SN Computer Science (2020) 1:17
https://doi.org/10.1007/s42979-019-0017-9

ORIGINAL RESEARCH

Improvement of Malware Classification Using Hybrid Feature Engineering

Emmanuel Masabo[1,2] · **Kyanda Swaib Kaawaase**[1] · **Julianne Sansa-Otim**[1] · **John Ngubiri**[1] · **Damien Hanyurwimfura**[2]

Received: 1 July 2019 / Accepted: 5 August 2019 / Published online: 16 August 2019

Abstract
Polymorphic malware has evolved as a major threat in Computer Systems. Their creation technology is constantly evolving using sophisticated tactics to create multiple instances of the existing ones. Current solutions are not yet able to sufficiently address this problem. They are mostly signature based; however, a changing malware means a changing signature. They, therefore, easily evade detection. Classifying them into their respective families is also hard, thus making elimination harder. In this paper, we propose a new feature engineering (NFE) approach for a better classification of polymorphic malware based on a hybrid of structural and behavioural features. We use accuracy, recall, precision, and *F* score to evaluate our approach. We achieve an improvement of 12% on accuracy between raw features and NFE features. We also demonstrated the robustness of NFE on feature selection as compared to other feature selection techniques.

Keywords Malware classification · Polymorphic malware · Machine learning · Feature engineering

Introduction

Malicious software is written with the purpose to destroy the information and attempt to break into systems' privacy and data integrity [1]. There were more than 430 million pieces of malware injected into the Internet in 2015 [2]. In quarter 3 of 2017, Kaspersky detected 277,646,376 malicious attacks from the Internet in 185 countries all over the world [3]. A number of these threats were new variants of existing malware families, in spite of the fact that new malware types could be detected [3]. Current malware is able to change themselves in several numbers of forms, thus exhibiting polymorphic behaviours with the aim of confusing and defeating the detection systems [4].

There exist signature-based and non-signature-based malware detection techniques today [5]. Signature-based systems are the most popular. They keep the patterns of known malware in the database, which thereafter are used to compare against new suspicious files. When a pattern similarity is successfully established, the files are considered to be malicious [6]. Their detection accuracy is high with very low false positives when detecting the known malware [5]. Nevertheless, this technique has some weaknesses. First, it cannot detect new malware or variants of existing ones, and second, the database needs to be frequently updated.

Most non-signature techniques rely on behaviours observed, while the file of interest is running and can determine if this file is malicious or not based on the predefined patterns [6]. They can detect new malware as well as the variants of existing ones. The main drawback these non-signature-based systems is a high likelihood of false positives.

A successful classification approach relies upon the analysis procedures and feature-processing techniques in the development of the classification model [7]. Features are the inner attributes that characterize the malware [8]. Highly significant features must be extracted and processed well to yield good results. Feature engineering plays a key role in the success of the machine-learning model to be built. It is one of the most important elements vital to a successful machine-learning project [9]. It is about how data are presented and munged in terms of cleaning messy data, transforming data from raw format to another more reliable form. Feature engineering also consists of other tasks related to data pre-processing. Poorly engineered features can have

✉ Emmanuel Masabo
masabem@gmail.com

[1] Makerere University, Kampala, Uganda

[2] University of Rwanda, Kigali, Rwanda

a negative impact on the results of predictive modelling. With well engineered features, the learning process will be easier and the performance will be very good [10]. Well-engineered features produce good results at low resource costs to the computer system. Therefore, there is a need for ideal features that strongly describe the data [10]. In this research, a feature engineering process based on a hybrid of structural and behavioural features is proposed. Key features have been taken into consideration to widely cover high- and intermediate-level goals of polymorphic malware [11].

With further processing, the impact factors of each feature on the outcome were assessed and appropriate techniques were applied to retain the most useful ones. The data set was composed of more than six thousand malicious binaries grouped in different families.

In this paper, we propose a new feature engineering approach that uses both structural and behavioural features. A prototype evaluation model is created to test the efficiency of the proposed approach.

Key contributions of this research are as follows: (1) we propose a new feature engineering (NFE) technique based on a hybrid feature set obtained through static and dynamic analyses. (2) We use NFE obtained features to build a classification model based on KNN, Linear Discriminant Analysis and Gradient Boosting Classifiers. (3) We evaluate the effectiveness of NFE on the performance of classifiers as well as its robustness on feature selection.

The rest of this paper is organized as follows. Section 2 presents the background and discussion of the related work. Details about the data set and feature extraction is described in Sect. 3, followed by Sect. 4 which focuses on the steps taken to create a new feature engineering approach. Experiments and results are discussed in Sect. 5. The general discussion is given in Sect. 6. Finally, conclusion and future work suggestions are made in Sect. 7.

Related Works

Static Analysis

With static analysis, features are extracted without running the malware [12]. Naidu et al. [13] used syntactic structures of a malware obtained by extracting hex dump sequences using sigtool from ClamAV and converting them into binary codes. These binary codes were then converted into DNA sequences using the following rule: '00' → 'A'; '11' → 'T'; '10' → 'G'; and '01' → 'C'. The DNA sequences were input into JAligner tool, where a Smith-Waterman algorithm (SWA) was implemented. Common substrings or patterns were extracted. Pairwise local alignment (PLA) was performed using SWA with different substitution matrices. Only substrings with the highest percentage of identities and similarities were extracted after PLA. 161 substrings were retained through 6 substitution matrices. These 161 substrings have been considered as meta-signatures and used to detect all known polymorphic variants of JS.Cassandra. T-Coffee tool was used to perform multiple sequence alignment on the generated meta-signatures and generated consensuses. Rules were generated using a data-mining classification algorithm called PRISM [14] and this allowed the generation of 47 super signature substrings. Therefore, those 47 super-signatures and 161 meta-signatures were converted back to hexadecimal and tested against JS.Cassandra. This method was able to detect 100% of JS.Cassandra malware and all its known variants.

Drew et al. [15] introduced a detection approach that follows the logic of biological gene sequence mutations. They used a tool developed for gene classification called the Super Threaded Reference-Free Alignment-Free N-sequence Decoder (STRAND) for classifying polymorphic malware. Their approach was evaluated on a 500 GB malware data set provided by Kaggle-Microsoft Malware Classification Challenge (BIG 2015) [15]. Features were developed for STRAND classifier from given sample bytes, where hex values were considered and missing values removed. This created a single string/strand sequence containing hex contents extracted from malware file. Sequences of words of length k or (kmers) were generated by strand. Word length of 10 characters and 2400 minihash values were used. Jaccard similarities were computed. Training time was less than 7 h and strand classification accuracy was 95%, of which nine classes of polymorphic malware in the given data set were detected.

In the extension of work [15], Drew et al. [15] presented a new approach that detects polymorphic malware using sequence classification methods and ensembles. STRAND algorithm was used for the classification task. Features extracted from bytes (.bytes) data and assembly data (.asm) as provided by Microsoft. Ensembles were created using both the ".asm" and ".bytes" features from Strand. The minihash scores of created models were computed and the highest detection accuracy reached was 98%.

