Machine Learning and Network Traffic to Distinguish Between Malware and Benign Applications

Laith Abualigah[1,2,3,4,5(✉)], Sayel Abualigah[6], Mothanna Almahmoud[6], Agostino Forestiero[7], Gagan Sachdeva[8], and Essam S. Hanandeh[9]

[1] Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman, Jordan
aligah.2020@gmail.com
[2] Faculty of Information Technology, Middle East University, Amman, Jordan
[3] Applied Science Research Center, Applied Science Private University, Amman, Jordan
[4] School of Computer Sciences, Universiti Sains Malaysia, George Town, Pulau Pinang, Malaysia
[5] Computer Science Department, Prince Hussein Bin Abdullah, Faculty for Information Technology, Al Al-Bayt University, Mafraq 25113, Jordan
[6] Department of Computer Information Systems, Jordan University of Science and Technology, Ar-Ramtha, Jordan
[7] Performance Computing and Networking, National Research Council of Italy, 87036 Rende, Italy
agostino.forestiero@icar.cnr.it
[8] First Abu Dhabi Bank, Abu Dhabi, UAE
[9] Department of Computer Information System, Zarqa University, Zarqa, Jordan
hanandeh@zu.edu.jo

Abstract. Virus detection software is widely used for servers, systems, and devices that seek to maintain security and reliability. Although these programs provide an excellent safety level, the traditional defense methods fail to detect new Malware. The more advanced approach relies on predicting malicious behavior with dynamic analysis of the process executed. This paper presents a new method for detecting malware using machine learning algorithms applied to data obtained from the Cuckoo sandbox. The Cuckoo sandbox isolates the file being analyzed, providing detailed dynamic analysis reports. The machine learning algorithms were compared and the most important features were identified. The results were obtained using six popular classifiers, including SVM, Random Forest, and LightGBM, and the XGBOOST algorithm had the highest accuracy, at an average of 97%. However, the research on machine learning-based malware analysis is limited in terms of computational complexity and detection accuracy.

Keywords: Machine Learning · XGBOOST · Malware · Network Traffic · Classification

C. Comito and D. Talia (Eds.): PerSOM 2022, LNICST 494, pp. 96–108, 2023.
https://doi.org/10.1007/978-3-031-31469-8_7

1 Introduction

Malware with particular types is one of the most common viruses studied in the last years in dynamic or static environments using various processing algorithms [1]. The detection of viruses such as Malware, spyware, or other complex tasks for a person or device to deal with in dynamic or static environments because the nature of these viruses is difficult to understand easily [2]. The continuous growth of Malware generates various information and security threats [3, 4].

A malware virus is a malicious program that some users create on our devices to obtain personal information, destroy programs on the devices, or obtain financial gain [5]. Many types of malware files exist, such as dynamic-link libraries (DLLs), executable files, or assembly-level instructions [6]. Therefore, it is essential to discover this type of virus, it can detect some types through the set of powers that it requires or through the behavior that it follows, but the method of detecting new types of viruses using this method may not be safe and accurate to a large extent, so the implementation of these applications. A dynamic environment in different operating systems, such as the cloud environment and the virtual machine (for example, VMware, sandbox), enables us to understand the nature of these programs and our ability to quickly and accurately detect them and identify malicious programs from them [7, 8].

Android, being one of the most widely used mobile operating systems in the world, has become a target for malware attacks due to its open-source code and the ability to install third-party applications without central control. Researchers and developers have been working on various security solutions, including static analysis, dynamic analysis, artificial intelligence, and data science to improve cybersecurity. Advancements in these fields have greatly improved the ability to predict malicious activities by using analytical models based on data. The goal of this research is to extract features for building prediction models using both static and dynamic analysis techniques.

