, precision, and recall, and we added F1-score because we tested two types of balanced datasets that can be measured using accuracy. In another hand, imbalanced datasets need to be measured using F1-score and accuracy. Equations from (4) to (7) show how these metrics are calculated [28].

F1-score mainly considers the values of both Precision and Recall, while Accuracy represents the percentage of the number of correct predictions in the model to the total number of inputs.

$$F1 - Score = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

$$Accurcy = \frac{TP + TN}{TP + FP + FN + TN} \quad (5)$$

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

$$Recall = \frac{TP}{TP + FP} \quad (7)$$

5 Experimental results

In this section, we present the results of our experiment to evaluate the findings of detection over ten datasets.

Based on all experiments, we evaluated the detection model and summarized the results of the study in the conclusion section.

Table 5 shows 14 classifiers that were used in the proposed detection model after applying five feature selection methods in ten labeled malware datasets.

Based on the literature review examining the performance of classifiers, we used 14 classifiers shown in Table 5. We selected these classifiers depending on the efficiency of the literature review. We chose them based on 1) the most common classifier, 2) the least efficient classifier to test our approach, and 3) the most efficient classifier. The diversity of this chosen standardization helps us studying the proposed detection system. In our study, we applied our approach to build our model using four main steps: pre-processing for data cleaning, using augmentation technique for imbalanced datasets, using five-feature selection methods, and applying the data on

Table 5: Classifiers used in proposed model.

NO.	Classifiers	Alias name used in charts
1	Ada Boost.M1	AdaBM1
2	Ada Boost.M1 (4)	AdaBM1(4)
3	AdaBoost	AdaB
4	CatBoost	CatBoost
5	Decision Stump	DStump
6	Hoeffding Tree	HTree
7	k Nearest Neighbors	KNN
8	Naive Bayes	NB
9	Random Committee	RComm
10	Random Committee (4)	RComm4
11	Random Forest	RF
12	Random Forest4	RF4
13	Random Tree	RT
14	Support vector Machines	SVM

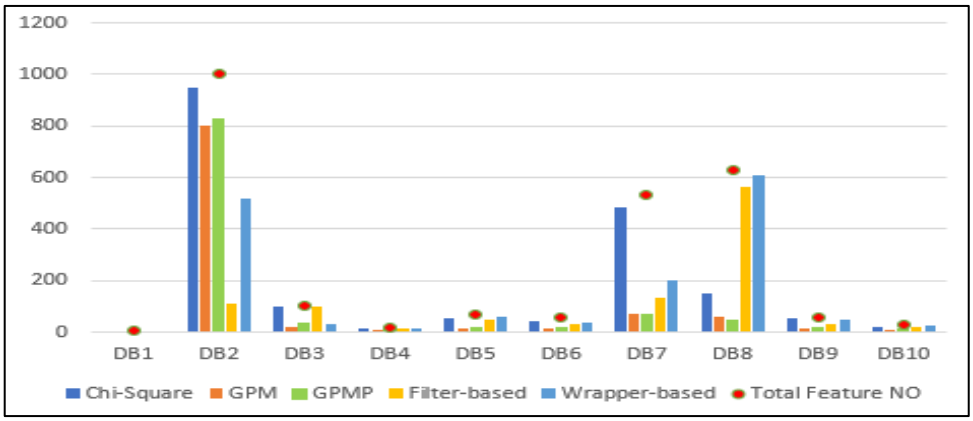

Figure 1: The number of features is used in all Datasets based on the FS methods.

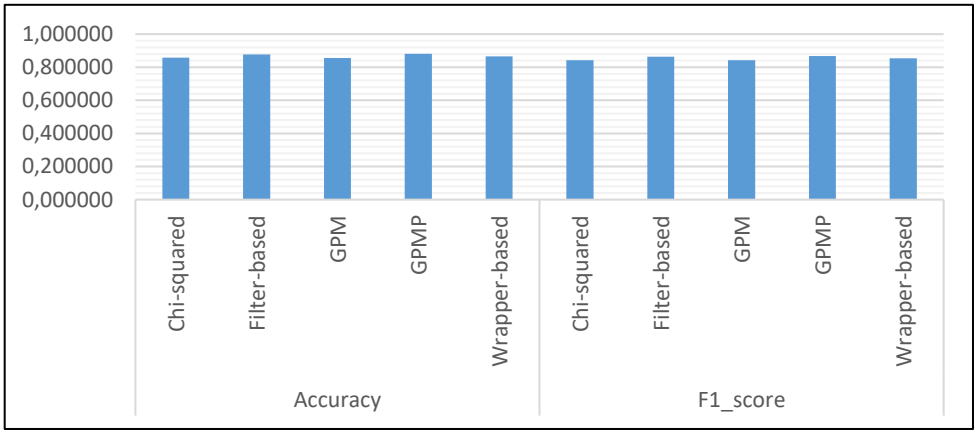

Figure 2: Average accuracy and F1-score summary for ten DS using 14 classifiers.

the model using 14 classifiers.The main objectives of this study focus on:

First: knowing if the new proposed feature selection methods affect the overall performance of the detection model.

