

A comparative study of machine learning tools for detecting Trojan horse infections in cloud computing environments

Hasan Kanaker¹, Monther Tarawneh², Nader Abdel Karim³, Adeb Alsaaidah⁴, Maher Abuhamdeh⁵, Osama Qtaish⁶, Essam Alhroob¹, Zaid Alhalhouli⁷

¹Department of Cyber Security, Faculty of Information Technology, Isra University, Amman, Jordan

²Department of Information Technology, College of Information Technology, Tafila Technical University, Altafila, Jordan

³Department of Intelligent Systems, Faculty of Artificial Intelligence, Al-Balqa Applied University, Al-Salt, Jordan

⁴Department of Networks and Cyber Security, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan

⁵Department of Computer Information System, Faculty of Information Technology, Isra University, Amman, Jordan

⁶Department of Software Engineering, Faculty of Information Technology, Isra University, Amman, Jordan

⁷Department of Data Science and Artificial Intelligence, Faculty of Information Technology, Isra University, Amman, Jordan

Article Info

Article history:

Received May 2, 2024

Revised Jul 20, 2024

Accepted Aug 6, 2024

Keywords:

Cloud computing

Control lab

Cybersecurity

Machine learning

Python

Trojan horse infection

Weka

ABSTRACT

Cloud computing offers several advantages, including cost savings and easy access to resources, it is also could be vulnerable to serious security attacks such as cloud Trojan horse infection attacks. To address this issue, machine learning is a promising approach for detecting these threats. Thus, different machine learning tools and models have been employed to detect Trojan horse infection such as Weka and Python Colab. This study aims to compare the performance of Weka and Python Colab, as popular tools for building machine learning models. This study evaluates the recall, accuracy, and F1-score of machine learning models built with Weka and Python Colab and compares their computational resources required employing several machine learning algorithms. The dataset collected and analyzed using dynamic analysis of Trojan horse infection in control lab environment. The findings of this study can help determine the decision about which tool to use to detect Trojan horse infections and provide insights into the strengths and limitations of Weka and Python Colab for building machine-learning models in general.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Hasan Kanaker

Department of Cyber Security, Faculty of Information Technology, Isra University

Amman, 11622, Jordan

Email: hasan.kanaker@iu.edu.jo

1. INTRODUCTION

Cloud computing, an innovative method of information technology, uses the internet and remote servers to deliver a common set of computing resources and applications to meet customer requirements [1]. The advent of cloud computing technologies in the first decade of this century ushered in a new era in the development of information technology (IT) infrastructure, as users can now obtain software and computing power over the Internet or networks. This has also led to the emergence of new models for hosting and distributing online services [2], [3]. Cloud computing allows users to easily access services, facilitating seamless access to data and program execution on a large number of connected computers. Moreover, it removes the requirement for users to install software on their personal computers and instead provides access to resources and applications via an internet connection at any time and from any location [2], [4].

Cloud computing has been widely adopted by public-sector businesses in many countries, including Australia, the United Kingdom (UK), the United State (US), and various European countries [5]–[7]. The

government cloud (G-Cloud) infrastructure was introduced by the UK government in 2010, saving an estimated £3.2 billion [5], [8], [9]. The cloud computing mall was also introduced by the US government in 2009 [8]. Cloud computing offers numerous benefits such as shared asset pooling, tremendous versatility, cost savings, flexibility, pay-as-you-go pricing, self-provisioning of resources, and multi-tenancy [10]–[12]. However, there are still security concerns and risks associated with cloud computing [13], particularly the risk of security attacks. As a result, both service providers and clients are concerned about these attacks [14].

Various forms of attacks pose a threat to cloud computing, including phishing, authentication, service denial, man-in-the-middle, and malware insertion, among others [2], [15]–[18]. Among them, malware attacks are a significant risk to the cloud computing ecosystem. One such attack is the "cloud Trojan injection attack," which involves injecting a malicious program into cloud services to cause harm. These programs can be disguised as normal commands and executed as such [19]. Cloud Trojan injection attacks can introduce malicious services, virtual machines, and applications into cloud systems, which can affect cloud functionality by interfering with or altering it [20], [21]. Attackers upload a malicious version of application, virtual machine or service to a cloud system so that it will think it is a real instance of that application virtual machine or service. When a normal user requests an instance of the malicious service to run, the malicious code is executed [20]. Trojan horses are challenging to detect using signature-based technologies, which is the most common anti-virus system detection method [22]. Furthermore, only well-known signatures can be used with signature-based anti-virus to achieve high accuracy. This type of detection has a drawback in that it frequently misses fresh attacks when malware completely changes its signature.

With the increasing number of cyber-attacks, the need for effective tools to detect malware and Trojan horse infections is more important than ever. To protect and reduce damage caused by Trojan horse infection, machine learning tools have been proven as an effective technique used to detect such threats by analyzing patterns in the data and identifying possible dangerous behavior. Weka and Python Colab are known tools to use and implement machine learning models. Weka is a Java-based program that provides a graphical user interface exploited ready-to-use machine learning models, while Python Colab is an online Jupyter notebook environment that allows users to utilize to implement and run Python machine learning libraries in the cloud. Although, it is possible to create and utilize machine learning models to identify Trojan horse infections using either Weka or Python Colab, there is limited researches comparing the performance of these two tools.

The goal of this study is to compare the performance of Weka and Python Colab in detecting Trojan horse infections. We will evaluate the recall, accuracy, and F1-score of machine learning models built with Weka and implanted using Python Colab. We will also compare the computational resources required to build and train the models, including processing time and memory usage. The results of this study help to determine the decision about which tool to use to detect Trojan horse infections. Furthermore, this study provides insights into the strengths and limitations of Weka and Python Colab for building machine-learning models in Trojan horse detection.

2. BACKGROUND AND RELATED WORKS

Trojan horses are a type of malicious software that pose as legitimate or helpful programs to gain access to users' computers. Cybercriminals frequently use social engineering tactics to trick users into unintentionally installing them [23]. Once installed, Trojan horses have the ability to remotely manipulate the victim's device, steal sensitive data, watch user activities, and corrupt, destroy, or modify system files [24], [25]. Unlike viruses and worms, Trojans require user interaction to propagate. They are considered among the most dangerous forms of malware due to their prevalence [24]–[26].

Trojan horses can come in two types: general and remote-access Trojans [24]. General Trojans perform a variety of malicious actions, including jeopardizing the data integrity of victim machines, directing users' workstations to specific websites via system files, and running additional harmful programs [26]. They have the ability to track user behavior and provide the information to the attacker. Remote-access Trojans, on the other hand, have a special capacity that allows hackers to remotely manipulate the target machine across the internet or a local area network (LAN). This type of Trojan is particularly dangerous as it can be used to steal confidential information from the victim's personal computer (PC) and carry out other nefarious activities.

Machine learning methods have become increasingly popular for malware detection in cloud environments, as well as in various other areas [27]–[32]. Support vector machines (SVM), decision trees (DT), random forest (RF), and naive Bayes (NB) are some of the machine learning techniques that have been used to successfully find malware in practical applications [30], [33], [34]. The size of the training dataset and the number of features that can be retrieved from it, however, determine how accurate these algorithms are.

