InfoTech Spectrum: Iraqi Journal of Data Science (IJDS)
ISSN: 3007-5467 , Vol 1(No.1), 1-15 (2024)
DOI: https://doi.org/10.51173/ijds.v1i1.4
https://ijds.mtu.edu.iq

ENHANCING MALWARE DETECTION THROUGH MACHINE LEARNING TECHNIQUES

Zeina S. Jassim [1*] , Mohamad M.Kassir [2]

[1,2] *Department of Computer Engineering, University of Qom, Qom, Iran.*

**Corresponding author E-mail: z.jassem@stu.qom.ac.ir (Zeina S. Jassim)*

RESEARCH ARTICLE

ARTICLE INFORMATION

SUBMISSION HISTORY:

Received: 15 April 2024

Revised: 25 May 2024

Accepted: 10 June 2024

Published: 30 June 2024

KEYWORDS:

Anomaly Detection;
Decision Tree; ID3;
Malware Detection;
Machine Learning

ABSTRACT

Malware detection is important to computer network security since it is the principal attack vector against modern enterprises. As a result, firms must remove viruses from computer systems. Using artificial intelligence, namely machine learning techniques, to function in real-time with an IT system is the ideal solution to this problem. This issue has yet to be fixed, but it is still significant because a lack of processing power and memory constrains these features. The most popular method for evaluating systems and intrusion detection models is using the Application Program Interface (API) calls via the KDD-CUP99 data set to give this solution. KDD-CUP99 has more than three hundred thousand samples, each with 54 features. However, the data set attributes were designed and chosen to provide us with a high malware detection rate. The quality of this data was lowered to produce results. To get the desired results, the attributes of this data were reduced. Data transformation and purification are used in this process. Inaccurate, unnecessary, duplicated, or missing information is eliminated by data cleansing. Data cleaning eliminates inaccurate, excessive, redundant, or lacking information. By comparing this study to earlier research that employed lengthy sequences of software interface (API) calls with the same machine-learning classifiers, data transformation includes discretization, which transforms the continuous process of discretizing continuous data into discrete forms is a type of data transformation. Using more advanced algorithms to do the task at hand with the best precision and the least expense increases accuracy and performance. The data set was divided into two categories using a Support Vector Machine (SVM), Decision Tree (DT), and Iterative Dichotomiser 3 (ID3). The findings revealed that little previous research uses a five-class classification strategy for malware detection. The accuracy of several works is comparable to the accuracy acquired in the proposed work.

1. INTRODUCTION

Financial motivations, political or military ambitions, and personal motives have increased computer network attacks. This has raised concerns about network security because firewalls, antivirus programs, and various other threats and malware detection tools cannot guarantee security. IDSs largely detect internal and external attackers' illicit usage and abuse. These systems are classifiers that consider the information source and the ID technique that was applied, such as anomaly or signature-based detection. Malware is a serious hazard. Thus, detecting it is crucial to safeguarding networks and computer systems. [1].

Using machine learning might be a method to enhance malware identification. Novelties may be found in massive data sets by computer systems trained in the data. Some oddities might indicate the presence of malicious software. Furthermore, these systems may adapt their operation

DOI: https://doi.org/10.51173/ijds.v1i1.4

immediately once they detect new malicious software types for several reasons, and it is critical to learn how to identify malicious software more effectively with these learning tools.

First, it increases the accuracy of identifying malicious software. Learning systems may analyze large data sets to uncover complex patterns that may have escaped the notice of traditional approaches. In light of this, taking this action may improve the efficiency of identifying malicious software and reduce errors. [1]. Second, research in this field allows for a quicker detection of malware. Machine learning algorithms can detect malware nearly instantaneously. Networks and computer systems can suffer significantly from malware, but these consequences can be mitigated with our quicker detection method [2]. Third, studies in this area provide hope for improved malware detection capabilities. Machine learning algorithms can modify strategies and tactics and integrate these modifications into their models in response to novel virus types [2]. Many studies have concentrated on malware detection systems based on machine learning. It has been suggested that feature selection techniques, especially evolutionary algorithms, can reduce expenses while increasing performance. However, these strategies have some drawbacks: neither their features nor overall comparability to alternative techniques have been carefully studied [3].

Information security malware detection is a top priority and will do so in the future due to the critical nature of the modern digital environment. This study aims to clarify the issue by concentrating on the detection and classification of malware. Additionally, the study's main objective is to improve the accuracy and efficiency of the proposed algorithm by correcting its shortcomings. The study aims to determine how effectively these algorithms deliver dependable results. There is no requirement to know the characteristics of beginning values before simulating using the provided technique. Instead, it divides the attributes at random to generate 20 beginning populations. The software selects the beginning population with the highest generation value. The most significant features were selected for this generation and used as a basis. This methodology is used in the study to demonstrate the utility of the Decision Tree (DT) algorithm for feature selection, the SVM algorithm for classification, and the superiority of ID3 due to its higher accuracy with the ultimate goal of increasing confidence in the acquired findings. SVM machines, ID3 methods, and DT trees with data from the KDD database employ optimal malware detection settings. After that, the data were divided into training and test sets, which were used to train and test the model. According to the test, the ID3 program detected malware with 97% accuracy. It performed better than others. SVM's highest accuracy rate was 88%, while DT trees got 96%. Python tools for creating machine-learning applications were delivered from Anaconda.

2. BACKGROUND and LITERATURE REVIEW

In 2013, Peiravian and Zhu discovered a method for identifying phoney Android applications[3] by looking at how the apps used permission requests and API calls to see if they were active. Different techniques were used to evaluate this part of the app's behavior, including support vector machine models, decision trees, and bagging.