Second: Knowing if the proposed methods give good performance of detection in balanced and imbalanced datasets.

Third: Determining which classifiers performs better using new FS methods and comparing them to other state-of-the-art performance methods.

Figure 1 shows the total number of features in all datasets compared to the number of features used in all FS methods in this study. As a Figure 1 appears almost in all datasets, chi-square and wrapper-based used many features in all datasets according to their calculation.

The proposed methods (GPM and GPMP) have a close result to the number of the used features, compared to filter-based. GPM and GPMP used fewer features than filter-based features in seven datasets. Table 4 shows the percentage of features used in ten datasets. GPM and GPMP have the minimum percentage of features used, with a value of 30% and 43%, respectively.

The highest number of features are used in Wrapper-based, in DS8 the percentage of features used are 97% that mean almost all the features are kept and used.

The highest number of features are used in Wrapper-based, and in DS8, the percentage of the used features are 97%. This means that almost all features are kept and used.

Based on the percentages shown in Table 4, and by applying FS on 14 classifiers, it can be noted, after conducting the initial analysis of the results, that the best results of F1-score and accuracy were found after applying the features that were selected by GPMP and Filter-based, with a little difference in values.

The first output of our results shows that the comparison between GPMP and Filter-based must be studied, while GPM gives less performance than these two FS methods.

This finding guided us to check if accuracy and F1-score were affected by these percentages. As shown in figures (3) to (12), the results of the experiment conducted for ten datasets show that we must study if these FS methods give the same performance in balanced and imbalanced datasets. Furthermore, we studied the overall behavior of the performance in all datasets, and we compared the values that were found in balanced and imbalanced datasets after applying SMOTE oversampling technique.

We noted, once we applied SMOTE augmentation technique, that prediction model is able to obtain the best performance based on F1-score and accuracy in the 14 classifiers that were used.

SMOTE is a common oversampling technique that is mainly used to handle the imbalanced datasets, but it may cause the model to need extra time for training and over-fitting. However, in this study, oversampling technique

Table 6: Average of accuracy and F1-score for ten DS using 14 classifiers and five FS methods.

	GPM		Filter-based		GPMP		Chi-squared		Wrapper-based		Avg F1-score
	Accuracy	F1_score	Accuracy	F1_score	Accuracy	F1_score	Accuracy	F1_score	Accuracy	F1_score	
AdaBoost Avg	**0.897007**	**0.892153**	**0.950888**	**0.950875**	0.912717	0.909771	**0.913025**	**0.912015**	0.905135	**0.910262**	**0.915015**
AdaBoost.M1 Avg	0.877519	0.875979	0.931936	0.931577	**0.933636**	**0.933926**	0.897579	0.897521	**0.911156**	0.910161	**0.909833**
AdaBoost.M1(4) Avg	0.889907	0.887123	0.920886	0.920435	0.917216	0.920035	0.870939	0.868993	0.902101	0.901588	**0.899635**
CatBoost Avg	0.844995	0.854525	0.855918	0.855751	0.885714	0.885961	0.898815	0.898688	0.857265	0.857820	**0.870549**
Decision Stump Avg	0.797667	0.790254	0.793439	0.775139	0.819049	0.812602	0.752604	0.732342	0.771943	0.756560	**0.773379**
Hoeffding Tree Avg	0.519706	0.381524	0.587115	0.442341	0.548830	0.396545	0.526053	0.386072	0.623355	0.525731	**0.426442**
KNN Avg	**0.904014**	**0.901189**	**0.932180**	**0.932396**	**0.954862**	**0.953708**	**0.932422**	**0.934905**	0.862516	0.852418	**0.914923**
NB Avg	0.768505	0.736326	0.712549	0.670044	0.735392	0.705738	0.700059	0.648922	0.789419	0.769453	**0.706096**
Random Committee Avg	0.882569	0.880216	0.908151	0.908101	0.906350	0.906522	0.884793	0.884288	0.825369	0.792877	**0.874401**
Random Committee(4) Avg	0.877746	0.873598	0.870887	0.871200	0.871380	0.871454	0.887017	0.885551	0.861022	0.858908	**0.872142**
Random Forest Avg	**0.957570**	**0.955435**	**0.976959**	**0.976194**	**0.979496**	**0.979747**	**0.945723**	**0.944170**	**0.959321**	**0.962125**	**0.963534**
Random Forest(4) Avg	**0.950536**	**0.947880**	**0.975251**	**0.975478**	**0.980718**	**0.979334**	**0.948085**	**0.944361**	**0.966235**	**0.966801**	**0.962771**
Random Tree Avg	**0.948175**	**0.942662**	**0.972579**	**0.972934**	**0.976031**	**0.974188**	**0.939855**	**0.939396**	**0.965424**	**0.962871**	**0.958410**
SVM Avg	0.880227	0.872036	0.898003	0.898045	0.913536	0.916118	0.907813	0.907274	**0.915233**	**0.919567**	**0.902608**
Avg	**0.856867**	**0.842207**	**0.877624**	**0.862894**	**0.881066**	**0.867546**	**0.857484**	**0.841750**	**0.865392**	**0.853367**	

