ANDROID MALWARE DETECTION USING MACHINE LEARNING TECHNIQUES

Vipin Kumar, Mr. Shyam Dwivedi

Vipin Kumar, M.Tech Scholar, Department of Computer Science and Engineering, Rameshwarm Institute of Technology & Management, Lucknow, India.
Shyam Dwevedi, Assistant Professor, Department of Computer Science and Engineering, Rameshwarm Institute of Technology & Management, Lucknow, India.

Abstract—Smart phone usage has exploded in recent years as a result of their low cost and digitalization of various services. With the increased use of smartphones, new security concerns have emerged, with threats from various malwares being the primary source of concern in this study. Various fraudulent mobile apps have exploded in popularity in recent years, notably on the Android platform, creating insurmountable barriers to identifying hazardous apps. Viruses writers generate new malware on a daily basis as the popularity of Android devices grows, posing a threat to the system's integrity and users' privacy. The goal of this thesis is to apply machine learning approaches to detect Android malware. A malware detection framework for Android is proposed, in which six machine learning models are employed for categorization of distinct malware, including Decision Trees, Support Vector Machines, Nave Bayes, Random Forests, K-Nearest Neighbors, Ensemble Methods / Extra-Tree Classifier. The proposed framework's performance is assessed using the CICMalAnal2017 Android malware dataset. The CICMalAnal2017 dataset contains a variety of malware, including adware, ransomware, and scareware. Feature Correlation, Random Forest Importance, Chi-Square Test, and Information Gain are among the four types of feature selection strategies used. Various machine learning methods, such as Decision Trees, Support Vector Machines, Nave Bayes, Random Forests, K-Nearest Neighbors, Ensemble Methods / Extra-Tree Classifier, are tested for malware classification. Ultimately, ML-based techniques for assessing source code vulnerabilities are described, because adding security after the app has been released may be more difficult. As a result, the goal of this study is to help academics gain a deeper understanding of the topic and identify prospective future research and development directions. Through experiments, it is observed that Random Forests and Extra-Tree Classifier have achieved 97% and 88% Accuracy, respectively and highest F1-Scoreachieved by the Random Forests and Extra-Tree Classifier, which is 97% and 86% , respectively.

Keywords— Malware detection; code vulnerability; machine learning; Android security.

I. INTRODUCTION

Malware, also known as malicious software or malicious code, is defined as "any code introduced, modified, or removed from a software system with the aim to intentionally inflict harm or disrupt the system's intended function." Malware can be a piece of code that is connected to a legal application, a standalone program, or a mix of the two.

In today's digital economy, most firms and individuals process and store digital content via computer networks and information systems. Not only are modern businesses digitizing their paper-based materials, but they are also developing new business models based on digital assets. Modern businesses such as Facebook, Netflix, and others are fantastic examples. As a result, when technological infrastructure is penetrated by hostile assaults, cyber risks offer a substantial problem. Malware is used in many of these assaults. The amount of reported security breaches due to viruses, trojans, ransom-wares, and other malware has increased dramatically in recent years, with stories of malware infestations hitting the news today more than ever. Almost every week, a new security flaw is discovered, which may be seen as a failure of the security industry to regulate and detect harmful information. Given the high volume of malware attacks and the growing popularity and widespread success of

Copyright @ 2022 Authors

machine learning (ML) methodologies in classification across a variety of domains, it's only natural to see these techniques used to supplement traditional malware detection methods, particularly supervised learning techniques. So far in 2016, more than 500 million malware samples have been discovered. In 2014, 317 million malware variants were discovered as new malware variants, up from 252 million malware variants in 2013, a 26 percent increase (Symantec Corporation 2015).

Malware Types: Malware gains access to a computer, obtains information, or damages it without the owner's knowledge. Malware, which is a major danger to computer security today, is used in a variety of attacks. Malware comes in many varieties, including:

Trojans: The name of this sort of malware comes from its resemblance to biological viruses. A virus is often made up of two subroutines. The first subroutine is in charge of infecting other programs by attaching the viral code, whereas the second subroutine is the real malware payload (i.e. virus payload). Viruses attach themselves to programs and spread when they are executed, either deliberately (by clicking on executable) or subconsciously (by using auto-run capabilities), relying on other programs (as hosts) and user involvement. In order for viruses to infect new computers, an infected file must be transferred between them by a user (e.g., USB pen, email attachment), emphasizing the user's participation in this mode of transmission.

Worm: A worm has the capacity to self-replicate like a virus, but it doesn't require user contact or a host. This is accomplished by making a worm a self-contained software that searches networks for susceptible computers to infect with a copy of itself. Worms, like viruses, can carry extra payloads that execute harmful operations on affected computers.

Spyware: A program that monitors a user's activities using various vectors (for example, capturing screenshots of the user's desktop/webcam).

Trojan: Viruses and worms are concerned with self-propagation; trojans, on the other hand, are concerned with tricking users into executing them without consideration for propagation. Trojans aim to get users to run the software by offering some beneficial feature. Trojans avoid the requirement for a host (in the case of a virus) or an exploit (in the case of a worm) by tricking users into running them.

Adware: A program that presents advertisements to the user automatically. The majority of them do not hurt someone directly.

Backdoor: Allows remote control of a system by installing or exploiting an unknown entry point into a user's system.

Bot: Short for "robot," a software that takes orders from a cybercriminal and executes them mindlessly. Compromised computers are known as zombies because of this trait.

Dropper: A program that makes it easier for other viruses to infiltrate a system**.**

Key-logger: To steal private information, a program monitors a user's input (e.g., keyboard strokes).

Password Stealer(PWS): A program that focuses on a user's personal data, such as user names and passwords.

Ransomware: Access to a system/files is restricted (typically by encryption) and a ransom is demanded to restore access.

Copyright @ 2022 Authors

Remote Administration Tool - RAT:A program that allows an administrator to operate a system remotely. Backdoor and Bot are similar, however on infected systems, Backdoor generally has more rights.

Rootkit: Program that hides the existence of other (harmful) program, making it difficult to identify malicious software.

Scareware: The user is tricked into paying for malware removal or downloading genuine malware by a program that seems to be malware**.**

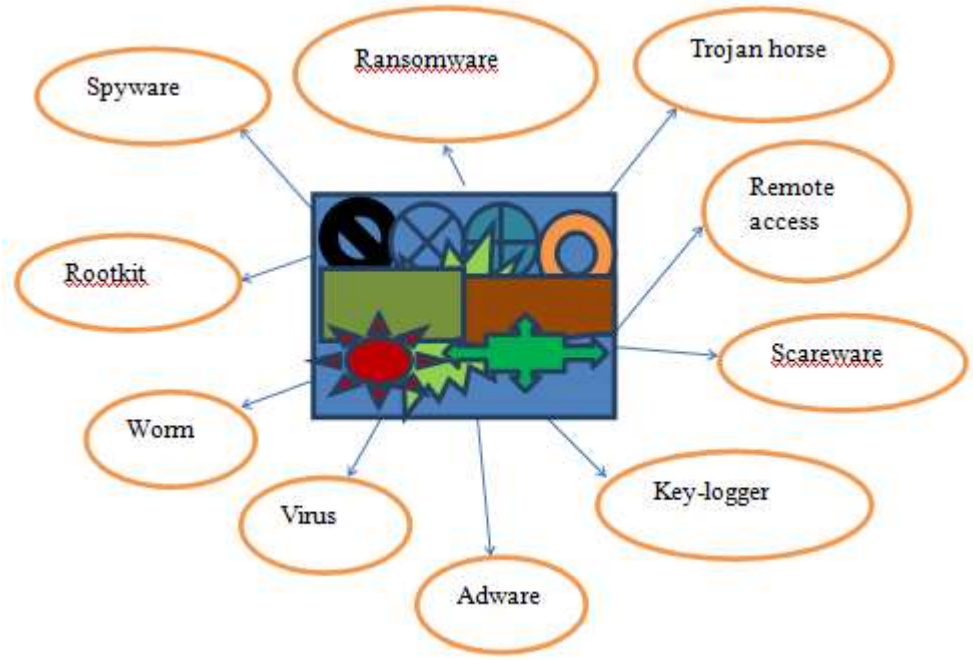


Figure 1: Types of Malware

II. MALWARE DETECTION APPROACHES

Anti-virus software is used to identify malware occurrences. Early detection methods merely searched for byte sequences to detect malware; hence these are also known as scanners. Before we get into the specifics of the various detection methods, it's important to note that scanners may be utilized as on-demand or on-access scanners. On-demand antivirus is only activated when the user requests it. On-access antiviral is memory-resident, which means it loads as a program and intercepts file and disc access operations. When files are opened, generated, or closed, the anti-virus software examines the altered item.
The knowledge of opaque malware security is a vital subject in malware recognition as per machine learning techniques [1] as consequence of the developing malware in innovation. Behavior-based and signature-based techniques are the two primary types of malware detection methods [2].

Signature-Based Malware Detection Approach: Signature is a malware characteristic that wraps the program structure and allows each virus to be identified individually. Commercial antivirus software frequently uses a signature-based detection method. This method detects known malware quickly and effectively, however it falls short of detecting new malware. Furthermore, by employing obfuscation techniques, malware from the same family can readily evade signature-based detection. The malware detection taxonomy based on machine learning approaches. According to the figure 2, the API calls features, assembly features, and binary features are existing approaches for malware detection method. These features use machine learning methods for predicting and detecting malicious files.

Copyright @ 2022 Authors

Table 1: Signature-Based malware detection approach has its own set of Benefits and Drawbacks

Malware Detection Approach	Benefits	Drawbacks
Signature-Based	For known malware, it's quick and effective. Used for a long time. Effective in detecting malware from the same family. It's simple to run. Accessible to a large audience.	The polymorphic malwares are not being detected. Information replication in a large database. Detection of new generation malware is insufficient.

Behavior-Based Malware Detection Approach: The behavior-based malware detection technique uses monitoring tools to examine the program's activity and decide if it is malicious or benign. Although the program's code is altered, the program's behavior remains consistent, allowing this technique to identify the vast majority of new malware [3]. Some malware binaries, on the other hand, do not execute well in a protected environment (virtual machine, sandbox). As a result, malware samples may be mislabeled as harmless.

Table 2: Behavior-Based malware detection approach has its own set of Benefits and Drawbacks

Malware Detection Approach	Benefits	Drawbacks
Behavior-Based	Detecting malware assaults that haven't been thought of before. Polymorphic malware detection is a difficult task. The malware's functioning is determined by this parameter. Detection of new malware is effective. Different variations of the same virus can be detected with ease [4,5].	For behavioral patterns, storage complexity is important. Complexity of time. Malware and benign samples have certain behaviors in common. It's impossible to define every conceivable behavior. It's difficult to categorize activity into harmful and non-malicious categories.