In 2018, Rathore, Agarwal et al. [4] concentrated on finding malware trends and used techniques like cross-validation to address the issue of unbalanced data classes. Several feature reduction strategies were investigated to counter the complexity brought on by dimensionality. In particular, deep neural networks (DNN-2L, DNN-4L, and DNN-7L) and Random Forest (RF) were compared. Data on malware detection showed that the random forest model performed better than the three deep neural network models. The accuracy achieved with a variance threshold and the random forest was 99.78%, 1.26 per cent greater than the previous best (which was achieved with random forests with fine-tuning). Xu, Z., Ray, S., Subramanian, et al. 2017 [5] showed a detailed method for detecting viruses that rely on machine learning assistance for online detection of memory access patterns. Examinations were carried out to tackle weaknesses in the kernel and user interfaces. The results showed that a kernel-level rootkit detection rate of 100% with less than 1% of false positives is possible. The results showed kernel-level rootkit identification can detect them with a 100% detection rate and a less than 1% false positive rate. The method produced fewer

than(50%) False Positives and a (99.0%) Detection Rate of attacks. Machine learning was one of the most significant advantages of not using human judgment to detect malware signatures.

In 2019, Sandeep employed an SNN architecture to detect malware during app installation [6]. Features were chosen using a random forest classifier, which performs better when dealing with categorical data. A classifier label was included in the dataset, which had 331 features in total. The bulk of the features were binary or categorical tags. Feng and Lin [7] 2020. Presented DL model training and feature extraction from resources collected from external sources, including application stores. Manifest characteristics, API calls, and opcode sequences fed a DNN into MobiDroid. This work built classifiers for malicious and benign Android applications using CNNs trained on seven primary feature categories. The accuracy, sensitivity, and recall values of all three characteristics were determined to be 96.87. With an accuracy rate of 88% when utilizing API calls, the MobiDroid technique surpassed the SVM method. Nonetheless, the latter attained a rate of 92%.

In a recent work by Lee et al. in 2021 [8], an experiment was conducted using a dataset of two classes and 122 characteristics. The classifier's performance by varying the number of nodes in the hidden layer could be compared, consisting of 52. A methodology for classifying malware based on PE headers was described in a report by Rkhouya and Chougdali [9] in 2021. Four machine-learning techniques were used for assessment, and the training dataset included around 130,000 files. Accuracy, execution speed, and UAC were used to evaluate these algorithms' effectiveness. Random Forest fared better than the other three techniques, obtaining an area under the ROC curve of 0.9933 and an accuracy higher than 99.64%.

In 2022 [10], Hussain et al. used six distinct machine-learning methods to detect malware in Windows executables. The experimental results included comparisons of the Accuracy, F1-score, recall, precision, and support of the results. Random Forest outperformed the other algorithms with an accuracy of (99.4%). The authors also described how the concept would be applied to the Flask framework in the future, enabling minimal processing resource consumption and real-time executable file scanning in the Windows environment. In 2022, Yateem, Yassen, and Shatnawi [11], according to the study findings, SVM performed better in terms of rates than KNN, NB, and other comparable classifiers, with API call features, the accuracy was 83%, and with authorization features, it was 94%. In addition to helping with research and defensive measures against expanding mobile information access, this approach aimed to increase virus detection rates.

Table 1: The majority of relevant works are summarized.

Reference Number.	The Year	Technique	Collection of data	Attributes	ACC
[3]	2013	SVM	2510 APK	130	96.4%
[4]	2018	RF&DNN	11, 688 Malicia Project 4. In Malicia	102	99.78%
[5]	2017	SVM&RF& GA	850 RIPE attacks	130	94.3%
[6]	2019	RF	Virus link in the Google Play Store	331	94.2%
[7]	2020	KNN NVG-RA SVM MLP	Pwnzen Infotech Inc., VirusShare, Drebin, Genom, Google Play Store, and Contagio Mobile Website.	100	87% 90% 60% 81%
(8)	2021	RF+DT +NB MLP+GA	SMS SPAM	100	98%
(9)	2021	SVM+DT and RF.	WIN Application	54	94.6%
[10]	2022	RF& SVM& DT& GNB	An online dataset repository	54	99.4% 98.5% 99% 98.9%
(11)	2022	SVM& KNN& Naive Bayes	CICI-AndMal-2019 Dataset.	14	94.36% 93.42% 84.33%
[12]	2023	RF SVM KNN	MalMernAnalasis &2022	710	99% 85% 98%

The proposed RF, SVM, and KNN (k-nearest-neighbors) algorithms were compared to the previously published works of Abdulwahed in terms of precision, accuracy, recall, and F-measure. In 2023 [12], Abdulwahed. Sh. Ban and Ali Al-Naji, in every instance, were able to extract the characteristics more precisely since the suggested model could include missing data without compromising accuracy. Table 1 provides a summary of relevant works.

3. METHOD

Security technologies such as intrusion detection systems (IDS) ensure the safety of computer networks and systems by continuously monitoring and identifying any unusual activity. With the expansion and complexity of modern networks, the frequency of cyberattacks has increased correspondingly. An Intrusion Detection System (IDS) is necessary to identify and alert the appropriate parties in case of a potential or significant security breach, such as unauthorized access, malware infection, etc., to improve and maintain network security. Although there are two primary kinds of intrusion detection systems—passive and aggressive—anomaly recognition is still necessary for identifying novel forms of assault. An intrusion detection system (IDS) actively monitoring the network can prevent or lessen the impact of an attack by, for example, shutting off a network connection.

3.1 Related Works

Much in-depth research on sophisticated malware detection has been conducted recently. For instance, [13] is proposing an unusual Bayesian Belief Network-based method to solve the malware detection issue. To improve accuracy and lower false positives in malware detection, the suggested method extracts distinctive characteristics from the static, dynamic, and event analysis of the malware sample. In [14] proposes a ground-breaking method that uses machine learning algorithms. Then, a deep learning technique is used to find hidden patterns that cannot be found using the machine learning approach. In [15], an effective model that makes use of autoencoders for anomaly detection in cloud computing networks is provided. The autoencoder attempts to recreate normal data with the least amount of error after being trained to understand a basic representation of it. In [16], bagging and Gradient Boosting Decision Tree (GBDT), two well-known ensemble approaches, are combined to create a dual ensemble model. A brand-new double-layer design for MIT (Man-in-the-Middle) detection is put out in [17]. A system for data integration, transformation, and sampling makes up the architecture's first layer. This method includes using a support vector machine (SVM) classifier's performance to evaluate and choose.