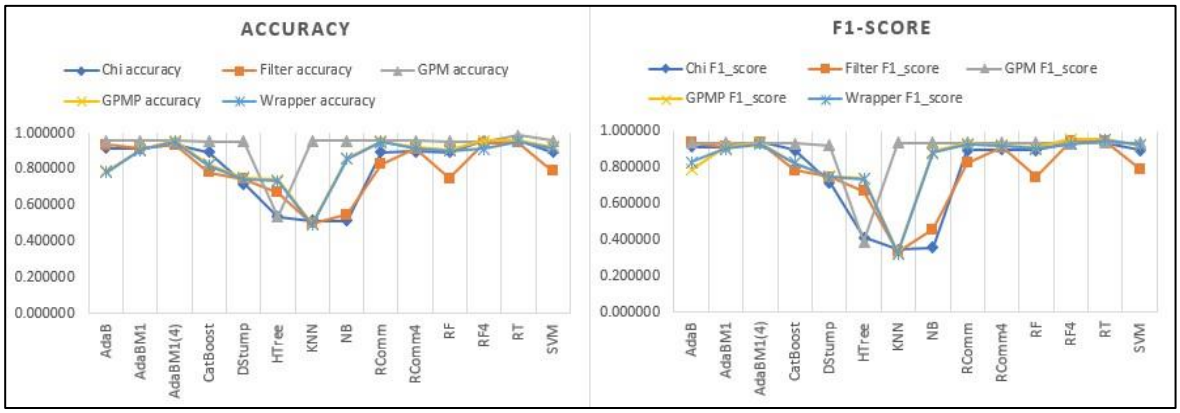

Figure 3: Accuracy and F1-score for DS1.

helps the model to give better performance when compared to balanced datasets.

Figures (3) to (12) illustrate the performance of all of our study objectives. In general, we can see that the balanced and imbalanced datasets are illustrated in similar shapes with little detailed differences occurred after applying SMOTE technique. This means that FS methods have a good result in all datasets regardless of whether they are balanced or imbalanced.

In the final step of our study, we tried to determine which classifier gives better detection performance using the five FS features over ten datasets (balanced and imbalanced).

After applying our approach on ten datasets, results were summarized by computing the average values for F1-score and accuracy for all experiments, as shown in Table 6 and Figure 2. The average of the highest calculated values of F1-score and accuracy shows that it is significant to rank the classifiers based on the efficiency.

We found that there were three datasets that held the best ranks in the average of all conducted experiments. Random Forest, Random Forest (4), and Random Tree are in the lead in accuracy and F1-score values. They are then followed by the other three classifiers, classified as group B of performance, namely: AdaBoost, AdaBoost.M1, and KNN. Additionally, both Hoeffding Tree and Decision Stump give the lowest values of F1-score and accuracy in all experiment. The remaining classifiers are categorized in the middle of giving good performance results scales.

Figure 2 summarizes the average values of accuracy and F1-score for ten DS using 14 classifiers. The average values for all experiments help us concluding our study by saying that GPMP and Filter-based give the best results in all experiments with the average of f1-score values that reach 0.867546 and 0.862894, respectively.

This finding leads us to examine the differences between FS methods. Figure 1 shows the number of features used in all datasets based on FS methods. The