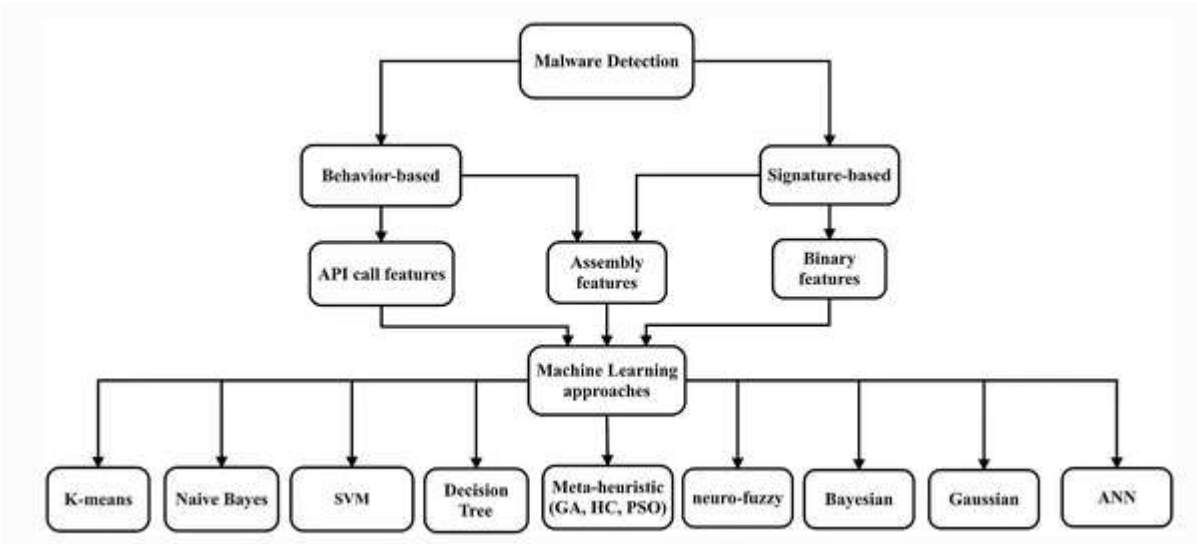


Figure 2: Taxonomy of Malware Detection Approaches

Copyright @ 2022 Authors

III. LITERATURE REVIEW

This section presents a brief review of the existing schemes for malware detection. We also talk about the gaps in the available literature and how we might close them when creating the malware detection model.

Schultz et al [6] presented a network packet-based malware detection methodology based on cloud computing. They utilized data mining methods to minimize the branching of packets by accumulating packet information, whether or not it is relevant for malware identification. Enck et al. [7] introduced the Kirin framework, which allows us to identify malicious programs by looking at the rights they seek throughout the installation process. Kirin is built on a set of principles that assist us in reducing the impact of malware in Android apps.

Wagener et al. [8] suggested an automated and adaptable method for extracting malware behavior from system calls. To compute related distances, the alignment approach was utilized to find similarities, and the Hellinger distance was determined. The research claims that disguised malware versions with identical characteristics can be discovered. The authors claim that a phylogenetic tree that reflects malware's common functions can help with categorization. The following are the missing aspects of the article:

• There is a lack of understanding about the malware dataset,

• There is no statistical evaluation of performance,

• There is no comparison of the suggested approach to other methods.

Furthermore, it is unclear how a phylogenetic tree may enhance performance.

Naval et al. [9] proposed a dynamic malware detection system that gathers system calls and creates a graph that discovers semantically meaningful pathways between them. A NP-complete issue is locating all semantically meaningful routes in a network. As a result, the authors assessed the most relevant pathways, which define malware behaviors that aren't seen in benign samples, to decrease the time complexity. According to the authors, the suggested approach beats its competitors because it can identify malware utilizing system-call injection assaults at a high rate, whereas other methods can't. The article contains a number of flaws, including performance overhead during path calculation, vulnerability to call-injection attacks, and the inability to effectively discover all semantically meaningful pathways. The performance may be improved if these constraints are removed.

Table 3: Summary of related works on malware detection approaches

Sr. No.	Paper	Publication	Goal	Year
1.	Schultz et al.[7]	IEEE	The detection of new malware	2001
2.	Kinder et al.[8]	SPRINGER	Find more about new and distinct types of malware	2005
3.	Karniket al.[9]	IEEE	Determine the many types of malware	2007
4.	Master et al.[10]	IEEE	Determine a variety of execution routes	2007
5.	Zhang et al.[11]	SPRINGER	Distinguish between typical malware and other types of malware	2007
6.	Wagener et al.[12]	SPRINGER	Find fresh malware as well as different types of malware	2008
7.	Beaucams and Matrionet al.[13]	IEEE	Determine the many types of malware	2009
8.	Cha et al.[14]	IEEE	With only one signature, you can catch different types of malware	2011

Copyright @ 2022 Authors

9.	Anderson et al.[15]	SPRINGER	96.41% correctly identify various types of malware	2011
10.	Song et al.[16]	SPRINGER	Recognize new malware	2012
11.	Baldangomboet al.[17]	ELSEVIER	With a success rate of 99%, it can identify known malware	2013
12.	Park et al.[18]	ELSEVIER	Determine the many types of malware	2013
13.	Islam et al.[19]	ELSEVIER	97% accuracy in detecting new and different types of malware	2013
14.	Naval et al.[20]	IEEE	Detect malicious code injection	2015
15.	Aashima and Bajaj et al.[21]	SPRINGER	With minimal cost, it is possible to anticipate the type of malware	2016
16.	Das et al.[22]	IEEE	Find fresh malware as well as different types of malware	2016
17.	Monireet al.[23]	HINDAWI	Recognize old and new malware	2016
18.	Cimitileet al.[24]	SPRINGER	Find more about new and distinct types of malware	2017
19.	Griffin et al.[25]	SPRINGER	Determine the many types of malware	2018

Kinder et al.[10] by suggested was STREAM, which automatically downloads, executes, and extracts features from Android applications. The collected characteristics are then utilized to build machine learning classifiers to identify malware in Android apps. STREAM has a drawback in that loading data requires a lot of system resources and time. Wei et al. [11] develop a malware detection algorithm based on abnormal Android app behavior. They created a model that took network information into account as a feature and used Naive Bayes and logistic machine learning techniques to obtain greater accuracy.

Ali et al. [12] proposed a Gaussian mixture-based malware detection approach. They gathered characteristics based on hardware use, such as CPU, memory, and battery, then used Gaussian mixture to train it. However, the approach they offer has a flaw: it necessitates the use of a distant server for calculation. Dixon et al. [13] created a model based on how a smartphone's battery life changes when it's infected with malware. However, the methodology they offer is incapable of detecting certain advanced viruses. Suarez-Tangil et al. [14] investigated whether technique, cloud-based detection or on-device detection, saves the most power. They proposed using machine learning techniques to compare the two ways using a power model. The use of a cloud-based detection approach is a more effective and preferable way for detecting malware. Chen et al. [15] presented a method that tracks how cellphones behave while transferring personal information to an external source. However, their study's solution is ineffective since it does not allow real-time detection.

Quan et al. [16] identify to malware from Android, examined three separate feature sets: native code, system calls, and API requests. The detection rate is determined by a predetermined threshold value. Ng et al. [17] used the dendritic cell approach to create a model that included the system call as a feature. Statistical techniques were used to pick the best characteristics, resulting in a better detection rate. Sheen et al. [18] presented a malware detection solution for Android based on API calls and permissions. They choose features by training three distinct classifiers with the relief algorithm: J48, SVM, and Naive Bayes. The detection rate is decent, however it takes a lot of resources and puts a lot of strain on the computer.

Tong and Yan [19] suggested a hybrid technique to detecting malware on Android that relied on individual and sequential system calls linked to file and network access. Their method can detect unexpected app behavior with a 91.76 percent detection rate. However, the proposed method has a flaw: it can't detect in real time. Fung et al. [20] introduced the RevMatch decision model, which uses the idea of malware detection history to determine whether an Android app is infected or not. This method does not allow for real-time detection. Abawajy and Kelarev [21] introduced ICFS, which incorporates feature selection methods and machine learning classifiers to identify malware on Android.

Tang et al. presented a bioinformatics approach for generating accurate polymorphic worm exploit-based signatures [22]. Multiple sequence alignment rewards consecutive substring extractions, noise reduction removes noise effects, and signature

Copyright @ 2022 Authors

transformation makes the reduced regular expression signature compatible with existing IDSs are the three phases in the approach.The authors claim that the proposed schema is noise-tolerant, as well as more accurate and exact than existing exploit-based signature generating schemas. This is because it extracts more polymorphic worm characters, such as one-byte invariants and invariant-by-byte distance constraints. The suggested schema, on the other hand, is confined to polymorphic worms and cannot be applied to other malware kinds.

Borojerdi and Abadi presented the MalHunter detection system, a novel approach based on sequence grouping and alignment [23]. For polymorphic malware, it automatically produces signatures depending on malware behavior. This is how the innovative technique works: The behavior sequences are constructed first from various malware samples. Different groups are then formed and kept in the database based on comparable behavioral sequences. Behavior sequences are acquired and compared to sequences already created and saved in the database in order to detect malware samples. The sample is classified as malware or benign based on its similarities.

According to the authors, the suggested schema is resistant to obfuscation techniques and may be used to identify various forms of polymorphic malware rather than being restricted to a single malware type. The authors also claim that the proposed system outperforms state-of-the-art signature generation approaches previously published in the literature, such as Tang et al., Newsome et al., and Perdisci et al.. The suggested approach is confined to polymorphic malware and has only been tested on a few hundred samples, which is insufficient to evaluate its effectiveness.

IV. PROPOSED ANDROID MALWARE DETECTION SCHEMES

The Python API includes the CSV module and the reader() method for loading CSV files. After loading the CSV data, transform it to a NumPy array and utilize it for machine learning.

Data Preprocessing:

Data Preprocessing is a process to convert raw data into meaningful data using different techniques [24-27].

Data in the real world is dirty like: Incomplete, noisy, inconsistent, duplicate.

Some technique in data preprocessing like: Missing values/Data, Data cleaning, Data integration, Data reduction, Data transformation, Data discretization.

Data cleaning

Data cleaning means fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.

Data integration

Data integration is a technique to merge data from multiple sources into a coherent data store, such as a data warehouse.

Data reduction

Data reduction is a technique used to reduce the data size by aggregating, eliminating redundant features, or clustering, for instances.

Data transformation

Data transformation means data are transformed or consolidating into forms appropriate for ML model training, such as normalization, may be applied where data are scaled to fall within a smaller range like 0.0 to 0.1.