Recent research, both theoretical and practical, has shown how serious a threat malware is to the digital world. Consequently, to improve security measures, virus mitigation strategies have experienced constant improvement. Malware was once often detected using conventional approaches that made use of heuristics and signatures [18]. Numerous in-depth studies have been conducted on the identification of feature-hybrid malware variants and ransomware attacks. For instance, [19] analyzes ransomware assaults and the methods employed to detect them. Initially, the detection methods are categorized based on how they operate. Then, a thorough comparison study is carried out to evaluate the suitability of these methods in different situations.

A suggested feature-hybrid method for malware variant detection is presented in [21]. The strategy integrates many feature kinds to address the issues. At first, opcodes are represented by a frequency vector, while a bi-gram model represents API calls. To improve these representations and accelerate the convergence, principal component analysis is used.

There are some limitations to traditional supervised learning techniques. Making and identifying crucial qualities of these things takes a lot of work and knowledge of the relevant topic. For this reason, there is increasing interest in using Deep Learning algorithms to detect malware.

3.2 Proposed Diagnostic System

This study uses an upgraded KDD-CUP99 dataset to build the Intrusion Detection System (IDS) detection phase, emphasizing categorized malware. Machine Learning algorithms are used to classify different types of malware and determine whether a program is typical. An extensive rundown of many malware. A flow diagram of the steps involved in malware diagnosis is shown in Fig 1.

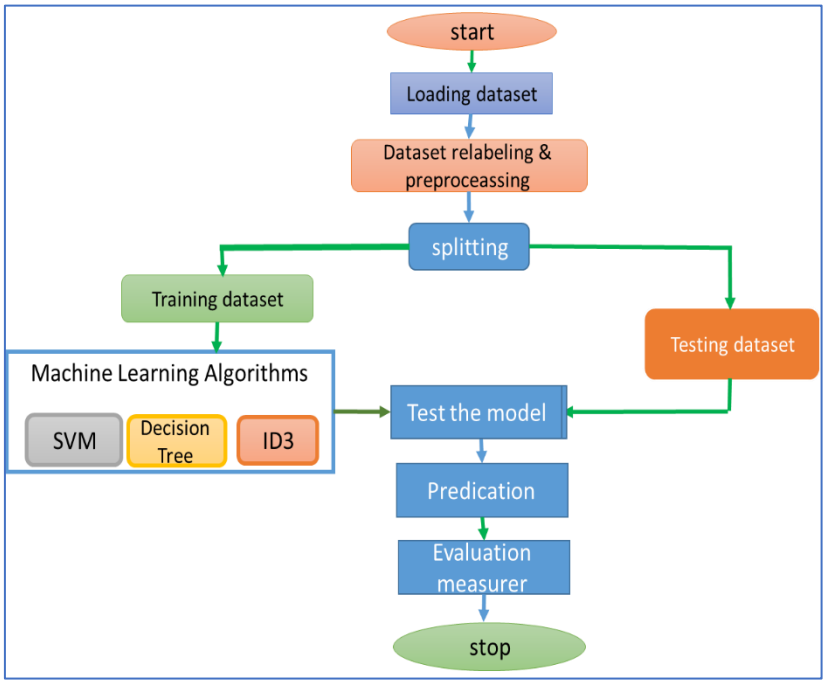

Figure 1: Flowchart of processes for diagnosing malware detection

Python is used to write the majority of the backend code for the web application. Algorithm execution and cloud data retrieval are only two of its numerous applications. The web app's creators evaluated several suggested algorithms before selecting the one that would offer the best performance and accuracy ratio. [22].

The system's foundation is an anomaly-based intrusion detection system. It aims to empower the system to differentiate between regular and deviant conduct by utilizing pertinent data and machine learning techniques. In the training phase, the system builds a decision tree using the ID3 method, a decision tree algorithm, and public data consisting of API call sequences classified into two classes, valid and malware, using the KDD-CUP99 dataset. Then, using this trained tree, data samples are classified into two groups: true, which indicates infiltration, and false, which represents typical behavior. The KDD-CUP99 information set is most frequently utilized to assess ID systems and models. Over 300,000 samples total—54 characteristics per sample—are present in KDD-CUP99.

- A total of 513 malicious files were employed.
- In contrast, 358 reliable files were used in total.

The first step in supervised machine learning is creating a dataset. The feature dataset is extracted using the files that were previously collected. FieldSize, addresses, entropy, and other parameters in machine learning are already primarily integer numbers. Integer or float data are used as recognition functions in machine learning. The PE parameters were retrieved using the Python "pefile" library. During extraction, pertinent aspects were considered [23]. Table 2 below displays the characteristics that were extracted:

Table 2: Features relevant extracted

features	The features details	Features	The features details
AddressOfEntryPoint	The entry point's address is indicated when the executable file loads into memory.	**Resources Mean Size**	Resources Mean Size
BaseOfCode	When it is loaded into memory, the address points to the start of the code section.	**Resources Min Entropy**	Resources Min Entropy
BaseOfData	The location is relative to the start of the data section when it is loaded into memory.	**Resources Min Size**	Resources Min Size
Characteristics	Flags denote the file's properties.	**Resources Nb**	Resources Number.
CheckSum	"The algorithm for computing the checksum is incorporated into IMAGHELP.DLL".	**Section Alignment**	The Alignment of Sections.
DllCharacteristics	"Dynamic Link library file characteristics"	**Section Max Raw size**	Section Max Raw size
ExportNb	"The Export number"	**Section Max Virtual size**	Section Max Virtual size
File Alignment	"The alignment factor (in bytes) that is used to align the raw data of sections".	**Sections Max Entropy**	Sections Max Entropy
Image base	When loaded into memory AND The preferred address of the first byte	**Sections Mean Entropy**	Sections Mean Entropy
Import NbDLL	Is a library (CODE &DATA)	**Sections Mean Raw size**	Sections Mean Raw size
Imported	The transformer's input should be a collection of characters or integers in an array.	**Sections Mean Virtualize**	Sections Mean Virtual size
ImportNbOrdinal	The data contains categorical data AND is encoded to numbers before it can fit and evaluate a model.	**SectionsMinEntropy**	Sections Min Entropy
LoadConfiguration Size	"The size of Load Configuration"	**Sections Min Rawsize**	Sections Min Raw size
LoaderFlags	Reserved, must be zero.	**Sections Min Virtualize**	Sections Min Virtual size
Machine	"The number that identifies the type of target machine"	**Sections Nb**	SectionNumber AND signed integer.
MajorLinker Version	"The linker major version number".	**Size OfCode**	The size (code section / the sum of all code sections)
MajorOperationsystem Version	" The major version number of the required operating system".	**SizeOfHeaders**	The combined size (an MS-DOS stub & PE header & section headers rounded up).
MajorSubsystemVersion	"The major version number of the subsystem".	**SIZEOfHEAP COMMIT**	SIZE of the STACK to heap commit & the size of the local heap to reserve."
MD5	Object(hash of files& string& arrays).	**SizeOfHeap Reserve**	
MinorImage Version	"The minor version number of the image".	**SizeOfImage**	The size (all headers of Section Alignment).
MinorLinker Version	"The linker minor version number".	**SizeOfInitialized Data**	The size of the initialized data section
Minor OperationSystemVersion	" The minor version number of the required operating system".	**Size Of Optional Header**	size of the code section
Minor Subsystem Version	" The minor version number of the subsystem".	**SizeOfUninitializedData**	"The size of the uninitialized data section (BSS)"
Name	object	**Size Of Stack Reserve**	(The size of the stack to reserve).
Number Of RvaAnd Sizes	"The number of data-directory entries in the remainder of the optional header".	**Win32VersionValue**	Reserved, must be zero.
ResourcesMeanEntropy	"an input device or a measured characteristic of an I/O device on a computer that supplies"	**SUBSYSTEM**	SUBSYSTEM that is required to run.
ResourcesMax ResourcesNb Entropy	The one with the largest entropy.	**Version Information Size**	It is version number."
Resources Max Size	Resources Max Size	**Resources Mean Entropy**	Resources Mean Entropy

Relevant feature selection aims to select a smaller subset of the 54 returned attributes necessary to differentiate between safe and malicious files. This entails utilizing libraries like

Pandas, Scikit, and Numpy for data processing to compare feature values between malicious and legitimate files. The dataset observations and features are the foundation for the Trees classifier, with the most important qualities being positioned at the top of the tree for easier comprehension. To reduce overfitting and raise prediction accuracy, the tree is finally trimmed to contain only the most crucial characteristics. Utilizing observations and traits taken from the dataset, the Trees classifier uses them. Higher subdivisions yield more information. Hence, the attributes chosen for the tree's top nodes are more significant for interpretation than those at the tree's end nodes. The Mistake was an error in the reference source. The relevant properties and their selection are shown in Fig 2.

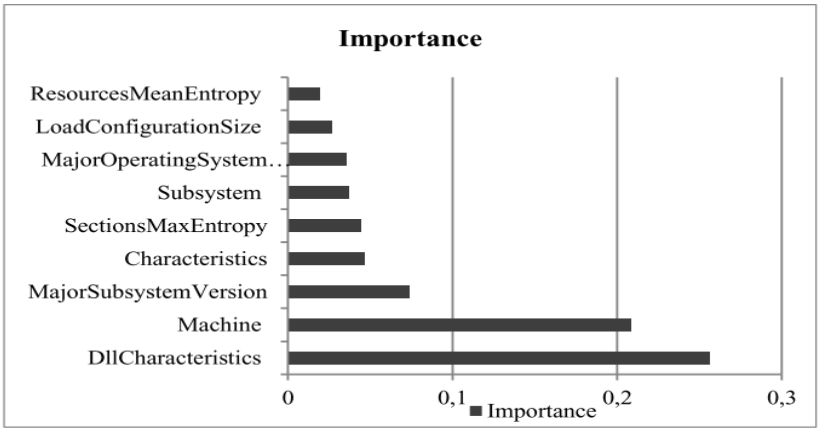

Figure 2: Significance of relevant traits [23].

3.3 Classification

These three categorization stage techniques were chosen with the best parameter sets and workable. The SVM, ID3, and DT were the names of them. Using the dataset, two arrays were made: The values that were valid for each row were all contained in the first, y; all the values that were not in Y were all contained in the second, X. By comparing the "X" and "y" characteristics, the categorization method search for trends that show how a certain set of values affects the legitimacy of the files. A training set and test comprise the arrays "X" and "y," respectively. Classifying this test set will allow us to determine whether our model functions as intended so that a model can be developed by training the classification algorithm on the training set of data. Python's time package monitored the time used to execute each algorithm.

3.4 Studying the Efficiency of Methods:

The three metrics used to evaluate the effectiveness of an algorithm are the execution time, accuracy rate, and false positive ratio. The model's accuracy can be assessed using the Python Scikit library score function. One way to find false positive and negative rates is to utilize the confusion matrix. Comparing different test findings and forecasts with the actual numbers is known as a confusion matrix. The acronyms TP, TN, FP, and FN can be found in a phone book. The statistics and machine learning fields, which primarily rely on AI (artificial intelligence) use cases based on Big Data technology, employ it extensively for data mining applications. Another function that generates a confusion matrix is found in the Scikit package. To calculate the efficiency numbers, utilize the following formulas:

Equation (1) can be used to calculate the False Positive rate, which is the ratio of all negative samples to mistakenly positive ones.

$$\boldsymbol{FPR} = \frac{\boldsymbol{False\ Positive}}{\boldsymbol{False\ positive+True\ Negative}} \qquad \textit{....(1)}$$

Equation (2) yields the percentage of positive test findings resulting in negative test results, known as the false negative rate.

$$FNR = \frac{False\ Negative}{False\ Negative+True\ Positive} \quad \text{.... (2)}$$

Equation (3) computes the accuracy, sometimes called the score, as the proportion of properly classified samples among all samples.

$$ACC = \frac{True\ Positive+True\ Negative}{True\ Positive+False\ Positive+False\ Negative+True\ Negative} \quad \text{... (3)}$$

3.5 Pre-processing of Datasets

Classification is divided into two stages:

PRE-PROCESSING includes (labeling, loading datasets, processing datasets, and dividing data into **(**Training and Testing sets).
Three components make up this process:

- Data transformation (normalization, attribute selection, concept hierarchy creation)
- Data reduction (aggregating data cubes, choosing attribute subsets, decreasing dimension)
- Data purification (managing missing data, applying the ID3 technique).