Naidu et al. [16] proposed an approach that is based on automatic signature extraction and uses Needleman–Wunsch and Smith–Waterman algorithms for detection mechanism. JS.Cassandra and W32.Kitti polymorphic malware were used in their experiments. Hex dumps were extracted and converted to DNA. Pairwise sequence alignments were performed, and meta- and super-signatures were generated. Finally, this method was able to detect 100% of JS.Cassandra and W32.Kitti known polymorphic variants.

The method proposed by Sharma et al. [17] uses signatures and pattern matching to detect polymorphic malware. A python module called pydasm is used to extract features and its report recorded. Opcodes were extracted from the

given instructions and used for further processing. K-NN algorithm was implemented and the method was able to detect polymorphic malware. The detection accuracy was not provided.

Dynamic Analysis

With dynamic analysis, features are extracted by executing the malware in a virtual environment [12]. Arshi et al. [18] proposed a machine-learning behavioural detection approach. They used 1270 malware programs of different file formats. Logic Model Tree and K-Means algorithms have been used for the task of classification and clustering, respectively. The results show that 18% of analysed malware were embedded with networking capabilities to connect to the outer world, while 82% aimed to corrupt the system locally or network resources. The proposed approach was able to cluster malware in their respective file types.

A behavioural-based sequential pattern was developed by Ahmadi et al. [19]. API calls were monitored during dynamic analysis and were extracted as features. Initial data set was then created from log data by only considering the repetitive patterns of the API calls. Feature selection was conducted using Fisher score algorithm. Support Vector Machines (SVM) and decision tree algorithms were used for malware classification. This method was evaluated on a sample of 806 malware and 306 benign files. A detection accuracy of 95% was successfully achieved.

Hybrid-Based Analysis

The hybrid technique combines features obtained through static and dynamic analyses [12]. Fraley et al. [20] developed a detection method based on the topological feature extraction using static and dynamic analysis techniques. IDA pro and cuckoo sandbox tools were used for feature extraction. Experiments were conducted on a data set of 3637 samples, of which 2400 were clean, 800 were malicious and 437 were unlabelled. They used function call graphs to trace instruction patterns. Belief propagation was used to uncover the properties of malicious files. 12 structural and 8 behavioural features were used as their feature set. Few examples of extracted features include MOV, ADD, LEA, file size, and compiler type. The created feature set was converted into the ARFF data format compatible with the Weka data-mining tool. Ensemble bagging algorithms were used in Weka for classification. Cross validation with tenfolds was used for validation. The overall accuracy was 99.9%, where the low false negative is 0.001.

Kaur et al. [21] developed a hybrid anomaly and signature detection system which was used to detect zero-day polymorphic worms in an active network flow. The system architecture had three components such as suspected traffic filter (STF), zero attack evaluation (ZAE), and signature generator (SG). Suspicious traffic was collected using STF, where known malware are blocked and logged. Zero-day attacks were redirected to the honeypot for further analysis. STF module was composed of two components: Honeynet and IDS/IPS to capture the traffic and compare network traffics. The Longest Common Prefix (LCP) algorithm was used in the comparison process. When IDS/IPS is found to have ignored an attack that was logged by honeynet, it meant that it was a new unknown attack. Analyses were done by ZAE component by which malicious strings were extracted such as NOP sleds, decryptor, shellcodes, and return addresses. Content-based signatures were generated by the SG module using invariant bytes found in a polymorphic malware. The method has been tested on a sample of 15,435 packets of which 734 were polymorphic. The detection rate was 96% with almost zero positive rates.

Saleh et al. [22] combined static-, dynamic-, and instruction-based features to create an integrated feature vector. They used Random Forest and Naive Bayes machine-learning algorithms to train classifiers and achieved a detection accuracy of 96.7%.

Farrokhmanesh et al. [23] proposed an approach that converts malware bytes to the audio signal and then employed Music Information Retrieval (MIR) techniques to develop a machine-learning model from audio signals. They were able to detect new and unseen instances of existing malware. Malware classification was conducted using Kaggle malware data set and obtained 95.80 *F* score with random forest classifier.

Gibert et al. [24] presented a deep learning system for classifying malware based on its visualization as grey-scale images. They claim that their solution can also detect malware in a real-time environment. Their method achieved an accuracy of 0.975 on a tenfold cross validation, using structural pattern features.

Gaps in Existing Works

Most of the previous works used accuracy as their main performance metric. This does not consider the cases, where the data set is unbalanced, thus causing a bias towards the class with the majority of instances. Another observation is that due to the size of the data set, the type and number of features used, the reported accuracy might be a result of overfitting. The techniques that use structural features only have a high rate of false negatives, whereas those that employ behavioural features are likely to have a high rate of false positives. On the other hand, the hybrid features might not provide the best-expected accuracy [12]. However, they offer significant benefits, as they provide a wide knowledge to the model useful for online and offline detection of the malware.

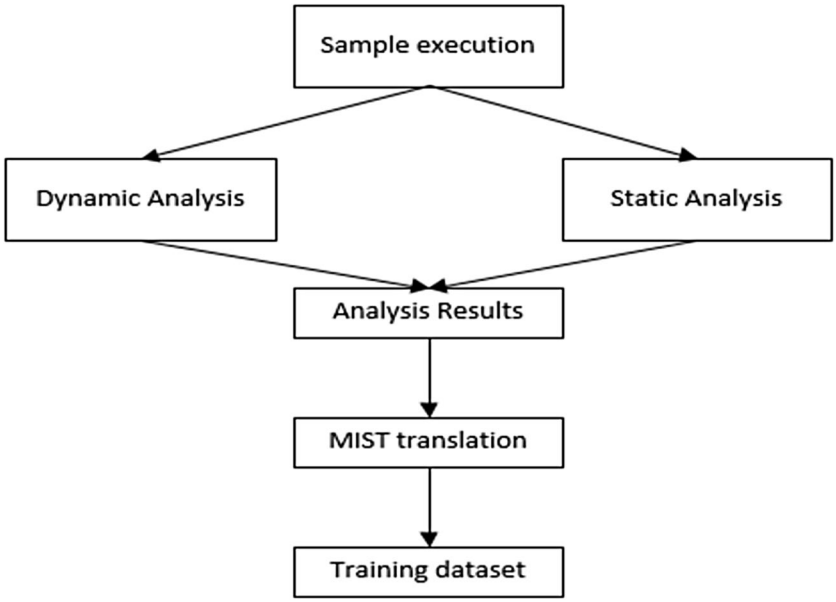

Fig. 1 Training data set generation [25]

Data Set and Feature Extraction

The used data set was provided by Marco Ramilli [25] and was named "*Malware Training Sets: A machine-learning data set for everyone*". It contained only malware samples of five different types. There are no benign files represented in this data set. In our future work, we will include benign samples. The types of malware in the given data set are:

Advanced persistence Threats (APT) malware use sophisticated techniques to exploit vulnerabilities in systems. Remote Administration Trojan (RAT) is widely used to achieve APT objectives. They can be controlled remotely and have the capability to persistently attack a specific target.

Crypto malware and locker malware a sub-category of ransomware, because both of them can blackmail the user. Crypto prevent data access from the targeted system. They use encryption to stop users from accessing their data and ask the user to pay for the encryption key.