Data Discretization

Data Discretization technique transforms numeric data by mapping values to interval or concept labels.

Label Encoding

Label Encoding apply on categorical variables. When the categorical variables convert into the number then it has mathematical value.

Copyright @ 2022 Authors

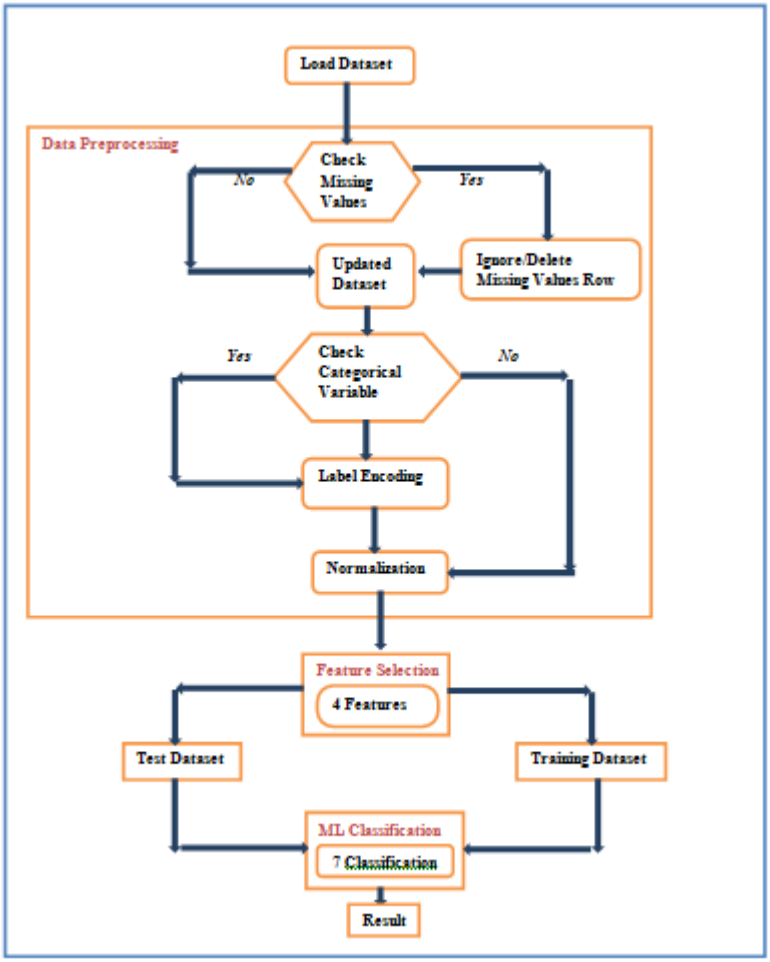


Figure 3: Block diagram of Android Malware Detection Schemes

Normalization: Normalization is there-scale features in the fixed range between 0 to1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalization also called as Min-Max Scaling. This last step involved in data preprocessing and before ML model training.

Feature Selection: When creating a predictive model, the technique of feature selection is used to reduce the number of input variables. Depending on the number of features produced in the preceding step, feature reduction may be required to increase the machine learning algorithm's accuracy and reduce processing overhead. I have used a variety of feature techniques, including information gain, Chi-square test, feature correlation, and random forest significance [28].

- Shorter training time
- Easier to interpret the model
- Reduce over-fitting
- Improves accuracy

Machine Learning Techniques: The field of Machine Learning (ML) focuses on the creation of computer algorithms that improve over time. When a program is given a task and some experience, and its performance at the task increases as a result of the experience, the program is said to be learning the task. We used machine learning skills and an increasing number of malware samples to study how to create a model that accounts for the aforementioned problems [29,30]. To do so, we used an incremental method to develop a malware detection model, allowing us to assess the effects of ground-truth (i.e. the grey region when it comes to malware) and how enforcing temporal consistent learning affects the final findings.

The following are the steps in a typical machine learning experiment:

Copyright @ 2022 Authors

1. A matrix of relevant observations and characteristics.

2. Processing the dataset after it has been prepared.

3.Creating training and testing sets from the data.

4.Selecting and training the classifier using the training set.

5.Determine the trained classifier's prediction accuracy using the testing set.

6. Using will-defined parameters, assess the accuracy of the classifier.

Once the classifier has been properly validated and certified for accuracy, it is implemented on large-scale systems. Some of the most widely used machine learning classifiers in proposals are Decision Trees Classifier, Support Vector Machine Classifier, Naïve Bayes Classifier, Random Forests Classifier, K-Nearest Neighbors, Ensemble Methods / Classifier.

After being tested and approved for accuracy, the classifier is implemented on large-scale systems. Researchers have been testing the accuracy of utilizing machine learning techniques to identify unknown malware using attributes obtained through static, dynamic, or hybrid methods [31,32]. To detect the malware, a android malware detection framework is proposed in which six machine learning models such as Decision Trees, Support Vector Machine, Naïve Bayes, Random Forests, K-Nearest Neighbors, Ensemble Methods / Extra-Tree Classifier are used for classification of different malware. Machine learning classifiers that have been used in a range of proposals include the following:

Decision Tree Classifier: The model is classified as supervised machine learning. Decision trees are versatile algorithms that can perform both classsification and regression tasks, and even multioutput tasks. Scikit-Learn uses the classification and regression tree algorithm to train decision trees. By checking for one characteristic at each node, it tries to divide the dataset into smaller groups. The algorithm divides the dataset into smaller chunks until all of the observations in a given subset are classified as "virus" or "benign." Which produces only binary trees: nonleaf nodes always have two children (i.e., questions only have yes/no answers).

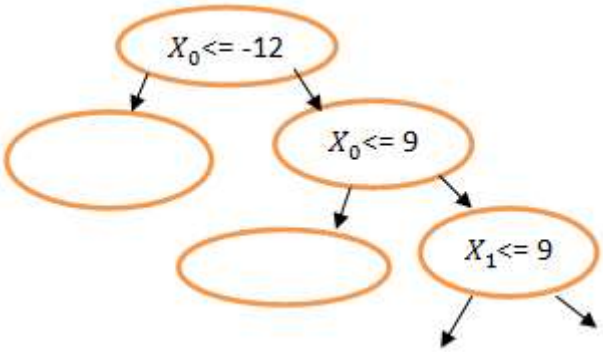


Figure 4: Decision tree classifier

Support Vector Machine (SVM) Classifier: To distinguish two classes with the greatest margin, SVM produces a hyper-plane. The hyper-plane and the nearest point are separated by that gap. As a result, the approach may be more broadly used.

Copyright @ 2022 Authors

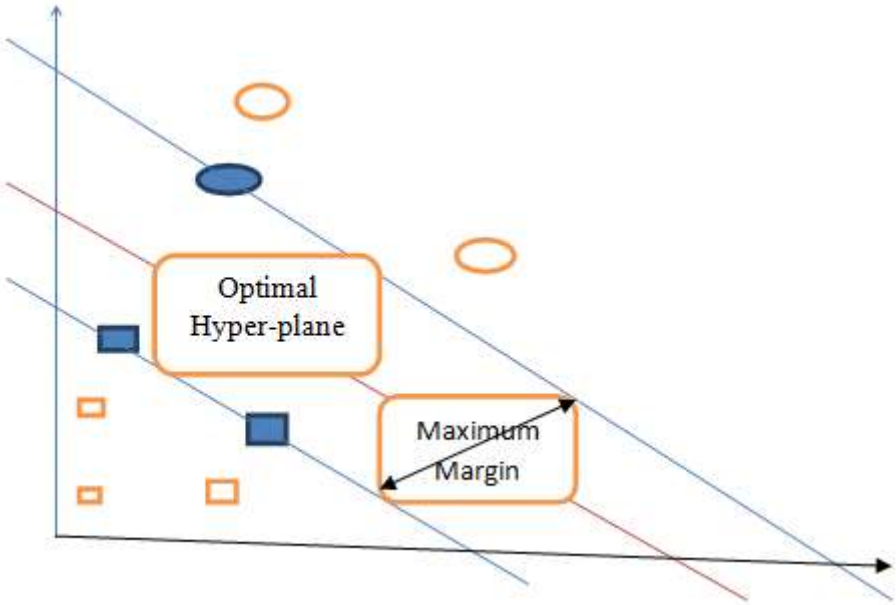


Figure 5: Support vector machine classifier

Naïve Bayes Classifier: Naïve bayes classifier is probabilistic supervised machine learning algorithms. It is use to solve classification problem. When dealing with two classes, Naive Bayes estimates a high probability for one class and a low probability for the other using Bayes' theorem [33,34]. Assume that all features/prediction in **"e"** are mutually independent, it is known as Naïve. We may distinguish between the two classes by establishing a threshold. The characteristics are assumed to be independent of one another by Naive Bayes.

Likelihood Prior

What is the probability of the evidence? Before seeing the proof,

What if our hypothesis is correct? How likely was our hypothesis?

$$P(H|e) = \frac{P(e|H)P(H)}{P(e)}$$

Posterior Marginal

Given the data, Under all potential scenarios,

How likely is our hypothesis? How likely is the new evidence?

(Not directly computable)

$$P(e) = \sum P(e|H_i)P(H_i)$$

Figure 6: Naïve bayes classifier

Random Forests Classifier: Is a set of decision tree classifiers that are run at random for a set number of times. Random forests or random decision forests are an ensemble learning method for classification, regression by making multiple decision tree using random samples from training data [35]. Given the training set processed through the classifier, each run selects the best correct results. The outcome is determined by the tree that received the most number of votes.