Machine learning algorithms are widely used in detecting malicious programs, and they have proven accurate and high- speed results. In this paper, machine learning algorithms were used, and six of the most powerful classifiers were chosen to implement and build the system on them: Support vector machine (SVM), Random Forest, LightGBM, Decision Tree, k-nearest neighbors (KNN), and Extreme Gradient Boosting (XGBoost). Where the data set is entered on these algorithms and a particular model is built for each one to reach the highest Accuracy and results, the comparison between them and the most influential features are also determined.

This paper's reminder is organized as follows: Sect. 2 presents the previous related work on malware detection. Section 3 describes the proposed methodology used. Section 4 discusses the data analysis and experimental results. The conclusion of this paper is explained in Sect. 5.

2 Literature Review

Many researchers in previous work study virus detection in a dynamic environment using machine learning algorithms. This paper focuses on malware detection in a dynamic environment using machine learning algorithms.

Poudyal et al. in [6] used machine learning algorithms to detect malware from assembly level instructions and dynamic link libraries (DLLs) and determine if the sample is a ransomware virus. The dataset used in this study consisted of 302 samples of malware from various sources including Virus Total, Virus Share, and the open-source malware repository, the zoo. The algorithms used were Bayesian Network (BN), Logistic Regression, and Sequential Minimal Optimization with Linear Kernel (SMO with LK), SMO with Poly kernel (PK), J48, Random Forest (RF), AdaboostM1 with J48, and AdaboostM1 with RF. Seven evaluation metrics were used to assess the algorithms: True positive rate (TPR), false positive rate (FPR), precision, recall, f-measure, and accuracy. The results showed that AdaboostM1 with RF had the highest accuracy for assembly level instructions (ASI) and DLLs at 97.8916% and 90.8012% respectively. For both assembly level instructions and DLLs, Random Forest (RF) had the best accuracy of 97.9532%.

Niveditha et al. in [9] proposed a framework where dynamic and static malware detection techniques are efficiently combined to classify and identify Malware at day zero with High Accuracy. The framework was tested and estimated on a sample of large data files from 0.1 million files, including clean files, to 0.03. It contains a variety of malware families in 0.13 million malicious binary files. Results showed that SVM had the best Accuracy of 93.03% for detecting Malware and benign species using 10× cross-validation. The proposed framework was intended to resolve issues and concerns about identifying zero-day Malware early.

Rabadi et al. in [10] proposed a new method for detecting and classifying malware using API-based dynamic feature extraction. This approach analyzes API calls and their arguments using machine learning algorithms. Two methods were designed for this purpose: the first method treats the complete list of arguments for each API call as a single feature, while the second method treats each argument for each API call separately as a single feature. To test its accuracy, they used datasets of 7,774 benign samples and 7,105 malicious samples of ten different types of malware. The results showed that their classification module had an accuracy of 98.0253%, and the malware detection module had an accuracy of more than 99.8992%, outperforming many existing API-based malware detection techniques.

Singh et al. in [11] conducted an analysis of seven techniques that use API calls for malware detection. They developed and evaluated the effectiveness of basic parameters of simple and advanced classification algorithms to improve the binary classification of binary files into benign or malicious programs. Dynamic API calls were used in the training of classifiers. Machine learning (ML) parameter tuning was performed to improve the accuracy of binary classification of binary files into malware or benign. Basic parameters such as k-value, kernel function, tree depth, loss function, partition criteria, learning rate, and number of capabilities were evaluated using API calls. They then evaluated the tuned machine learning algorithms using 6,434 benign and 8,634 malware samples. The results showed that Random Forest produced the highest accuracy of 99.1% in binary classification.

Kumar et al. in [12] used a methodology involving the combination of Fuzzy AHP and Fuzzy TOPSIS technology to assess the impact of different malware analysis techniques on a web application perspective. This study used different versions of the University's

web application to assess the impact of many current malware analysis technologies. The study shows that reverse engineering is the most effective method for analyzing complex Malware. Zhang et al. in [13] presented machine learning algorithms to detect the particular type of malware virus from eight ransomware families (cryptolocker, cryptowall, Cryrar, locky, petya, reveton, teslacrypt, and wannacry). There are 1787 ransomware samples that the authors used to detect if the sample is ransomware or not. After collecting the dataset, they extracted the features using Term Frequency-Inverse document frequency (TF-IDF) and n-gram. Naïve Bayes, Decision Tree, random forest, K-Nearest Neighbor, and Gradient Boosting Decision Tree are machine learning algorithms. The results shown that the random forest gave the best Accuracy of 91.43% when using the value of n-gram is 3.