Locker malware are mainly designed to lock and prevent access to the device interface, largely leaving data untouched.

Zeus malware are meant to steal banking information by collecting keystrokes logging and stilling information filled in forms.

Shadowblocker is an attacker group name which steals NSA developed cracking and backdoor creation tool. These malware are meant to exploit system vulnerabilities and can be monitored remotely. They use plugins to capture webcam and microphone output, log keystrokes as well as accessing other drives for surveillance purposes.

Table 1 Available malware types

Malware type	Number of samples
APT	292
Crypto	2020
Locker	431
Zeus	2019
Shadowbrokers	1270

Feature Extraction

Features were extracted using dynamic analysis and static analysis [25]. Malware Instruction Set (MIST) concepts provided by Trinius et al. [26] for behaviour-based analysis were followed to extract features. MIST is essentially an optimized representation for an efficient investigation of malware behaviours using data-mining and machine-learning techniques. After feature extraction, every sample's information was organized into a JSON format. Every feature characterizes the malicious act at some level. In the data set, pieces of evidence of each feature were identified for each malware sample and hashed using ELF[1] hash function. The overall process of training set generation is described in the flowchart, as shown in Fig. 1. The statistics of available malware types are given in Table 1.

[1] ELF is a mathematical function that converts a given input value into another cryptographic hash value of fixed length.

```
for /r . %%g in (*.json) do (
  set "var=%%g"
mongoimport --db test --collection test --file "%%g"
)
```

Fig. 2 Batch script to merge JSON chunk files in one file

Data Preparation

Merging JSON Files

The data set samples were in separate JSON files. We had to create one JSON file with all the samples together. We used MongoDB to achieve this objective. MongoDB [27] is a fast NoSQL cross-platform document database program that supports and uses JSON-based documents. It provides high availability, automatic scaling, and high performance. We created a simple batch script that was used to merge all the individual JSON sample files in one file, as shown in Fig. 2.

From JSON to CSV

We needed our data set in CSV format for further processing. Extracted features are shown in Table 2.

Exploratory Data Analysis

The exploratory analysis provides some useful insights about the main characteristics of the data such as feature correlations, or distributions. It gives a solid understanding of the data useful to prepare data for modelling as well as generating a hypothesis for further analysis [28]. The box plot in Fig. 3 describes some findings of this analysis. We see skewness and how the values in the data set are highly spread with many outliers. Outliers

Table 2 Extracted features

Features				
api_resolv	service_start	antivm_generic_disk	copies_self	fraudguard_threat_intel_api
cmd_exec	antianalysis_detectfile	antivm_generic_disk_setupapi	creates_largekey	infostealer_bitcoin
file_access	antiav_detectfile	antivm_generic_diskreg	creates_nullvalue	infostealer_browser
file_delete	antiav_detectreg	antivm_generic_system	critical_process	infostealer_ftp
file_drop	antiav_servicestop	antivm_vbox_files	cryptAM	infostealer_keylog
file_read	antidbg_devices	antivm_vbox_keys	deepfreeze_mutex	infostealer_mail
file_write	antidbg_windows	antivm_vbox_libs	deletes_self	injection_createremotethread
mutex_access	antiemu_wine_func	antivm_vmware_devices	deletes_shadow_copies	injection_runpe
net_con	antiemu_wine_reg	antivm_vmware_files	disables_browser_warn	injection_rwx
net_dns	antimalware_metascan	antivm_vmware_keys	disables_system_restore	internet_dropper
net_http	antisandbox_cuckoocrash	antivm_vpc_files	disables_uac	ipc_namedpipe
pe_imports	antisandbox_productid	antivm_vpc_keys	disables_windows_defender	mimics_agent
pe_sec_character	antisandbox_restart	antivm_xen_keys	disables_windowsupdate	mimics_filetime
pe_sec_entropy	antisandbox_sboxie_libs	api_spamming	downloader_cabby	mimics_icon
pe_sec_name	antisandbox_sleep	banker_zeus_mutex	dridex_behavior	modifies_certs
reg_access	antisandbox_sunbelt_libs	banker_zeus_url	driver_load	modifies_desktop_wallpaper
reg_delete	antisandbox_suspend	bcdedit_command	dropper	modifies_hostfile
reg_read	antisandbox_unhook	bootkit	encrypted_ioc	modify_proxy
reg_write	antivirus_virustotal	browser_security	exec_crash	modify_security_center_warnings
service_create	antivm_generic_bios	bypass_firewall	stealth_webhistory	modify_uac_prompt
recon_checkip	recon_fingerprint	clamav	stealth_window	virus
multiple_useragents	ransomware_files	network_http	ransomware_recyclebin	origin_langid
network_bind	ransomware_message	network_torgateway	rat_spynet	origin_resource_langid
network_cnc_http	ransomware_radamant	office_security	reads_self	packer_entropy
packer_upx	recon_programs	persistence_autorun	static_pe_anomaly	process_interest
packer_vmprotect	removes_zoneid_ads	persistence_service	static_versioninfo_anomaly	process_needed
persistence_ads	sniffer_winpcap	polymorphic	stealth_file	ransomware_extensions
str	recon_beacon	pony_behavior	stealth_hidden_extension	ransomware_file_modifications
stealth_hiddenreg	stealth_hide_notifications	stealth_network	stealth_timeout	

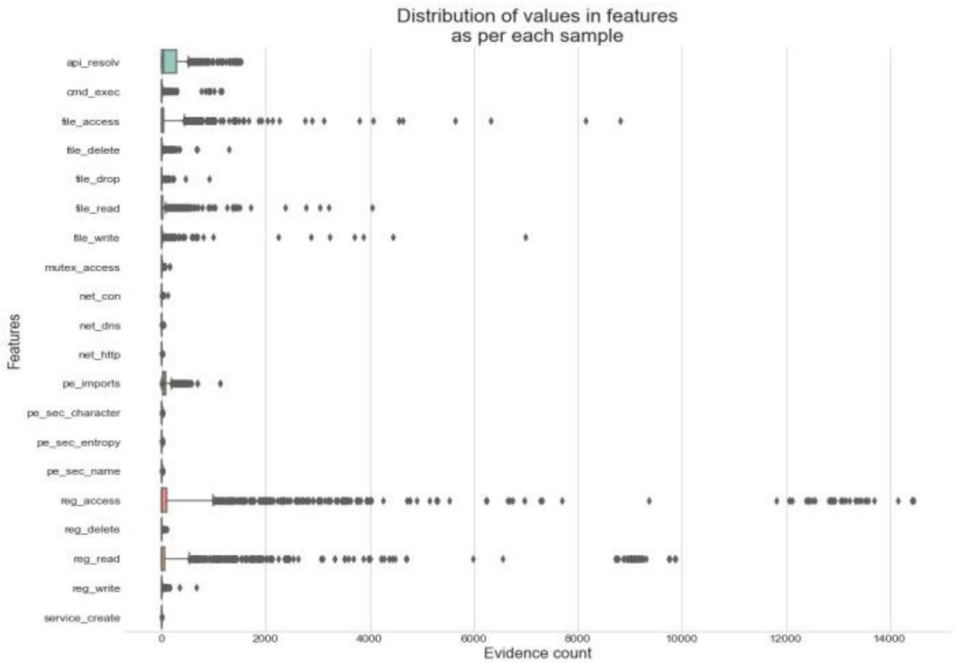

Fig. 3 Box plot analysis of the first 20 features

are values that drastically deviate from others in the data set. All observations below $Q1 - 1.5(Q3 - Q1)$ or above $Q3 + 1.5(Q3 - Q1)$ are outliers, where Q1 and Q3 are first and third quartiles, respectively. This situation once again implies the need for standardizing our data before further processing.