Copyright @ 2022 Authors

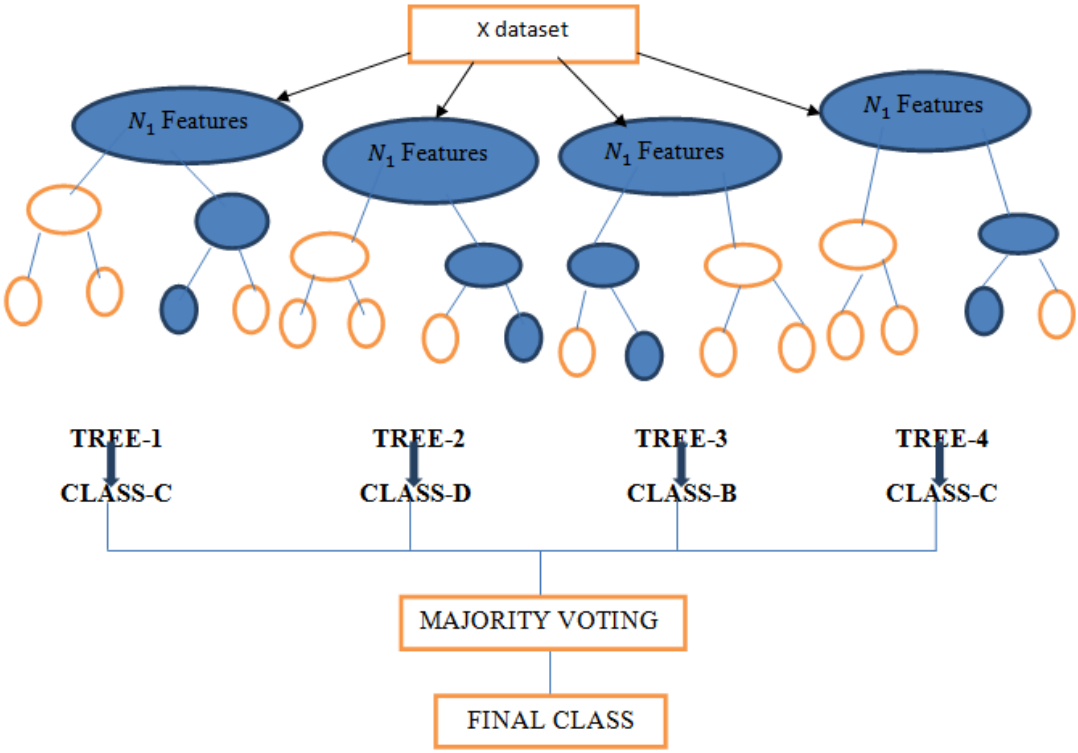


Figure 7: Random Forest classifier

K-Nearest Neighbors classifier: K Nearest Neighbor is supervised machine learning algorithms that can perform both classification and regression tasks using numbers (K) of neighbors (instances). KNN detects the K number of training observations that are closest to a test observation, counts how many observations in each class out of K, and returns an estimate of the likelihood that this test observation belongs to a given class.

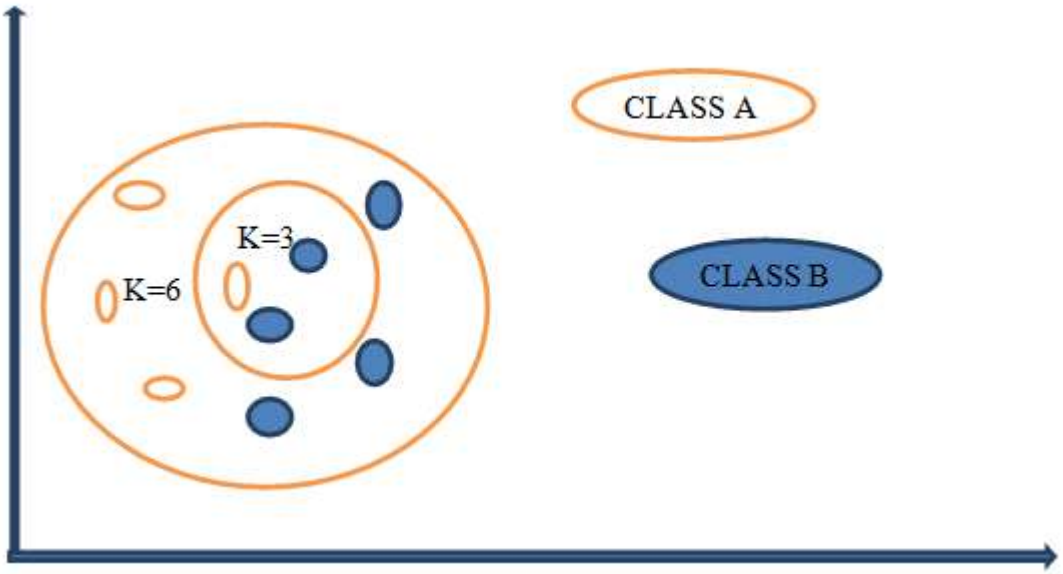


Figure 8: K-nearest Neighbors classifier

Copyright @ 2022 Authors

Ensemble Methods: Ensemble models are a group of weighted base classifiers that provide greater prediction performance than individual underlying classifiers [36]. Neural networks and random forests can be considered as an ensemble of underlying basic classifiers.

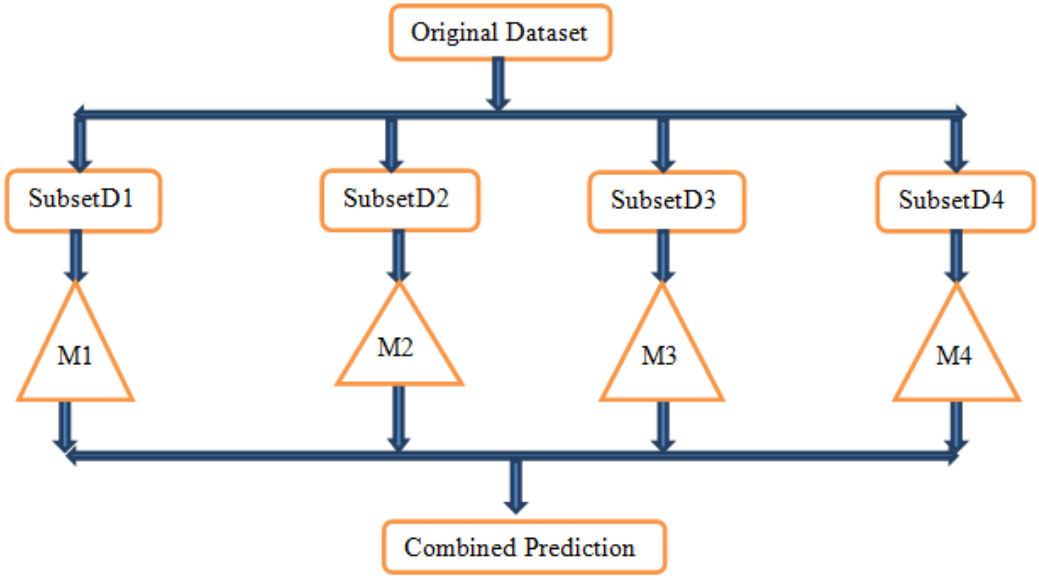


Figure 9: Ensemble classifier

V. RESULT ANALYSIS

The CICMalAnal2017 dataset is one of the few that includes real-time network traffic from both malicious and benign Android apps. The objective of this research is to use Machine Learning techniques in conjunction with the CICMalAnal2017 dataset to properly detect an application's intent from network traffic data gathered. The Canadian Institute for Cyber-security at the University of New Brunswick describes the dataset used in this study here. The dataset may be downloaded via the link at the bottom of their site's description:

https://www.unb.ca/cic/datasets/andmal2017.html

The download takes a long time because this is a large dataset (approximately 300 MB). The given 85 characteristics are utilized to obtain high accuracy in simpler machine learning methods, as reported by Arash et al. the dataset's makeup is indicated in the table below:

Categories of Traffic	***Total Count of Instances***
Benign	1,210,210
Malware	752,937

We can see the makeup more clearly if we break it down further:

Categories of Malware	***Total Count of Instances***
Adware	424,147
Ransom-ware	348,943
Scare-ware	401,165

Below is a list of both the categories of malware and the species of malware for each unique piece of malware:

Categories of Malware	***Species of Malware***	***Total Count of Instances***
Adware	DOWGIN	39,682
	EWIND	43,374

Copyright @ 2022 Authors

	FEIWO	56,632
	GOOLIGAN	93,772
	KEMOGE	38,771
	KOODOUS	32,547
	MOBIDASH	31,034
	SELFMITE	13,029
	SHAUNET	39,271
	YOUMI	36,035
Ransom-ware	CHARGER	39,551
	JISUT	25,672
	KOLER	44,555
	LOCKERPIN	25,307
	PLETOR	4,715
	PORNDROID	46,082
	RANSOMBO	39,859
	SIMPLOCKER	36,340
	SVPENG	54,161
	WANNALOCKER	32,701
Scare-ware	ANDROIDDEFENDER	56,440
	ANDROIDSPY	25,414
	AVFORANDROID	42,448
	AVPAS	40,776
	FAKEAPP	34,676
	FAKEAPPAL	44,563
	FAKEAV	40,089
	FAKEJOBOFFER	30,683
	FAKETAOBAO	33,299
	PENETHO	21,631
	VIRUSSHIELD	23,716
	Unlabeled	7,430

Performance metrics: We go through the essential definitions of the performance parameters we used to evaluate our suggested malware detection model. All of these characteristics are calculated using a confusion matrix. It is made up of real and detected categorization data derived from detection models. The malware detection model's confusion matrix may be seen in Table 4.

Table 4: Confusion matrix to classify a Android app is benign or malware (.apk)

	Benign	***Malware***
Benign	*Benign* → *Benign* (TN)	*Benign* → *Malware* (FP)
Malware	*Malware* → *Benign* (FN)	*Malware* → *Malware* (TP)

Precision: Precision is used to determine the number of positive class predictions that are actually positive class predictions.

$$Precision = \frac{N_{Malware \rightarrow Malware}}{N_{Malware \rightarrow Malware} + N_{Benign \rightarrow Malware}}$$

Recall: Recall is defined as the number of positive class predictions produced from all positive examples in a data set.

$$Recall = \frac{N_{Malware \rightarrow Malware}}{N_{Malware \rightarrow Malware} + N_{Malware \rightarrow Benign}}$$

Copyright @ 2022 Authors

Accuracy: Accuracy is calculated by dividing the number of correctly identified malware-infected apps by the total number of benign and malware-infected apps. It is available for machine learning techniques that are supervised, semi-supervised, or hybrid.

$$Accuracy = \frac{N_{Benign \rightarrow Benign} + N_{Malware \rightarrow Malware}}{N_{Classes}}$$

F1 – Measure: To develop distinct malware detection models in this work, we employed numerous machine learning approaches. As a result, comparing two models with high recall but low accuracy, or vice versa, is exceedingly difficult. As a result, the F1-measure was used to compare two different models in this investigation. F-measure is a tool that allows you to assess accuracy and recall at the same time. The F1-measure is defined by utilizing the harmonic mean rather than the arithmetic mean and punishing extreme values more heavily.