Singh et al. in [14] applied machine learning algorithms to detect one of the most famous viruses (Malware) in the dynamic analysis environment, called Cuckoo sandbox. They collected datasets about Malware from VirusShare (8524 malware samples) and VirusTotal (7239 malware samples) Malware dataset. The K-Nearest Neighbor, Naïve Bayes, Support Vector Machine, Decision Tree, random forest, Gradient Boosting, and AdaBoost. Are machine learning algorithms. They have shown that the AdaBoost obtained the best accuracy value of 0.9863 compared with the others.

Alzaylaee et al. in [15] proposed machine learning algorithms to detect Malware in the Android operating system. The dataset was collected from 2,444 Android applications; half of it is malware samples and 49 families of the Android malware genome project. The authors used machine learning algorithms are Support Vector Machine, Naive Bayes, Simple Logistic, Multilayer Perceptron, Partial Decision Trees, Random Forest, and J48 Decision Tree. They have shown that the RF achieved the best results in detecting malware from Malware from Android applications, around 0.931.

Kilgallon et al. in [16] proposed two techniques to predict the required time in malware detection in the sandbox and to classify the sample if it is Malware or not. The authors used machine learning algorithms in the first problem while using a novel approach in the second. The dataset used is about 3,320 malware samples for two problems. The classification algorithms applied in their experiments are Decision Tree, Random Forest, Support Vector Machine, Neural Network, and K-nearest Neighbors. The results in the classification problem show that the Neural Network gave the best Accuracy with 92.18%. While the Accuracy of prediction required time problem is equal to 90%.

Kumar et al. in [17] proposed a machine learning model to detect the malware threat in a cloud environment called a clustering model. They collected malware files (1344 files) and cleaning files (1436 files) from VirusShare and VirusTotal sites. The malware files contain many malware types like rootkit, adware, spyware, virus, Trojan, worm, backdoor, and hijacker. While the cleaning files contain PDF, PHP, and other clean files. The clustering model obtained the 95.7% as an accuracy values in detection process, where the number of clusters is 18 and the threshold value is 150. Other related machine learning methods can be found in [18–25]. Table 1 shows the summarization of these works.

Table 1. Related work

Ref	Year	Algorithm used	Dataset used	Dataset size	Results
[12]	2017	Support Vector Machine, Naive Bayes, Simple Logistic, Multilayer Perceptron, Partial Decision Trees	Android application	2,444 applications	The true positive rate of Random Forest gave = 0.931
[13]	2017	Decision Tree, Random Forest, Support Vector Machine, Neural Network, and K-nearest Neighbors	Malware detection dataset	3,320 samples	Accuracy of Neural Network in classification problem = 92.18%. While in the accuracy of prediction required time = 90%
[15]	2019	Proposed own network approach	They used four malware datasets: 1- Malgenome and Contagio 2- VirusShare site 3- Playdrone 4- Google Play store	1- 1.2k and 0.3k samples 2- 24k apps 3- 22k apps 4- 1.2k benign apps	Accuracy = 0.94
[16]	2019	• Random Forest • Logistic Regression • Naïve Bayes • Stochastic Gradient Descent • *K*-Nearest Neighbors • Support Vector Machine	They used three datasets: benign, ransomware, and malware files	300 benign files 1000 ransomware samples 900 malware samples	Accuracy of Random Forest = 99.53%

(*continued*)

Table 1. (*continued*)