From the observations made during the exploratory analysis, we discovered that the features were not well represented and that no algorithm could work properly with poorly designed features. The next section focuses on experiments to present a new feature engineering approach that will lead to the creation of a good model, capable of performing well in terms of classification accuracy and very short processing time.

Feature Engineering

It is all about features [10]. Informative, independent, and discriminating features are crucial for the creation of effective malware classification models. To get such features, the researcher created a new feature engineering (NFE) approach that uses the domain knowledge of the data to manipulate and create features that yield bigger gains in performance. The presented approach should be able to handle situations such as handling imbalanced data sets, avoid overfitting, and reduce high rates of false positives and false negatives. This approach should as well be able to handle features with incomplete domain knowledge representation and sparse features.

In this section, we present a set of steps used to create an NFE approach. This process comprises algorithms for advanced pre-processing and the design of the NFE approach.

Data Pre-processing

The data set has a lot of null entries as well as impurities. This task consists of a series of subtasks aiming at cleaning our data set and makes it ready for use. In each feature, pieces of evidence were provided. There were features with empty evidence. It means that a particular sample is not concerned with the specific behaviour represented by that feature. Some other samples have many pieces of evidence for various features. For example, one sample was having file access (d41d8cd9 913f9c49 96da6d37), which means that there were three pieces of evidence of file access activity recorded during the analysis of that particular sample. One of the data-cleaning tasks was to transform the data set by counting the number of pieces of evidence in each feature per sample.

The outcome was $E_{\text{num}} \geq 0$, where E_{num} is the number of pieces of evidence of each feature per sample. All null values were replaced with 0, because they represented the absence of evidence in a particular situation. The process is described by Algorithm 1. This step led to a new version of the data set composed of 4286 malware files and 89 features.

Algorithm 1: *Advanced Data pre-processing*

```
Require: DSunclean //raw dataset
Ensure: DSclean //cleaned dataset
PROCEDURE cleanData
nrows ← size(DSunclean)
nfeatures ← length(DSunclean)
nrows2 ← 0
nfeatures2 ← 0
if nrows = 0 then
  return NULL
else
  for row = 1 to nrows STEP 1 do
    if DSunclean[row] = nil then
      drop DSunclean[row]
      nrows2 ← size(DSunclean)
    end if
  end for
  for col = 1 to nfeatures STEP 1 do
    if DSunclean[col] = nil then
      drop DSunclean[col]
      nfeatures2 ← length(DSunclean)
    end if
  end for
  for row = 1 to nrows2 STEP 1 do
    for col = 1 to nfeatures2 STEP 1 do
      if DSunclean[row][col] = nil then
        pass
      else
        w ← DSunclean[row][col].split(" ")
// Counting evidences per feature and per sample
        Enum ← length(w)
        DSclean[row][col] ← Enum
      end if
    end for
  end for
end if
RETURN DSclean
```

NFE Approach Design

The steps involved in designing a new Feature Engineering (NFE) approach are as follows:

1. Given a data set DS of dimension d and size s, where d is the total number of features and s is the number of samples in the data set. We denote the initial feature set as:

$$F = \{f_0, f_1, f_2, f_3, \ldots, f_i\}, \text{ where } 0 \leq i < d. \quad (1)$$

Given the feature f_i, count its cumulative total number of occurrences in the data set.

We denote total occurrence as $occ(f_i)$. Occurence is computed as follows:

Algorithm 2: Computing occurances

```
Occ ← 0
for col = 1 to d STEP 1 do
  for row = 1 to s STEP 1 do
    if DS[col][row] =!nil then
      occ ← occ + 1
    else
      pass
    end if
  end for
end for
```

2. Compute initial feature importance for each feature f_i based on Gradient Boost algorithm. We denote initial importances as:

$$imp(f_i). \quad (2)$$

3. Create initial feature ranking according to their importances. We denote initial ranks as:

$$rank(f_i). \quad (3)$$

4. Compute the estimator value $e(f_i)$ for each feature. This value is computed based on feature occurrences and their ranks.

$$e(f_i) = \frac{occ(f_i)}{rank(f_i)}. \quad (4)$$

5. Find the strength of the association between initial feature importance $imp(f_i)$ and their estimators $e(f_i)$ represented by the correlation $p(I, E)$ as shown below:

$$p(I,E) = \frac{\sum_{i=1}^{d} \left(imp(f_i) - \overline{imp}\right)\left(e(f_i) - \bar{e}\right)}{\left(\sum_{i=1}^{d} \left(imp(f_i) - \overline{imp}\right)^2 \sum_{i=1}^{d} \left(e(f_i) - \bar{e}\right)^2\right)^{1/2}}, \quad (5)$$

where $\overline{imp}$ is the mean of all initial feature importances. $\bar{e}$ is the mean of all feature estimators. d is the dimension of the data set.

Table 3 Top 11 important features and their ranking according to GB and NFE

GB features					NFE features				
GB Features	GB ranking	NFE ranking	Importances (GB)	Importances (NFE)	NFE features	GB ranking	NFE ranking	Importances (GB)	Importances (NFE)
api_resolv	1	1	0.133	0.290	api_resolv	1	1	0.133	0.290
pe_imports	2	2	0.115	0.187	pe_imports	2	2	0.115	0.187
str	3	3	0.101	0.128	str	3	3	0.101	0.128
antivirus_virustotal	4	4	0.100	0.072	antivirus_virustotal	4	4	0.100	0.072
clamav	5	13	0.052	0.016	packer_entropy	7	5	0.034	0.043
packer_entropy	6	7	0.037	0.032	reg_read	8	6	0.032	0.036
reg_access	7	5	0.034	0.043	file_read	6	7	0.037	0.032
file_read	8	6	0.032	0.036	file_access	9	8	0.028	0.031
reg_read	9	8	0.028	0.031	pe_sec_name	11	9	0.022	0.026
static_detection	10	22	0.026	0.003	persistence_autorun	17	10	0.014	0.023
file_access	11	9	0.022	0.026	file_drop	21	11	0.011	0.018
…					…				

6. Compute the new NFE feature importances as follows:

$$imp_{nfe}(f_i) = f\left(e(f_i), p(I,E)\right) = \frac{e(f_i) * p(I,E)}{d}, \quad (6)$$

where d is the dimension or total number of features.

7. Normalize individual features to the range between 0 and 1.

$$Norm\left(imp_{nfe}(f_i)\right) = \frac{imp_{nfe}(f_i)}{\sum_{i=1}^{d} imp_{nfe}(f_i)}, \quad (7)$$

where $0 \leq i < d$.

