Discover Internet of Things

Discover

Case Study

Is iterative feature selection technique efficient enough? A comparative performance analysis of RFECV feature selection technique in ransomware classification using SHAP

Rawshan Ara Mowri[1] · **Madhuri Siddula**[1] · **Kaushik Roy**[1]

Received: 29 May 2023 / Accepted: 15 November 2023
Published online: 30 November 2023

Abstract
The realm of cybersecurity places significant importance on early ransomware detection. Feature selection is critical in this context, as it enhances detection accuracy, mitigates overfitting, and reduces training time by eliminating irrelevant and redundant data. However, iterative feature selection techniques tend to select the best-performing subset of features through an iterative process which leaves chance for a crucial feature not being selected and the number of selected features may not always be the optimal or the most suitable for a given problem. Hence, this study aims to conduct a performance comparison analysis of an iterative feature selection technique- Recursive Feature Elimination with Cross-Validation (RFECV) with six supervised Machine Learning (ML) models to evaluate its efficiency in classifying ransomware utilizing the Application Programming Interface (API) call and network traffic features. The study employs an Explainable Artificial Intelligence (XAI) framework called SHapley Additive exPlanations (SHAP) to derive the crucial features when RFECV is not integrated with the ML models. These features are then compared with RFECV-selected features when it is integrated. Results show that without RFECV the ML models achieve better classification accuracies on two datasets. Again, RFECV falls short of selecting impactful features, leading to more false alarms. Moreover, it lacks the capability to rank the features based on their importance, reducing its efficiency in ransomware classification overall. Thus, this study underscores the importance of integrating explainability techniques to identify critical features, rather than solely relying on iterative feature selection methods, to enhance the resilience of ransomware detection systems.

Keywords Dynamic analysis · Feature engineering · Explainable artificial intelligence · Ransomware · Cyber security

Abbreviations

API	Application programming interface
AIS	Artificial immune system
CFG	Call flow graph
CF-NCF	Class frequency non-class frequency
DoS	Denial-of-Service
DGA	Domain generation algorithm
DNS	Domain name system
XAI	Explainable artificial intelligence

Madhuri Siddula and Kaushik Roy contributed equally to this work.

✉ Rawshan Ara Mowri, rmowri@aggies.ncat.edu; Madhuri Siddula, msiddula@ncat.edu; Kaushik Roy, kroy@ncat.edu | [1]Department of Computer Science, North Carolina A &T State University, Greensboro, NC 27411, USA.

EDA	Exploratory data analysis
FN	False negative
FNR	False negative rate
FPR	False positive rate
FP	False positives
KNN	K-nearest neighbor
LR	Logistic regression
LSTM	Long short-term memory
ML	Machine learning
MLP	Multilayer perceptron
NB	Naïve Bayes
OvO	One-vs-One
OvR	One-vs-Rest
PCAP	Packet capture
RF	Random forest
RFECV	Recursive feature elimination with cross-validation
SHAP	SHapley Additive exPlanations
SDN	Software defined networking
SGD	Stochastic gradient descent
SVM	Support vector machine
TCP	Transmission control protocol
TN	True negative
TP	True positives
VM	Virtual machine
PE	Windows portable executable

1 Introduction

Ransomware is a harmful software that applies symmetric and asymmetric cryptography to inscribe user information and poses a Denial-of-Service (DoS) attack on the intended user [1]. The unique functional process of ransomware attacks makes it more harmful than any malware attacks and causes irreversible losses. Crypto-viral Extortion', which is the functional process of ransomware, includes three main steps [2] as depicted in Fig. 1. In the initial step, the attacker creates a key pair that incorporates a private key K1 and a public key K2, puts the public key K2 in the ransomware, then, at that point, launches the ransomware. After entering a computer, in the second step, the ransomware activates itself and produces an arbitrary symmetric session key K3 to encrypt the victim's files or data. Next, the ransomware utilizes K2 to encrypt K3 and to create a small irregular ciphertext E1. Then, the ransomware zeroizes K3 and the plaintext from the person's drive. A communication bundle P1 containing previously generated E1, a payment note M, and a medium to contact the attacker, is then created. After that, the ransomware informs the victim of the attack and demands payment via a transaction medium within a set amount of time in order to decrypt the files by displaying the payment note M. At the final step, as the payment is completed, the communication bundle P1 is adjusted to P2 containing just the deviated ciphertext E1 and steered back to the attacker. The attacker gets P2, decrypts E1 with K1, and gets K3 which is then sent back to the victim to decrypt the files. Finally, upon receiving K3, the victim decrypts the files. Usually, the victim pays the ransom using untraceable cryptocurrency [3]. However, paying the ransom doesn't guarantee that the decryption key could secure the encrypted files, which could be the worst scenario of any type of ransomware attack [4].

Supported by a report by Symantec in 2015, there are two types of ransomware [5]-

- Locker ransomware: denies access to the system or device
- Crypto ransomware: denies access to the files or data

However, according to [6], based on the functionalities, ransomware is categorized into four groups-

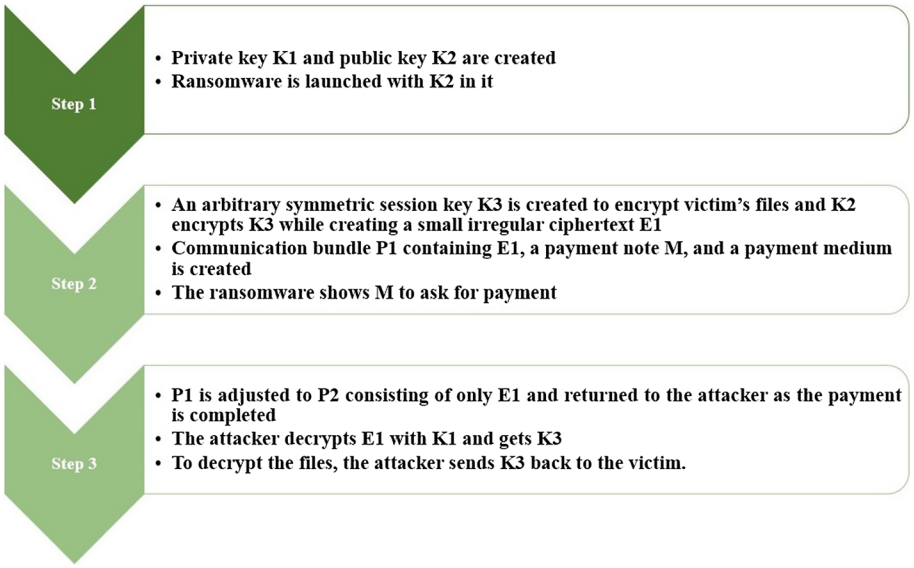


Fig. 1 Workflow of a ransomware

- Encrypting ransomware: encrypts and denies access to the victim's files and data (i.e., AIDS Trojan, CryptoLocker, WannaCry, CryptoWall) [6]
- Non-encrypting ransomware: doesn't do encryption but rather threatens to try if the ransom is not paid (i.e., WinLock, NotPetya) [6]
- Leak-ware: doesn't do encryption instead claims to reveal stolen information from the victim's system if the ransom is not paid [7]
- Mobile ransomware: targets the Android platform [8]

All these categories of ransomware are playing a vital role in the recent upsurge in the incidence of ransomware attacks. Due to the increasing number of ransomware variants and ransomware attacks, researchers have been earnestly involving themselves in looking for efficient ways to improve the scenarios. While some researchers are analyzing the distinctive behaviors of ransomware by executing it in a secure environment called Dynamic Analysis [1, 9–13], some researchers are analyzing the ransomware without any execution, referred to as Static Analysis [14–16]. However, a good number of researchers are combining these two approaches and adopting a Hybrid Analysis Approach [17–19]. Although the static analysis technique takes less analysis time and facilitates the researchers by not requiring the execution of malicious files, this technique struggles to trace new ransomware variants because of the ever-evolving code obfuscation technique. On the other hand, although a dynamic analysis approach might take a longer time to process and analyze the ransomware program, this approach can detect ransomware with higher accuracy as it executes the ransomware program in a secure virtual environment and does real-time behavioral analysis. The main idea is that despite the changes in the new ransomware variants, they will still show the same behavioral patterns. Therefore, for this study, we have opted for the dynamic analysis approach for its ability to detect and classify ransomware families based on behavioral patterns regardless of the code obfuscation techniques deployed by the ransomware programmers [20, 21]. Among the broad range of different behavioral characteristics obtained from the dynamic analysis approach, selecting critical features for a robust ransomware classification or detection system has been a constant challenge. Although feature selection techniques play a pivotal role in this regard, investigating whether those selected features are crucial for the corresponding system has always been ignored and so has the efficiency of the feature selection technique. Especially, when applying an iterative feature selection technique, there is always a chance for a crucial feature not being included in the best-performed subset of features and the number of selected features may not always be optimal for a given problem. Hence, the prime objective of this study is to investigate the efficiency of RFECV- an iterative feature selection technique by incorporating XAI in our work and to demonstrate how XAI can be employed to derive highly contributing features that would facilitate building robust ransomware detection systems. Again, regardless of the relentless effort of the investigated research works, improvement scenarios are still existing in the manual data collection process, not having a large or diverse dataset that focuses on both Crypto and Locker types of ransomware [1, 10, 12, 22–26], using the Cuckoo sandbox environment that often falls short of providing in-depth and accurate analysis reports [1, 9–13], and

not presenting the highly contributing features to the output of each ML model. Therefore, this study also automates the data collection process by developing a Web-Crawler, focuses on both the Crypto and Locker types of ransomware, utilizes an advanced sandbox environment, namely, Falcon Sandbox for the dynamic analysis of ransomware, and presents the highly contributing features to the output of each ML model.

The main contributions of this study are:

- Developing a Web-Crawler, 'GetRansomware' to automate collecting the Windows Portable Executable (PE) files of 15 different ransomware families from the VirusShare repository. The Web-Crawler is essential to automate searching and downloading the samples and cutting down the manual workload.
- Constructing two different ransomware datasets by analyzing two types of binaries, namely, Windows Portable Executables (PE) and Packet Capture (PCAP) files of both Crypto and Locker types of ransomware.
- Examining and comparing the performance of six Supervised ML models in identifying ransomware families, both with and without the use of the RFECV feature selection approach. Since our approach includes utilizing RFECV for selecting the optimum number of features and RandomSearchCV for selecting the optimum hyperparameter values for each classifier, this study attempts to optimize each model's performance in both scenarios before the comparison is made.
- Presenting the efficiency of the RFECV feature selection technique in ransomware classification. For this task, first, we utilize 'SHapley Additive exPlanations' to obtain the highly contributing features from the without feature selection scenario. Next, we obtain the RFECV-selected features from the with feature selection scenario. Finally, we report how the important set of features varies for each ML model in two scenarios and how they affect the final outcome. Thus, this study also demonstrates the application of SHAP to identify the critical features that significantly contribute to the classification of ransomware.

The rest of this paper is structured as follows: Sect. 2 presents the related works. Section 3 details our methodology. The experimental results and discussions are illustrated in Sect. 4. Section 5 concludes the paper with the direction for future works.