In a recent study [23], an approach based on convolutional neural networks (CNN) for virus detection in cloud platforms has been suggested. For identifying malware, the authors used a CNN model with two dimensions (2D) and a CNN model with three dimensions (3D). Also, they conducted experiments by installing various infections on simulated PCs and achieved 79% accuracy with the 2D CNN model and 90% accuracy with the 3D CNN model. However, this study did not compare CNN to conventional machine learning techniques; it only focused on CNN [29].

The efficacy of various machine learning methods, such as DT and SVM, was proved using a system based on machine learning for malware detection [30]. The researchers employed the Cuckoo Sandbox to test different malware kinds in a simulated environment, producing an analysis report depending on how the malware samples act in the setting. The Cuckoo Sandbox has been utilized by numerous researchers to examine malware [30], [35]–[38] examined the viability of using CNN, RF, and k-nearest neighbors (KNN) models and used features from application programming interface (API) calls to identify malware. Additionally, Watson *et al.* [31] utilized the random forest classifier to keep track of a virtual machine's process activity.

Weka has been used in studies on malware detection by numerous security researchers, including [24], [26], [39], [40]. On the other hand, several researchers, including [40]–[43] have utilized Python in their studies. Kumar *et al.* [44] reportedly demonstrated a machine learning-based approach to accurately and efficiently determine whether a sample is malware or benign. Weka was utilized by the authors to implement six classification algorithms. As a result of applying 10-fold cross-validation, the random forest classifier was able to achieve an accuracy of 98.4%.

J45, logistic model tree (LMT), naive Bayes (NB), random forest (RF), multilayer perceptron (MLP) algorithm, random tree (RT), reduced error pruning tree (REPTree), Bagging, AdaBoost, KStar, simple logistic, lazy k-nearest neighbor (IBK), locally weighted learning (LWL), support vector machine (SVM), and radial basis function (RBF) network are among the fifteen different machine learning algorithms used, this report [26] provided a comparative analysis of malware identification. The experiment was carried out in the Weka environment using the classification of malware with portable executable (PE) headers (ClAMP) dataset. The accuracy rate is still poor even if the RF method performs better than the other techniques.

In accordance with a study [24], this study examined eight machine learning classifiers to detect Trojan horses in cloud environments. Investigations examining the accuracy of the cloud Trojan horse detection rate have been conducted using the data mining tool Weka. The most effective classifiers for identifying Trojan horses in a cloud-based setting have been shown to be the sequential minimal optimization (SMO) and multilayer perceptron based on the studies that have been done. With a 95.86% accuracy rate, SMO and multilayer perceptron have the greatest rate.

According to their research paper, Kanaker *et al.* [24] proposed a hybrid machine learning algorithm that merged KNN and NB for detecting Trojan horses, and they executed their work using Python in the Colab environment, where certain machine-learning techniques are tested with 99.5% accuracy. The empirical results of this study demonstrate that the hybrid algorithm is the most effective algorithm for detecting a Trojan horse.

Sethi *et al.* [45] develop a machine learning-based malware analysis framework for rapid and accurate malware detection. A Python package was developed for feature selection, feature extraction, and the building of training and testing datasets. For the detection and classification of the provided dataset, they made use of numerous machine learning techniques offered by the Python Scikit-learn framework. A decision tree with an accuracy of 99.37% has a high detection rate, according to experimental results.

In a publication [46], a unique machine learning approach is suggested and implemented in Python and MATLAB to recognize the Mirai malware. This study compares the performance of artificial neural network (ANN) and RF models using a dataset created by combining the benign and malicious datasets for seven internet of things (IoT) devices. Because of the greatest performance, which had an accuracy of 92.8%, the results are deemed to be reliable and accurate.

The first step in the malware detection procedure is malware analysis. Malware cannot be discovered until it is observed and its behavior is understood. As a result, it is easy to add security measures to malware detectors. There are two categories of malware analysis techniques: static and dynamic, depending on the amount of time and technology required to complete the investigation.

Without actually running them, static analysis examines malware executable files in a controlled environment [47]–[49]. Static data is extracted from the code and used to determine whether the software contains malicious code [47]. The executable file contains a lot of static features, like memory compactness. To do static analysis, a variety of tools can be employed, including decompilers, disassemblers, source code analyzers, and debuggers [47]. Furthermore, only known malware signatures can be reliably detected via static analysis. Because of this, it could rarely be unable to assess malware signatures that are unknown and

not in its database. Regular updating and production by humans with specific knowledge are required for the signatures used in standard static analysis [4], [7], [24], [39]. It is not employed in this study due to the reasons mentioned above.

Dynamic analysis evaluates malware behavior in a dynamically controlled environment. In addition to starting in privileged mode, the virus modifies the registry as it runs. When the malware switches to privilege mode, the operating system will be completely under its control. All resources are completely under the control of dynamic analytic software. It is able to function in a secure environment as a result. The program can operate in debug mode and alter computer registry keys in a controlled environment. The dynamic environment returns to the initial snapshot that was taken at the start of environment development after executing and analyzing a malware sample. Before looking at another malware sample, this makes sure the area is clean.

By strengthening threat identification and prediction, automating and streamlining threat response, and fortifying and adapting security measures, the literature in the field of machine learning and cybersecurity advances our understanding and capacities. It advances the discipline by introducing novel approaches and offering useful case studies that illustrate their effectiveness in the real world. This helps to better protect digital infrastructures by providing both theoretical understanding and useful tools.

3. RESEARCH METHOD

In order to compare the effectiveness of Weka and Python Colab in detecting Trojan horse infections, we will use a dataset consisting of known malware and benign files. We utilized several tools, including Sandbox, NEWT Pro, PromiscDetect, ProcessExplorer, PortMon, and Wireshark. Dynamic analysis was used in this study since it is more effective, powerful, and reliable. The analysis was carried out in four stages:

- Data collection: We gathered benign and Trojan horse samples from *VirusShare.com* and VirusTotal [4], [24], [30], [39], [50], [51]. One of the largest publicly available virus repositories on the internet is called VirusShare, whereas VirusTotal serves as a repository for malware samples. We obtained 3,000 executable Trojan horses from samples made available on VirusTotal and VirusShare, which were detected by powerful antivirus systems such as Avast, F-Secure, Kaspersky, Comodo, Avira, Bitdefender, and others. Machine learning techniques will be applied on this dataset for further malware research.
- Data analysis: A controlled lab environment is necessary for Trojan horse analysis. To establish a completely controlled, segregated environment and prevent the spread of Trojan horses, the physical network link must be severed. However, without a network connection it is impossible to do dynamic analysis on Trojan horse samples. A virtual cloud environment is created using virtualization technologies, and V_{mnet} is used to link a dissimilar server (monitoring host and the attacker) to the cloud. This controlled lab architecture is comparable to that used by [2], [24], [25], [30]
- Feature extraction: This stage involves determining the significance of the dataset's current features. It retains the crucial components while eliminating the superfluous ones [52]. Feature extraction can be used to achieve higher accuracy rates.
- Testing framework: In this stage, we perform machine learning algorithms on the dataset using Weka and Python independently after feature extraction.