$$F1 - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

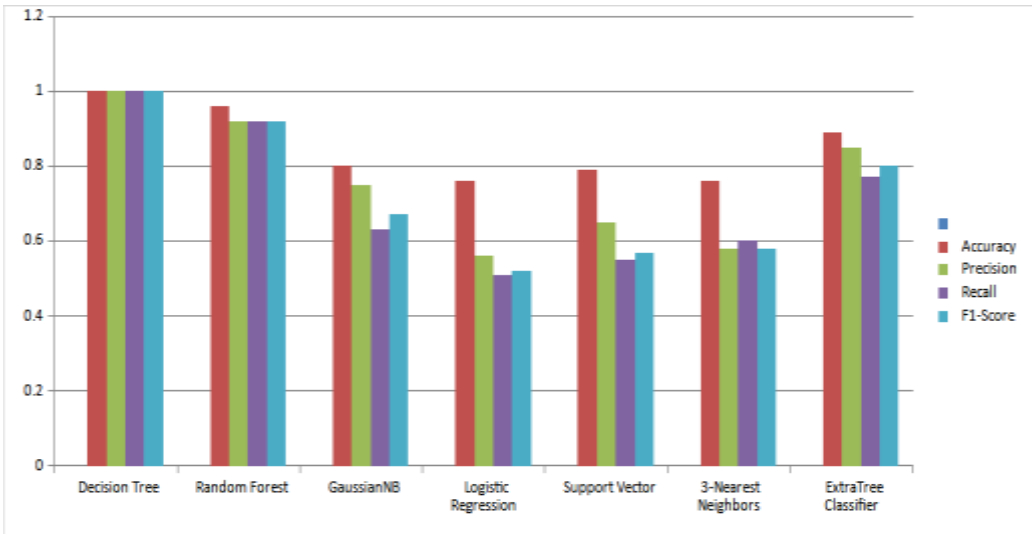
$$F1 - measure = \frac{2 * N_{Malware \rightarrow Malware}}{2 * N_{Malware \rightarrow Malware} + N_{Benign \rightarrow Malware} + N_{Malware \rightarrow Benign}}$$

The classifiers were trained using a 10-fold cross-validation approach that was performed three times in each trial. The data was then divided into two sets training set (70%) and testing set (30%). The following are the seven classifiers that were used Decision Tree (DT) classifier, Random Forest (RF) classifier, Gaussian Naïve-Bayes (GNB) classifier, Logistic Regression (LR) classifier, Support Vector Machine (SVM) classifier, K-Nearest Neighbors (KNN) classifier, Extra-Tree (Ex.-T) classifier.

Performance Analysis of the Proposed Adware Malware dataset based on ML classifiers

In this section, we analyze the performance of proposed Adware dataset with respect to **feature correlation** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 8** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 874,254 executable in all, including 424, 147 malware and 450, 107 benign ones.

Feature Correlation				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.96	0.92	0.92	0.92
GNB	0.80	0.75	0.63	0.67
LR	0.76	0.56	0.51	0.52
SVM	0.79	0.65	0.55	0.57
3-NN	0.76	0.58	0.60	0.58
Ex.-T	0.89	0.85	0.77	0.80



Copyright @ 2022 Authors

Figure 8: Performance metrics by feature correlation based on Adware Malware dataset

In this section, we analyze the performance of proposed Adware dataset with respect to **random forest importance** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 9** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 874,254 executable in all, including 424, 147 malware and 450, 107 benign ones.

Random Forest Importance				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.97	0.94	0.93	0.93
GNB	0.81	0.76	0.65	0.69
LR	0.76	0.54	0.50	0.51
SVM	0.80	0.67	0.58	0.59
3-NN	0.76	0.58	0.60	0.58
Ex.-T	0.90	0.86	0.78	0.81

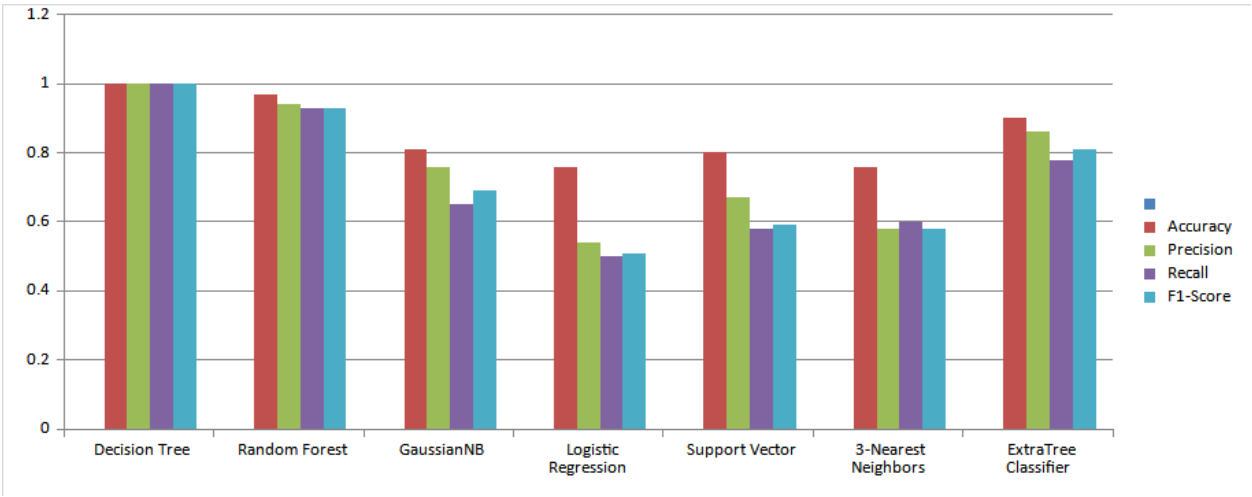


Figure 9: Performance metrics by Random forest importance based on Adware Malware dataset

In this section, we analyze the performance of proposed Adware dataset with respect to chi-square test feature technique existing machine learning classifiers using Anaconda supports python 3.8. In Figure 10 illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 874,254 executable in all, including 424, 147 malware and 450, 107 benign ones.

Chi-square test				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.96	0.92	0.92	0.92
GNB	0.80	0.75	0.63	0.67
LR	0.76	0.56	0.51	0.52

Copyright @ 2022 Authors

SVM	0.80	0.67	0.57	0.58
3-NN	0.76	0.58	0.60	0.58
Ex.-T	0.89	0.84	0.77	0.80

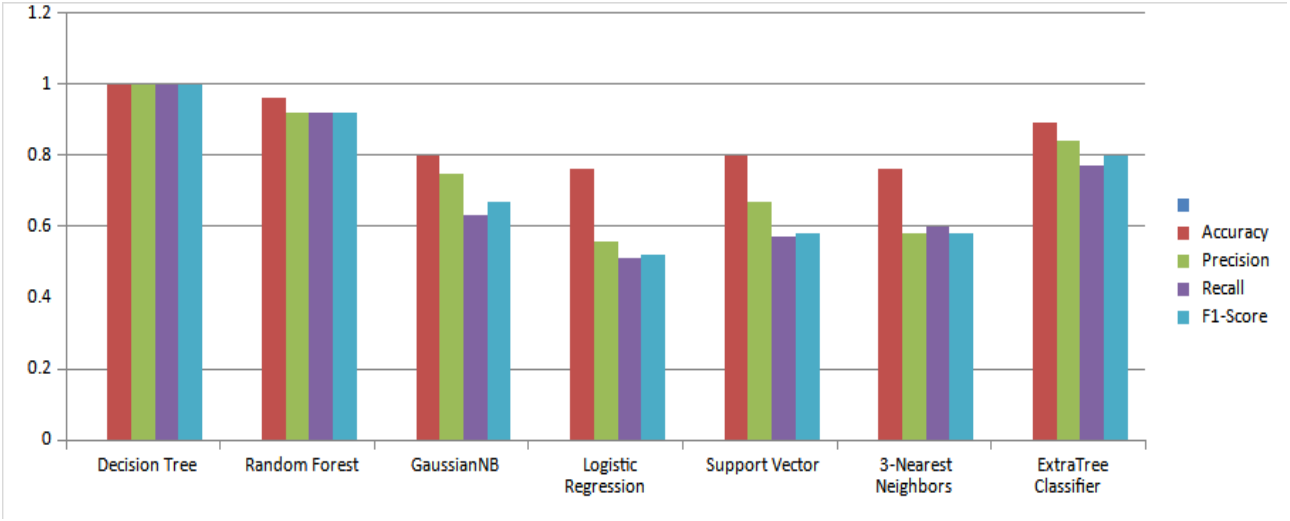
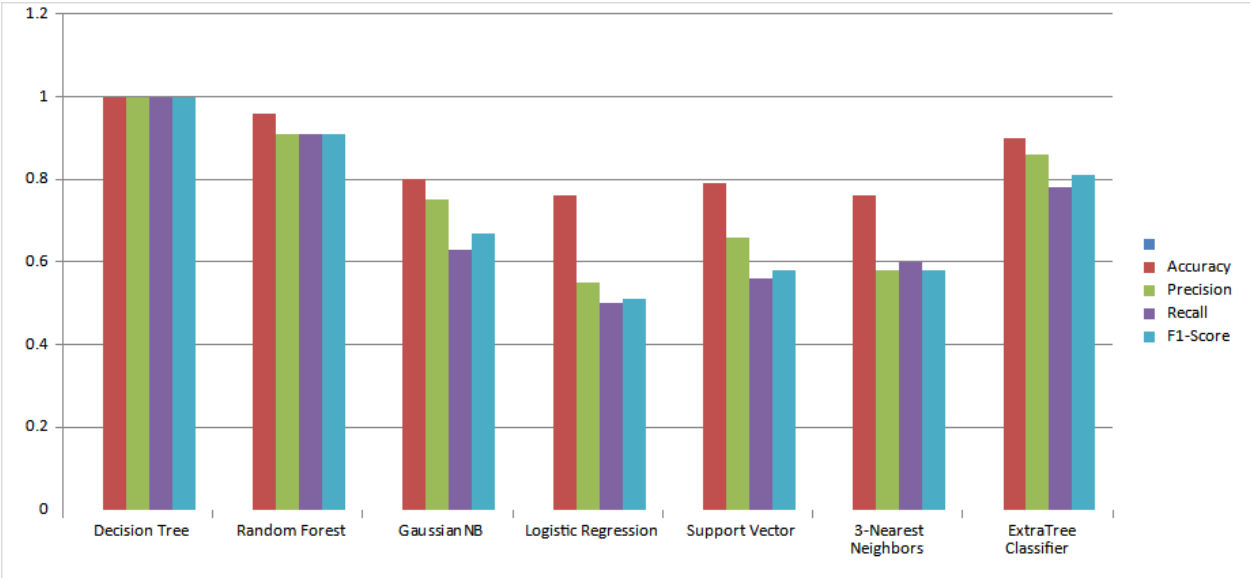


Figure 10: Performance metrics by Chi-square test based on Adware Malware dataset

In this section, we analyze the performance of proposed Adware dataset with respect to **information gain** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 11** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 874,254 executable in all, including 424, 147 malware and 450, 107 benign ones.