2 Related works

In this section, we present several prior approaches to ransomware detection or classification. Although malware of a particular kind is called ransomware and many of the previous approaches include ransomware families in the malware dataset, our investigation mainly focuses on the binary and multiclass classification of ransomware through the dynamic analysis approach. First, we present recent research on API sequence and frequency-based ransomware detection and classification techniques. Next, we introduce a few investigations on network traffic features-based methods. Then, we mention several works that combine other significant features along with API call features and network traffic features towards ransomware detection and classification. All of these approaches are similar to our method since we consider both the API call features and network traffic features for comparing the performance of ML models with and without the RFECV feature selection technique.

A good number of researchers analyzed API call behaviors and proposed ransomware detection or classification methods based on the API call sequences or frequencies. Maniath et al. [10] analyzed the API call behavior of 157 ransomware and presented Long Short-Term Memory (LSTM)-based ransomware detection that focuses on API call sequence and compensates for the ransomware that causes execution delays. However, this work lacks complete information about the ransomware families/variants and the number of benign software used for the experiment. Vinayakumar Kumar et al. [11] proposed a Multilayer Perceptron (MLP)-based ransomware detection method focusing on API call frequency but they deployed a simple MLP network that failed to distinguish CryptoWall and Cryptolocker. Chen et al. [23] used API Call Flow Graph (CFG) generated from the extracted API sequence using the API monitor tool for detecting ransomware. Regardless, the work is based on a smaller dataset that includes only four ransomware families. Also, graph-similarity analysis requires higher computational power that some systems may not provide. Takeuchi et al. [12] used API call sequences to identify zero-day ransomware attacks and the work involved kernel tricks for tuning Support Vector Machine (SVM). However, the accuracy of this work decreases while using standardized vector representation because of the less diverse dataset. Bae et al. [27] extracted the API call sequences using the Intel Pin Tool. Their sequential process includes generating an n-gram sequence, input vector, and Class Frequency Non-Class Frequency (CF-NCF) for every sample before fitting

their model. Nevertheless, their work lacks complete information about the ransomware families/variants used for the experiment, and the work's accuracy can be improved with the help of deception-based techniques. Hwang et al. [13] analyzed API calls and used two Markov chains, one for ransomware and another for benign software to capture the API call sequence patterns. By using Random Forest (RF), they compensate Markov Chains and control False Positive Rate (FPR) and False Negative Rate (FNR) to achieve better performance. However, their model produces high FPR that can be improved with the help of signature-based techniques.

In contrast to the API call behaviors, some researchers analyzed network traffic behaviors of different ransomware families. Cabaj et al. [24] proposed two real-time Software Defined Networking (SDN) based mitigation methods that were developed using OpenFlow to ensure a prompt reaction to the threat while not decreasing the overall network performance. However, the proposed method is only based on the features of CryptoWall ransomware. Tseng et al. [25] proposed a method that can identify specific network traffic types and detect in-network behavior sequences. Their approach detects ransomware before encryption starts. Regardless, the work lacks complete information about the ransomware families/variants as well as benign software used for the experiment. Alhawi et al. [26] used TShark for capturing and analyzing malicious network traffic activities followed by utilizing the WEKA ML tool to detect ransomware based on only 9 extracted features. Nonetheless, because of using fewer features of only 210 ransomware, the proposed method may fall short of recognizing the new ransomware variants. Almashhadani et al. [22] built a dedicated testbed for executing and capturing the network traffic of the sample ransomware and proposed a multi-classifier that works on two different levels: packet-based and flow-based classifiers. Their method employed a language-independent algorithm that can detect domain names from general sonic axioms. However, the proposed method is only based on the Locky ransomware. In Almashhadani et al. [28], thoroughly analyze ransomworm network traffic, focusing on WannaCry and NotPetya. They extract 21 informative features from session-based and time-based flow levels to distinguish compromised host propagation traffic. Two machine learning classifiers are built based on these features. Moreover, they developed MFMCNS, a multi-feature and multi-classifier network system, which shows 99.8% detection accuracy. Nevertheless, the research relies heavily on WannaCry traffic analysis due to the greater availability of WannaCry PCAP files compared to those of NotPetya. Singh et al. [29] present SINN-RD, an innovative Neural Network-based Ransomware Detection System employing Spline Interpolation. They outline data normalization and feature generation from log files. Security analysis confirms SINN-RD's robustness against potential threats. The practical application assesses its impact on key performance metrics, including accuracy, precision, recall, and F1-score. Furthermore, comparative analysis demonstrates that SINN-RD outperforms existing schemes, achieving an impressive 99.83% accuracy.

Instead of considering only API call behavior or only the network traffic behavior, some researchers combined these two categories of behavior along with other malicious indicators (i.e., registry key operations, file extensions, files/directory operation, etc.) for their models. D. Sgandurra et al. [9] analyzed API calls, registry key operations, embedded strings, file extensions, files/directory operations, and dropped file extensions prior to developing their model. The features were selected using the mutual information criterion and their proposed method 'EldeRan' was able to deal with sophisticated encryption methods of ransomware at an early stage. However, the limitation of 'EldeRan' is that it produces a higher False Positive Rate. Continella et al. [30] analyzed filesystem operations and presented two models: process-centric trained on each process and system-centric trained on the whole system. They developed 'ShieldFS'-a software on OS that can detect malicious file activities and roll back from the attack. However, their system-centric model produces high false positives, and the system may face performance degradation due to the add-on driver on the OS. Lu et al. [31] analyzed API calls, network features, registry operations, file operations, directory operations, and memory usage for developing a ransomware detection method based on the Artificial Immune System (AIS). They applied real-valued detector generation based on the V-detector negative selection while optimizing the AIS parameter (i.e., hypersphere detector distribution) to improve the ransomware detection rate. Regardless, their system also produces higher false alarms. Hasan et al. [1] considered API calls, network features, registry key operations. process operations, function length frequency, and printable string information for their model. They proposed a framework- 'RansHunt' that takes a hybrid approach to identify potential static and dynamic features for the SVM classifier that outperforms traditional AV tools. However, the proposed method only focuses on the Crypto category. So, it may not be effective for the Locker category. In [32], Zahoora et al. analyzed API requests, file directory setups, file extensions, file processes, registry keys setups, strings, and dropped file records and introduced CSPE-R, a Cost-Sensitive Pareto Ensemble strategy for detecting new Ransomware attacks. Initially, an unsupervised deep Contractive Auto Encoder is used to transform the feature space. CSPE-R explores different semantic spaces and uses a novel Pareto Ensemble-based estimator selection strategy to balance false positives and false negatives. The experimental results demonstrate 93% accuracy against zero-day ransomware attacks, although the dataset includes only 11 crypto-ransomware families. Masum et al. [33] introduce a unique feature selection-based

framework, incorporating various machine learning algorithms, particularly neural network-based classifiers, for efficient ransomware classification and detection. The framework uses variance threshold and VIF threshold as feature selection tools to eliminate low-variant and highly correlated features. The models' performance was evaluated through a comprehensive comparative analysis of DT, RF, NB, LR, and NN classifiers. The experimental findings indicate that the Random Forest classifier outperforms other classifiers, demonstrating the highest 99% accuracy.

Table 1 presents the synopsis of the previous research works conducted on the analysis, detection, and classification of ransomware.

3 Methodology

The methodology of this study consists of three subsequent steps as illustrated in Fig. 2: Data Collection, Feature Engineering, and Classification.

3.1 Data collection

We have developed a Web-Crawler- 'GetRansomware' to automate collecting the Windows Portable Executable (PE) files of 15 different ransomware families from the VirusShare repository [34]. We have also shared the Web-Crawler on our GitHub repository for public access [35]. About 95% of the PE files were collected from VirusShare using GetRansomware. The rest of the PE files were collected from theZoo [36] and Hybrid-Analysis.com [37]. In addition, we have collected the Packet Capture (PCAP) files of those ransomware families from the malware-traffic-analysis [38]. Table 2 presents the number of collected samples.

3.2 Feature engineering

The scarcity of the ransomware dataset is one of the major challenges that hinder the research work in this area [39]. Therefore, for this study, we construct two different datasets from two types of binaries through separate feature engineering processes. In the first process, we create the first dataset by analyzing the PE files while in the second process, we create the second dataset by analyzing the PCAP files.

3.2.1 Process 1: creation of the first dataset- 'Data1'

The feature engineering step for the first process is composed of two phases. The phases are:

- Phase 1: Feature Extraction
- Phase 2: Feature Selection

Phase 1: feature extraction From the wide range of distinct behavioral features, we have considered utilizing API call frequencies for our study. API calls are made by the application or program running at a user level to request services as depicted in Fig. 3. It is the method through which data or information is exchanged between the sending device and the receiving device. The OS performs the requested services by issuing these calls, and the outcomes are returned to the caller user applications. Thus, API calls made by the ransomware program allow the attackers to explore and obtain control of the system and perform malicious activities. Since analyzing API call behavior leads researchers to better understand the program's behavior [40, 41], therefore, we have considered extracting the API call frequency by executing the PE files of the ransomware.

We have analyzed the PE files with the help of Hybrid-Analysis.com [37], powered by the CrowdStrike Falcon Sandbox [42]. To automate submitting malicious binaries, pull the analysis report after the analysis, and perform advanced or required search queries on the database, Falcon Sandbox provides a free, convenient, and efficient API key that one can obtain from an authorized user account. For analysis, we have used our API key and Falcon Sandbox Python API Connector- VxAPI wrapper [43] to automatically submit the binaries from the system. After submission, Falcon Sandbox runs the binaries in a Virtual Machine (VM) and captures the run-time behaviors as illustrated in Fig. 4. Later, it shows the analysis results on the web interface.

Table 1 Synopsis of the literature review

References	Dataset	Classifier	Accuracy
API call features			
Maniath et al. [10]	157 Ransomware, Unspecified number of benign software	Long Short-Term Memory	96.67%
Vinayakumar Kumar et al. [11]	755 Ransomware, 219 Benign Software	Multilayer Perceptron	100% (Binary), 98% (Multi-class)
Chen et al. [23]	83 Ransomware, 85 Benign	Simple Logistic	98.2%
Takeuchi et al. [12]	276 Ransomware, 312 Benign Software	Support Vector Machine	97.48%
Bae et al. [27]	1000 Ransomware, 900 Malware, 300 Benign software	Random Forest	98.65%
Hwang et al. [13]	1909 Ransomware, 1139 Benign software	Markov Chain, Random Forest (Two-stage detection model)	97.3%
Network features			
Cabaj et al. [24]	359 CryptoWall samples	N/A	N/A
Tseng et al. [25]	155 Ransomware, Unspecified number of benign software	Deep Neural Network	93.92%
Alhawi et al. [26]	210 Ransomware, 264 Benign software	J48	97.1%
Almashhadani et al. [22]	Locky ransomware, Unspecified number of benign software	Bayes Net	99.83%
Almashhadani et al. [28]	83661 KB Ransomware, 1264185 KB Benign Software	MFMCNS	99.8%
Singh et al. [29]	6441 Ransomware, 6441 Benign Software	Neural Network	99.83%
API call features, network features, and other features			
Sgandurra et al. [9]	582 Ransomware, 942 Benign Software	Regularized Logistic Regression	96.34%
Continella et al. [30]	383 Ransomware, 2245 Benign Software	Random Forest	97.70%
Lu et al. [31]	1000 Ransomware, 1000 Benign Software	V-detector	90%
Hasan et al. [1]	360 Ransomware, 532 Malware, 460 Benign Software	Support Vector Machine	97.10%
Zahoora et al. [32]	582 Ransomware, 942 Benign Software	CSPE-R Ensemble	93%
Masum et al. [33]	96633 Ransomware, 41414 Benign Software	Random Forest	99%