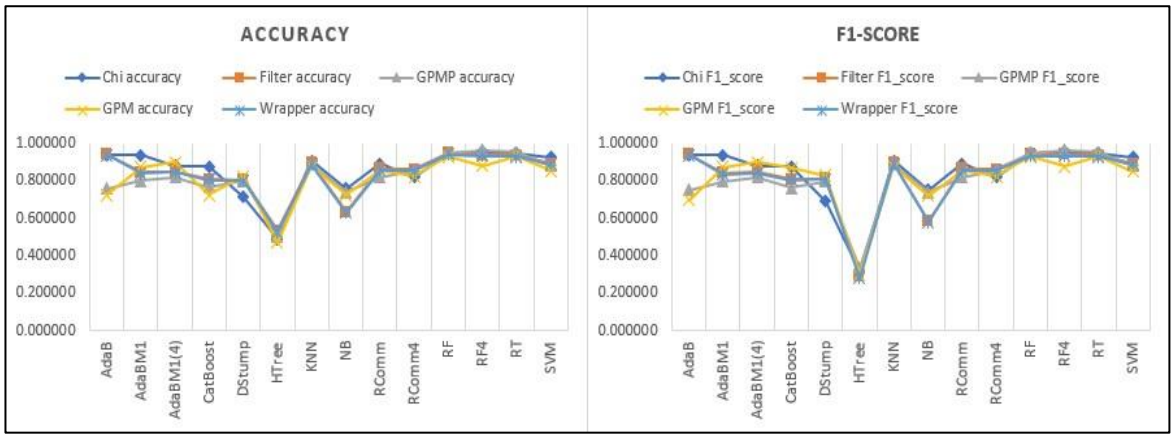

Figure 4: Accuracy and F1-score for DS2.

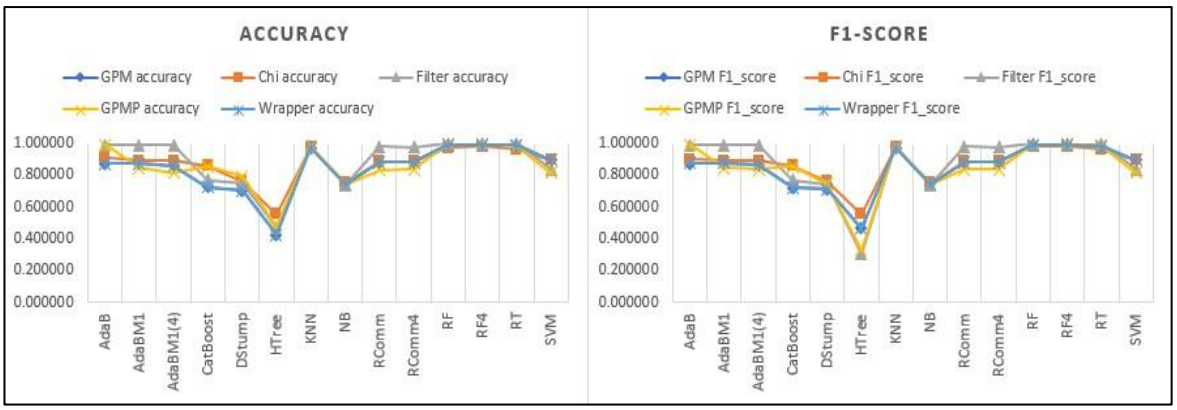

Figure 5: Accuracy and F1-score for DS3.

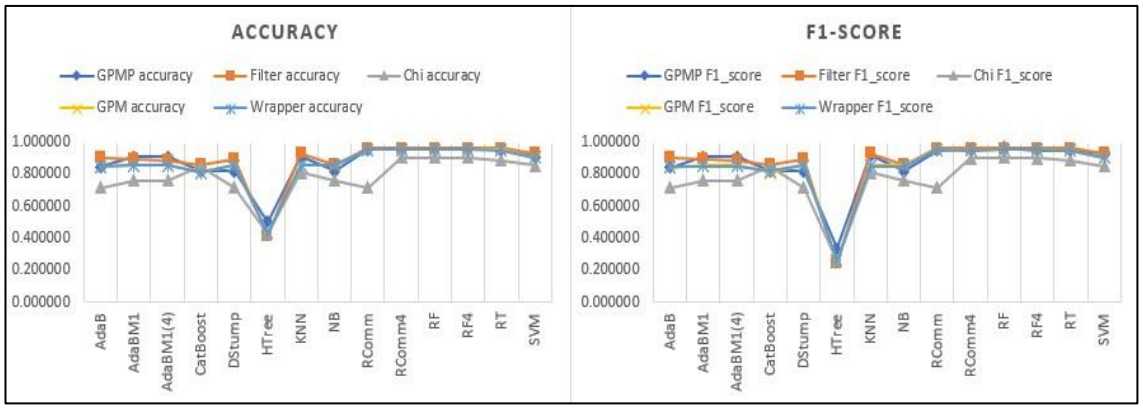

Figure 6: Accuracy and F1-score for DS4.