4. DATA ANALYSIS

Figure 1 demonstrates a controlled lab setup [2], [24] where a real device is used to host a virtualized version of the cloud, isolated from the internet. The attacker initiates the attack from outside the cloud situation, whereas the server in the cloud has monitoring tools installed to track activities such as process monitoring, file monitoring, network monitoring, and registry monitoring. Dynamic analysis and behavior monitoring through the Cuckoo Sandbox were used to assess the detection of Trojan horses in the controlled lab environment. A similar approach has been adopted in [2], [24], [39], [45]. The accuracy of detection was calculated separately using two different environments: Python Colab and the Weka data mining tool.

The dynamic and Cuckoo Sandbox analysis have been performed on all files. Each file is executed and its runtime operations are examined to determine the behavior of the virus in a newly installed operating system. The Trojan horse samples are tested in a controlled environment mode for dynamic analysis. Once the testing is complete, the Deepfreeze computer program is used to restore the controlled environment to its original uninfected state. During the dynamic analysis stage, the behavior of each injected Trojan sample is observed on the host machine. The monitoring tools used in this stage include listening ports, registry, network, random-access memory (RAM), files, transmission control protocol (TCP), processes, and dynamic-link libraries (DLLs). Any suspicious behavior detected by these tools, the sample is identified and labeled as Trojan.

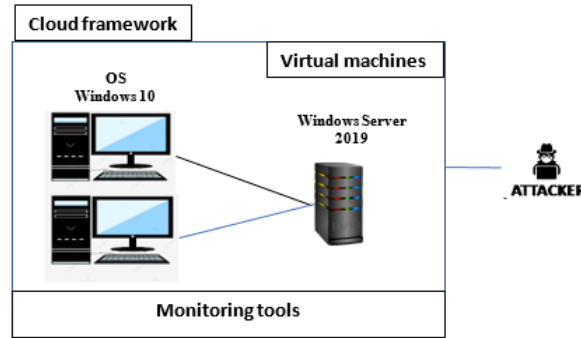


Figure 1. Controlled laboratory setup [2], [24]

4.1. Feature extraction

The process of evaluating an existing feature's relevance in a dataset is called feature extraction. The salient characteristics are kept while the irrelevant ones are removed [24]. Selecting features aids in improving accuracy rates. For this study, we have collected 3000 Trojan horse samples from VirusTotal and VirusShare. The process of feature extraction employed the Cuckoo Sandbox analysis report and dynamic analysis. Through analysis carried out in a controlled lab environment, the qualities that were recognized as properties of each Trojan horse were identified, and a relevant dataset was established. This phase is finished with the creation of an extensive output analysis report. This report contains details on the creation, modification, deletion, and access to files and registry keys as well as DLLs, RAM, and networking protocols. Nine features make up the output file, which has been translated to the suitable file format arff. The output data is encoded in this file format so that it may be used as input data in the machine learning simulation environments of Python Colab and WEKA. The trojan horse dataset was successfully classified in this study using a 10-fold cross-validation along with a number of classifiers, including NB, IBK, RF, J48, and regression.

4.2. Testing framework

The framework depicted in Figure 2 is developed using Python in the Colab environment and Weka machine learning technique, where certain machine-learning algorithms are tested and implemented. The Trojan horse dataset is classified using classifiers including 10-fold cross-validation, J48, RF, MLP, IBK, regression, NB, and others.

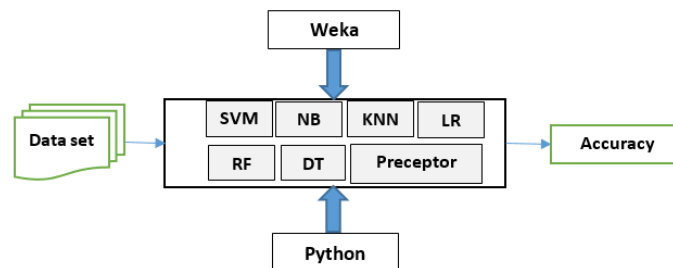


Figure 2. Framework for testing trojan attack detection

5. EXPERIMENTAL RESULTS AND DISCUSSION

In this experiment, a variety of machine-learning techniques were applied using both the Weka tool and Python Colab. Python is a widely used programming language in data science, scientific computing, and machine learning, and has an extensive library of tools for these fields [38]. Weka, on the other hand, is a Java-based tool designed specifically for data mining and includes a range of machine learning techniques [53]. Researchers have used both Weka and Python in clustering, classification, and detection studies [24], [26], [39]–[43].

For this experimentation, a 10-fold cross-validation dataset was generated using both Weka and Python. An approach that is frequently used to evaluate the error rate of a learning strategy on a particular

dataset is the 10-fold cross validation method [54], [55]. The dataset was divided into ten parts, or "folds," and each fold was utilized nine times for training and once for testing. The usage of a 10-fold cross-validation approach has two main advantages: Because it uses data as possible for both the training and testing phases, it is more accurate [24].

Various established performance metrics, including accuracy rate, F-measure, precision, and recall, have been employed to assess the classifiers' efficacy. Table 1 shows the performance metrics of various machine learning algorithms applied in the experiment using both WEKA and Python. The algorithms include J48, IBK, NB, regression, RF, SMO and MLP. From subsections 5.1 to 5.4, explains the experimental results in detail.

Table 1. Trojan horse detection results using Weka and Python various machine learning methods

Algorithm	WEKA				Python			
	Precision	Recall	F-Measure	Accuracy (%)	Precision	Recall	F-Measure	Accuracy (%)
Naïve Bayes	0.957	0.957	0.953	95.6	1	0.823	0.903	84.5
Random forest	0.957	0.957	0.953	95.6	0.958	0.977	0.968	94.3
IBK	0.957	0.957	0.953	95.6	0.962	1	0.981	96.6
Regression	0.957	0.957	0.953	95.6	0.876	1	0.934	87.6
J48	0.957	0.957	0.953	95.6	0.953	0.977	0.974	95.4
Multilayer perceptron	0.959	0.959	0.955	95.8	0.876	1	0.934	87.6
SMO	0.959	0.959	0.955	95.8	0.876	1	0.934	87.6

5.1. Precision rate results in Weka and Python

Precision provides an explanation for the percentage of genuine positive classifications in all positive findings. It is the quantity of samples that are correctly recognized and do not represent false positives. Equation (1) states that precision is calculated using the TP and FP rates. Greater precision is indicated by a higher TP.

$$\text{Precision} = \frac{TP}{(TP+FP)} \tag{1}$$

The precision findings for each of the classifiers used in this experiment are displayed in Table 1 and Figure 3. For WEKA, all the algorithms have the same precision rate value of 0.957% except for SMO and Multilayer Perceptron, which have a slightly higher precision rate value of 0.959%. However, for Python, the naïve Bayes algorithm has the highest precision rate value at 100%.