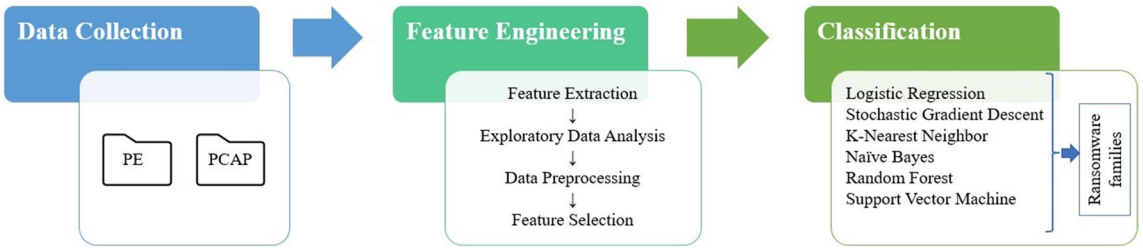
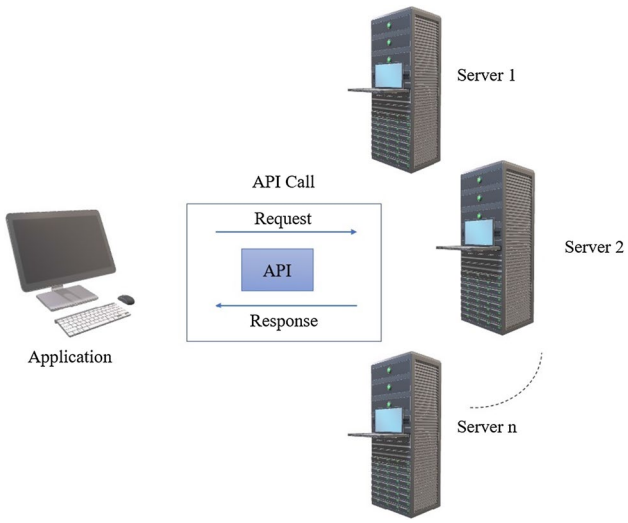


Fig. 2 Process overview of our methodology

Table 2 Number of collected samples

Ransomware	PE file	PCAP file
Cerber (c0)	95	58
CryptoLocker (c1)	95	55
CryptoWall (c2)	97	55
Eris (c3)	98	55
Hive (c4)	100	56
Jigsaw (c5)	95	60
Locky (c6)	95	60
Maze (c7)	100	55
Mole (c8)	100	56
Sage (c9)	100	56
Satan (c10)	100	60
Shade (c11)	98	57
TeslaCrypt (c12)	97	59
Virlock (c13)	95	57
WannaCry (c14)	95	57
Total	1460	856

Fig. 3 Communication through the API call



Contrary to the prior works where the analysis tasks were done using the Cuckoo Sandbox [1, 9–13], we have analyzed the PE files using the Falcon Sandbox that uses a VM (Windows 7 64-bit) to execute the PE files. Falcon Sandbox incorporates many other services, such as VirusTotal, Thug honeyclient, OPSWAT Metadefender, TOR, NSRL (Whitelist), Phantom, and a large number of antivirus engines to provide an integrated and in-depth analysis reports compared to other Sandboxes. While executing the binaries, we have set run-time to the maximum available duration in the Falcon

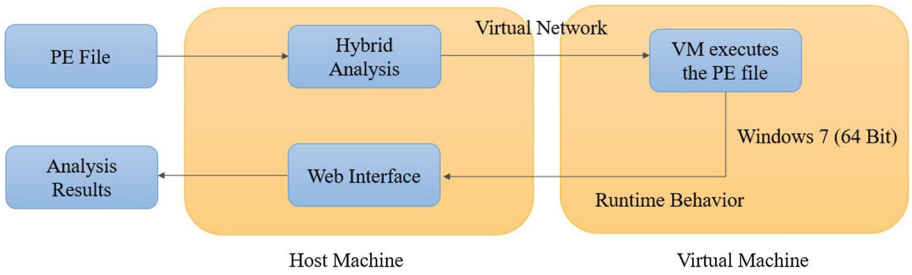


Fig. 4 Block diagram of the PE file execution process

Sandbox to deal with the delayed execution techniques deployed by the attackers. The total time for the analysis was (1460 PE files * 7 min) = 170 h = 7 days approximately. Next, we obtained the analysis report by using the API key from which we have only sorted and computed the frequency of each API call. At the end of the PE files analysis process, we obtained our first dataset- 'Data1' consisting of the different frequencies of 68 distinct API calls associated with the 15 ransomware families as presented in Table 3.

Phase 2: feature selection At the beginning of the feature selection phase, we have evenly divided (stratified train-test split) our dataset into train data (80%) and test data (20%) to avoid data leakage. Next, we have applied Recursive Feature Elimination with Cross-Validation (RFECV) [44] to our train data. RFECV is a wrapper-style feature selection method that wraps a given ML model as depicted in Fig. 5 and selects the optimal number of features for each model by recursively eliminating 0-n features in each loop. Next, it selects the best-performing subset of features based on the accuracy or the score of cross-validation. RFECV also removes the dependencies and collinearity existing in the model. By using RFECV, we have selected 6 distinct subsets of features for 6 ML classifiers. These features have been selected by setting 'min features to select' as 34 (half of the features), cv=5, and 'scoring'='accuracy' so that RFECV would select at least half of the features based on the optimum accuracy over the 5-fold cross-validation.

3.2.2 Process 2: creation of the second dataset- 'Data2'

The feature engineering step for the second process is composed of four phases. The phases are:

- Phase 1: Feature Extraction
- Phase 2: Exploratory Data Analysis (EDA)
- Phase 3: Data Preprocessing
- Phase 4: Feature Selection

Phase 1: feature extraction We have considered utilizing network traffic features for the second dataset for our study. The Transmission Control Protocol (TCP) refers to the set of standardized communication protocols that specify how computers communicate over the network. According to our literature review, the communication between the infected host machine (source) and the attacker (destination) is conducted through the transport layer [45]. Besides, HTTP GET or POST methods are also used to send back the information to the attacker [22]. Hence, we have opted for capturing the TCP traffic and the HTTP traffic information by analyzing the PCAP files of the ransomware.

Again, ransomware often spreads through spam emails containing malignant attachments as macro-enabled word documents. By executing a script, these attachments download the executable file of that ransomware from a URL and install it on the system. After the installation, the ransomware continuously tries to search and connect to its C & C servers to exchange the encryption key and launch the attack session. Firstly, it utilizes an encrypted list of IP addresses for creating a TCP session with the C & C servers. Upon failure due to the unreachable or blacklisted IP addresses or disrupted session, the ransomware then opts to find out its C & C server by executing the Domain Generation Algorithm (DGA) and recurrently produces a good number of pseudo-random domain names. Then, the ransomware continues sending the Domain Name System (DNS) request to those domain names until the actual C &C server is found as illustrated in Fig. 6. Here, DNS converts human-readable domain names to machine-readable IP addresses. Upon successful establishment of a TCP session, the attacker guides the victim in delivering the payload. The characteristic of dispatching an extensive number of DNS requests looking for a real C &C server looks like an arbitrary set of characters. Meaningful statistical information can be derived from these requested domain names as well as the pattern of randomness found in them [46]. If the ransomware detection method can trace the randomness

Table 3 List of features in the 'Data1' dataset

API Call Features	
1. FindWindowExW	35. NtProtectVirtualMemory
2. LdrGetDllHandle	36. NtQueryAttributesFile
3. NtAdjustPrivilegesToken	37. NtQueryDefaultLocale
4. NtAlertThread	38. NtQueryDirectoryFile
5. NtAllocateVirtualMemory	39. NtQueryInformationFile
6. NtAlpcSendWaitReceivePort	40. NtQueryInformationProcess
7. NtConnectPort	41. NtQueryInformationToken
8. NtCreateEvent	42. NtQueryKey
9. NtCreateFile	43. NtQueryObject
10. NtCreateKey	44. NtQuerySystemInformation
11. NtCreateKeyEx	45. NtQueryValueKey
12. NtCreateMutant	46. NtQueryVirtualMemory
13. NtCreateSection	47. NtQueryVolumeInformationFile
14. NtCreateThreadEx	48. NtReadFile
15. NtCreateUserProcess	49. NtReadVirtualMemory
16. NtDelayExecution	50. NtRequestWaitReplyPort
17. NtDeleteValueKey	51. NtResumeThread
18. NtDeviceIoControlFile	52. NtSetContextThread
19. NtEnumerateKey	53. NtSetInformationFile
20. NtEnumerateValueKey	54. NtSetInformationKey
21. NtFsControlFile	55. NtSetInformationProcess
22. NtGetContextThread	56. NtSetInformationThread
23. NtMapViewOfSection	57. NtSetSecurityObject
24. NtNotifyChangeKey	58. NtSetValueKey
25. NtOpenDirectoryObject	59. NtTerminateProcess
26. NtOpenEvent	60. NtTerminateThread
27. NtOpenFile	61. NtUnmapViewOfSection
28. NtOpenKey	62. NtWaitForMultipleObjects
29. NtOpenKeyEx	63. NtWriteFile
30. NtOpenMutant	64. NtWriteVirtualMemory
31. NtOpenProcess	65. NtYieldExecution
32. NtOpenProcessToken	66. OpenSCManager
33. NtOpenSection	67. OpenServiceW
34. NtOpenThreadToken	68. SetWindowsHookEx

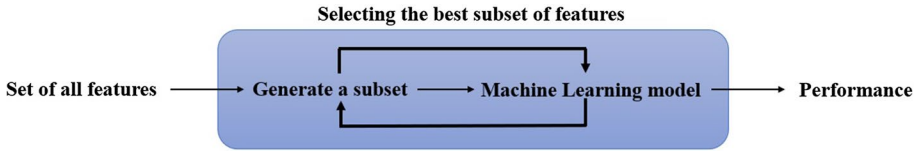


Fig. 5 RFECV feature selection technique

that occurs before finding out the actual C &C server, it can be stopped before the ransomware begins encrypting files. This is an efficient approach in case of a zero-day attack as deriving the information from the known ransomware is not required in this case. Therefore, we have opted for extracting DNS traffic information by analyzing the PCAP files of the ransomware.

We have analyzed the PCAP files using Wireshark- a network protocol analyzer [47, 48]. This manual process involved three identical systems with Wireshark installed and 2 volunteers for analyzing the PCAP files. We have extracted 18 network traffic features that according to [49], convey important statistical information that enhances the ability of the classification algorithms to classify ransomware. Then, these features have been merged resulting in 'Data2'. Table 4 presents the list of network traffic features.