This procedure involves data transformation and purification. Erroneous, excessive, redundant, or incomplete information is eliminated by data cleansing.

POST-PROCESSING is a kind of data transformation that uses methods to improve accuracy and performance by transforming continuous data into a discrete form.

3.6 Using K-Fold Cross-Validation to Split Data

K-Fold cross-validation is a method that is widely used in machine learning. The present project has made use of five-fold cross-validation. Five-fold cross-validation was applied in the current project. This approach assures that each subset is utilized precisely once for testing and training. Five distinct subsets exist within the KDD-CUP99 dataset. The K-fold cross-validation procedure is described below, and Algorithm 1 depicts the algorithm.

Algorithm 1: K_fold Cross-Validation Method

K_fold Cross_Validation Algorithm
Input: download the dataset of *KDD-CUP99* **Output:** Divided all the Data into Training and Testing sets. **Start:** **Step 1**: Divide Data set D into subgroups(K) of equal size. **Step 2**: The first subset was used as the testing set AND the other subsets as the Training set. **Step 3:** K times the second step was repeated. **Step 4:** The formula following may be used to get the average prediction error of the entire model performance or K. $CV_K = \frac{1}{k} \sum_{i=1}^{k} EU_i$ ... (4) *Method End*

3.7 Classification Methods

This section goes deep into a thorough analysis of three proposed methods. First and foremost, this research needs to find the algorithm that works better than the others, shows remarkable accuracy, and is thus the best option for malware detection.

3.7.1 Decision Tree

Decision trees are used for regression and classification in machine learning. Until subsets are homogenous, they partition data recursively according to discriminative traits. Decision trees are useful for their readability, simplicity, and capacity to handle various kinds of data. Trees can be trimmed to remove extraneous branches or nodes to prevent overfitting. Pruning a tree involves cutting off extra branches or nodes to prevent overfitting. In Python, decision trees are often created using the sci-kit-learn module. A Decision Tree Classifier object is constructed, loaded, and split into parts.

3.7.2 ID3 Method

Before C4.5, ID3 splits datasets iteratively according to the characteristic that yields the maximum information gain to create decision trees. Measured before and after division by a particular characteristic, information gain quantifies the change in dataset impurity (entropy). The algorithm chooses each tree node's splitting criterion based on which attribute has yielded the most information. Despite being simple and effective, ID3 cannot manage missing data and may cause overfitting.

3.7.3 Support vector Machine Learning

Regression estimation, malware categorization, and pattern recognition are among the many applications for SVMs [24]. Maximizing the distance between various samples is their goal. Classifications in SVM might be linear or non-linear. The optimal plane to separate patterns is found using linear SVM [24]. The training samples {(xi, yi)} N i=1 are considered, where yi represents the associated target output, and xi is the input pattern of the ith occurrence. The linearly separable to create patterns with yi = +1 and yi = -1. Here is how a hyperplane depicts the separation equation:

$$W^T X + b = 0 \quad \ldots (5)$$

Non-linear decision boundaries are found using two steps in SVM. The first step involves kernel functions to convert the input data into a higher dimensional space. A linear separation hyperplane must be located in the converted space as the second step [24]. Diverse datasets with few training samples are easily handled by SVM classification. Unfortunately, certain disadvantages include choosing kernel function parameters and speed and size limitations during testing and training. [25].

3.8 Evaluation of the proposed model

The Machine Learning approaches of the proposed system are presented and evaluated in this section. It contrasts the accuracy results with previous studies and evaluates how well the system's classification algorithm performs compared to alternative approaches. It is critical to consider various Machine Learning techniques and other security measures to improve cybersecurity.

3.8.1 Detecting and Defending

After a firewall, an intrusion detection system (IDS) watches over all network traffic and serves as a backup filter for harmful packets. It looks for variations from regular behavior and known attack signatures. An intrusion detection system can recognize Malware and scanning assaults, asymmetric routing, buffer overflow, protocol-specific attacks, and network overloads through pattern correlation. The system sounds an alert when it finds an abnormality. This alert might be anything from a message for an IT administrator to a notation in an audit record. The issue's root cause was found in the IDS team's investigation. The four states that emerge from the classification of data by intrusion detection systems are true positive (an intrusion that has been appropriately identified), false positive (a false alert), true negative (an incursion that has been correctly recognized as normal) and false negative (an intrusion that has been overlooked).

3.8.2 Evaluation criteria

Table (3) provides the Intrusion Detection System (IDS) rating criteria. IDS solutions' efficacy and performance are evaluated using these standards. They provide standards by which to compare the efficacy and dependability of intrusion detection systems (IDS) in identifying and thwarting different kinds of network vulnerabilities and attacks.