Information Gain				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.96	0.91	0.91	0.91
GNB	0.80	0.75	0.63	0.67
LR	0.76	0.55	0.50	0.51
SVM	0.79	0.66	0.56	0.58
3-NN	0.76	0.58	0.60	0.58
Ex.-T	0.90	0.86	0.78	0.81



Copyright @ 2022 Authors

Figure 11: Performance metrics by Information Gain based on Adware Malware dataset

Performance Analysis of the Proposed Ransom-ware Malware dataset based on ML classifiers

In this section, we analyze the performance of proposed Ransom-ware dataset with respect to **feature correlation** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 12** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 700, 080executable in all, including 348, 943 malware and 351, 137 benign ones.

Feature Correlation				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.98	0.97	0.96	0.96
GNB	0.88	0.82	0.79	0.79
LR	0.72	0.56	0.47	0.47
SVM	0.74	0.49	0.46	0.45
3-NN	0.72	0.53	0.60	0.56
Ex.-T	0.90	0.88	0.81	0.84

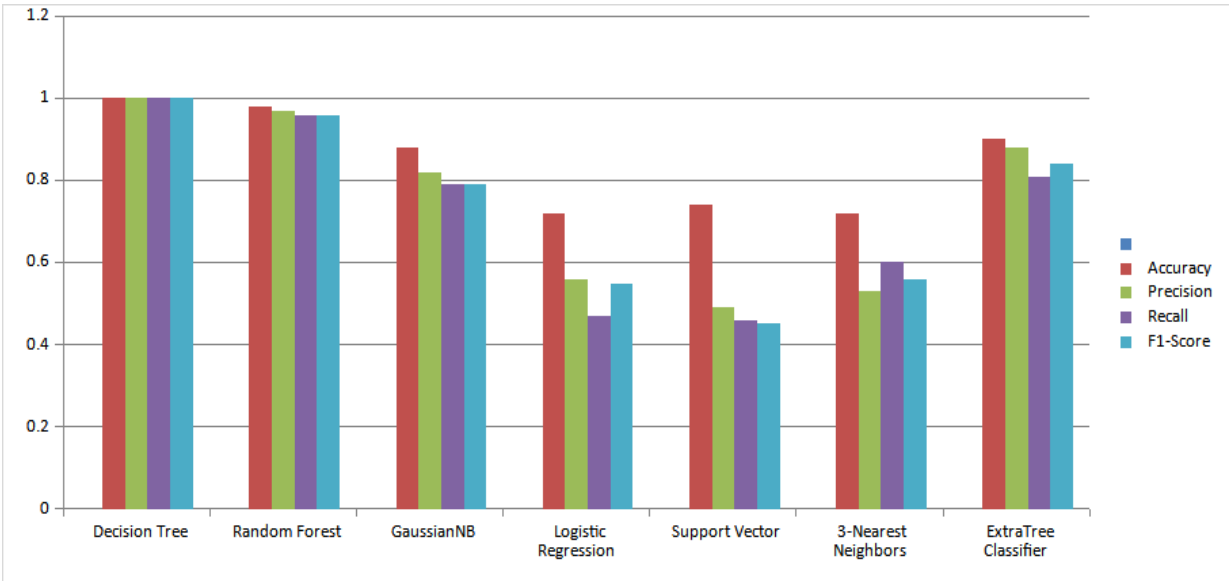


Figure 12: Performance metrics by Feature correlation based on Ransom-ware Malware dataset

In this section, we analyze the performance of proposed Ransom-ware dataset with respect to **random forest importance** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 13** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 700, 080executable in all, including 348, 943 malware and 351, 137 benign ones.

Random forest importance				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.99	0.98	0.97	0.97
GNB	0.89	0.83	0.81	0.81
LR	0.73	0.54	0.46	0.46
SVM	0.75	0.52	0.49	0.48
3-NN	0.72	0.53	0.60	0.56
Ex.-T	0.90	0.89	0.82	0.86

Copyright @ 2022 Authors

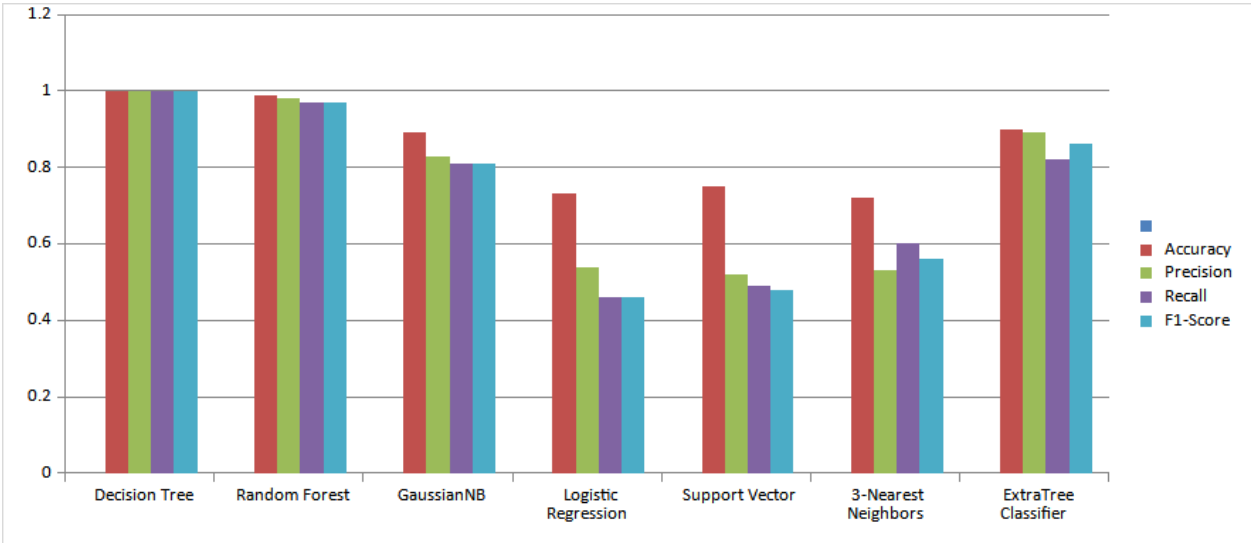
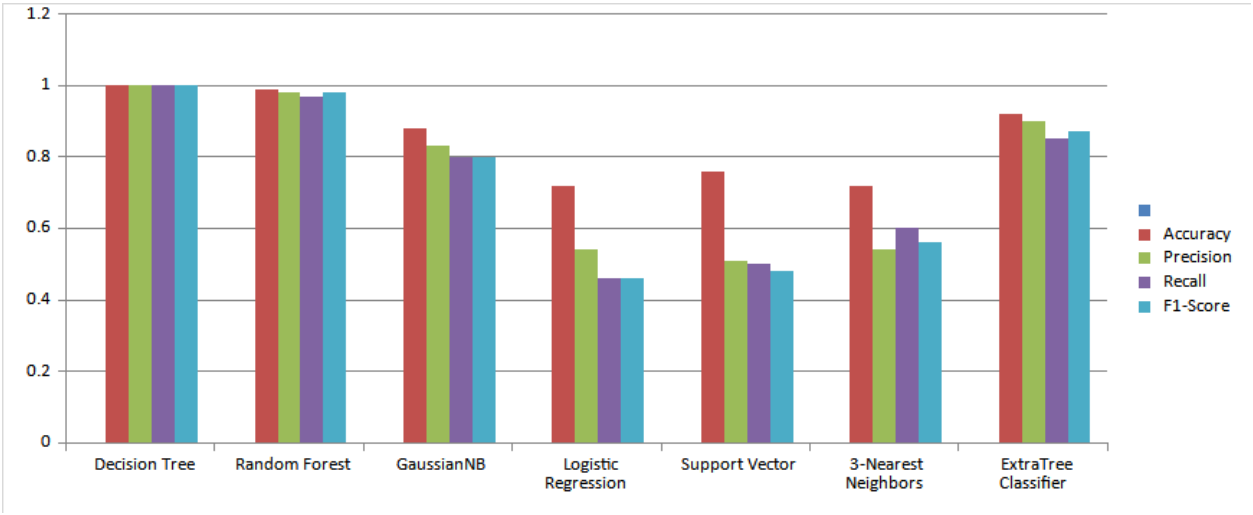


Figure 13: Performance metrics by Random forest importance based on Ransom-ware Malware dataset

In this section, we analyze the performance of proposed Ransom-ware dataset with respect to **chi-square test** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 14** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 700, 080executable in all, including 348, 943 malware and 351, 137 benign ones.

Chi-square test				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.99	0.98	0.97	0.98
GNB	0.88	0.83	0.80	0.80
LR	0.72	0.54	0.46	0.46
SVM	0.76	0.51	0.50	0.48
3-NN	0.72	0.54	0.60	0.56
Ex.-T	0.92	0.90	0.85	0.87



Copyright @ 2022 Authors

Figure 14: Performance metrics by Chi-square test based on Ransom-ware Malware dataset

In this section, we analyze the performance of proposed Ransom-ware dataset with respect to **information gain** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 15** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 700, 080 executable in all, including 348, 943 malware and 351, 137 benign ones.

Information gain				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.99	0.97	0.97	0.97
GNB	0.90	0.84	0.82	0.82
LR	0.72	0.55	0.47	0.47
SVM	0.75	0.51	0.48	0.47
3-NN	0.72	0.53	0.60	0.56
Ex.-T	0.89	0.87	0.81	0.84

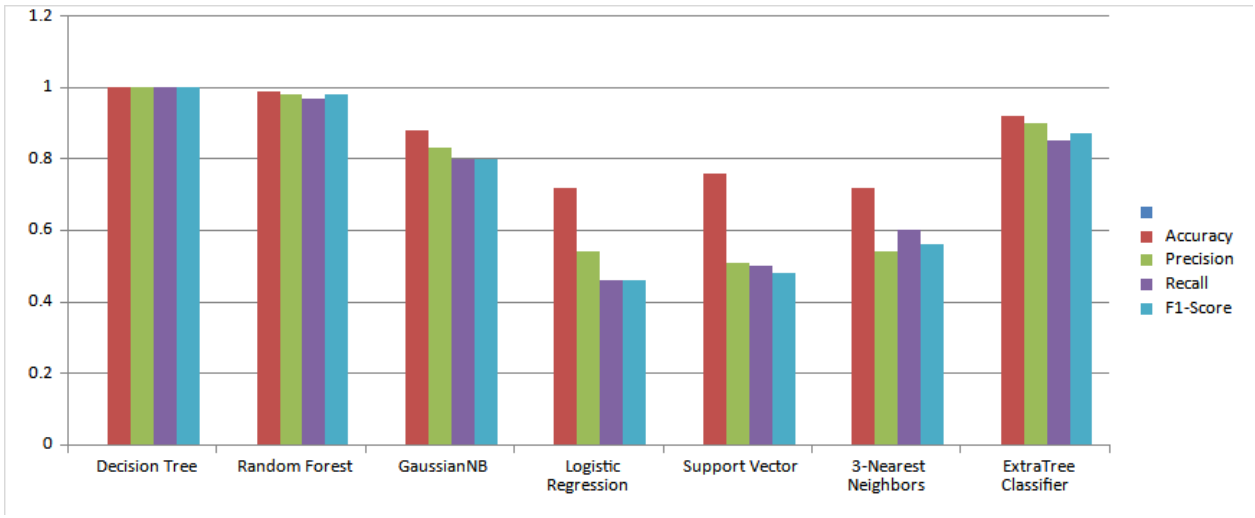


Figure 15: Performance metrics by Information gain based on Ransom-ware Malware dataset

Performance Analysis of the Proposed Scare-ware Malware dataset based on ML classifiers

In this section, we analyze the performance of proposed Scare-ware dataset with respect to **feature correlation** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 16** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 824, 302executable in all, including 401, 165 malware and 423, 137 benign ones.