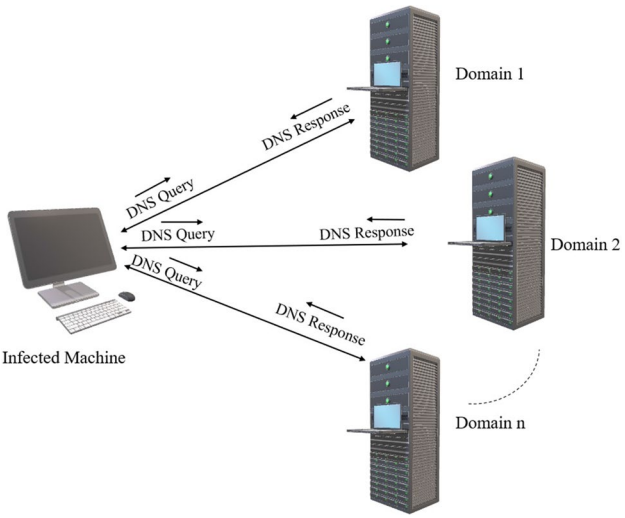


Fig. 6 Finding out the actual C & C server by sending DNS requests [Author's own processing]

Phase 2: exploratory data analysis (EDA) At the beginning of Phase 2, we have evenly divided (stratified train-test split) the dataset into train data (80%) and test data (20%) to avoid data leakage. Next, we have done exploratory data analysis to better understand the raw data so that the data could be preprocessed as per requirement. The findings from this phase are:

- Categorical data: We have found 11 features containing categorical data. They are the IP and port of the client, IP and port of the server, Bytes sent from the client to the server, Bytes sent from the server to the client, HTTP method GET or POST of the HTTP requests, Response code to the HTTP requests, URL requested in the HTTP request, IP and port of the client.1, IP and port of the DNS server, DNS request, and DNS response. These categorical data need to be encoded into numerical values since the classifiers require the data to be understandable so that they can be trained on and make predictions.
- Random missing values: Since different ransomware families create different numbers of conversations over the network, the number of instances captured from the PCAP files was different for each ransomware sample. Hence,

Table 4 List of features in the 'Data2' dataset

Network traffic features
1. IP and port of the client
2. IP and port of the server
3. Bytes sent from the client to the server
4. Bytes sent from the server to the client
5. RSTs in the TCP connection from client to server
6. RSTs in the TCP connection from server to client
7. FINs in the TCP connection from client to server
8. FINs in the TCP connection from server to client
9. Number of HTTP requests present in the connection
10. HTTP method (GET or POST) of the HTTP requests
11. Response code to the HTTP requests
12. URL requested in the HTTP request
13. Timestamp of the DNS request
14. IP and port of the client in the DNS request
15. IP and port of the DNS server
16. RCode of the DNS response (It is sent by the server indicating whether it was able to settle the request or not)
17. DNS request
18. DNS response

we have observed missing values in network traffic information. Handling missing values is an essential part of the feature engineering process as the ML models may generate biased or inaccurate results if the missing values are not handled properly. There are two ways of dealing with missing values, such as deleting the missing values and imputing the missing values. Since deleting the missing values ends up deleting the entire row or column that contains the missing values, there is a probability of losing useful information in the dataset. So, we have opted for imputing the missing values.

Phase 3: data preprocessing In the data preprocessing phase, firstly, we have encoded the categorical data into numerical data for which we have applied One-Hot Encoding [50] by using the '.get_dummies' attribute of Pandas data frame package that generates the dummy variables of those 11 features. For preventing the 'Dummy Variable Trap', we have set 'True' as 'drop_first' parameter. To normalize the data and to prevent the imputer from producing biased numerical replacements for the missing data, we have scaled the numerical values between 0 and 1. After normalizing the data, we have used Scikit-Learn's Impute package to apply KNNImputer to fill up the missing values.

Phase 4: feature selection We have selected the network traffic features using RFECV by setting 'min_features_to_select' as 9 (half of the features), cv=5, and 'scoring'= 'accuracy' so that RFECV would select at least half of the features based on the optimum accuracy over the 5-fold cross-validation applied on our train data.

3.3 Classification

We have employed Supervised Machine Learning algorithms to classify 15 ransomware families into corresponding categories. Supervised learning algorithms are trained on the labeled dataset to make a decision in response to the unseen test dataset. These algorithms are generally of two types, such as classification-based and regression-based. The classification-based algorithms are used to accomplish both binary and multi-class classification where the instances from the test dataset are classified into one among an array of known classes, such as Naïve Bayes, Random Forest, K-Nearest Neighbor, etc. On the other hand, regression-based algorithms consider the relationship between independent features or input variables and dependent target class or continuous output variables to make a prediction, such as Linear Regression, Neural Network Regression, Lasso Regression, etc. As this study focuses on classifying 15 ransomware families, the following algorithms have been employed that are widely used for both binary and multi-class classification as per requirement:

- Logistic Regression (LR): is a type of statistical analysis that predicts the probability of a dependent variable from a set of independent variables using their linear combination.
- Stochastic Gradient Descent (SGD): is an optimization algorithm to find the model parameters by updating them for each training data so that the best fit is reached between predicted and actual outputs.
- K-Nearest Neighbor (KNN): estimates the likelihood of a new data point being a member of a specific group by measuring the distance between neighboring data points and the new data point.
- Naïve Bayes (NB): is based on Bayes' theorem and predicts the probability of an instance belonging to a particular class.
- Random Forest (RF): constructs multiple decision trees during the training phase and finally determines the class selected by the maximum number of trees.
- Support Vector Machine (SVM): takes one or more data points from different classes as inputs and generates hyperplanes as outputs that best distinguish the classes.

Since this study focuses on multi-class classification and some classifiers are only designed for binary classification problems (i.e., Logistic Regression, Support Vector Machine, etc.), these cannot be directly applied to multi-class classification problems. Therefore, Heuristic Methods [51] can be applied to divide a multi-class classification problem into several binary classification problems. There are two types of heuristic methods as illustrated in Fig. 7. The methods are:

- One-vs-Rest (OvR) which splits the dataset into one class against all other classes each time [52].
- One-vs-One (OvO) which splits the dataset into one class against every other class each time [53].

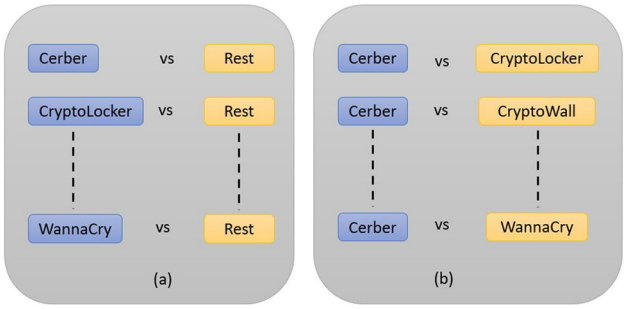


Fig. 7 Heuristic methods: **a** One-vs-Rest and **b** One-vs-One

We have applied the OvR method for our experiment to reduce the time and computational complexities. All these classifiers are built along with 'RandomSearchCV' [54]- a hyperparameter optimization technique, to find the best combination of hyperparameters for maximizing the performance of the models' output in a reasonable time. Instead of exhaustively searching for the optimal values of the hyperparameters through a manually determined set of values (i.e., Grid Search), RandomSearchCV randomly searches the grid space and selects the best combination of hyperparameter values based on the accuracy or the score of cross-validation. Since we have used RFECV for feature selection and RandomSearchCV for hyperparameter optimization, the Nested Cross-Validation technique has been implemented in the pipeline to build each model.

4 Experimental results and discussions

4.1 Experimental results

We have evaluated the models in terms of Precision, Recall, F1-score, and Accuracy. These performance metrics are measured as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100$$

where, TP = True Positives, FP = False Positives (Type 1 Error), TN = True Negative, FN = False Negative (Type 2 Error).

Table 5 presents the performance comparison of Machine Learning models with and without feature selection for the 'Data1' dataset. It shows that with and without feature selection LR outperforms other classifiers securing 98.20% and 99.30% overall accuracy respectively. Although there is a slight performance degradation in all the classifiers in the with-feature selection scenario, remarkable improvement in the processing time has been observed. As shown in Table 6, with-feature selection, the average processing time of all the classifiers has been improved by 26.97%. We present the

Table 5 Performance comparison between LR, SGD, KNN, NB, RF, and SVM with respect to the with-feature selection and without-feature selection using the 'Data1' dataset (P(avg)= Average performance, w FS= With-Feature Selection, and wo FS= Without-Feature Selection

P(avg)	LR		SGD		KNN		NB		RF		SVM	
	w FS	wo FS	w FS	wo FS	w FS	wo FS	w FS	wo FS	w FS	wo FS	w FS	wo FS
Accuracy	98.20	99.30	90.43	92.45	89.62	90.52	97.17	97.46	91.51	92.78	94.34	95.58
Precision	98.53	99.37	98.86	100	94.33	94.08	97.82	98.77	100	99.79	99.15	99.21
Recall	98.22	99.30	91.36	92.45	89.62	90.52	97.17	97.46	91.51	92.78	94.34	95.58
F1- score	98.20	99.29	94.61	95.85	90.78	91.27	97.21	98.02	95.23	95.87	96.40	97.19

Table 6 Classifier's processing time comparison without-feature selection and with-feature selection using the 'Data1' dataset

Classifier	Without-feature selection (in seconds)	With-feature selection (in seconds)	Improvement (%)
LR	79.21	58.44	26.22
SGD	78.43	57.62	26.53
KNN	78.00	51.25	34.29
NB	76.39	55.67	27.12
RF	75.41	58.28	22.71
SVM	79.19	59.43	24.95

Average processing time improvement (%)

classification accuracy for each class of the best-performed supervised machine learning model from these classifiers in two different scenarios. Figure 8 illustrates the normalized confusion matrix of the LR classifier. As shown in Fig. 8a, when the features are not selected, among 15 classes, the classifier could distinguish 13 classes with 100% accuracy. However, the classifier produces 1% false negatives classifying CryptoLocker ransomware and 11% false positives classifying Shade ransomware. On the other hand, Fig. 8b shows the confusion matrix of the LR classifier with feature selection. Although the classifier could distinguish 10 classes with 100% accuracy, the classifier produces 1% false negatives classifying Cerber, 22% false positives classifying CryptoLocker, 10% false positives classifying Mole, 10% false positives classifying Sage, and 11% false positives classifying Shade ransomware.

Table 7 presents the performance comparison of Machine Learning models with and without feature selection for the 'Data2' dataset. It shows that with and without feature selection NB outperforms other classifiers securing 97.89% and 98.95% overall accuracy respectively. Even though all of the classifiers in the with-feature selection scenario show a minor performance deterioration, a notable improvement in processing time has been seen. As shown in Table 8, with-feature selection, the average processing time of all the classifiers has been improved by 34.72%. We present the classification accuracy for each class of the best-performed supervised machine learning model from these classifiers in two different scenarios. Figure 9 illustrates the normalized confusion matrix of the NB classifier.