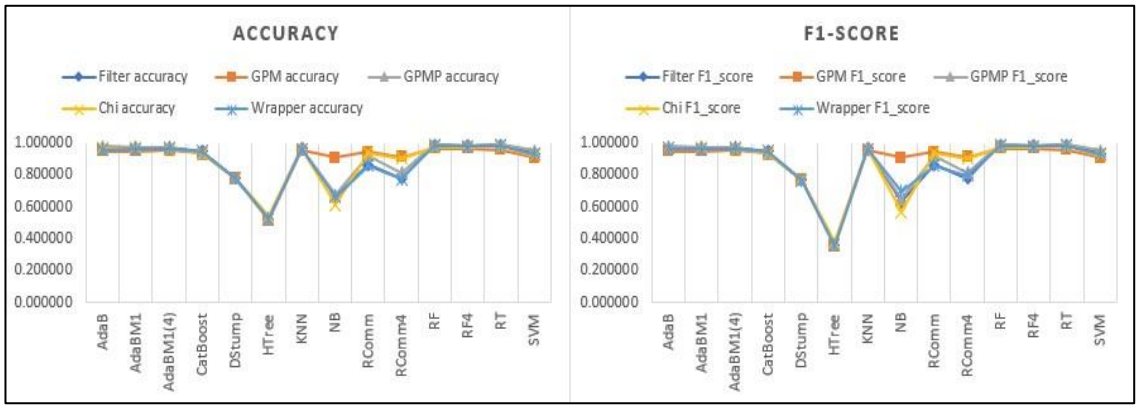

Figure 7: Accuracy and F1-score for DS5.

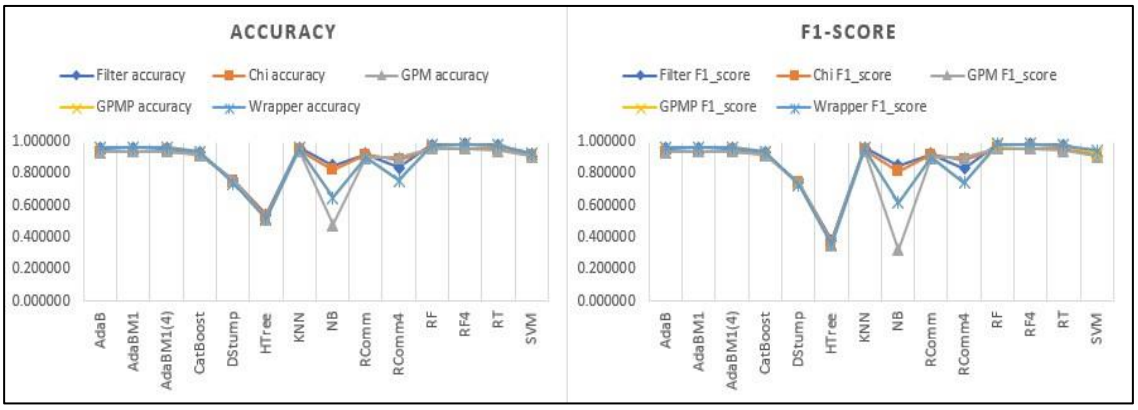

Figure 8: Accuracy and F1-score for DS6.

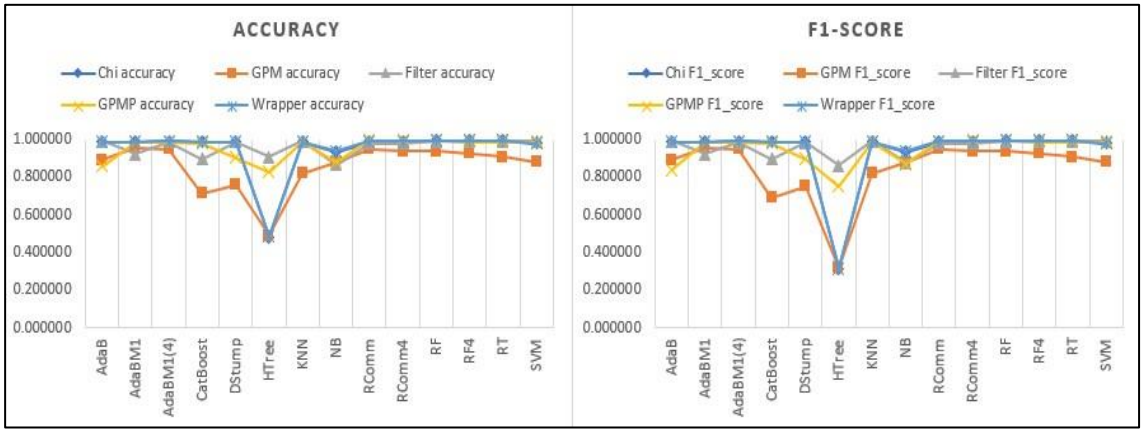

Figure 9: Accuracy and F1-score for DS7.