8. Create the new feature ranking $rank_{nfe}(f_i)$ based on newly computed importances $imp_{nfe}(f_i)$.

   The higher $imp_{nfe}(f_i)$, the higher $rank_{nfe}(f_i)$.

9. If $c < d > 30$:

   Select first k top ranking NFE features $f_0^{'}, f_1^{'}, f_2^{'}, \ldots, f_k^{'}$ based on $imp_{nfe}(f_i)$, where c is the number of classes, d is the total number features, and $k <= 30$.

   New NFE feature set is:

$$F_{nfe} = \{f_0^{'}, f_1^{'}, f_2^{'}, \ldots, f_k^{'}\} \quad (8)$$

   For a fast training process, we propose that the new features to be selected should not be more than 30 if they provide a good performance.

10. Use the selected features in step 9 to create the model. If the performance of the model is poor, derive more features as shown in the next steps. The researcher assumes that the accuracy less than 90% can be considered as poor performance in malware classification and detection problems.

At step 6 above, the new importances will produce a new ranking. This NFE approach should be able to automatically select the best features based on assumptions in steps 10.

Depending on the reported performance, features obtained at step 9 can be retained if a satisfactory level was achieved.

Experiments and Results

Impact of NFE on Features

NFE computes new feature importances, thus creating a new ranking and making it easy to select the most important features in a different way. The top 11 important features and their ranking according to GB and NFE are described in Table 3. Figure 4 presents a comparison between initial feature importances based on GB and new feature importances produced by NFE technique on the data set.

To further understand the changes in features brought in by NFE, Table 4 shows the % of affected features. Figure 5 shows the number of features replaced when applying NFE to initial features. Figure 6 shows the changes in feature ranking and the total % affected.

To verify if the given GB and NFE rankings differ significantly, Wilcoxon's signed-rank test and the Spearman's rank correlation coefficient have been calculated using data in Tables 3 and 4. The null hypothesis (H0) stipulates that there is no strong difference between rankings of both GB and NFE. The alternative hypothesis on the other hand stipulates

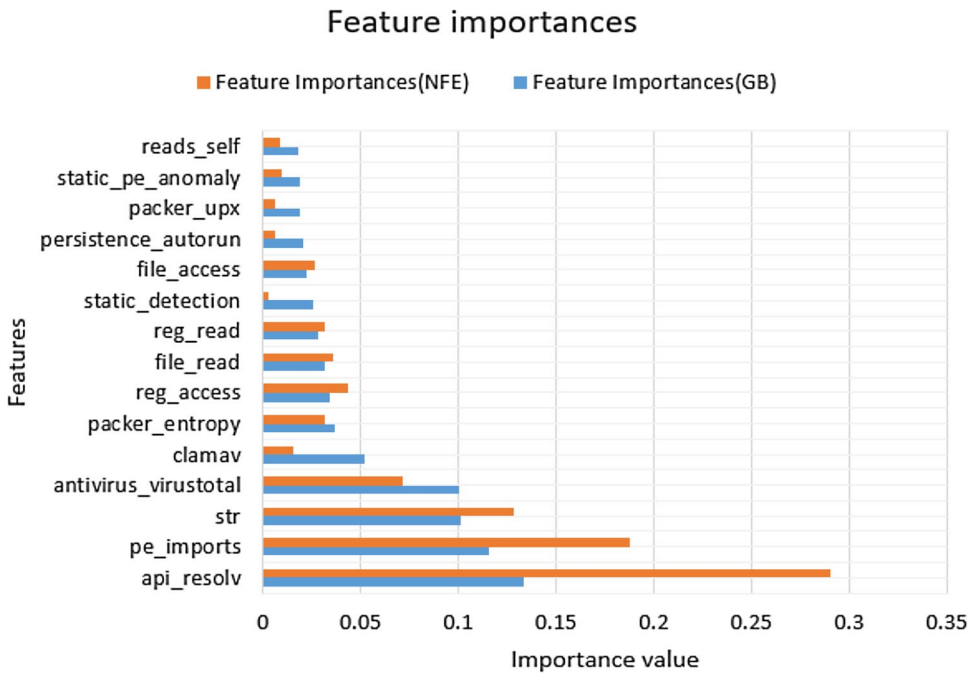

Fig. 4 Comparison between NFE and GB feature importances

Table 4 % of features affected by NFE

Top initial features by GB	Number of replaced features by NFE	Number of unaffected rankings	Number of features unaffected by NFE	% of affected feature ranking by NFE
5	1	4	4	20.0
10	2	4	8	60.0
15	3	6	12	60.0
20	3	6	17	70.0
25	4	6	21	76.0
30	3	6	27	80.0
35	4	6	31	82.9
60	8	6	52	90.0
70	6	6	64	91.4
80	2	6	78	92.5
89	0	6	89	93.3