As shown in Fig. 9a, when the features are not selected, among 15 classes, the classifier could distinguish 10 classes with 100% accuracy. However, the classifier produces 2% false negatives classifying CryptoLocker and 1% false negatives classifying Maze ransomware. On the other hand, Fig. 9b shows the confusion matrix of the NB classifier with feature

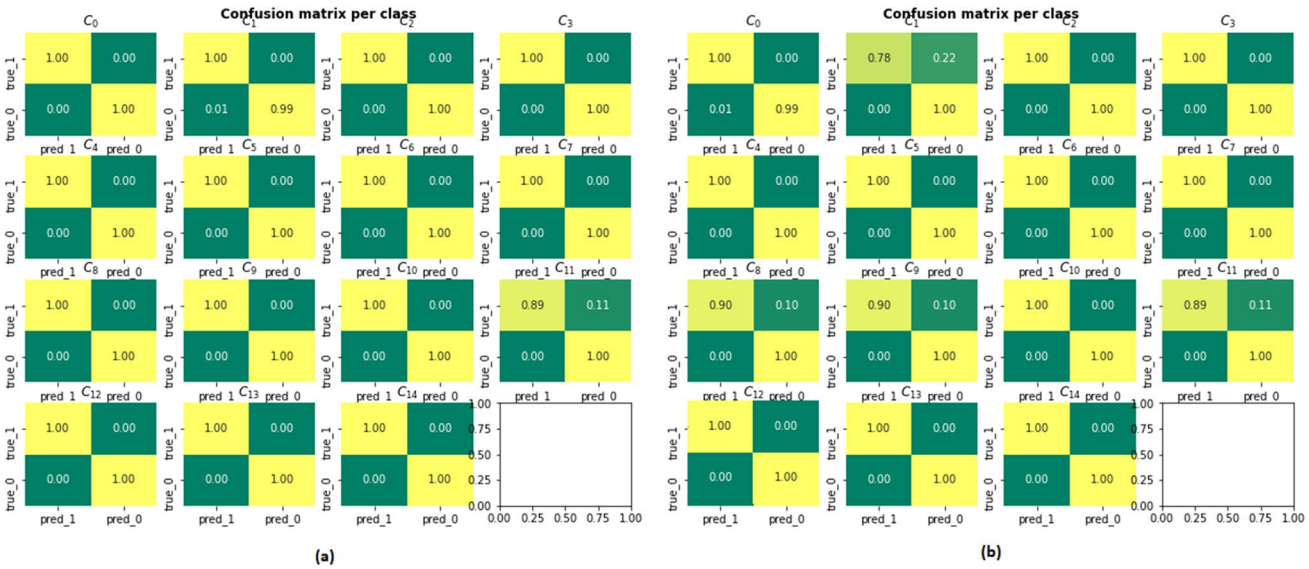


Fig. 8 Confusion matrix of (**a**) Logistic Regression without feature selection, and (**b**) Logistic Regression with feature selection for the 'Data1' dataset [Author's own processing]

Table 7 Performance comparison between LR, SGD, KNN, NB, RF, and SVM with respect to the with-feature selection and without-feature selection using the 'Data2' dataset P (avg) = Average performance, w FS = With-Feature Selection, and wo FS = Without-Feature Selection

P (avg)	LR		SGD		KNN		NB		RF		SVM	
	w FS	wo FS	w FS	wo FS	w FS	wo FS	w FS	wo FS	w FS	wo FS	w FS	wo FS
Accuracy	92.25	94.04	81.69	82.76	80.99	83.25	97.89	98.95	78.87	79.96	92.25	93.90
Precision	98.21	97.81	90.96	93.27	92.05	92.56	98.21	99.05	100	99.90	98.73	98.89
Recall	92.25	94.04	88.03	87.53	80.99	83.25	97.89	98.95	78.87	79.96	92.25	93.90
F1- score	94.81	95.67	88.98	89.91	84.13	85.96	97.92	98.95	86.99	87.64	95.01	96.06

Table 8 Classifier's processing time comparison without-feature selection and with-feature selection using the 'Data2' dataset

Classifier	Without-feature selection (in seconds)	With-feature selection (in seconds)	Improvement (%)
LR	88.19	56.88	35.5
SGD	85.31	54.66	35.9
KNN	85.44	54.30	36.4
NB	84.13	51.29	35.5
RF	85.27	56.78	33.4
SVM	83.18	56.93	31.6
Average processing time improvement (%)			

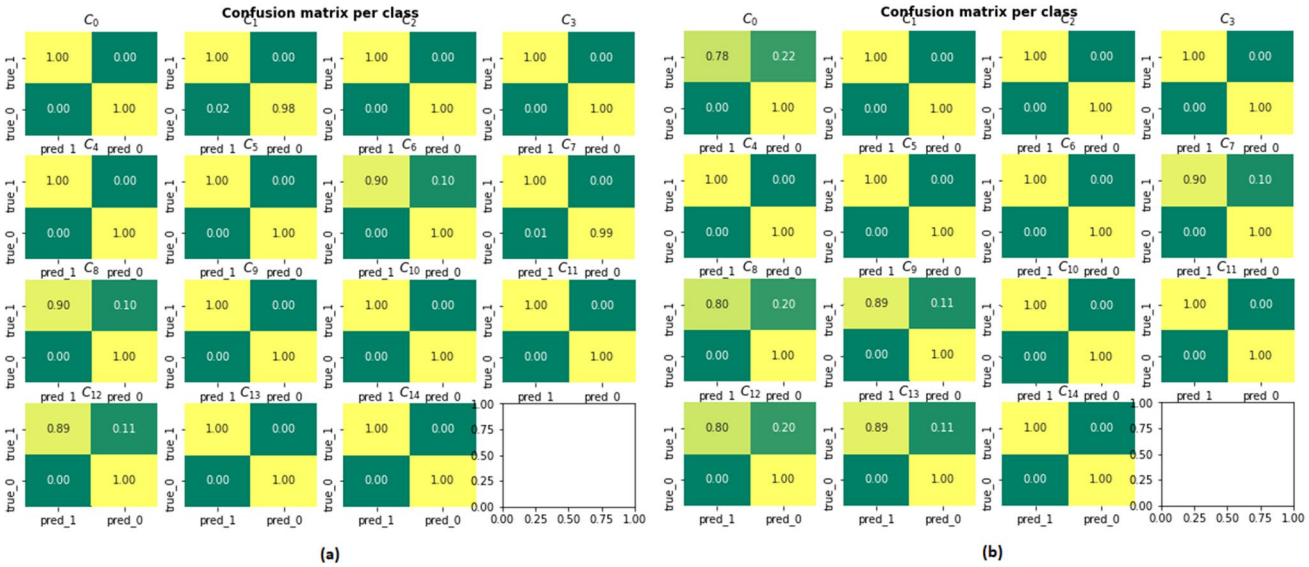


Fig. 9 Confusion matrix of (**a**) Naïve Bayes without feature selection, and (**b**) Naïve Bayes with feature selection for the 'Data2' dataset [Author's own processing]

selection. The classifier could distinguish 9 classes with 100% accuracy with no false negatives. However, with feature selection, the classifier produces higher false positives as compared to that without-feature selection.

4.2 Discussions

In this section, we present the comparison between the RFECV-selected features in the with-feature selection scenario and the highly contributing features in the without-feature selection scenario to examine the efficiency of the RFECV feature selection

technique toward ransomware classification. For this task, we apply 'Shapley Additive exPlanations', a tool for visualizing data that helps explain the results of machine learning models. SHAP is based on the coalition game theory that measures each feature's individual contribution to the final output while conserving the sum of contributions being the same as the final result [55]. Unlike other explanation techniques that are limited to explaining specific models, SHAP values can be used to explain a wide variety of models, such as DeepExplainer to explain Deep Neural Networks (i.e., Multi-Layer Perceptron, Convolutional Neural Networks, etc.), TreeExplainer to explain tree-based models (i.e., Random Forest, XGBoost, etc.), and KernelExplainer to explain any model, etc. [56, 57]. For our study, we have used TreeExplainer to obtain highly contributing features from the Random Forest classifier, while for the other classifiers, we have used KernelExplainer.

In the context of the classification model, the SHAP value is represented as a two-dimensional array. Each column corresponds to a feature used in the model, while each row represents an individual prediction made by the model. The SHAP value in this array indicates the contribution of a specific feature to the output of the corresponding prediction. Positive SHAP values indicate that a feature has a positive influence on pushing the model output toward the base value or expected value. Conversely, negative SHAP values indicate that a feature has a negative influence on pushing the base value toward the model output. The base value, or the average model output, is calculated based on the training data. To visualize this explanation for a single prediction, the Force plot can be utilized as illustrated in Fig. 10. In Fig. 10, the features with higher SHAP values (highlighted in red) positively contribute to pushing the base value toward the model output, while the features with lower SHAP values (highlighted in blue) negatively contribute to pushing the base value toward the model output.

Passing the array of SHAP values to a 'summary plot' function creates a feature importance plot as shown in Fig. 11. Here, we illustrate 40 highly contributing features (as RFECV selects the highest 40 features for the KNN classifier) for each classifier in the without-feature selection scenario for the 'Data1' dataset. Here, the x-axis denotes the mean of the absolute SHAP value for each feature which indicates the total contribution of the feature to the model and the y-axis denotes the features used for the classification. The features are organized in descending order from top to bottom by how strongly they influence the model's decision. As illustrated in Fig. 11, the set of highly contributing features and their order varies for each classifier. However, for our study, we only examine the variation of the RFECV-selected features with the highly contributing features of the corresponding classifiers. Table 9 presents the set of optimum features selected by RFECV for each ML classifier from the 'Data1' dataset and Table 10 presents the list of RFECV-selected features for each ML classifier that is not present in the top 40 highly contributing features. By comparing these two tables, we get the features that are causing performance deterioration in the with-feature selection scenario and producing higher false alarms as compared to that without-feature selection.

Similarly, for the 'Data2' dataset, we present the comparison between the RFECV-selected features in the with-feature selection scenario and the highly contributing features in the without-feature selection scenario. Figure 12 illustrates the features of the 'Data2' dataset in descending order from top to bottom by how strongly they influence the model's decision. For each classifier, the order of the features varies except for the 'Bytes sent from the client to the server' feature. However, similar to the previous step, we only examine the variation of the RFECV-selected features. Table 11 presents the set of optimum features selected by RFECV from the 'Data2' dataset for each ML classifier, and Table 12 presents the list of features that were not selected by the RFECV. By comparing these two tables, we get the features that are causing performance deterioration even with the best-performed ML classifier in the with-feature selection scenario and produce higher false alarms as compared to that without-feature selection.

Although SHAP importance shows the effect of a given feature on the model output while disregarding the exactness of the prediction, our study, by comparing the highly contributing features in the without feature selection scenario and the RFECV selected features in the with feature selection scenario finds out that the RFECV feature selection technique often fails to select the crucial features that have a high impact on the model output resulting in both Type 1 and Type 2 error. Again, for two different ransomware datasets, the selected features have been ranked 1, while the not-selected features have been ranked greater than 1. Hence, the order of the selected features based on their importance remains unknown in the RFECV feature selection technique. In addition, this study also reveals that RFECV falls short of improving the performance of our ML models. Our ML models secure better classification accuracies without RFECV (For 'Data1' dataset, with and without feature selection LR secures 98.20% and 99.30%