Table 3: Performance evaluation standards

Evaluation type	Criterion Evaluation	Standard Description	Scientific form of the Formula
Diagnosis criteria	The rate of detection	DR = $\frac{TP}{TP+FN}$	DR = $\frac{TP}{TP+FN}$
	ACC	This criterion determines the degree of accuracy of normal and abnormal in all states. These standards are crucial for assessing and contrasting various groups.	ACC= $\frac{TP+TN}{TP+FP+FN+TN}$
Performance standards	Recall	This metric represents the portion of the accuracy metric that is absent or indicates the proportion of actual assault samples the classifier covers and detects. This metric is comparable to the detection rate (DR) and the correlation between DR and recall. It remains the same.	detection rate
	Rate of Precision	This measure shows the proportion of data samples that tested positive (positive infiltrations) relative to all positive samples that were considered positive (total infiltrations, including false positives and real positives).	Pre = $\frac{TP}{TP+FN}$ .. (7)
	F_Measure	The value of F-Measure represents the harmonic average (recall and accuracy).	F-Measure = $\frac{2 \times Pre \times recall}{Pre+recall}$ .. (8)
	The importance of positive prediction	A measure of a test's accuracy is its positive predictive value.	PPV

4. RESULTS

Intrusion Detection Systems (IDSs) will get better categorization results with increased performance or efficiency. KDD-CUP99 is the source of the testing data used in this investigation. The outcomes discussed in this section are relevant only to the algorithm's evaluation when randomly chosen data from the training dataset is used. When assessing the model, several variables were considered, including the detection rate, correctness, accuracy, positive predictive value (PPV), and F-measure.

4.1 Decision Tree (DT) Performance Measures:

The Decision Tree(DT) algorithm performance with 5-fold cross-validation is displayed in Table (4). Three distinct dataset proportions were used for testing and training: 50%, 35%, and 65%. Using 35% of the dataset for testing and 65% for training (358 and 668 samples, respectively) produced the best results (96.6% accuracy). The accuracy decreased to 95.0% when training and testing were conducted on half of the dataset (513 samples). The performance metrics decreased because more testing samples were in the first scenario (513 vs. 358 in the second case).

Table 4: Five-fold Decision Tree (DT) Performance Measures

Test samples	TN	FN	Recall	specificity	PPV	F-measure	Accuracy	Execution time
	FP	TP						
513	2609	986	0.91	0.96	90.52	1.79	95.0%	0.76 sec
358	2796	664	0.91	0.97	91.42	1.81	96.6%	0.52sec

4.2 Performance Measures for the ID3

The system-dependent ID3 method performs best when the test sample makes up 35% of the dataset (358 samples). The algorithm's 5-fold validation performance is shown in Table (5). Initially, 513 samples, or 50% of the dataset, were used to train and test the model. When the training sample is 65% and the test sample is 35%, the greatest accuracy of 97.7% is obtained (358 and 668 samples, respectively). It performs better than other algorithms regarding recall, specificity, F-measurement, positive predictive value (PPV), and execution time. Accuracy drops to 96.9%, along with other metrics, when half of the dataset is used for testing and training. On the other hand, there is a notable difference in the execution time: in the first example, testing took 0.63 seconds and 513 samples, while for the greatest accuracy (97.7%), it took 0.49 seconds and 358 samples. The first instance had more testing samples (513 vs. 358 in the second example), which may have contributed to the performance decline.

Table 5: ID3 Performance metrics for five times

Test samples	TN FP	FN TP	Recall	specificity	PPV	F-measure	Accuracy	Execution time
513	355 7	7 94	0.931	0.981	93.06	1.843	96.9%	0.63 sec
358	281 4	4 66	0.942	0.985	94.28	1.865	97.7%	0.49sec

4.3 Performance Measures for the SVM

Regarding malware detection, the SVM algorithm performs admirably. As Table (6) shows, if 50% of the dataset (513 samples) is tested, the classifier attains an accuracy of 87.9%. The reasons for this poorer accuracy are the random selection of training and testing samples and a high number of testing samples. Still, the system accuracy increases to 88.7% when testing uses 35% of the dataset (358 samples). Furthermore, using 35% and 65% of the dataset for training and testing improves the performance of metrics like recall, specificity, accuracy, F-measure, positive predictive value (PPV), and execution time. Simply put, Table (6) displays the 5-fold cross-validation performance metrics for the SVM model. At 35% for the test sample and 65% for the training sample, the greatest accuracy (88.7%) is attained. The dataset 50% of (513 samples) used for training and 50% for testing yields the lowest accuracy (87.9%).

Table 6: SVM Performance measures for 5-fold

Test samples	T_N F_P	F_N T_P	Recall	specificity	PPV	F-measure	Accuracy	Execution time
513	355 30	30 71	0.931	0.922	70.29	1.390	87.9%	0.67 sec
358	265 20	20 66	0.714	0.929	71.42	1.414	88.7%	0.63sec

4.4 The Comparison Result of The Malware Detection

The recommended SVM, decision tree, and ID3 classifiers for malware detection are evaluated Based on test length, specificity, accuracy, recall, and precision. When 50% of the data (513 samples) are used for training and testing, the comparison results are shown in Fig.3, and ID3 outperforms the other classifiers with a high specificity of 98.1%. The decision tree (95.0%) and SVM classifier (87.9%) have the highest accuracy, while ID3 (96.9%) has the best performance. In comparison to the other classifiers, ID3 also shows faster diagnostic results. In conclusion, ID3 is the most dependable recommended classifier for malware detection.

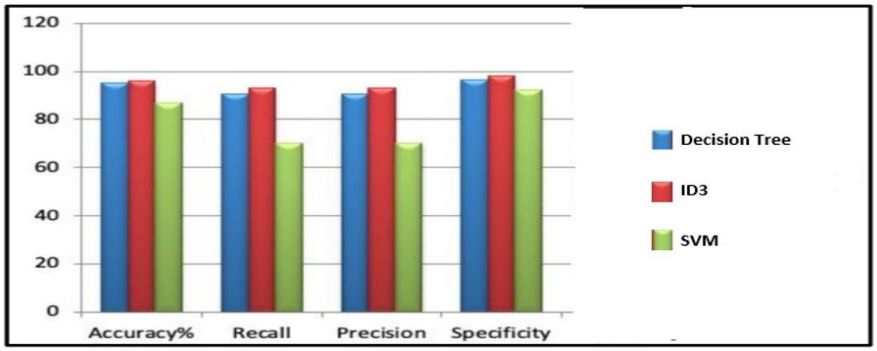

Figure 3:. The KDD-CUP99 dataset

Fig .4 compares the performance of classifiers utilizing 35% of the data (358 samples) and 65% (668 samples) from the KDD dataset for accuracy evaluation. ID3s obtain the greatest accuracy of 97.7% of the two classifiers, beating the accuracy of the decision tree (96.6%) and SVM (88.7%). Regarding recall, accuracy, specificity, and execution time, ID3s perform better than the other classifiers.