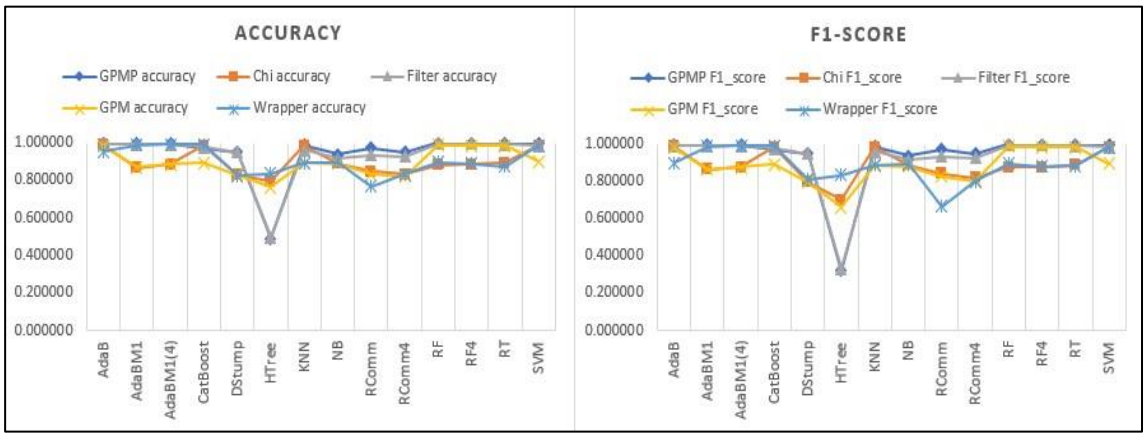

Figure 10: Accuracy and F1-score for DS8.

Figure 1 shows that in most of the datasets, the GPMP used fewer features than Filter-based. This means that the computation used in the model used less time in GPMP than on Filter-based.

Figures (3) to (12) show the F1-score and accuracy of all datasets. The analysis of the figures values shows the same results summarized in Table 6. In all figures, Random Forest, Random Forest (4), and Random Tree are at the top of all experiments. The values of AdaBoost.M1, and KNN are approximately similar, but the values of the Hoeffding Tree and the decision stump are shown in all figures below. These findings can be generalized for all datasets, whether they are balanced or imbalanced, as previously discussed.

To check the effectiveness of our study, we have implemented our model on ten datasets to get the big picture of our study and the reasons why the proposed model is more effective and efficient.

It is difficult to compare the results of the proposed model with other models because most of the models use a limited number of malware detection features and because there are other limitations such as using a single dataset to make a comparison between the results. This study also covers both balanced and imbalanced datasets and applies the proposed model to them. Most of the

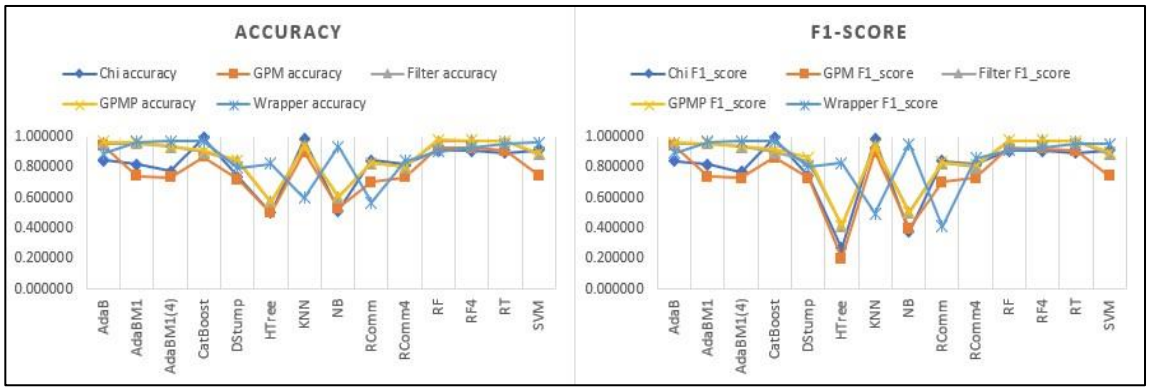

Figure 11: Accuracy and F1-score for DS9.