Fig. 10 Force plot for single instance of 'Data1' dataset

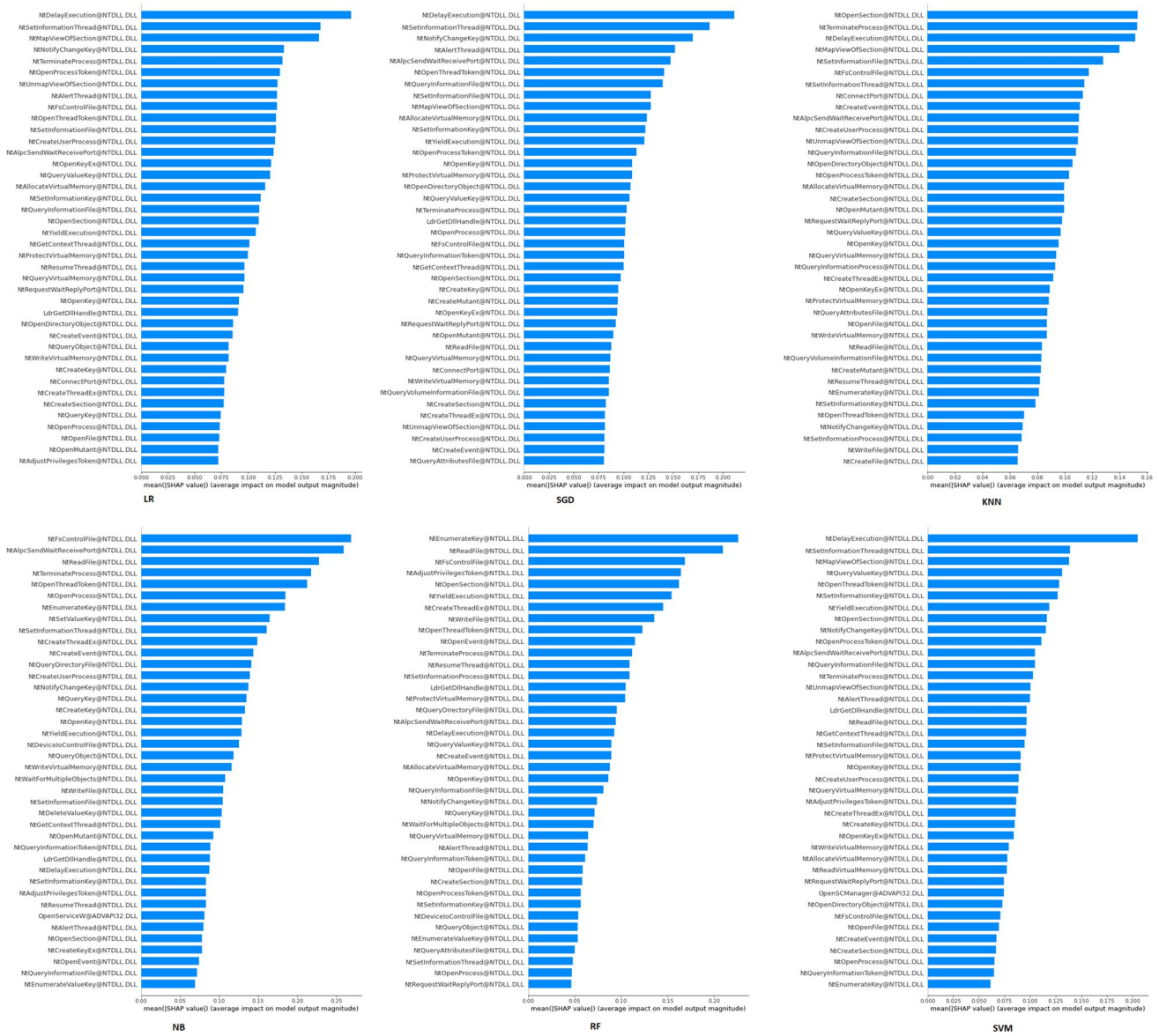


Fig. 11 Summary plot showing the top 40 highly contributing features of the 'Data1' dataset for each ML classifier in the without feature selection scenario

overall accuracy respectively. For 'Data2' dataset, with and without feature selection NB secures 97.89% and 98.95% overall accuracy respectively.) and thus this study also substantiates the performance of our ML models over existing literature. While using API call features, although VinayaKumar et al. [11] achieved 100% exactness for binary classification, the model secured 98% accuracy doing multiclass classification utilizing only 7 classes. Again, regardless of having a good detection rate by utilizing network traffic features, Almashhadani et al. [22] did not extend their work for multiclass classification, and in [29], they rely heavily on WannaCry traffic analysis due to the greater availability of WannaCry PCAP files compared to those of NotPetya. In contrast to these prior approaches, while conducting our study, we improved the data collection process by developing a Web-Crawler to automate collecting 15 different ransomware families and created two different ransomware datasets based on API call features ('Data1') and network traffic features ('Data2') from 2 types of binaries ('PE' and 'PCAP' files respectively) of both the Crypto and Locker types of ransomware. Also, our LR and NB models offer comparative performance over existing literature with explainability that demonstrates the application of SHAP to identify the critical features that significantly contribute to the classification of ransomware.

Table 9 Set of optimum features selected by RFECV from the 'Data1' dataset

Classifier: LR			
Total Selected Features: 38			
NtDelayExecution	NtSetinformationThread	NtMapViewOfSection	NtNotifyChangeKey
NtTerminateProcess	NtOpenProcessToken	NtUnmapViewOfSection	NtAlertThread
NtFsControlFile	NtOpenThreadToken	NtSetinformationFile	NtCreateUserProcess
NtAlpcSendWaitReceivePort	NtOpenKeyEx	NtAllocateVirtualMemory	NtSetInformationKey
NtQueryInformationFile	NtYieldExecution	NtProtectVirtualMemory	NtResumeThread
NtQueryVirtualMemory	NtRequestWaitReplyPort	LdrGetDllHandle	NtEnumerateKey
NtOpenEvent	NtQueryInformationProcess	NtOpenDirectoryObject	NtCreateEvent
NtWriteVirtualMemory	NtCreateKey	NtConnectPort	NtCreateThreadEx
NtCreateSection	NtQuerykey	NtOpenProcess	NtOpenFile
NtOpenMutant	NtAdjustPrivilegesToken		
Classifier: SGD			
Total Selected Features: 38			
NtDelayExecution	NtSetinformationThread	NtNotifyChangeKey	NtAlertThread
NtAlpcSendWaitReceivePort	NtOpenThreadToken	NtQueryInformationFile	NtSetInformationFile
NtMapViewOfSection	NtAllocateVirtualMemory	NtSetinformationKey	NtYieldExecution
NtOpenProcessToken	NtOpenkey	NtProtectVirtualMemory	NtQueryValueKey
NtTerminateProcess	NtOpenProcess	NtFsControlFile	NtQueryInformationToken
NtOpenSection	NtCreateKey	NtCreateMutant	NtRequestWaitReplyPort
NtOpenMutant	NtReadFile	NtConnectPort	NtWriteVirtualMemory
NtCreateSection	NtCreateThreadEx	NtUnmapViewOfSection	NtCreateEvent
NtQueryAttributesFile	NtEnumerateKey	NtOpenEvent	NtQueryInformationProcess
NtSetContextThread	NtQueryDirectoryFile		
Classifier: KNN			
Total Selected Features: 40			
NtOpenSection	NtTerminateProcess	NtDelayExecution	NtMapViewOfSection
NtSetinformationFile	NtFsControlFile	NtSetinformationThread	NtCreateEvent
NtAlpcSendWaitReceivePort	NtCreateUserProcess	NtUnmapViewOfSection	NtQueryInformationFile
NtOpenDirectoryObject	NtAllocateVirtualMemory	NtCreateSection	NtOpenMutant
NtQueryValueKey	NtOpenKey	NtQueryInformationProcess	NtCreateThreadEx
NtProtectVirtualMemory	NtQueryAttributesFile	NtOpenFile	NtWriteVirtualMemory
NtReadFile	NtQueryVolumeInformation- File	NtCreateMutant	NtResumeThread
NtEnumerateKey	NtSetInformationKey	NtOpenThreadToken	NtNotifyChangeKey
NtSetinformationProcess	NtWriteFile	NtCreateFile	NtDeviceIoControlFile
NtAlertThread	NtGetContextThread	NtQueryDirectoryFile	NtSetContextThread
Classifier: NB			
Total Selected Features: 38			
NtAllocateVirtualMemory	NtFsControlFile	NtAlpcSendWaitReceivePort	NtReadFile
NtTerminateProcess	NtOpenThreadToken	NtOpenProcess	NtEnumerateKey
NtSetValueKey	NtSetinformationThread	NtCreateThreadEx	NtCreateEvent
NtQueryDirectoryFile	NtCreateUserProcess	NtNotifyChangeKey	NtQueryKey
NtCreateKey	NtOpenkey		
NtYieldExecution	NtDeviceIoControlFile	NtQueryObject	NtWriteVirtualMemory
NtWaitForMultipleObjects	NtDeleteValueKey	NtGetContextThread	NtOpenMutant
NtQueryInformationToken	NtDelayExecution	NtSetInformationKey	NtAdjustPrivilegesToken
NtResumeThread	OpenServiceW	NtAlertThread	NtOpenSection

Table 9 (continued)

Classifier: NB			
Total Selected Features: 38			
NtCreateKeyEx	NtOpenEvent	NtQueryInformationFile	NtEnumerateValueKey
Classifier: RF			
Total Selected Features: 38			
NtEnumerateKey	NtReadFile	NtFsControlFile	NAdjustPrivilegesToken
NtOpenSection	NtYieldExecution	NtCreateThreadEx	NtWriteFile
NtOpenThreadToken	NtOpenEvent	NtTerminateProcess	NtResumeThread
NtSetinformationProcess	NtProtectVirtualMemory	NtQuery DirectoryFile	NtAIpcSendWaitReceivePort
NtDelayExecution	NtCreateEvent	NtAllocateVirtualMemory	NtQueryInformationFile
NtQueryKey	NtWaitForMultipleObjects	NtQueryVirtualMemory	NtAlertThread
NtOpenFile	NtOpenProcess Token	NtSetinformationKey	NtDeviceloControlFile
NtQueryobject	NtEnumerateValueKey	NtQueryAttributesFile	NtSetinformationThread
NtOpenProcess	NtRequestWaitReplyPort	NtCreateUserProcess	NtGetContextThread
NtQueryInformationProcess	NtMapViewOfSection		
Classifier: SVM			
Total Selected Features: 37			
NtDelayExecution	NtSetinformationThread	NtMapViewOfSection	NtQueryValueKey
NtOpenThreadToken	NtSetinformationKey	NtOpenSection	NtNotifyChangeKey
NtOpenProcessToken	NtAIpcSendWaitReceivePort	NtTerminate rocess	NtUnmapViewOfSection
NtAlertThread	LdrGetDIlHandle	NtGetContextThread	NtSetinformationFile
NtOpenKey	NtCreateUserProcess	NtQueryVirtualMemory	NtAdjustPrivilegesToken
NtCreateThreadEx	NtCreateKey	NtWriteVirtualMemory	NtAllocateVirtualMemory
NtReadVirtualMemory	NtRequestWaitReplyPort	OpenSCManager	NtOpenDirectoryObject
NtFsControlFile	NtOpenFile	NtCreateEvent	NtCreateSection
NtOpenProcess	NtQueryInformationToken	NtEnumerateKey	NtSetContextThread
NtDeviceIoControlFile			

4.2.1 Advantages of SHAP over iterative feature selection technique

SHAP is a powerful tool in explainable AI that can be useful for ransomware detection in the following ways:

- Feature Importance: SHAP helps determine the importance of each feature in a machine learning model's decision-making process. By analyzing the SHAP values assigned to each feature, one can identify which features contribute the most to the prediction of ransomware attacks. This information can aid in understanding the key indicators or patterns associated with ransomware.
- Model Interpretability: Ransomware detection models are often complex, involving various algorithms and techniques. SHAP provides a way to interpret and explain the predictions made by these models. It can help cybersecurity experts, analysts, and investigators understand the factors that contribute to a system being classified as potentially affected by ransomware. By analyzing the explanations provided by SHAP, they can gain insights into the decision-making process of the model.
- Anomaly Detection: Ransomware attacks often exhibit anomalous behavior compared to normal system usage. SHAP can help identify these anomalies by providing explanations for individual predictions. If a particular prediction has a high SHAP value for certain features, it indicates that those features strongly contributed to the model's decision. Unusual values or combinations of features can then be flagged as potential indicators of ransomware activity.
- Early Warning System: By training a model with historical ransomware attack data, SHAP can provide valuable insights into early warning signs. It can identify the specific indicators that are most indicative of ransomware