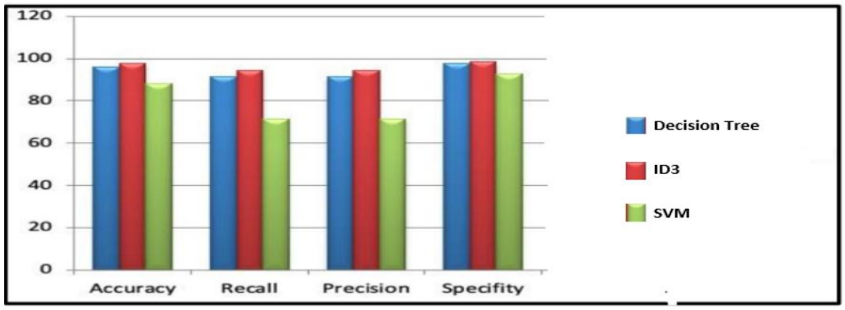

Figure 4: Comparing the four-class malware detection performance

4.5 Performance Comparison with Previous Work

Several research studies have used the KDD dataset to improve malware detection accuracy. However, to the researcher's knowledge, only a few prior studies have explicitly used a five-class categorization method for malware identification. The accuracy attained in several works is outlined in Table (8) and compared to the accuracy achieved in the suggested work.

Table 8: Result of comparison with previous work of malware detection

Reference	The method	Training %	Testing %	Accuracy%	Our work: The method	Our work: Training set %	Our work: Testing set%	Our work: Accuracy%
[5]	SVM+RF+ GA	82%	19%	94%	SVM+ ID3 + DT	65%	35%	88% 97% 96%
[7]	SVM + MLP + KNN	50%	50%	60% 81% 87%	SVM+ DT +ID3	50%	50%	87% 95% 96%

In [5], a dataset used the KDD repository with SVM, RF, and GA to detect malware. 94% of the samples were used for testing, and eighty-two % were used for training. The accuracy of the SVM, ID3, and decision tree (DT) approached 88%, 97%, and 96%, respectively. When contrasting the precision above with that of the suggested task the important to note that 35% of the samples in the proposed study were used as a test set and 65% of the samples were used as a training set. Despite having a lower percentage of training samples than in the research study[5], the accuracy was higher for all algorithms used in the proposed study (decision tree, ID3, and SVM).

In [7], 50% of the samples were used for training and 50% for testing, as SVM, MLP, and KNN were employed to detect malware. With an accuracy of 87%, the KNN outperformed SVM and MLP, which came in at 60% and 81%, respectively. The decision tree (DT) and ID3 obtained 95% and 96% accuracy, respectively, when comparing the accuracy of the proposed system to that reported above for the same proportion of training and test samples (50%). SVM obtained the lowest accuracy of 87%.

5. CONCLUSION AND FUTURE WORKS

In computer security, this article describes the functions of malware detection and intrusion detection systems (IDS). These systems may be rendered more precise and efficient by utilizing Machine Learning techniques, such as the ID3 algorithm. Machine learning requires feature engineering in four crucial phases to improve malware detection. Algorithmic selection Propagate dataset. Cross-checking, observing, and informing the claim that IDSs are divided into two categories of store-based information sources—network or host-based—has always been true. Anomaly recognition (network) and object-based ID methods fall into two groups. Intelligent security systems (IDS) employ data mining and machine learning to distinguish between odd and normal activity patterns, which allows them to spot attacks with high accuracy. The ID3 machine learning system, which relies on decision trees for training, has the potential to detect a noteworthy proportion of threats accurately. The ID3 algorithm is a great choice for completing intrusion detection jobs because of its ability to classify and detect anomalies. In conclusion, it proved how important intrusion detection systems (IDS) are for network security and how methods like ID3 may improve their effectiveness. The research suggested the following approaches to improve malware detection using AI and ML:

- This is the first piece of useful advice: a malware detection model quality may be greatly increased by selecting features using selection techniques.
- Collective approaches are a second recommendation. This method may increase accuracy and performance by integrating many models.
- The third strategy is CNNs and RNNs, two deep learning-based techniques. They were identified as malware, thanks to their complex patterns.
- Attending to adversarial assaults is a crucial component of malware detection, as indicated in the fourth recommendation. Methods investigated in this article are used to identify and lessen the effects of harmful assaults on machine learning models.
- The fifth step is to ensure that everything is thoroughly recorded using interpretable machine-learning techniques so that the decision-making process is understood and explained.
- The sixth is transfer learning, or the application of information from one field to another, a suggestion for increasing the effectiveness of malware detection models.
- Seventh, real-time detection techniques are shown to emphasize fast thinking and the danger that malware quicksands pose.
- The eighth is that to assess the effectiveness of malware detection programs, appropriate assessment metrics must be used.
- Ninth, the size of the training dataset can be increased by employing data augmentation approaches to improve machine learning models for generalization.
- The tenth is continuous learning, which is a piece of advice. A model may progressively improve over time while holding onto what it has already learned if it continuously learns from new data.

Taken together, though, these 10 suggestions provide practitioners and academics with the means to develop more effective Machine Learning models for malware detection. Using other

datasets with higher-quality software examples would be helpful and fascinating for the next initiatives. Furthermore, research indicates that machine learning properties provide a substantial advantage over current malware detection systems, adding to the exciting growth in this field. There's still a lot of opportunity for more research in this area, judging by the continuous development of new machine-learning techniques.

CONFLICT OF INTEREST

The authors declare that there is *no conflict of interest* regarding the publication of this paper.