Ref	Year	Algorithm used	Dataset used	Dataset size	Results
[17]	2019	Logistic regression Decision Tree Random Forest Bagging Classifier AdaBoost Classifier Tree Classifier Gradient Classifier	They used benign and malware files	800 benign files and 2200 malware files	Accuracy of Gradient Classifier = 94.64%
[5]	2019	BN, Logistic Regression, SMO with LK, SMO with PK, J48, RF, AdaboostM1 with J48, and AdaboostM1 with RF	Malware dataset from virus resources	302 samples	Accuracy of AdaboostM1 with RF = 97.8916%, 90.8012% in ASI, DLLs, respectively Accuracy of RF = 97.9532% in both ASI, DLLs datasets
[14]	2020	Clustering model	Malware dataset from VirusShare and VirusTotal sites	2780 sample files	Accuracy = 95.7%
[10]	2020	Naïve Bayes, Decision Tree, random forest, K-Nearest Neighbor, and Gradient Boosting Decision Tree	Ransomware dataset	1787 samples	Accuracy of Random forest = 91.43%
[11]	2020	K-Nearest Neighbor, Naïve Bayes, Support Vector Machine, Decision Tree, random forest, Gradient Boosting, and AdaBoost	Two sores dataset: VirusShare and VirusTotal	8524 and 7239 samples	AdaBoost obtained the best accuracy value of 0.9863

3 The Proposed Method

This paper proposes a method for detecting Malware that relies on machine learning algorithms. Figure 1 illustrates the overall process for our proposed method. After cleaning and manipulating the data, a discovery model is constructed using six machine learning algorithms to classify unknown binary samples into Malware or benign files. Then a set of evaluation procedures is used to determine and compare each model's Accuracy.

Figure 1 shows the system architecture of the proposed methodology used in this paper. The following section explains the components of the system architecture.

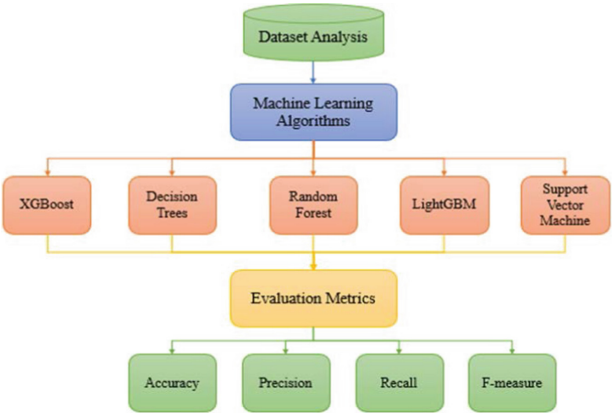

Fig. 1. System Architecture

3.1 Description of Churn Dataset

The dataset used was extracted by Urcuqui and Christian. And Navarro and Andres. (2016), where the data was obtained by creating a binary vector of the permissions used for each application analyzed (1 = used, 0 = unused). The malware/benign samples were then divided by "type". 1 malware and 0 non-malware. The authors note that an essential topic of work is building a good variety of Malware.

Data extracted from the Android Genome Project (MalGenome), an active dataset from 2012 through the end of 2015, this batch of Malware has a size of 1,260 apps, aggregated into a total of 49 families. A batch of pcap files from the integrated Droid-Collector project was used by 4705 benign and 7846 malicious applications. A particular script processed all files to extract the features, and this analysis aims to know the possibility of distinguishing between Malware and benign applications using network traffic (Table 2).

3.2 Data Analysis

One critical stage is data cleaning and processing, and we searched for existing data to find 7845 missing data in duration, avg_local_pkt_rate, and avg_remote_pkt_rate. The ideal solution in a situation like this is to delete these features. Since most of the data we have is digital, it was essential to extract the statistical analysis of these features to understand the data deeply, and we concluded that the best network features are:

- (R1): TCP packets refer to the number of packets sent and received during TCP communication.