Table 10 List of RFECV-selected features from the 'Data1' dataset for each ML classifier that is not present in the top 40 highly contributing features

Classifier	API call features	Total	Average performance decrease with RFECV-selected features (%)
LR	NtEnumerateKey NtOpenEvent NtQueryInformationProcess	3	1.10
SGD	NtEnumerateKey NtOpenEvent NtQueryInformationProcess NtSetContextThread NtQueryDirectoryFile	5	2.02
KNN	NtDeviceIoControlFile NtAlertThread NtGetContextThread NtQueryDirectoryFile NtSetContextThread	5	0.9
NB	NtAllocateVirtualMemory	1	0.29
RF	NtCreateUserProcess NtGetContextThread NtQueryInformationProcess NtMapViewOfSection	4	1.27
SVM	NtSetContextThread NtDeviceIoControlFile	2	1.24

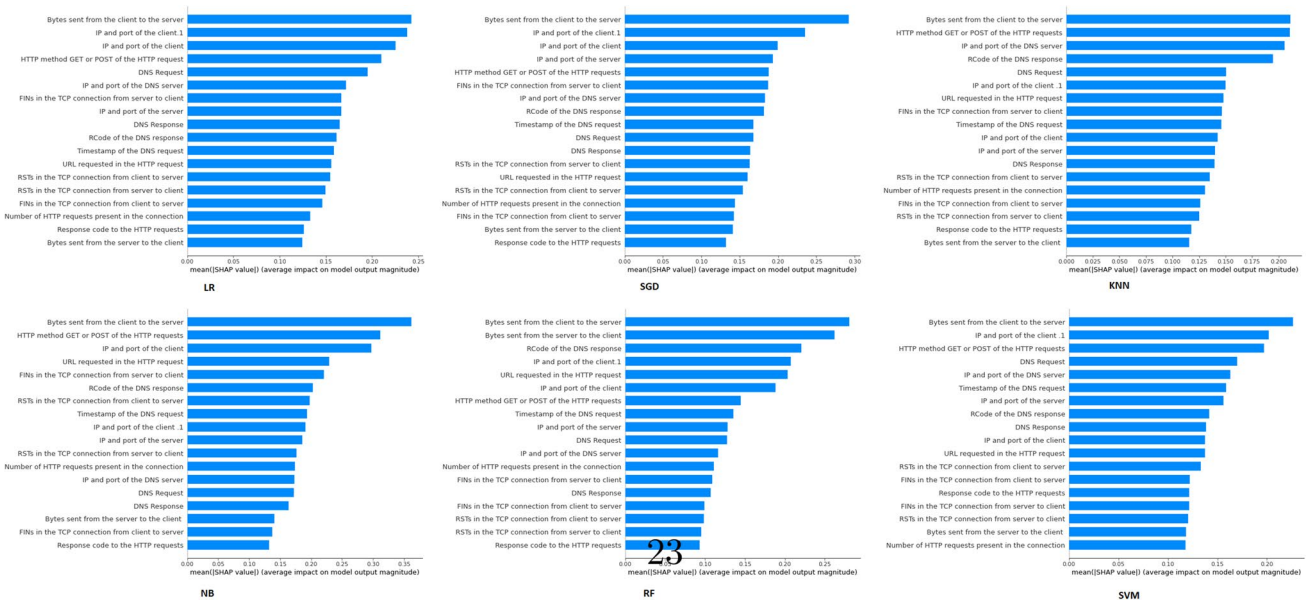


Fig. 12 Summary plot showing the features of the 'Data2' dataset in descending order based on their contribution to each ML classifier's decision

attacks, allowing organizations to proactively monitor and detect potential threats. This can help security teams respond quickly and prevent or mitigate the impact of ransomware attacks.

- Vulnerability Assessment: SHAP can be used to assess the vulnerability of a system to ransomware attacks. By analyzing the contributions of different features to the model's predictions, security professionals can identify

Table 11 Set of the optimum number of features selected by RFECV from the 'Data2' dataset

Classifier: LR	
Total Selected Features: 14	
RSTs in the TCP connection from server to client	RSTs in the TCP connection from client to server
FINs in the TCP connection from client to server	FINs in the TCP connection from server to client
Bytes sent from the client to the server	Bytes sent from the server to the client
DNS Request	DNS Response
IP and port of the client.1	IP and port of the client
HTTP method GET or POST of the HTTP request	IP and port of the server
Timestamp of the DNS request	RCode of the DNS response
Classifier: SGD	
Total Selected Features: 10	
IP and port of the client.1	IP and port of the client
IP and port of the server	FINs in the TCP connection from server to client
URL requested in the HTTP request	RSTs in the TCP connection from client to server
Bytes sent from the client to the server	HTTP method GET or POST of the HTTP requests
Number of HTTP requests present in the connection	Response code to the HTTP requests
Classifier: KNN	
Total Selected Features: 10	
Bytes sent from the client to the server	Bytes sent from the server to the client
HTTP method GET or POST of the HTTP requests	Response code to the HTTP requests
IP and port of the DNS server	RCode of the DNS response
URL requested in the HTTP request	IP and port of the client
RSTs in the TCP connection from client to server	Number of HTTP requests present in the connection
Classifier: NB	
Total Selected Features: 16	
RCode of the DNS response	RSTs in the TCP connection from client to server
IP and port of the server	RSTs in the TCP connection from server to client
Number of HTTP requests present in the connection	IP and port of the DNS serve
DNS Request	DNS Response
Bytes sent from the server to the client	FINs in the TCP connection from client to server
Response code to the HTTP requests	Bytes sent from the client to the server
HTTP method GET or POST of the HTTP requests	IP and port of the client
URL requested in the HTTP request	FINs in the TCP connection from server to client
Classifier: RF	
Total Selected Features: 13	
Timestamp of the DNS request	IP and port of the DNS server
Number of HTTP requests present in the connection	FINs in the TCP connection from server to client
Bytes sent from the client to the server	Bytes sent from the server to the client
RCode of the DNS response	IP and port of the client.1
URL requested in the HTTP request	IP and port of the client
HTTP method GET or POST of the HTTP requests	DNS Response
RSTs in the TCP connection from server to client	
Classifier: SVM	
Total Selected Features: 13	
RSTs in the TCP connection from client to server	RSTs in the TCP connection from server to client

Table 11 (continued)

Classifier: SVM	
Total Selected Features: 13	
Bytes sent from the client to the server	Bytes sent from the server to the client
Number of HTTP requests present in the connection	IP and port of the client .1
HTTP method GET or POST of the HTTP requests	DNS Request
IP and port of the DNS server	Timestamp of the DNS request
IP and port of the server	RCode of the DNS response
DNS Response	

Table 12 List of features from the 'Data2' dataset that were not selected by the RFECV

Classifier	Not selected network traffic features	Total	Average performance decrease with RFECV-selected features (%)
LR	IP and port of the DNS server	4	1.79
	URL requested in the HTTP request		
	Number of HTTP requests present in the connection		
	Response code to the HTTP requests		
SGD	Timestamp of the DNS request	8	1.07
	DNS Request		
	DNS Response		
	RSTs in the TCP connection from server to client		
	IP and port of the DNS server		
	RCode of the DNS response		
	FINs in the TCP connection from client to server		
	Bytes sent from the server to the client		
KNN	FINs in the TCP connection from client to server	8	2.26
	RSTs in the TCP connection from server to client		
	IP and port of the server		
	DNS Response		
	FINs in the TCP connection from server to client		
	Timestamp of the DNS request		
	DNS Request		
	IP and port of the client .1		
NB	Timestamp of the DNS request	2	1.06
	IP and port of the client .1		
RF	FINs in the TCP connection from client to server	5	1.09
	RSTs in the TCP connection from client to server		
	IP and port of the server		
	DNS Request		
	Response code to the HTTP requests		
SVM	FINs in the TCP connection from client to server	5	1.65
	Response code to the HTTP requests		
	FINs in the TCP connection from server to client		
	IP and port of the client		
	URL requested in the HTTP request		

the weak points in their systems. They can then focus on improving the security measures for those vulnerable areas, reducing the risk of successful ransomware attacks.

Overall, SHAP can enhance ransomware detection by providing interpretability, feature importance analysis, anomaly detection, and a proactive approach to identifying and mitigating potential threats. Thus, SHAP offers advantages over iterative feature selection techniques. By leveraging SHAP, organizations can better understand the factors driving ransomware predictions and strengthen their cybersecurity defenses accordingly.

5 Conclusion

Early detection of ransomware is a key research area in cybersecurity. While feature selection techniques aim to improve detection accuracy while reducing overfitting and time complexity, the selected features must be crucial to support the technique's effectiveness. This research thoroughly analyzes the performance of an iterative feature selection technique-Recursive Feature Elimination with Cross-Validation (RFECV) with widely utilized Supervised Machine Learning models on two different ransomware datasets. By employing the SHapley Additive exPlanations (SHAP) framework, critical features are determined when RFECV is not integrated with the ML models and then compared to RFECV-selected features. The study reveals that without RFECV the classification accuracies are better than with RFECV (For 'Data1' dataset, with and without feature selection LR secures 98.20% and 99.30% overall accuracy respectively. For 'Data2' dataset, with and without feature selection NB secures 97.89% and 98.95% overall accuracy respectively). Again, RFECV occasionally fails to select impactful features from both datasets, leading to both Type 1 and Type 2 errors. Moreover, the RFECV approach fails to disclose the importance-based order of selected features, reducing its efficacy in ransomware classification. Consequently, the study highlights the significance of integrating explainability techniques to identify highly contributing features, as relying solely on iterative feature selection techniques is not sufficient for strengthening ransomware detection systems. However, the research exclusively concentrates on the RFECV feature selection technique and does not assess the performance of Deep Learning models. Therefore, future investigations should explore other iterative feature selection methods and incorporate Deep Learning models to expand this research further.

Acknowledgements This study was partially funded by NetApp.

Author contributions The contents of the article were experimented and edited by Rawshan Ara Mowri with the supervision of Dr. Maduri Siddula and Dr. Kaushik Roy.

Data availability The first dataset- 'Data1' has been generated by analyzing the Windows Portable Executable (PE) files collected from VirusShare repository: https://virusshare.com/ (last accessed March 20, 2023) [34], theZoo: https://github.com/ytisf/theZoo (last accessed April 1, 2023) [36], and Hybrid-Analysis.com: https://www.hybrid-analysis.com/ (last accessed April 1, 2023) [37]. The Web-Crawler- 'GetRansomware' to automate collecting the PE files of the ransomware families from the VirusShare repository is available here: https://github.com/rmowri/GetRansomware (last accessed June 7, 2023) [35]. The second dataset- 'Data2' has been generated by analyzing the Packet Capture (PCAP) files collected from the malware-traffic-analysis: https://www.malware-traffic-analysis.net/. (last accessed May 28, 2023) [38].

Declarations

Competing interests The authors declare no competing interests.

References

1. Hasan MM, Rahman MM. Ranshunt: A support vector machines based ransomware analysis framework with integrated feature set. In: 2017 20th International Conference of Computer and Information Technology (ICCIT), 2017;1–7. https://doi.org/10.1109/ICCITECHN.2017.8281835.