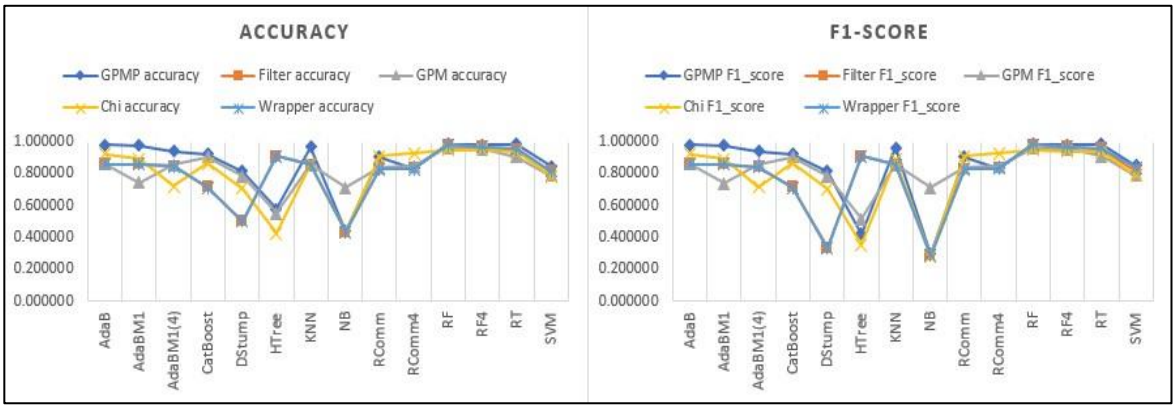

Figure 12: Accuracy and F1-score for DS10.

related works measure accuracy as a performance measurement, but our study does the measures using accuracy and F1-score because we use an imbalanced dataset. However, the results of the proposed model can be evaluated along with other related works by checking the result of F1-score of 0.9635 while we use Random Forest in the average of ten datasets, and this is considered a good value for the detection rate.

We have proposed a malware detection model using 14 classifier algorithms and five feature selection methods, two of which are proposed. Our feature selection methods are compared to other recent methods by applying them to the same datasets to check the differences in accuracy. We found our proposed method to be very effective for distinguishing between benign and harmful programs in relation to their detection.

6 Conclusion

This paper presents a model for detecting malware to enhance the detection rate by using five feature selection methods in ten malware datasets and 14 classifiers.

This study examines if this proposed detection method gives better detection value for balanced and imbalanced datasets. The experiments shown throughout the study have no difference in detection values while using balanced and imbalanced datasets after applying SMOTE overfitting technique in imbalanced datasets.

The results of this experiment have confirmed that the proposed GPMP feature selection methods attained high detection values in accuracy and F1-score.

The overall rankings of feature selection methods depending on accuracy and F1-score in this experiment are GPMP, Filter-based, Wrapper-based, and chi-square, respectively.

Results show that GPMP methods used fewer features than other methods with a percentage of 43% in the average of ten datasets. Filter-based that compete GPMP in detection rate used 63% features in an average of ten datasets. This shows how Filter-based affects the complexity and computation in the detection model. The average values of detection rate summarize the performance when using FS methods by saying that GPMP and Filter-based give average F1-score values of 0.867546 and 0.862894, respectively.

The final findings in this study focus on performance ranks for 14 classifiers in an average of all experiments. Random Forest, Random Forest (4), and Random Tree have the highest experiment results in accuracy and F1-score values. The values for these classifiers in F1-score are 0.963534, 0.962771, and 0.958410, respectively.

These values are followed by the values of AdaBoost, AdaBoost.M1, and KNN, while Hoeffding Tree and Decision Stump in all experiments give lower values for F1-score and accuracy.

We intend, in our future work, to apply this presented method in this model on android malware detection in order to study the features of the datasets and the performance of classifiers.

Reference

[1] "The number of new malicious files detected every day increases by 5.2% to 360,000 in 2020 | Kaspersky." https://www.kaspersky.com/about/press-releases/2020_the-number-of-new-malicious-files-detected-every-day-increases-by-52-to-360000-in-2020 (accessed Jun. 14, 2021).

[2] Y. Jian, X. Dong, and L. Jian, "Detection and recognition of abnormal data caused by network intrusion using deep learning," Inform., vol. 45, no. 3, pp. 441–445, 2021, doi: 10.31449/inf.v45i3.3639.

[3] O. F.Y, A. J.E.T, A. O, H. J. O, O. O, and A. J, "Supervised Machine Learning Algorithms: Classification and Comparison," Int. J. Comput. Trends Technol., vol. 48, no. 3, pp. 128–138, 2017, doi: 10.14445/22312803/ijctt-v48p126.

[4] A. Chaudhuri, "Parallel fuzzy rough support vector machine for data classificatin in cloud environment," Inform., vol. 39, no. 4, pp. 397–420, 2015.

[5] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A Survey on Machine Learning Techniques for Cyber Security in the Last Decade," IEEE Access, vol. 8, no. 01, pp. 222310–222354, 2020, doi: 10.1109/ACCESS.2020.3041951.

[6] O. Savenko, A. Nicheporuk, I. Hurman, and S. Lysenko, "Dynamic signature-based malware detection technique based on API call tracing," CEUR Workshop Proc., vol. 2393, pp. 633–643, 2019.