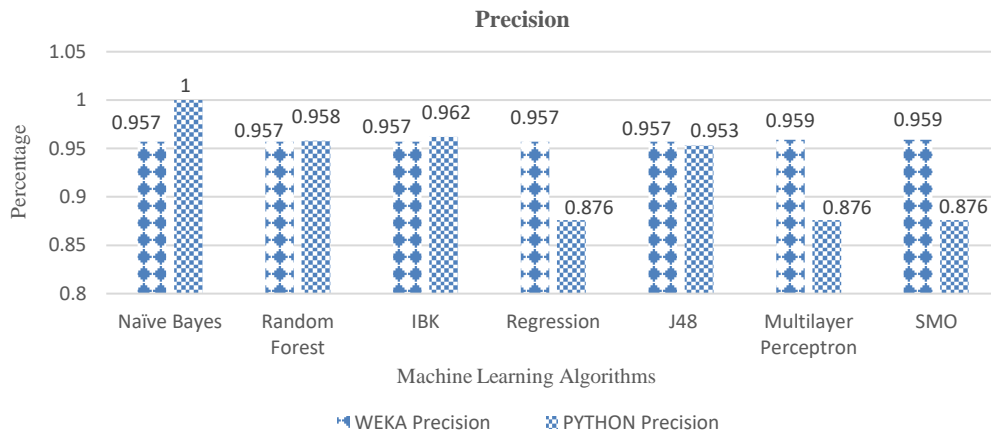


Figure 3. Precision rate of numerous classification algorithms

5.2. Recall rate results in Weka and Python

Recall is a statistic that quantifies the frequency with which a machine learning model properly selects positive samples (true positives) from all of the real positive samples in the dataset. On the other hands, truly positive (TP) samples are those from predicted clouds trojan horse samples that are correctly categorized and tagged as dangerous. Calculating recall is done using (2).

$$Recall = TPR = \frac{TP}{(TP+FN)} \tag{2}$$

From the experimental results in Table 1 and Figure 4, it is clearly seen that for WEKA, all the algorithms have the same recall rate value of 0.957%, except for MLP and SMO, which have a slightly higher recall rate value of 0.959%. For Python, IBK, regression, SMO and the multilayer perceptron algorithm have the highest recall rate value at 100%.

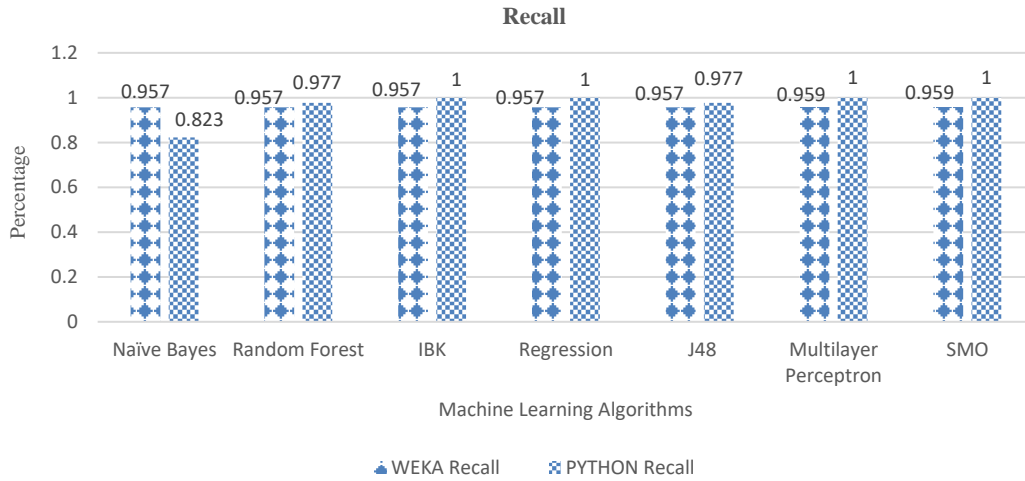


Figure 4. Recall rate of numerous classification algorithms

5.3. F-Measure rate results in Weka and Python

The F-Measure is a system performance metric that aggregates recall and precision into a single number. The calculation of the F-measure is shown in (3).

$$F - Measure = \frac{2 \times Recall \times precision}{Recall + precision} \tag{3}$$

The experimental results in Table 1 and Figure 5 show that for WEKA, all the algorithms have the same F-measure rate value of 0.953%, except for MLP and SMO, which have slightly higher F-measure rate values of 0.955%. While, for Python, IBK has the highest F-measure rate value at 0.981% and naïve Bayes algorithm has the lowest F-measure rate value at 0.903%.

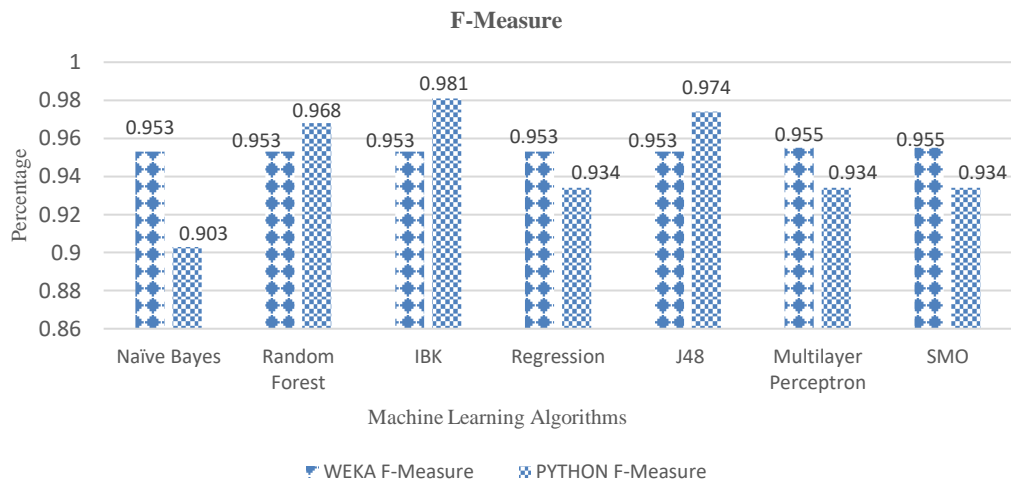


Figure 5. F-Measure rate of numerous classification algorithms

5.4. Accuracy rate results in Weka and Python

Accuracy is often referred to as correct classification. The percentage of accurate predictions is expressed by the performance statistic accuracy. Table 1 and Figure 6 show the accuracy rates for the different classifiers.

The experimental results for the various classifiers in this study have shown that for WEKA, multilayer perceptron and SMO have slightly higher accuracy rates at 95.8%. In contrast, all other classifiers accuracy rates are equal, and their accuracy rates are equal to 95.6%. While, for Python, the experimental results have shown that the IBK has the highest accuracy rate value of 96.6% and the naïve Bayes has the lowest accuracy rate value at 84.5%.