REFERENCES

[1] M. Abdelsalam, R. Krishnan, Y. Huang and R. Sandhu, "Malware Detection in Cloud Infrastructures Using Convolutional Neural Networks," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2018, pp. 162-169, doi: 10.1109/CLOUD.2018.00028.

[2] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse and T. Yagi, "Malware Detection with Deep Neural Network Using Process Behavior," 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, USA, 2016, pp. 577-582, doi: 10.1109/COMPSAC.2016.151.

[3] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls," 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 2013, pp. 300-305, doi: 10.1109/ICTAI.2013.53.

[4] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, “Malware detection using machine learning and deep learning,” in Lecture notes in computer science, 2018, pp. 402–411. doi: 10.1007/978-3-030-04780-1_28.

[5] Z. Xu, S. Ray, P. Subramanyan and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, Switzerland, 2017, pp. 169-174, doi: 10.23919/DATE.2017.7926977.

[6] J. Lee, H. Jang, S. Ha, and Y. Yoon, “Android Malware Detection Using Machine Learning with Feature Selection Based on the Genetic Algorithm,” Mathematics, vol. 9, no. 21, p. 2813, Nov. 2021, doi: 10.3390/math9212813.

[7] S. HR, "Static Analysis of Android Malware Detection using Deep Learning," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 841-845, doi: 10.1109/ICCS45141.2019.9065765.

[8] R. Feng, S. Chen, X. Xie, G. Meng, S. -W. Lin and Y. Liu, "A Performance-Sensitive Malware Detection System Using Deep Learning on Mobile Devices," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1563-1578, 2021, doi: 10.1109/TIFS.2020.3025436.

[9] S. Rkhouya and K. Chougdali, “Malware detection using a Machine-Learning based approach,” International Journal of Information Technology and Applied Sciences IJITAS), vol. 3, no. 4, pp. 167–171, Oct. 2021, doi: 10.52502/ijitas.v3i4.172.

[10] Hussain, M. Asif, M. B. Ahmad, T. Mahmood, and M. A. Raza, “Malware detection using machine learning algorithms for Windows Platform,” in Lecture notes in networks and systems, 2022, pp. 619–632. doi: 10.1007/978-981-16-7618-5_53.

[11] S. Shatnawi, Q. Yassen, and A. Yateem, “An Android malware detection approach based on static feature analysis using machine learning algorithms,” Procedia Computer Science, vol. 201, pp. 653–658, Jan. 2022, doi: 10.1016/j.procs.2022.03.086.

[12] N. B. S. Abdulwahed, N. A. Al-Naji, N. I. Al-Rayahi, N. A. Yahya, and N. A. G. Perera, "Automated Computer Vision System for urine color detection," Journal of Techniques, vol. 5, no. 1, pp. 66–73, Apr. 2023, doi: 10.51173/jt.v5i1.896.

[13] Sharma, B. B. Gupta, A. K. Singh, and V. K. Saraswat, "Multi-dimensional Hybrid Bayesian belief network based approach for APT malware detection in various systems," in Lecture notes in networks and systems, 2023, pp. 177–190. doi: 10.1007/978-3-031-22018-0_16.

[14] K. S. Sangher, A. Singh, and H. M. Pandey, "Signature based ransomware detection based on optimizations approaches using RandomClassifier and CNN algorithms," International Journal of Systems Assurance Engineering and Management, vol. 15, no. 5, pp. 1687–1703, Jul. 2023, doi: 10.1007/s13198-023-02017-9.

[15] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical autoencoder based anomaly detection by using vector reconstruction error," Cybersecurity, vol. 6, no. 1, Jan. 2023, doi: 10.1186/s42400-022-00134-9.

[16] M. H. L. Louk and B. A. Tama, "Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system," Expert Systems With Applications, vol. 213, p. 119030, Oct. 2022, doi: 10.1016/j.eswa.2022.119030.

[17] S, S. D, and P. G, "Malicious insider threat detection using variation of sampling methods for anomaly detection in cloud environment," Computers & Electrical Engineering, vol. 105, p. 108519, Dec. 2022, doi: 10.1016/j.compeleceng.2022.108519.

[18] G. M and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," Computer Science Review, vol. 47, p. 100529, Dec. 2022, doi: 10.1016/j.cosrev.2022.100529.

[19] M. Ashraf, M. Asif, M. B. Ahmad, A. Ayaz, A. Nasir and U. Ahmad, "Towards Classification and Analysis of Ransomware Detection Techniques," 2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2023, pp. 1-5, doi: 10.1109/iCoMET57998.2023.10099204.

[20] J. Zhang, Z. Qin, H. Yin, L. Ou, and K. Zhang, "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding," Computers & Security, vol. 84, pp. 376–392, Apr. 2019, doi: 10.1016/j.cose.2019.04.005.

[21] J. Zhang, Z. Qin, H. Yin, L. Ou, and K. Zhang, "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding," Computers & Security, vol. 84, pp. 376–392, Apr. 2019, doi: 10.1016/j.cose.2019.04.005.

[22] W. Zhao, I. Abdelaziz, J. Dolby, K. Srinivas, M. Helali, and E. Mansour, "Serenity: library based Python code analysis for code completion and automated machine learning," arXiv (Cornell University), Jan. 2023, doi: 10.48550/arxiv.2301.05108.

[23] S. Rkhouya and K. Chougdali, "Malware detection using a Machine-Learning based approach," International Journal of Information Technology and Applied Sciences (IJITAS), vol. 3, no. 4, pp. 167–171, Oct. 2021, doi: 10.52502/ijitas.v3i4.172.

[24] F. Kazemi, N. Asgarkhani, and R. Jankowski, "Predicting seismic response of SMRFs founded on different soil types using machine learning techniques," Engineering Structures, vol. 274, p. 114953, Oct. 2022, doi: 10.1016/j.engstruct.2022.114953.

[25] Roy and S. Chakraborty, "Support vector machine in structural reliability analysis: A review," Reliability Engineering & System Safety, vol. 233, p. 109126, Jan. 2023, doi: 10.1016/j.ress.2023.109126.