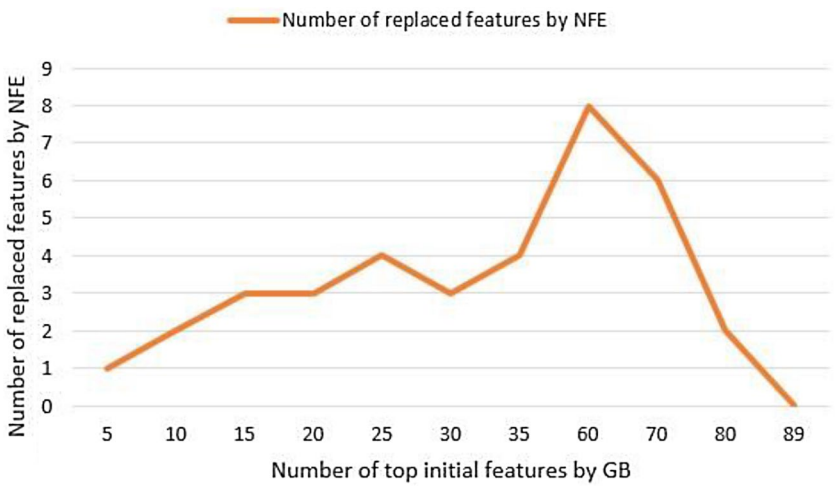

Fig. 5 Impact of NFE on features

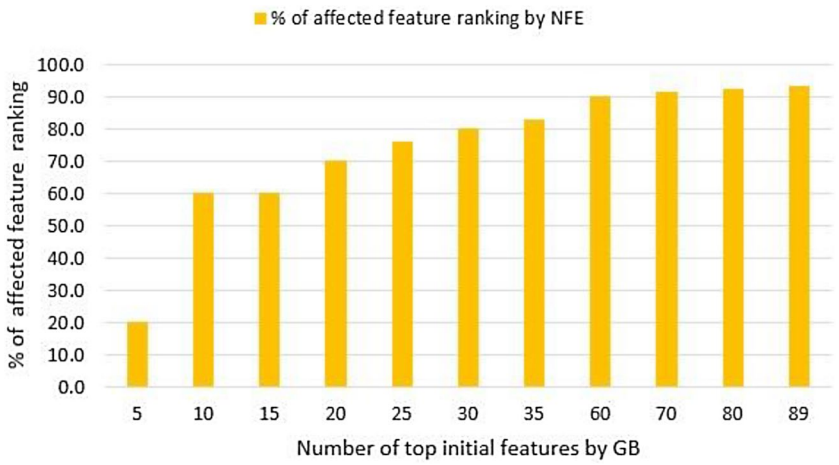

Fig. 6 NFE feature ranking

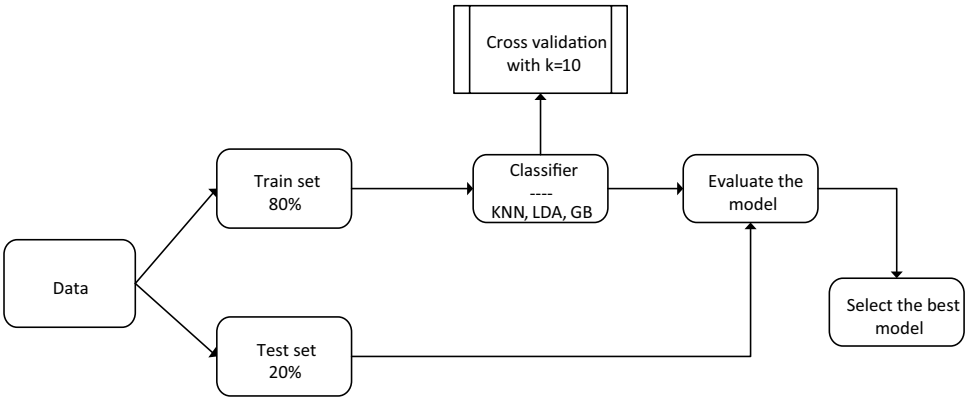

Fig. 7 Schematic overview of the classification model building process

that there is a significant statistical difference between the rankings of features produced by GB and NFE.

First, in Table 3, both GB and NFE rankings are given for only top 11 features among all 89 features. However, the statistical tests have been run on all 89 features. To perform Wilcoxon's signed-rank test, the following parameters were used: $n = 89$ (total number of features) and $\alpha = 0.05$ (confidence level). We calculated the test static $T_{stat} = 1612$. We used the Wilcoxon Signed-Ranks Table [29] to find the critical values for the T statistic at $n = 83$, because there were 6 features, whose differences in ranks were 0. From this table, we find that $T_{crit} = 1311$ (two-tail test). Since $T_{crit} = 1311 < 1612 = T_{stat}$, we cannot reject the null hypothesis (i.e., $p \geq 0.05$), and so conclude that there is no significant statistical difference between the rankings provided by GB and NFE. We also calculated the Spearman's rank correlation coefficient ($r = 0.81$) which shows a strong correlation among the rankings of GB and NFE.

Second, to further supplement this analysis, we repeated both tests using observations in Table 4. This table shows changes that occurred during the ranking process in given intervals. We tried to understand if there is a significance between initial GB features and the number of features unaffected by NFE. We use $\alpha = 0.05$ and $n = 11$. We calculated the test static $T_{stat} = 1$. From the Wilcoxon Signed-Ranks Table, we find that $T_{crit} = 10$ (two-tail test) at $n = 11$. Since $T_{crit} = 10 > 1 = T_{stat}$, we reject the null hypothesis, and so conclude that there is a significant statistical difference between the changes produced by rankings of both GB and NFE. The computed Spearman's rank correlation coefficient ($r = 0.24$) which shows a weak correlation. Further experiments were conducted to verify if these changes make a difference in improving classification performance.

We also conducted other experiments to classify the malware and assess the performance of the classifier, given different conditions of the data set and features. Our problem falls in the multi-class categories, where more than two prediction values were provided. There were five classes of malware to be predicted. We adopted supervised machine-learning algorithms that have the capabilities of handling multi-class prediction problems such as Linear Discriminant Analysis (LDA), KNN, and the Gradient Boosting Classifier. The reason for using different algorithms in our experiments is to try to test different options and adopt the one that offers the best results in a particular situation.

Table 5 Performance using raw features on the unbalanced data set

Model	Accuracy	Precision	Recall	f1-score
GB	**0.82**	**0.82**	**0.84**	**0.81**
KNN	0.59	0.6	0.6	0.6
LDA	0.756	0.76	0.78	0.75

The bold was showing the best performances achieved by the best classifier in various experiments

Table 6 Performance using raw features on the balanced data set

Model	Accuracy	Precision	Recall	f1-score
GB	**0.87**	**0.82**	**0.84**	**0.81**
KNN	0.74	0.6	0.59	0.59
LDA	0.67	0.67	0.76	0.67

The bold was showing the best performances achieved by the best classifier in various experiments

Table 7 Performance using NFE features on the unbalanced data set

Model	Accuracy	Precision	Recall	f1-score
GB	**0.93**	**0.93**	**0.93**	**0.93**
KNN	0.68	0.7	0.69	0.69
LDA	0.77	0.74	0.77	0.73

The bold was showing the best performances achieved by the best classifier in various experiments

Table 8 Performance using NFE features on the balanced data set

Model	Accuracy	Precision	Recall	f1-score
GB	**0.94**	**0.94**	**0.94**	**0.94**
KNN	0.79	0.66	0.66	0.66
LDA	0.68	0.7	0.81	0.74

The bold was showing the best performances achieved by the best classifier in various experiments

The given data set was split into two parts: 20% for testing and 80% for training. The 20% test set was hidden from the learning algorithm and later used for the final model evaluation. To validate the performance of our model during the training phase, we applied a tenfold cross-validation technique on the training set. These concepts are highlighted in Fig. 7.

Experiments were done using raw features and new features generated by our NFE approach. Given that the data set was imbalanced, i.e., samples for each category of malware were not distributed proportionally, we also considered the two scenarios (unbalanced and balanced). To create a balanced version of the data set, we adopted the Synthetic Minority Over-Sampling Technique (SMOTE). This technique helped to oversample by generating synthetic data and increasing the number of minority classes.

Experiment 1: Using all Features

This experiment was conducted using all original features in the data set. On the imbalanced data set, the results obtained are shown in Table 5. Using all features on a balanced data set slightly improved the performances, as shown in Table 6. The accuracy of Gradient Boost Classifier has increased by 5%.

Experiment 2: Using NFE Features

Given that the performance attained in the previous experiment 1 is practically unacceptable, especially in malware classification situation, we adopted NFE approach to get better performance. The performances achieved both on the imbalanced and balanced data sets are shown in Tables 7 and 8, respectively.

In Fig. 8, we show the positive impact of NFE features, where the performance Gradient Boost has been significantly improving in all the experiments' scenarios.