2. Young AL, Yung M. Cryptovirology: the birth, neglect, and explosion of ransomware. Commun ACM. 2017;60(7):24–6. https://doi.org/10.1145/3097347.
3. Moussaileb R, Cuppens N, Lanet J-L, Le Bouder H. Ransomware network traffic analysis for pre-encryption alert. In: Benzekri A, Barbeau M, Gong G, Laborde R, Garcia-Alfaro J, editors. Foundations and practice of security. Cham: Springer; 2020. p. 20–38.
4. Young A, Yung M. Cryptovirology: extortion-based security threats and countermeasures. In: Proceedings 1996 IEEE Symposium on Security and Privacy, 1996;129–140. https://doi.org/10.1109/SECPRI.1996.502676.
5. Savage K, Coogan P, Lau H. The evolution of ransomware. https://docs.–broadcom.com/doc/the-evolution-of-ransomware-15-en (accessed on 10 March 2023).
6. Gane B. 9 Scariest Ransomware Viruses. Available. http://www.e92plus.com/blog/ e92plus/2017/06/02/9-scariestransomware-viruses (accessed on 29 June 2017).
7. Young A, Yung M. Malicious cryptography: exposing cryptovirology. Hoboken: John Wiley & Sons Inc; 2004.
8. Yang T, Yang Y, Qian K, Lo DC-T, Qian Y, Tao L. Automated detection and analysis for android ransomware. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, 2015;1338–1343. https://doi.org/10.1109/HPCC-CSS-ICESS.2015.39.
9. Sgandurra D, Muñoz-González L, Mohsen R, Lupu EC. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. 2016.
10. Maniath S, Ashok A, Poornachandran P, Sujadevi VG, Sankar AU, P, Jan S. Deep learning lstm based ransomware detection. In: 2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE), pp. 2017;442–446. https://doi.org/10.1109/RDCAPE.2017.8358312.
11. Vinayakumar R, Soman KP, Senthil Velan KK, Ganorkar S. Evaluating shallow and deep networks for ransomware detection and classification. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017;259–265 https://doi.org/10.1109/ICACCI.2017.8125850.
12. Takeuchi Y, Sakai K, Fukumoto S. Detecting ransomware using support vector machines. In: Workshop Proceedings of the 47th International Conference on Parallel Processing. ICPP Workshops '18. Association for Computing Machinery, New York, NY, USA. 2018https://doi.org/10.1145/3229710.3229726.
13. Hwang J, Kim J, Lee S, Kim K. Two-stage ransomware detection using dynamic analysis and machine learning techniques. Wireless Pers Commun. 2020;112:2597–609.
14. Zhang H, Xiao X, Mercaldo F, Ni S, Martinelli F, Sangaiah AK. Classification of ransomware families with machine learning based onn-gram of opcodes. Futur Gener Comput Syst. 2019;90:211–21. https://doi.org/10.1016/j.future.2018.07.052.
15. Baldwin J, Dehghantanha A. In: Dehghantanha, A., Conti, M., Dargahi, T. (eds.) Leveraging Support Vector Machine for Opcode Density Based Detection of Crypto-Ransomware, 2018;107–136. Springer, Cham. https://doi.org/10.1007/978-3-319-73951-9_6.
16. Khammas BM. Ransomware detection using random forest technique. ICT Express. 2020;6(4):325–31. https://doi.org/10.1016/j.icte.2020.11.001.
17. Subedi KP, Budhathoki DR, Dasgupta D. Forensic analysis of ransomware families using static and dynamic analysis. In: 2018 IEEE Security and Privacy Workshops (SPW), 2018;180–185. https://doi.org/10.1109/SPW.2018.00033.
18. Shaukat SK, Ribeiro VJ. Ransomwall: a layered defense system against cryptographic ransomware attacks using machine learning. In: 2018 10th International Conference on Communication Systems & Networks (COMSNETS), 2018;356–363. https://doi.org/10.1109/COMSNETS.2018.8328219.
19. Ferrante A, Malek M, Martinelli F, Mercaldo F, Milosevic J. Extinguishing ransomware—a hybrid approach to android ransomware detection. In: Imine A, Fernandez JM, Marion J-Y, Logrippo L, Garcia-Alfaro J, editors. Foundations and Practice of Security. Cham: Springer; 2018. p. 242–58.
20. Roundy KA, Miller BP. Binary-code obfuscations in prevalent packer tools. ACM Comput Surv. 2013. https://doi.org/10.1145/2522968.2522972.
21. Coogan K, Debray S, Kaochar T, Townsend G. Automatic static unpacking of malware binaries. In: 2009 16th Working Conference on Reverse Engineering, 2009;167–176. https://doi.org/10.1109/WCRE.2009.24.
22. Almashhadani AO, Kaiiali M, Sezer S, O'Kane P. A multi-classifier network-based crypto ransomware detection system: A case study of Locky ransomware. IEEE Access. 2019;7:47053–67. https://doi.org/10.1109/ACCESS.2019.2907485.
23. Chen Z-G, Kang H-S, Yin S-N, Kim S-R. Automatic ransomware detection and analysis based on dynamic api calls flow graph. In: Proceedings of the International Conference on Research in Adaptive and Convergent Systems. RACS '17, 2017;196–201. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3129676.3129704.
24. Cabaj K, Mazurczyk W. Using software-defined networking for ransomware mitigation: the case of cryptowall. IEEE Network. 2016;30(6):14–20. https://doi.org/10.1109/MNET.2016.1600110NM.
25. Aragorn T, Yun-chun C, YiHsiang K, Tsungnan L. Deep Learning for Ransomware Detection. https://www.semanticscholar.org/paper/Deep-Learning-for-Ransomware-Detection-Aragorn-Yun-chun/cc3a41b37230861cfe429632744e0d1db19256b7 (accessed on 11 March 2023).
26. Alhawi OMK, Baldwin J, Dehghantanha A. Leveraging machine learning techniques for windows ransomware network traffic detection, 2018;93–106 https://doi.org/10.1007/978-3-319-73951-9_5.
27. Bae SI, Lee GB, Im EG. Ransomware detection using machine learning algorithms. Concurrency Comput Pract Exp. 2020;32(18):5422. https://doi.org/10.1002/cpe.5422.
28. Almashhadani AO, Carlin D, Kaiiali M, Sezer S. Mfmcns: a multi-feature and multi-classifier network-based system for ransomworm detection. Comput Secur. 2022;121: 102860. https://doi.org/10.1016/j.cose.2022.102860.
29. Singh J, Sharma K, Wazid M, Das AK. Sinn-rd: spline interpolation-envisioned neural network-based ransomware detection scheme. Comput Electr Eng. 2023;106: 108601. https://doi.org/10.1016/j.compeleceng.2023.108601.

30. Continella A, Guagnelli A, Zingaro G, De Pasquale G, Barenghi A, Zanero S, Maggi F. Shieldfs: a self-healing, ransomware-aware filesystem. In: Proceedings of the 32nd Annual Conference on Computer Security Applications. ACSAC '16, pp. 336–347. Association for Computing Machinery, New York, NY, USA 2016. https://doi.org/10.1145/2991079.2991110.
31. Lu T, Zhang L, Wang S, Gong Q. Ransomware detection based on v-detector negative selection algorithm. In: 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), 2017;531–536. https://doi.org/10.1109/SPAC.2017.8304335.
32. Zahoora U, Khan A, Rajarajan M, Khan SH, Asam M, Jamal T. Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive pareto ensemble classifier. Sci Rep. 2022. https://doi.org/10.1038/s41598-022-19443-7.
33. Masum M, Hossain Faruk MJ, Shahriar H, Qian K, Lo D, Adnan MI. Ransomware classification and detection with machine learning algorithms. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022;0316–0322. https://doi.org/10.1109/CCWC54503.2022.9720869.
34. VirusShare.com—Because Sharing is Caring. http://virusshare.com (accessed on 8 October 2022).
35. rmowri/GetRansomware. https://github.com/rmowri/GetRansomware (accessed on 8 October 2022).
36. ytisf/theZoo. http://github.com/ytisf/theZoo (accessed on 8 October 2022).
37. Free Automated Malware Analysis Service—powered by Falcon Sandbox. https://www.hybrid-analysis.com/ (accessed on 8 October 2022).
38. malware-traffic-analysis.net Homepage. https://www.malware-traffic-analysis.net/ (accessed on 11 March 2023).
39. Al-rimy BAS, Maarof MA, Shaid SZM. Ransomware threat success factors, taxonomy, and countermeasures: a survey and research directions. Comput Secur. 2018;74:144–66. https://doi.org/10.1016/j.cose.2018.01.001.
40. Al-Bakri AM, Hussein HL. Static analysis based behavioral api for malware detection using markov chain. Comput Eng Intel Syst. 2014;5:55–63.
41. Amro SA, Cau A. Behavioural api based virus analysis and detection. 2012.
42. Falcon Sandbox: Automated Malware Analysis Tool - CrowdStrike. https://www.crowdstrike.com/products/threatintelligence/falconsandbox-malware-analysis (accessed on 10 May 2022).
43. PayloadSecurity. https://github.com/PayloadSecurity/VxAPI (accessed on 9 October 2022).
44. Recursive Feature Elimination. https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html#:~:text=Recursive%20feature%20elimination%20(RFE)%20is,number%20of%20features%20is%20reached (accessed on 30 October 2022).
45. Narudin FA, Feizollah A, Anuar NB, Gani A. Evaluation of machine learning classifiers for mobile malware detection. Soft Comput. 2016;20(1):343–57. https://doi.org/10.1007/s00500-014-1511-6.
46. Berrueta E, Morato D, Magaña E, Izal M. A survey on detection techniques for cryptographic ransomware. IEEE Access. 2019;7:144925–44. https://doi.org/10.1109/ACCESS.2019.2945839.
47. Wireshark. https://www.wireshark.org/ (accessed on 9 October 2022).
48. Wireshark User Guide. https://www.wireshark.org/docs/wsug_html/ #ChapterIO (accessed on 9 October 2022).
49. Berrueta E, Morato D, Magaña E, Izal M. Open repository for the evaluation of ransomware detection tools. IEEE Access. 2020;8:65658–69. https://doi.org/10.1109/ACCESS.2020.2984187.
50. Pandas get_dummies (One-Hot Encoding) Explained. https://datagy.io/pandas-get-dummies/ (accessed on 30 October 2022).
51. One-vs-Rest and One-vs-One for Multi-Class Classification. https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/ (accessed on 1 December 2022).
52. sklearn.multiclass.OneVsRestClassifier. https://scikit-learn.org/stable/ modules/generated/sklearn.multiclass. OneVsRestClassifier.html (accessed on 1 December 2022).
53. sklearn.multiclass.OneVsOneClassifier. https://scikit-learn.org/stable/ modules/generated/sklearn.multiclass. OneVsOneClassifier.html (accessed on 1 December 2022).
54. sklearn.model_selection.RandomizedSearchCV. https://scikit-learn.org/stable/ modules/generated/sklearn.model_selection. RandomizedSearchCV.html (accessed on 1 December 2022).
55. Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17, 2017;4768–4777. Curran Associates Inc., Red Hook, NY, USA.
56. Molnar C. Chapter 6 Model-Agnostic Methods. https://christophm.github.io/ interpretable-ml-book/agnostic.html (accessed on 11 March 2023).
57. Welcome to the SHAP Documentation. https://shaplrjball.readthedocs.io/en/ latest/ index.html (accessed on 9 October 2022).