Depending on the findings in Table 1 and Figures 3 to 6, it can be concluded that for WEKA, all the algorithms have the same precision, recall, F-measure, and accuracy rate values of 95.7% except for multilayer perceptron and SMO, which have slightly higher precision, recall, and F-measure rate values at 95.9% and accuracy rate values at 95.8%. On the other hand, for Python, naïve Bayes has the highest precision rate value at 100%, while IBK, regression, SMO and multilayer perceptron, have the highest recall rate value at 100%. IBK has the highest F-measure and accuracy rate values at 98.1% and 96.6%, respectively. However, regression, multilayer perception, and SMO have the lowest precision rate value at 87.6%. Naïve Bayes has the lowest recall, F-measure, and accuracy rate values at 82.3%, 90.3%, and 84.5%, respectively.

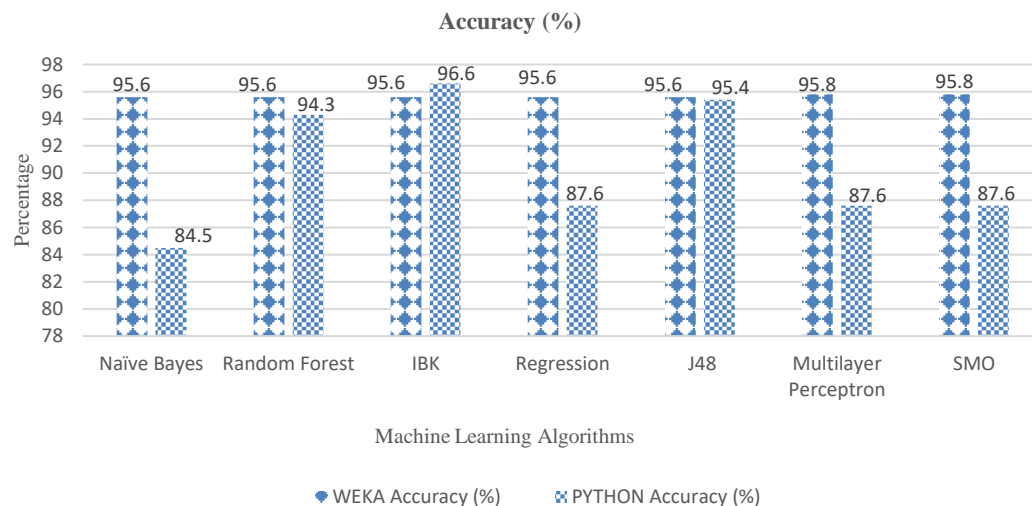


Figure 6. Accuracy rate of numerous classification algorithms

Different machine learning algorithms perform differently in Trojan horse detection. Multilayer perceptron and sequential minimal optimization classifiers have shown better results in Weka, while IBK and J48 classifiers have performed better in Python. Overall, the findings show that machine learning techniques are effective at finding Trojan horses, with high recall, precision, and F-Measure scores. The choice of algorithm and tool (WEKA or Python) may depend on the dataset, specific requirements and constraints of the application.

A comparison between Python and Weka for various machine learning algorithms in term of the execution time is shown in Table 2. We can see that Python has outperform Weka in lower for most of the algorithms. Python execution time for naïve Bayes, random forest, IBK, J48, SMO is smaller than Weka's. But the multilayer perceptron algorithm has shown more execution time in Weka. The reason for that because Weka's implementation of multilayer perceptron has complex computations than in Python. Overall, the experimental findings demonstrate that Python is better in terms of execution time for the majority of machine learning algorithms. However, we should not forget that the execution time can be affected by the hardware and software configurations of the computer systems used for the analysis.

This research employs the default parameter configurations for both Python-Colab and Weka. The Tables 3 to 9 display the parameter settings for machine learning algorithms used in this study. They show naïve Bayes, random forest, IBK, regression, J48, multilayer perceptron, and sequential minimal optimization classifiers using Python-Colab and Weka, respectively.

Table 2. Time comparison between Python and Weka

Machine learning algorithm	Python execution time (s)	Weka execution time (s)
Naïve Bayes	0.003	0.02
Random forest	0.001	0.05
IBK	0.002	0.01
Regression	0.004	0.07
J48	0.002	0.01
Multilayer perceptron	0.003	0.79
SMO	0.002	0.03

Table 3. Naive Bayes machine learning algorithm parameters in Python and Weka

Python		Weka	
Parameter	Value	Parameter	Value
priors	None	<i>UseKernelEstimator</i>	False
<i>var_smoothing</i>	1e-9	<i>UseSupervisedDiscretization</i>	False

Table 4. Random forest machine learning algorithm parameters in Python and Weka

Python		Weka	
Parameter	Value	Parameter	Value
<i>n_estimators</i>	500	<i>bagSizePercent</i>	100
<i>criterion</i>	gini	<i>maxDepth</i>	0
<i>max_depth</i>	None	<i>numFeatures</i>	0
<i>min_samples_split</i>	2	<i>seed</i>	1
<i>min_samples_leaf</i>	1	<i>numExecutionSlots</i>	1
<i>min_weight_fraction_leaf</i>	0	<i>batchSize</i>	100
<i>max_features</i>	auto	<i>debug</i>	False
<i>max_leaf_nodes</i>	None	<i>doNotCheckCapabilities</i>	False
<i>min_impurity_decrease</i>	0	<i>breakTiesRandomly</i>	False
<i>bootstrap</i>	True	<i>printClassifiers</i>	False
<i>oob_score</i>	False	<i>printTrees</i>	False
<i>n_jobs</i>	None	<i>inBagOutOfBagEvaluation</i>	False
<i>random_state</i>	1		
<i>verbose</i>	0		
<i>warm_start</i>	False		
<i>class_weight</i>	None		
<i>ccp_alpha</i>	0.0		
<i>max_samples</i>	None		

Table 5. IBK machine learning algorithm parameters in Python and Weka

Python		Weka	
Parameter	Value	Parameter	Value
<i>n_neighbors</i>	5	<i>KNN</i>	1
<i>weights</i>	uniform	<i>DistanceWeighting</i>	No distance weighting
<i>algorithm</i>	auto	<i>WindowSize</i>	0
<i>leaf_size</i>	30	<i>DistanceFunction</i>	EuclideanDistance
<i>p</i>	2	<i>CrossValidate</i>	False
<i>metric</i>	minkowski	<i>MeanSquared</i>	False
<i>metric_params</i>	None	<i>Debug</i>	False
<i>n_jobs</i>	None	<i>doNotCheckCapabilities</i>	False

Table 6. Regression machine learning algorithm parameters in Python and Weka

Python		Weka	
Parameter	Value	Parameter	Value
<i>penalty</i>	l2	<i>Ridge</i>	1.0E-8
<i>dual</i>	False	<i>MaxIts</i>	-1
<i>tol</i>	1e-4	<i>Debug</i>	False
<i>C</i>	100	<i>doNotCheckCapabilities</i>	False
<i>fit_intercept</i>	True	<i>BatchSize</i>	100
<i>intercept_scaling</i>	1	<i>NumDecimalPlaces</i>	2
<i>class_weight</i>	None		
<i>random_state</i>	1		
<i>solver</i>	lbfgs		
<i>max_iter</i>	100		
<i>multi_class</i>	ovr		
<i>verbose</i>	0		
<i>warm_start</i>	False		
<i>n_jobs</i>	None		
<i>l1_ratio</i>	None		