[7] S. Euh, H. Lee, D. Kim, and D. Hwang, "Comparative analysis of low-dimensional features and tree-based ensembles for malware detection systems," IEEE Access, vol. 8, pp. 76796–76808, 2020, doi: 10.1109/ACCESS.2020.2986014.

[8] S. S. Alotaibi, "Regression coefficients as triad scale for malware detection," Comput. Electr. Eng., vol. 90, no. December 2019, p. 106886, 2021, doi: 10.1016/j.compeleceng.2020.106886.

[9] B. Cheng et al., "MoG: Behavior-Obfuscation Resistance Malware Detection," Comput. J., vol. 62, no. 12, pp. 1734–1747, 2019, doi: 10.1093/comjnl/bxz033.

[10] M. N. Sakib, C. T. Huang, and Y. D. Lin, "Maximizing accuracy in multi-scanner malware detection systems," Comput. Networks, vol. 169, p. 107027, 2020, doi: 10.1016/j.comnet.2019.107027.

[11] F. Manavi and A. Hamzeh, "A new approach for malware detection based on evolutionary algorithm," GECCO 2019 Companion - Proc. 2019 Genet. Evol. Comput. Conf. Companion, pp. 1619–1624, 2019, doi: 10.1145/3319619.3326811.

[12] A. G. Kakisim, M. Nar, N. Carkaci, and I. Sogukpinar, Analysis and evaluation of dynamic feature-based malware detection methods, vol. 11359 LNCS. Springer International Publishing, 2019.

[13] C. H. Lo, T. C. Liu, I. H. Liu, J. S. Li, C. G. Liu, and C. F. Li, "Malware classification using deep learning methods," Proc. Int. Conf. Artif. Life Robot., vol. 2020, pp. 126–129, 2020, doi: 10.5954/ICAROB.2020.OS4-4.

[14] "Malware Analysis Datasets: PE Section Headers | Kaggle." https://www.kaggle.com/ang3loliveira/malware-analysis-datasets-pe-section-headers (accessed Mar. 07, 2021).

[15] "Malware Analysis Datasets: Top-1000 PE Imports | IEEE DataPort." https://ieee-dataport.org/open-access/malware-analysis-datasets-top-1000-pe-imports (accessed Mar. 07, 2021).

[16] "Malware Analysis Datasets: API Call Sequences | IEEE DataPort." https://ieee-dataport.org/open-access/malware-analysis-datasets-api-call-sequences (accessed Mar. 07, 2021).

[17] "Windows Malware Detection | Kaggle." https://www.kaggle.com/sidneylima/rewema (accessed Mar. 07, 2021).

[18] Microsoft, "Microsoft Malware Classification Challenge (BIG 2015) | Kaggle," 2018. https://www.kaggle.com/c/malware-classification/data. (accessed Mar. 07, 2021).

[19] A. Kumar, "ClaMP (Classification of Malware with PE headers)," vol. 1, 2020, doi: 10.17632/XVYV59VWVZ.1.

[20] "Malware Executable Detection | Kaggle." https://www.kaggle.com/piyushrumao/malware-executable-detection (accessed Mar. 07, 2021).

[21] "GitHub - rewema/REWEMA." https://github.com/rewema/REWEMA (accessed Mar. 07, 2021).

[22] "Malware Classification | Kaggle." https://www.kaggle.com/kallolkumarpaul/malware-classification (accessed Mar. 07, 2021).

[23] "Malware Goodware Dataset | Kaggle." https://www.kaggle.com/arbazkhan971/malware-goodware-dataset (accessed Mar. 07, 2021).

[24] N. Iqbal and M. Islam, "Machine learning for dengue outbreak prediction: A performance evaluation of different prominent classifiers," Informatica, vol. 43, no. 3, 2019, doi: 10.31449/inf.v43i3.1548.

[25] S. A. Alsaif and A. Hidri, "Impact of data balancing during training for best predictions," Inform., vol. 45, no. 2, pp. 223–230, 2021, doi: 10.31449/inf.v45i2.3479.

[26] J. L. P. Lima, D. MacEdo, and C. Zanchettin, "Heartbeat Anomaly Detection using Adversarial Oversampling," Proc. Int. Jt. Conf. Neural Networks, vol. 2019-July, no. July, pp. 1–7, 2019, doi: 10.1109/IJCNN.2019.8852242.

[27] A. Elyasaf and M. Sipper, "Software review: The HeuristicLab framework," Genet. Program. Evolvable Mach., vol. 15, no. 2, pp. 215–218, 2014, doi: 10.1007/s10710-014-9214-4.

[28] E. Amer and I. Zelinka, "A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence," Comput. Secur., vol. 92, 2020, doi: 10.1016/j.cose.2020.101760.