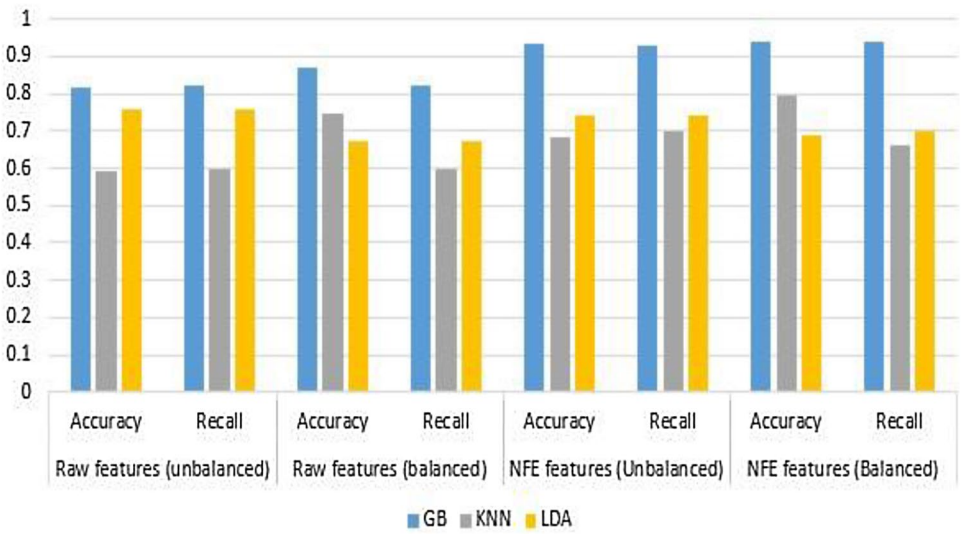

Fig. 8 Impact of NFE features on improving model classification performance

Impact of NFE on Malware

The number of correctly and incorrectly classified by GB model based on NFE is shown in Table 9. Based on results reported in Table 10, we see that shadowbroakers are the more affected by NFE, where among all samples classified as shadowbroakers, the model has the ability to specifically classify the real ones with a high precision of 99%. Again, based on the recall, we see that the model reached a level of 97%. This means that our model is very sensitive on classifying the real shadowbroakers, with the low False Positive rate of 0.03.

APT are the least affected by NFE, where among all samples classified as APT, the model has the ability to specifically classify the real APT with a high precision of 77%. Again, based on the recall, we see that the model reached a level of 87%. This means that our model is not very sensitive on classifying the real APT, with the False Positive rate of 0.22.

Comparing NFE with Other Feature Selection Techniques

One of NFE's strengths is the selection of features. This approach was compared to other three feature selection techniques to assess its robustness. The chosen techniques are Analysis of Variance (ANOVA), Recursive Feature Elimination (RFE), and Principal Component Analysis (PCA).

Analysis of variance is used to analyse the statistical difference among group means in a sample. In the context of feature selection, ANOVA analyzes the significance between each feature and the target vector. If features are categorical, ANOVA is done using Chi square. If features are numerical, F scores are computed for each feature, where the higher the score, the more important the feature. To implement ANOVA, this research used F scores, because the processed data set features were numeric.

RFE technique uses the model accuracy to find the features that have more prediction power. It works by recursively removing features and building a model on the remaining ones.

PCA is a data reduction technique that transforms a data set into a more compressed form using linear algebra. The number of principal components chosen corresponds to the new number of features.

Results obtained are shown in Table 11. This table shows classification performance after selecting the top 30 relevant features as given by different techniques. The given results were obtained using GB classifier.

Of the four techniques presented, NFE achieved the best results for the 30 most discriminating features. NFE is slightly better than ANOVA in terms of precision, recall, and F score, but more accurate than ANOVA. PCA gave poor performances in all measurements. All these differences are illustrated in Fig. 9.

To further assess the effectiveness of NFE on feature selection, we conducted experiments using features selected by GB itself. The top 10, 20, 30, and 40 features were used. GB best classifier was used to build the model. Comparison results are given in Table 12. It can be seen that NFE has made a slight improvement in various feature selection settings. 94% has been the highest achievement for all performance metrics. NFE first reached this value achieved on precision, recall, and F score after only 20 top features. The same value has also been first reached by NFE after 30 to features. After 30 features, no much improvement has been noted on the highest performance.

Table 9 Confusion matrix showing correctly and incorrectly classified samples

	Predicted				
	APT	Crypto	Locker	Shadow brokers	Zeus
APT	41	2	0	0	4
Crypto	3	364	2	1	11
Locker	6	9	78	0	3
Shadow brokers	1	0	3	120	0
Zeus	2	3	4	0	201

Discussion

The above reported results showed the improvement in the classification performance of malware. The performance of the best model increased up to around 12% on both accuracy and recall, as shown in Tables 5 and 8. The proposed NFE approach allowed to achieve such results, because the success of machine learning depends mostly on the features used.

Table 10 Various performance metric's scores on each malware family

	FN	FP	% RECALL	Precision	FN%	FP%
APT	6	12	0.872	0.774	0.128	0.226
Crypto	17	14	0.955	0.963	0.045	0.037
Locker	18	9	0.813	0.897	0.188	0.103
Shadow brokers	4	1	0.968	0.992	0.032	0.008
Zeus	9	18	0.957	0.918	0.043	0.082

Table 11 Comparison of GB classifier performances between NFE, ANOVA, RFE, and PCA

Feature selection technique	Accuracy	Precision	Recall	*F* score
ANOVA	91.1	93	93	93
RFE	73.7	78	76	75
PCA	62.9	56	55	54
NFE	**94**	**94**	**94**	**94**

The bold was showing the best performances achieved by the best classifier in various experiments

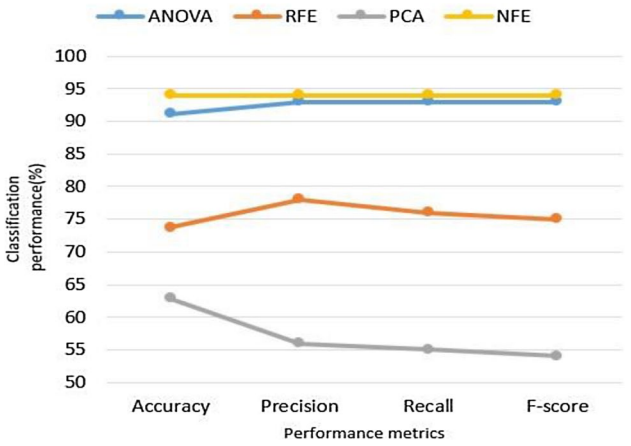

Fig. 9 Comparison between NFE, ANOVA, PCA, and RFE on feature selection performances

Most previous studies used accuracy as the main performance metric. However, this might not be enough depending on the nature of the data set. For example, when the data set is imbalanced, where the class proportions are largely different, accuracy will not be a good metric for evaluation, because high values will be by default given to the dominating class. To deal with this problem, we introduced an over-sampling technique that allowed us to create a balanced data set for better representation of all given classes. This, therefore, contributed to more stable values of accuracies. We also used recall in our evaluation to measure the sensitivity of the built classifier in classifying relevant classes of malware. This helps evaluate the rate of false negatives. As of given results, the GB classifier has been able to classify malware with an accuracy of 0.94% and a recall of 94%. This has been possible using features created by a new technique referred to as NFE in this paper.