Feature correlation				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.96	0.97	0.93	0.95
GNB	0.83	0.81	0.74	0.76
LR	0.62	0.36	0.33	0.31
SVM	0.63	0.39	0.34	0.30
3-NN	0.64	0.42	0.54	0.46

Copyright @ 2022 Authors

Ex.-T	0.87	0.91	0.79	0.84

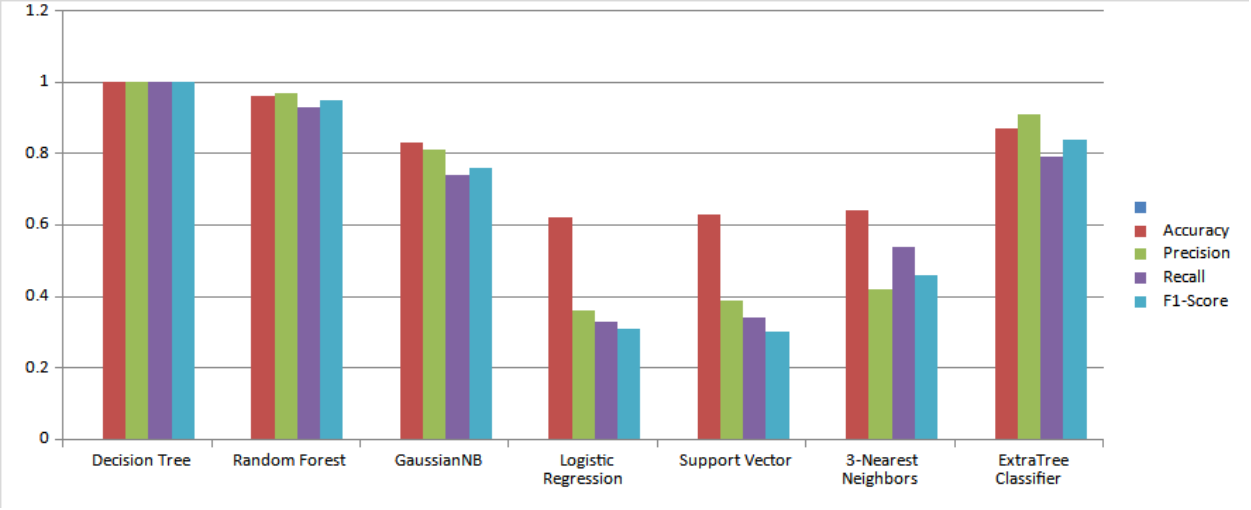


Figure 16: Performance metrics by Feature Correlation based on Scare-ware Malware dataset

In this section, we analyze the performance of proposed Scare-ware dataset with respect to **random forest importance** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 17** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 824, 302executable in all, including 401, 165 malware and 423, 137 benign ones.

Random forest importance				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.96	0.97	0.93	0.95
GNB	0.83	0.81	0.74	0.76
LR	0.62	0.35	0.32	0.31
SVM	0.64	0.40	0.35	0.31
3-NN	0.64	0.42	0.54	0.46
Ex.-T	0.89	0.93	0.82	0.86

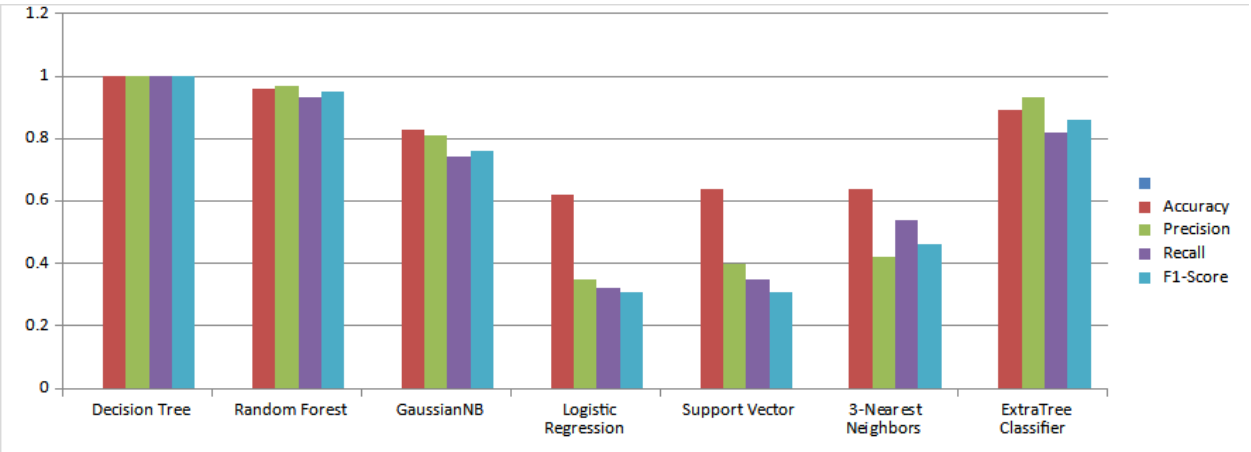


Figure 17: Performance metrics by Random forest importance based on Scare-ware Malware dataset

In this section, we analyze the performance of proposed Scare-ware dataset with respect to **chi-square test** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 18** illustrate performance of existing

Copyright @ 2022 Authors

classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 824, 302executable in all, including 401, 165 malware and 423, 137 benign ones.

Chi-square test				
Model	Accuracy	Precision	Recall	F1-Score
DT	1	1	1	1
RF	0.97	0.98	0.95	0.97
GNB	0.83	0.82	0.74	0.77
LR	0.62	0.36	0.32	0.31
SVM	0.64	0.42	0.35	0.32
3-NN	0.65	0.45	0.56	0.48
Ex.-T	0.88	0.92	0.79	0.84

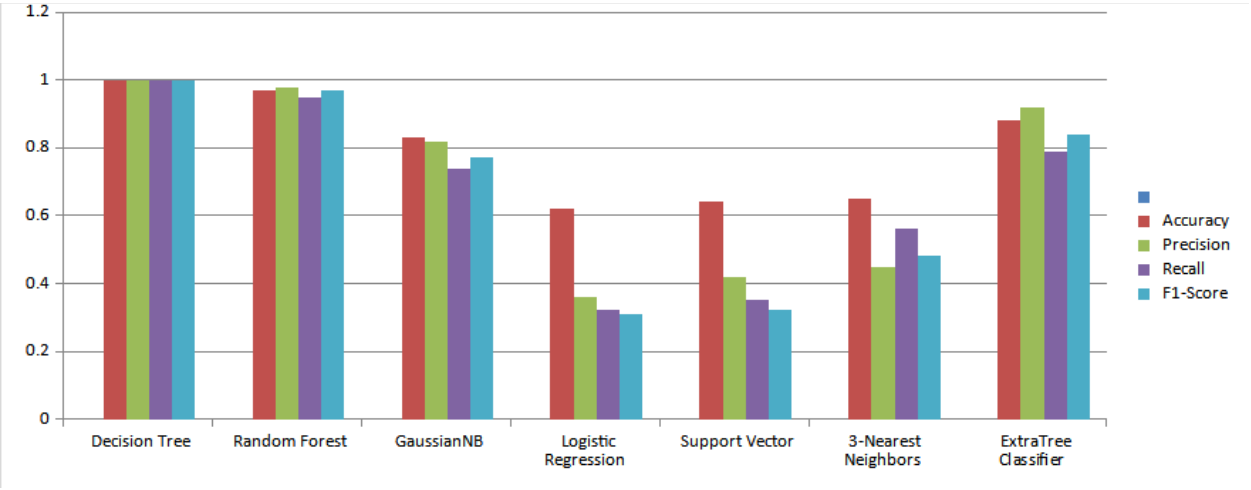


Figure 18: Performance metrics by Chi-square test based on Scare-ware Malware dataset

In this section, we analyze the performance of proposed Scare-ware dataset with respect to **information gain** feature technique existing machine learning classifiers using Anaconda supports python 3.8. In **Figure 19** illustrate performance of existing classifiers such as DT, RF, GNB, LR, SVM, KNN and Ex.-T classifier. In this experiment, There are 824, 302executable in all, including 401, 165 malware and 423, 137 benign ones.