Table 7. J48 machine learning algorithm parameters in Python and Weka

Python		Weka	
Parameter	Value	Parameter	Value
Criterion	gini	Criterion	entropy
Splitter	best	Max depth	None
Max depth	None	Min samples split	2
Min samples split	2	Min samples leaf	1
Min samples leaf	1	CCP alpha	0.25
Min weight fraction leaf	0.0	Subtree raising	True
Max features	None	Use Laplace	False
Random state	None	Binary splits	False
Max leaf nodes	None	Save instance data	False
Min impurity decrease	0.0		
Class weight	None		
CCP alpha	0.0		

Table 8. Multilayer perceptron machine learning algorithm parameters in Python and Weka

Python		Weka	
<i>penalty</i>	None	<i>learningRate</i>	0.3
<i>alpha</i>	0.0001	<i>momentum</i>	0.2
<i>fit_intercept</i>	True	<i>hiddenLayers</i>	'a'
<i>max_iter</i>	1000	<i>trainingTime</i>	500
<i>tol</i>	1e-3	<i>validationThreshold</i>	20
<i>shuffle</i>	True	<i>seed</i>	0
<i>verbose</i>	0	<i>learningRateDecay</i>	False
<i>eta0</i>	0.1	<i>convergeEpochs</i>	0
<i>n_jobs</i>	None	<i>convergeThreshold</i>	0.001
<i>random_state</i>	1	<i>normalizeAttributes</i>	True
<i>early_stopping</i>	False	<i>normalizeClass</i>	True
<i>validation_fraction</i>	0.1	<i>decay</i>	False
<i>n_iter_no_change</i>	5	<i>reset</i>	True
<i>class_weight</i>	None	<i>nominalToBinaryFilter</i>	True
<i>warm_start</i>	False	<i>debug</i>	False

Table 9. SMO machine learning algorithm parameters in Python and Weka

Python		Weka	
<i>C</i>	1.0	<i>C</i>	1.0
<i>kernel</i>	linear	<i>kernel</i>	PolyKernel
<i>degree</i>	3	<i>filterType</i>	2
<i>gamma</i>	'scale'	<i>toleranceParameter</i>	0.001
<i>coef0</i>	0.0	<i>epsilon</i>	1.0E-12
<i>shrinking</i>	True	<i>numFolds</i>	-1
<i>probability</i>	False	<i>randomSeed</i>	1
<i>tol</i>	1e-3	<i>buildLogisticModels</i>	False
<i>cache_size</i>	200	<i>debug</i>	False
<i>class_weight</i>	None	<i>doNotCheckCapabilities</i>	False
<i>verbose</i>	False		
<i>max_iter</i>	-1		
<i>decision_function_shape</i>	'ovr'		
<i>break_ties</i>	False		
<i>random_state</i>	1		

6. CONCLUSION

This study explored the security risks and benefits associated with cloud computing and the usage of machine learning for detecting Trojan horse infections. Weka and Python Colab, two well-known machine learning tools, were tested in the study to see how well they performed at identifying Trojan horse infections. The results show that both tools are effective in detecting Trojan horse infections, However, the instrument selected will rely on the particular needs and resources offered. Furthermore, the study evaluated the computational resources required by each tool and found that Python Colab generally had faster execution times than Weka. This study helps researchers and practitioners in cybersecurity field to adopt the suitable machine learning tools to detect the malware infection effectively based on performance and accuracy detection rate using this study insights into the strengths and limitations of Weka and Python Colab for building machine learning models and detecting Trojan horse infections.

This study limited by extracting nine features from the samples. However, it is crucial to emphasize that additional study is required, such as by increasing feature extraction, in order to increase the success rate

of Trojan horse identification in the cloud computing environment. Future research can build upon this study by exploring other machine learning algorithms and tools for detecting security threats in cloud computing environments. Moreover, this research can be extended by comparing the Weka and Python Colab with tuning and utilize the features selection to enhance and compare the performance and accuracy rate.