We also found out that the real comparison with earlier works is not easy due to the fact that data sets or features used are different. However, the primary contribution of this research was to find a feature engineering approach that uses a hybrid of structural and behavioural features (see Table 2)

Table 12 Comparison of GB classifier performances between top selected GB and NFE features

Selected features	GB classifier performances			
	Accuracy	Precision	Recall	*F* score
GB features				
Top 10	0.83	0.81	0.81	0.8
Top 20	0.93	0.93	0.93	0.93
Top 30	0.93	0.94	0.94	0.94
Top 40	0.94	0.94	0.94	0.94
NFE features				
Top 10	0.91	0.92	0.92	0.92
Top 20	0.93	**0.94**	**0.94**	**0.94**
Top 30	**0.94**	0.94	0.94	0.94
Top 40	0.94	0.94	0.94	0.94

The bold was showing the best performances achieved by the best classifier in various experiments

to increase the performance in classifying malware, including polymorphic malware. NFE performed well on feature selection compared to other techniques, as shown in Tables 11 and 12.

Conclusion and Future Work

A good malware classification system should be able to classify even the polymorphic variants that can appear in any form. Machine-learning works well in building the classification models. However, for any algorithm to perform at its best, it requires a data set that is well balanced, highly preprocessed and cleaned. This can be accomplished through an advanced feature engineering that produces optimal input features. This work proposed a new feature engineering approach (NFE), which yielded good results on a complex data set in classifying polymorphic malware and can as well be generalized to metamorphic malware. The achieved performances in multiple scenarios of the experiments by the Gradient Boosting Classifier were 81%, 87%, 93%, and 94%. There was an improvement of 12% in accuracy. NFE also outperformed ANOVA, PCA, RFE, and GB on feature selection. In the future, we will improve NFE to increase current performance limits. NFE will also be boosted with feature derivation capabilities. We will also apply NFE on a data set composed of both malware and benign samples, in order create improved malware detection models.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bhuiyan ZA, Wang T, Hayajneh T, Weiss GM. Maintaining the Balance between Privacy and Data Integrity in Internet of Things. In: Proceedings of the 2017 international conference on management engineering, software engineering and service sciences, 2017.
2. McKenna B. Symantec's Thompson pronounces old style IT security dead. Netw Secur. 2016;2:1–3.
3. Unuchek R, Sinitsyn F, Parinov D, Liskin A. IT threat evolution Q3 2017. Statistics. 2017. https://securelist.com/it-threat-evolution-q3-2017-statistics/83131/. Accessed 27 Nov 2017.
4. Chau M, Alan Wang G, Chen H. A syntactic approach for detecting viral polymorphic malware variants. Lecture notes computer science (including its subseries lecture notes in artificial intelligence (LNAI) and lecture notes in bioinformatics), vol. 9650, no. April, 2016.
5. Masabo E, Kaawaase KS, Sansa-otim J, Ngubiri J. A state of the art survey on polymorphic malware analysis and detection techniques. ICTACT J Soft Comput 2018;8(4):1762–74.
6. Kumar A, Kuppusamy KS, Aghila G. A learning model to detect maliciousness of portable executable using integrated feature set. J King Saud Univ Comput Inf Sci 2019;31(2):252–65.
7. Jiang Q. A feature selection method for malware detection. In: Proceeding IEEE International Conference on Information and Automation, no. June, pp. 890–895, 2011.
8. Lin C-T. Feature selection and extraction for malware classification. J Inf Sci Eng. 2015;31:965–92.
9. VanderPals J. Python data science handbook | python data science handbook. Sebastopol: O'Reilly; 2016.
10. Feffer S. It's all about the features. 2017. https://www.reality.ai/single-post/2017/09/01/It-is-all-about-the-features. Accessed 22 Nov 2017.
11. Dornhack H, Kadletz K, Luh R, Tavolato P. Malicious behavior patterns. In: 2014 IEEE 8th international symposium, pp. 384–389, 2014.
12. Damodaran A, Di Troia F, Visaggio CA, Austin TH, Stamp M. A comparison of static, dynamic, and hybrid analysis for malware detection. J Comput Virol Hacking Tech. 2017;13(1):1–2.
13. Naidu V. Using different substitution matrices in a string-matching technique for identifying viral polymorphic malware variants. In: 2016 IEEE congress on evolutionary *computation* (CEC), pp. 2903–2910, 2016.
14. Narayanan A, Chen Y, Pang S, Tao B. The effects of different representations on static structure analysis of computer malware signatures. Sci World J. 2013;2013:671096.
15. Drew J, Hahsler M, Moore T. Polymorphic malware detection using sequence classification methods and ensembles. EURASIP J Inf Secur. 2017;2017(1):2.
16. Naidu V, Narayanan A. Needleman–Wunsch and Smith–Waterman Algorithms for Identifying Viral Polymorphic Malware Variants. In: 2016 IEEE 14th international conference on dependable, Autonomic and Secure Computing, 14th international conference on pervasive intelligence and computing, 2nd international conference on big data intelligence and computing and cyber science and technology congress, no. August, pp. 326–333, 2016.
17. Sharma P, Kaur S, Arora J. An advanced approach to polymorphic/metamorphic malware detection using hybrid clustering approach. Int Res J Eng Technol. 2016;3(6):2229–32.
18. Arshi D, Singh M. Behavior analysis of malware using machine learning. In: 2015 eighth international conference on contemporary computing (IC3), 2015, pp. 481–486.
19. Ahmadi M, Sami A, Rahimi H, Yadegari B. Malware detection by behavioural sequential patterns. Comput Fraud Secur. 2013;2013(8):11–9.
20. Fraley JB, Figueroa M. Polymorphic malware detection using topological feature extraction with data mining. SoutheastCon. 2016;2016:1–7.
21. Kaur R, Singh M. Efficient hybrid technique for detecting zero-day polymorphic worms. In: Souvenir of the 2014 IEEE international advance computing conference, IACC, no. September 2011, pp. 95–100, 2014.
22. Saleh M, Li T, Xu S. Multi-context features for detecting malicious programs. J Comput Virol Hacking Tech. 2018;14(2):181–93.
23. Farrokhmanesh M, Hamzeh A. Music classification as a new approach for malware detection. J Comput Virol Hacking Tech. 2018;15:77–96.
24. Gibert D, Mateu C, Planes J, Vicens R. Using convolutional neural networks for classification of malware represented as images. J Comput Virol Hacking Tech. 2018;15:15–28.
25. Ramilli M. Malware training sets: a machine learning dataset for everyone. 2016. http://marcoramilli.blogspot.it/2016/12/malware-training-sets-machine-learning.html. Accessed 05 Oct 2017.
26. Trinius P, Willems C, Holz T, Rieck K. A malware instruction set for behavior-based analysis. In: Sicherheit Schutz und Zuverlässigkeit SICHERHEIT, no. TR-2009-07, pp. 1–11, 2011.
27. Truică CO, Boicea A, Trifan I. CRUD Operations in MongoDB. In: International conference on advanced computer science and information systems (ICACSEI 2013), no. ICACSEI, pp. 347–350, 2013.
28. Willems K. Python exploratory data analysis tutorial. https://www.datacamp.com/community/tutorials/exploratory-data-analysis-python. Accessed 30 Nov 2017.
29. Zaiontz C. Wilcoxon signed-ranks test. 2019. http://www.real-statistics.com/non-parametric-tests/wilcoxon-signed-ranks-test/. Accessed 25 Jul 2019.