Information gain				
Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
DT	1	1	1	1
RF	0.95	0.97	0.92	0.94
GNB	0.83	0.81	0.74	0.76
LR	0.62	0.36	0.33	0.31
SVM	0.63	0.39	0.34	0.30
3-NN	0.64	0.42	0.54	0.48
Ex.-T	0.87	0.91	0.79	0.84

Copyright @ 2022 Authors

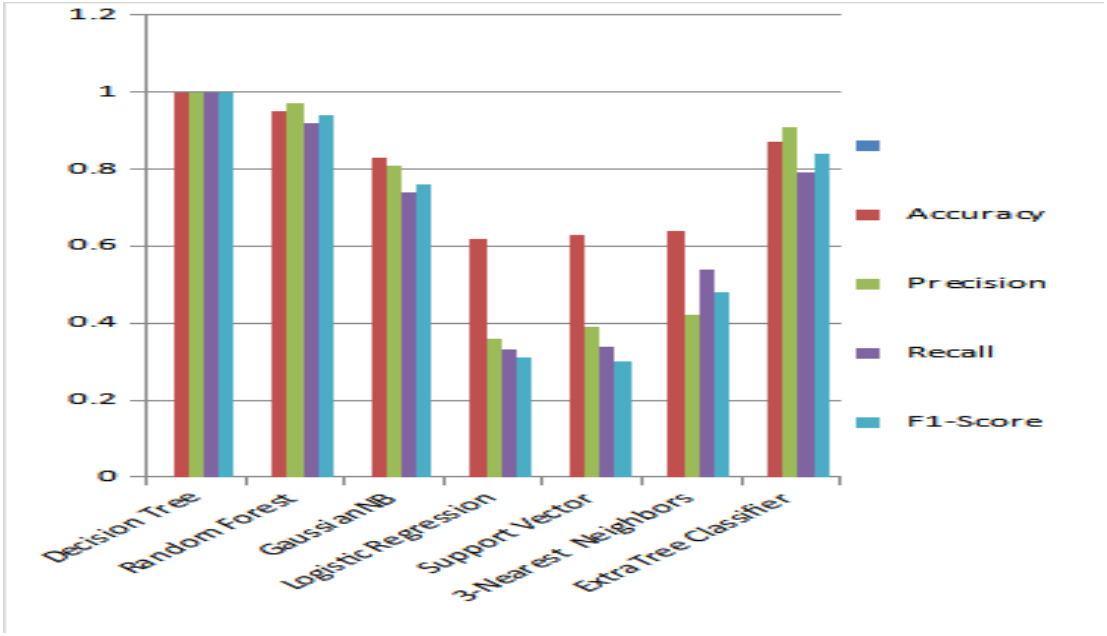


Figure 19: Performance metrics by Information gain based on Scare-ware Malware dataset

VI. CONCLUSIONS

The purpose of this dissertation is to examine relevant works in malware detection using machine learning, re-evaluate a few of them, and reach a judgment on the accuracy promises made by their authors. The survey found that the authors' claimed results were influenced by several internal factors under their control, which may or may not have affected the accuracy of the results. Some of these variables have to do with sample size, feature count, or feature extraction technique. As a result, given the underlying elements and assumptions, there was no conclusive conclusion that the performance of one classifier was superior to the others. This dissertation uses the same dataset to re-evaluate and compare the techniques of other researchers. The following is the data: 1) large enough to cover all observations, 2) diverse enough to encompass the majority of well-known malware kinds, and 3) balanced enough to avoid skewing the results in one direction or the other. As a result, all subjectivity toward the produced conclusions is eliminated. To do this, two studies were re-evaluated using a standardized dataset and a publicly available open-source feature extraction technique. The findings of the re-evaluation indicated that a larger dataset produced better results, but they also revealed inefficiencies in certain classifiers' handling of feature sets that are either too sparse or linearly disconnected, causing some to underperform. In this paper, we looked at how machine learning techniques fit into the context of malware detection, as well as how the dataset used could affect the classifier's results. We've proven that temporal consistent techniques can be used to categorize malware samples with only a minor loss in accuracy. We also concluded that we can minimize the size of the training dataset to avoid having to train with all previously seen samples, and we debated how much smaller it can be without sacrificing ideal outcomes. We modified our base model for better results after learning about the effects of temporal consistency in the task of malware identification.

We feel that the critical topic of how much should we aspire for spectacular results on ML approaches used to identify malware requires further debate, particularly when it leads to classifiers that do not perform better in practical scenarios. The goal we set for ourselves was not without its difficulties, but we believe our work shows that machine learning can identify malware not just conceptually, but also practically

Copyright @ 2022 Authors

REFERENCES

[1] Qaisar, Z.H.; Li, R. Multimodal information fusion for android malware detection using lazy learning. Multimed. Tools Appl. 2021, 1–15.
[2] Mahindru, A.; Sangal, A. MLDroid—Framework for Android malware detection using machine learning techniques. Neural Comput. Appl. 2021, 33, 5183–5240.
[3] Xu, K.; Li, Y.; Deng, R.H.; Chen, K. Deeprefiner: Multi-layer android malware detection system applying deep neural networks. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018; pp. 473–487.
[4] JADX. Available online: https://github.com/skylot/jadx/ (accessed on 19 May 2021).
[5] McLaughlin, N.; Martinez del Rincon, J.; Kang, B.; Yerima, S.; Miller, P.; Sezer, S.; Safaei, Y.; Trickel, E.; Zhao, Z.; Doupé, A.; et al. Deep android malware detection. In Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, Scottsdale, AZ, USA, 22–24 March 2017; pp. 301–308.
[6] Amin, M.; Tanveer, T.A.; Tehseen, M.; Khan, M.; Khan, F.A.; Anwar, S. Static malware detection and attribution in android byte-code through an end-to-end deep system. Future Gener. Comput. Syst. 2020, 102, 112–126.
[7] Alzaylaee, M.K.; Yerima, S.Y.; Sezer, S. DL-Droid: Deep learning based android malware detection using real devices. Comput. Secur. 2020, 89, 101663.
[8] Vu, L.N.; Jung, S. AdMat: A CNN-on-Matrix Approach to Android Malware Detection and Classification. IEEE Access 2021, 9, 39680–39694.
[9] Millar, S.; McLaughlin, N.; del Rincon, J.M.; Miller, P. Multi-view deep learning for zero-day Android malware detection. J. Inf. Secur. Appl. 2021, 58, 102718.
[10] Acar, Y.; Stransky, C.; Wermke, D.; Weir, C.; Mazurek, M.L.; Fahl, S. Developers need support, too: A survey of security advice for software developers. In Proceedings of the 2017 IEEE Cybersecurity Development (SecDev), Cambridge, MA, USA, 24–26 September 2017; pp. 22–26.
[11] Mohammed, N.M.; Niazi, M.; Alshayeb, M.; Mahmood, S. Exploring software security approaches in software development lifecycle: A systematic mapping study. Comput. Stand. Interfaces 2017, 50, 107–115.
[12] Weir, C.; Becker, I.; Noble, J.; Blair, L.; Sasse, M.A.; Rashid, A. Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers. Softw. Pract. Exp. 2020, 50, 275–298.
[13] Alenezi, M.; Almomani, I. Empirical analysis of static code metrics for predicting risk scores in android applications. In Proceedings of the 5th International Symposium on Data Mining Applications, Cham, Switzerland, 29 March 2018; Springer: Cham, Switzerland, 2018; pp. 84–94.
[14] Palomba, F.; Di Nucci, D.; Panichella, A.; Zaidman, A.; De Lucia, A. Lightweight detection of android-specific code smells: The adoctor project. In Proceedings of the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, Austria, 20–24 February 2017; pp. 487–491.
[15] Pustogarov, I.; Wu, Q.; Lie, D. Ex-vivo dynamic analysis framework for Android device drivers. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 1088–1105.
[16] Amin, A.; Eldessouki, A.; Magdy, M.T.; Abdeen, N.; Hindy, H.; Hegazy, I. AndroShield: Automated android applications vulnerability detection, a hybrid static and dynamic analysis approach. Information 2019, 10, 326.
[17] Tahaei, M.; Vaniea, K.; Beznosov, K.; Wolters, M.K. Security Notifications in Static Analysis Tools: Developers' Attitudes, Comprehension, and Ability to Act on Them. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; pp. 1–17.
[18] Goaër, O.L. Enforcing green code with Android lint. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering Workshops, Melbourne, VIC, Australia, 21–25 September 2020; pp. 85–90.
[19] Habchi, S.; Blanc, X.; Rouvoy, R. On adopting linters to deal with performance concerns in android apps. In Proceedings of the 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE), Montpellier, France, 3–7 September 2018; pp. 6–16.
[20] Wei, L.; Liu, Y.; Cheung, S.C. OASIS: Prioritizing static analysis warnings for Android apps based on app user reviews. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, Paderborn, Germany, 4–8 September 2017; pp. 672–682.
[21] Luo, L.; Dolby, J.; Bodden, E. MagpieBridge: A General Approach to Integrating Static Analyses into IDEs and Editors (Tool Insights Paper). In Proceedings of the 33rd European Conference on Object-Oriented Programming (ECOOP 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 15–19 July 2019.
[22] Wang, Y.; Xu, G.; Liu, X.; Mao, W.; Si, C.; Pedrycz, W.; Wang, W. Identifying vulnerabilities of SSL/TLS certificate verification in Android apps with static and dynamic analysis. J. Syst. Softw. 2020, 167, 110609.
[23] Gupta, A.; Suri, B.; Kumar, V.; Jain, P. Extracting rules for vulnerabilities detection with static metrics using machine learning. Int. J. Syst. Assur. Eng. Manag. 2021, 12, 65–76.
[24] Norouzi, Monire, AlirezaSouri, and MajidSamadZamini. "A data mining classification approach for behavioral malware detection." Journal of Computer Networks and Communications 2016 (2016).

Copyright @ 2022 Authors

[25] Cimitile, Aniello, et al. "Talos: no more ransomware victims with formal methods." International Journal of Information Security 2018.

[26] Malhotra, Aashima, and Karan Bajaj. "A hybrid pattern based text mining approach for malware detection using DBScan." CSI transactions on ICT 2016.

[27] Park, Younghee, Douglas S. Reeves, and Mark Stamp. "Deriving common malware behavior through graph clustering." computers & security 39 (2013): 419-430.

[28] Singh, Prabhat K., and ArunLakhotia. "Static verification of worm and virus behavior in binary executables using model checking." IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop, 2003..IEEE, 2003.

[29] Rahman, Sheikh Shah Mohammad Motiur, and Sanjit Kumar Saha. "StackDroid: evaluation of a multi-level approach for detecting the malware on android using stacked generalization." International Conference on Recent Trends in Image Processing and Pattern Recognition.Springer, Singapore, 2018.

[30] Dixon, Bryan, and Shivakant Mishra. "Power based malicious code detection techniques for smartphones." 2013 12th IEEE international conference on trust, security and privacy in computing and communications.IEEE, 2013.

[31] Suarez-Tangil, Guillermo, et al. "Power-aware anomaly detection in smartphones: An analysis of on-platform versus externalized operation." Pervasive and Mobile Computing 2015.

[32] Chen, Patrick Shicheng, Shu-Chiung Lin, and Chien-Hsing Sun. "Simple and effective method for detecting abnormal internet behaviors of mobile devices." Information Sciences 2015.

[33] Baldangombo, Usukhbayar, NyamjavJambaljav, and Shi-Jinn Horng. "A static malware detection system using data mining methods." arXiv preprint arXiv2013.

[34] Ng, Deniel V., and Jen-Ing G. Hwang. "Android malware detection using the dendritic cell algorithm." 2014 International conference on machine learning and cybernetics.Vol. 1.IEEE, 2014.

[35] Sheen, Shina, R. Anitha, and V. Natarajan. "Android based malware detection using a multifeature collaborative decision fusion approach." Neurocomputing 2015.

[36] Tong, Fei, and Zheng Yan. "A hybrid approach of mobile malware detection in Android." Journal of Parallel and Distributed computing 2017.

.

Copyright @ 2022 Authors