REFERENCES

- [1] N. A. Karim *et al.*, "Performance comparison of hyper-V and KVM for cryptographic tasks in cloud computing," *Computers, Materials & Continua*, vol. 78, no. 2, pp. 2023–2045, 2024, doi: 10.32604/cmc.2023.044304.
- [2] H. M. Kanaker, M. M. Saudi, and M. F. Marhusin, "Detecting worm attacks in cloud computing environment: Proof of concept," in *2014 IEEE 5th Control and System Graduate Research Colloquium*, Aug. 2014, pp. 253–256, doi: 10.1109/ICSGRC.2014.6908732.
- [3] M. G. Avram, "Advantages and challenges of adopting cloud computing from an enterprise perspective," *Procedia Technology*, vol. 12, pp. 529–534, 2014, doi: 10.1016/j.protcy.2013.12.525.
- [4] Y. Sun, J. Zhang, Y. Xiong, and G. Zhu, "Data security and privacy in cloud computing," *International Journal of Distributed Sensor Networks*, vol. 10, no. 7, Jul. 2014, doi: 10.1155/2014/190903.
- [5] H. M. Kanaker, M. M. Saudi, and M. F. Marhusin, "A systematic analysis on worm detection in cloud-based systems," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 3, pp. 1405–1412, 2015.
- [6] S. Jones, Z. Irani, U. Sivarajah, and P. E. D. Love, "Risks and rewards of cloud computing in the UK public sector: a reflection on three organisational case studies," *Information Systems Frontiers*, vol. 21, no. 2, pp. 359–382, Apr. 2019, doi: 10.1007/s10796-017-9756-0.
- [7] V. Kundra, *Federal Cloud Computing Strategy*, U.S: CreateSpace Independent Publishing Platform, 2021. Accessed: Mar. 24, 2024. [Online]. Available: <https://www.amazon.com/Federal-Computing-Strategy-Information-Officer/dp/1477475567#>
- [8] I. Bojanova, J. Zhang, and J. Voas, "Cloud computing," *IT Professional*, vol. 15, no. 2, pp. 12–14, Mar. 2013, doi: 10.1109/MITP.2013.26.
- [9] B. Kepes, "A cautionary government cloud story--UK's G-cloud. does the 'G' stand for gone?," *FORBES*, 2015. <https://www.forbes.com/sites/benkepes/2015/01/27/a-cautionary-government-cloud-story-uks-g-cloud-does-the-g-stand-for-gone/> (accessed Mar. 24, 2024).
- [10] S. P. Amjad Hussain Bhat and D. Jena, "Machine learning approach for intrusion detection on cloud virtual machines," *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, vol. 2, no. 6, pp. 57–66, 2013.
- [11] T. Campbell, "Cloud computing security," *Practical Information Security Management*, pp. 193–204, 2016, doi: 10.1007/978-1-4842-1685-9_12.
- [12] M. Tarawneh, F. AlZyoud, Y. Sharrab, and H. Kanaker, "Secure e-health framework in cloud-based environment," in *2022 International Arab Conference on Information Technology (ACIT)*, Nov. 2022, pp. 1–5, doi: 10.1109/ACIT57182.2022.9994164.
- [13] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," *arXiv preprint arXiv:1609.01107*, 2016.
- [14] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, Jan. 2011, doi: 10.1016/j.jnca.2010.07.006.
- [15] N. A. Karim, H. Kanaker, S. Almasadeh, and J. Zarqou, "A robust user authentication technique in online examination," *International Journal of Computing*, pp. 535–542, Dec. 2021, doi: 10.47839/ijc.20.4.2441.
- [16] N. A. Karim, O. A. Khashan, H. Kanaker, W. K. Abdurraheem, M. Alshinwan, and A.-K. Al-Banna, "Online banking user authentication methods: a systematic literature review," *IEEE Access*, vol. 12, pp. 741–757, 2024, doi: 10.1109/ACCESS.2023.3346045.
- [17] N. A. Karim, H. Kanaker, W. K. Abdurraheem, M. A. Ghaith, E. Alhroob, and A. M. F. Alali, "Choosing the right MFA method for online systems: a comparative analysis," *International Journal of Data and Network Science*, vol. 8, no. 1, pp. 201–212, 2024, doi: 10.5267/j.ijdns.2023.10.003.
- [18] N. A. Karim *et al.*, "Using interface preferences as evidence of user identity: a feasibility study," *International Journal of Data and Network Science*, vol. 8, no. 1, pp. 537–548, 2024, doi: 10.5267/j.ijdns.2023.9.003.
- [19] T.-S. Chou, "Security threats on cloud computing vulnerabilities," *International Journal of Computer Science and Information Technology*, vol. 5, no. 3, pp. 79–88, Jun. 2013, doi: 10.5121/ijcsit.2013.5306.
- [20] D. Jamil and H. Zaki, "Security issues in cloud computing and countermeasures," *International Journal of Engineering Science and Technology*, vol. 3, no. 4, pp. 2672–2676, 2011.
- [21] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of Cloud computing," *The Journal of Supercomputing*, vol. 63, no. 2, pp. 561–592, Feb. 2013, doi: 10.1007/s11227-012-0831-5.
- [22] Y.-F. Liu, L.-W. Zhang, J. Liang, S. Qu, and Z.-Q. Ni, "Detecting Trojan horses based on system behavior using machine learning method," in *2010 International Conference on Machine Learning and Cybernetics*, Jul. 2010, pp. 855–860, doi: 10.1109/ICMLC.2010.5580591.
- [23] N. L. Yer Fui, A. Asmawi, and M. Hussin, "A dynamic malware detection in cloud platform," *International Journal of Difference Equations*, vol. 15, no. 2, pp. 243–258, Dec. 2020, doi: 10.37622/IJDE/15.2.2020.243-258.
- [24] H. Kanaker, N. Abdel Karim, S. A.B. Awwad, N. H.A. Ismail, J. Zraqou, and A. M. F. Al ali, "Trojan horse infection detection in cloud based environment using machine learning," *International Journal of Interactive Mobile Technologies (IJIM)*, vol. 16, no. 24, pp. 81–106, Dec. 2022, doi: 10.3991/ijim.v16i24.35763.
- [25] A. M. Abuzaid, M. M. Saudi, B. M. Taib, and Z. H. Abdullah, "An efficient Trojan horse classification (ETC)," *IJCSI International Journal of Computer Science Issues*, vol. 10, no. Issue 2, No 3, pp. 96–104, 2013.
- [26] Y. J. H. Emmanuel Gbenga Dada, Joseph Stephen Bassi and A. H. Alkali, "Performance evaluation of machine learning algorithms for detection and prevention of malware attacks," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 21, no. 3, pp. 18–27, 2019.
- [27] M. Abdelsalam, R. Krishnan, and R. Sandhu, "Online malware detection in cloud auto-scaling systems using shallow convolutional neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11559 LNCS, pp. 381–397, 2019, doi: 10.1007/978-3-030-22479-0_20.
- [28] M. Abdelsalam, R. Krishnan, Y. Huang, and R. Sandhu, "Malware detection in cloud infrastructures using convolutional neural networks," *IEEE International Conference on Cloud Computing, CLOUD*, vol. 2018-July, pp. 162–169, 2018, doi:




- 10.1109/CLOUD.2018.00028.
- [29] J. C. Kimmell, M. Abdelsalam, and M. Gupta, "Analyzing machine learning approaches for online malware detection in cloud," in *Proceedings - 2021 IEEE International Conference on Smart Computing, SMARTCOMP 2021*, 2021, pp. 189–196, doi: 10.1109/SMARTCOMP52413.2021.00046.
- [30] R. Kumar, K. Sethi, N. Prajapati, R. R. Rout, and P. Bera, "Machine learning based malware detection in cloud environment using clustering approach," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul. 2020, pp. 1–7, doi: 10.1109/ICCCNT49239.2020.9225627.
- [31] M. R. Watson, N. U. H. Shirazi, A. K. Marnierides, A. Mauthe, and D. Hutchison, "Malware detection in cloud computing infrastructures," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 192–205, 2016, doi: 10.1109/TDSC.2015.2457918.
- [32] A. Mousa, F. Alali, and H. Kanaker, "Advances in the processing and analysis of medical images using neural networks," *NeuroQuantology*, vol. 20, no. 6, 2023.
- [33] S. Joshi, H. Upadhyay, L. Lagos, N. S. Akkipeddi, and V. Guerra, "Machine learning approach for malware detection using random forest classifier on process list data structure," in *ACM International Conference Proceeding Series*, 2018, pp. 98–102, doi: 10.1145/3206098.3206113.
- [34] A. Utku, İ. A. Dođru, and M. A. Akcayol, "Decision tree based android malware detection system," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, May 2018, pp. 1–4, doi: 10.1109/SIU.2018.8404151.
- [35] C. T. Lin, N. J. Wang, H. Xiao, and C. Eckert, "Feature selection and extraction for malware classification," *Journal of Information Science and Engineering*, vol. 31, no. 3, pp. 965–992, 2015.
- [36] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in *Proceedings - International Computer Software and Applications Conference*, 2016, vol. 2, pp. 577–582, doi: 10.1109/COMPASAC.2016.151.
- [37] R. S. Pircoveanu, S. S. Hansen, T. M. T. Larsen, M. Stevanovic, J. M. Pedersen, and A. Czech, "Analysis of malware behavior: type classification using machine learning," in *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, Jun. 2015, pp. 1–7, doi: 10.1109/CyberSA.2015.7166115.
- [38] Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," *Expert Systems with Applications*, vol. 52, pp. 16–25, 2016, doi: 10.1016/j.eswa.2016.01.002.
- [39] H. Kanaker, M. M. Saudi, and N. Azman, "Evaluation of EWCDMCC cloud worm detection classification based on statistical analysis," *Advanced Science Letters*, vol. 23, no. 6, pp. 5365–5369, 2017, doi: 10.1166/asl.2017.7377.
- [40] K. Sethi, S. K. Chaudhary, B. K. Tripathy, and P. Bera, "A novel malware analysis framework for malware detection and classification using machine learning approach," in *Proceedings of the 19th International Conference on Distributed Computing and Networking*, Jan. 2018, pp. 1–4, doi: 10.1145/3154273.3154326.
- [41] S. Raschka and V. Mirjalili, "Python machine learning: Machine learning and deep learning with python. Scikit-Learn, and TensorFlow," in *Taiwan Review*, 2017, pp. 1–595.
- [42] S. Raschka, J. Patterson, and C. Nolet, "Machine learning in python: main developments and technology trends in data science, machine learning, and artificial intelligence," *Information (Switzerland)*, vol. 11, no. 4, 2020, doi: 10.3390/info11040193.
- [43] V. Koutsokostas and C. Patsakis, "Python and malware: developing stealth and evasive malware without obfuscation," in *Proceedings of the 18th International Conference on Security and Cryptography*, 2021, pp. 125–136, doi: 10.5220/0010541500002998.
- [44] A. Kumar, K. S. Kuppusamy, and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set," *Journal of King Saud University - Computer and Information Sciences*, vol. 31, no. 2, pp. 252–265, 2019, doi: 10.1016/j.jksuci.2017.01.003.
- [45] K. Sethi, R. Kumar, L. Sethi, P. Bera, and P. K. Patra, "A novel machine learning based malware detection and classification framework," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, Jun. 2019, pp. 1–4, doi: 10.1109/CyberSecPODS.2019.8885196.
- [46] T. G. Palla and S. Tayeb, "Intelligent Mirai malware detection in IoT devices," in *2021 IEEE World AI IoT Congress (AIIoT)*, May 2021, pp. 420–426, doi: 10.1109/AIIoT52608.2021.9454215.
- [47] R. Tahir, "A study on malware and malware detection techniques," *International Journal of Education and Management Engineering*, vol. 8, no. 2, pp. 20–30, Mar. 2018, doi: 10.5815/ijeme.2018.02.03.
- [48] A. G. Akintola *et al.*, "Performance analysis of machine learning methods with class imbalance problem in android malware detection," *International Journal of Interactive Mobile Technologies (IJIM)*, vol. 16, no. 10, pp. 140–162, May 2022, doi: 10.3991/ijim.v16i10.29687.
- [49] M. Ijaz, M. H. Durad, and M. Ismail, "Static and dynamic malware analysis using machine learning," in *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, Jan. 2019, pp. 687–691, doi: 10.1109/IBCAST.2019.8667136.
- [50] "virusshare," *virusshare.com*. <https://virusshare.com/about> (accessed Apr. 04, 2024).
- [51] "VirusTotal," *virustotal.com*. <https://www.virustotal.com/gui/home/upload> (accessed Apr. 04, 2024).
- [52] A. R. Muhsen, G. G. Jumaa, N. F. A. L. Bakri, and A. T. Sadiq, "Feature selection strategy for network intrusion detection system (NIDS) using meerkat clan algorithm," *International Journal of Interactive Mobile Technologies*, vol. 15, no. 16, pp. 158–171, 2021, doi: 10.3991/ijim.v15i16.24173.
- [53] M. Hall, G. Holmes, B. Pfahringer, P. Reutemann, E. Frank, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2014, Accessed: Apr. 04, 2024. [Online]. Available: <https://www.researchgate.net/publication/221900777>.
- [54] R. R. Bouckaert *et al.*, "WEKA manual for version 3-9-1," University of Waikato, 2015.
- [55] H. N. K. AL-Behadili and K. R. Ku-Mahamud, "Fuzzy unordered rule using greedy hill climbing feature selection method: an application to diabetes classification," *Journal of Information and Communication Technology*, vol. 20, no. 3, pp. 391–422, 2021, doi: 10.32890/JICT2021.20.3.5.

BIOGRAPHIES OF AUTHORS






Hasan Kanaker    received his PhD. degree in cybersecurity from USIM University, Malaysia, in 2018. Currently, he is an assistant professor at the Cybersecurity Department, Faculty of Information Technology, Isra University, Jordan. His research interests include security, intrusion detection, machine learning, malware, deep learning, malware detection, network security, user authentication, cloud computing security, and information security. He can be contacted at email: hasan.kanaker@iu.edu.jo.






Monther Tarawneh    received his PhD. in computing from the University of Sydney, Australia in 2009. Currently, he is an assistant professor at the Information Technology Department, College of Information Technology, Tafila Technical University, Jordan. His research interests include deep learning, security, eHealth, and IoT. He can be contacted at email: mtarawneh@ttu.edu.jo.






Nader Abdel Karim    received his PhD. degree in cybersecurity from UKM University, Malaysia, in 2017. Currently, he is an assistant professor at the Cybersecurity Department, College of Artificial Intelligence, Al-Balqa Applied University, Jordan. In the areas of user authentication, cyber security, human-computer interaction (HCI), and online learning, he has very strong experience. He has also participated in a number of research projects, including ones on virtual privacy techniques and preferences-based authentication. He can be contacted at email: nader.salameh@bau.edu.jo.






Adeeb Alsaaidah    received his PhD. degree in computer network from USIM University, Malaysia, in 2018. Currently, he is an assistant professor at the Network and Cybersecurity Department, Faculty of Information Technology, Al-Ahliyya Amman University, Jordan. His research interests include network performance, multimedia networks, network quality of service (QoS), the IoT, network modeling and simulation, network security, and cloud security. He can be contacted at email: A.alsaaidah@ammanu.edu.jo.






Maher Abuhamdeh    received his PhD. degree in computer information systems from the Arab Academy for Management, Banking and Financial Sciences, Jordan, in 2010. Currently, he is an assistant professor at the Computer Information Systems Department, Faculty of Information Technology, Isra University, Jordan. His research interests include computer communications (networks), software engineering, big data, artificial intelligence, data mining, and mobile learning. He can be contacted at email: maher.abuhamdeh@iu.edu.jo.






Osama Qtaish    received his PhD. degree in software engineering from Utara University, Malaysia. In 2014. Currently, he is an assistant professor at the Software Engineering Department, Faculty of Information Technology, Isra University, Jordan. His research interest includes CRM, service-oriented architecture, web service, QoS, SOC, web service composition, data mining and optimization. He can be contacted at email: osama.qtaish@iu.edu.jo.



Essam Alhroob    received his PhD. degree in artificial intelligence from UMP University, Malaysia, in 2020. Currently, he is an assistant professor at the Cybersecurity Department, Faculty of Information Technology, Isra University, Jordan. His research interests include neural networks, artificial intelligence, machine learning, pattern classification, and fuzzy neural networks. He can be contacted at email: essam.alhroob@iu.edu.jo.



Zaid Alhalhouli    received his PhD. degree in information and communication technology from UNITEN University, Malaysia, in 2015. Currently, he is an associate professor at the Data Science and Artificial Intelligence Department, Faculty of Information Technology, Isra University, Jordan. His research interests include machine learning, data mining, database, NLP, knowledge sharing, and social network. He can be contacted at email: zaid.alhalhouli@iu.edu.jo.