Table 2. Features of dataset

Type	Features
Categorical Features	name
	Type
Numerical Features	dist_port_tcp
	external_ips
	vulume_bytes
	udp_packets
	tcp_urg_packet
	source_app_packets
	remote_app_packets
	source_app_bytes
	remote_app_bytes
	duracion
	avg_local_pkt_rate
	avg_remote_pkt_rate
	source_app_packets.1
	dns_query_times
	tcp_packets

- (R2): Different TCP packets represent the total number of packets that are different from TCP.
- (R3): External IP denotes the number of external addresses that the application attempted to communicate with.
- (R4): Volume of bytes refers to the number of bytes sent from the application to external sites.
- (R5): UDP packets, represent the total number of packets transmitted via UDP during communication.
- (R6): Packets of the source application, refer to the number of packets sent from the application to a remote server.
- (R7): Remote application packages, indicate the number of packages received from external sources.
- (R8): Bytes of the application source, represent the volume (in bytes) of communication between the application and the server.
- (R9): Bytes of the application remote, refer to the volume (in bytes) of data sent from the server to the emulator.
- (R10): DNS queries, refer to the number of DNS requests.

3.3 Machine Learning Methods

Extreme Gradient Boosting (XGBoost) is one of the most powerful and effective grading algorithms in the last few years and is the one that will be used to predict the class label. XGBoost performance was tested and compared with various popular classification techniques.

- First: The data set is used in two parts. The first part will be used for tuning, while the second will train and test the developed models. Stratified sampling is used because the data set is not balanced, and they must have the same share of category labels in both segment samples.
- Second: XGBoost parameters are set. This step helps greatly improve the classifier's performance and maximize it over the rest of the experiments. The second part of the dataset is used to test and test the algorithm with a 10-fold validation technique. In this way, nine folds are used to train the pattern, and 1fold is used to test the pattern. This process is repeated 10 times. Then the results are averaged.
- Third: XGBoost was trained using redundant sample data and tested on non-exhaustive test data.
- Fourth: XGBoost performance is evaluated using standard rating scales: Accuracy, recall, and F1.
- Fifthly, other classifiers such as Decision Trees (DT), Random Forest, LightGBM, and Support Vector Machine (SVM) are used, and compare their results with XGBOOST.

3.4 Evaluation Measure

To evaluate XGBoost, selectors, and classifiers in Label Prediction, the performance criteria for Accuracy, recall, precision, and F-measure are used. Binary performance evaluation criteria are calculated based on the confusion matrix. The false positive and positive current states are referred to as FP and TP, while the false negative and true negative states are abbreviated as FN and TN, as shown in Table 3.

Table 3. Confusion matrix of class label

		Production Class	
		0	1
Actual Class	0	TP	FP
	1	FN	FP

The precision is the percentage of correctly predicted student labels. It is computed using the following equation

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

Recall expresses the percentage of the correctly predicted student label. It is calculated using the equation.

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

Accuracy represents the percentage of the sum of correct predictions. The following equation gives it:

In this stage, the prediction model is built and tested with the following steps:

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

Neither recall nor Accuracy alone can describe a classifier's efficiency because good performance according to one of these indicators does not necessarily mean good performance according to the other. For this reason, the F- measure, a common combination of these two measures, is used as one metric to evaluate classifier performance. This measure is defined as the harmonic mean of recall and Accuracy and is calculated using the following equation.

$$F - meadure = \frac{2 * Precision * Recall}{Precision * Recall}$$

4 Experiments and Results

Our proposed method detects malware applications and distinguishes them from benign ones. We conducted several experiments to show the efficacy and Accuracy of the classifiers used.

The six algorithms chosen in this paper are representative and widely used for binary classification. However, in this research, we aim to classify Malware and good ware into two categories (0 and 1). We chose the following six algorithms for our experiments:

Random Forest (RF) is an ensemble learning technique that can be used for regression, classification, and feature importance analysis. It produces results by combining the outputs of multiple decision trees that are created through training processes.

K-Nearest Neighbors (KNN) is a machine learning algorithm that can be used for both classification and regression. It utilizes a pattern recognition approach to classify an element by determining the class of the K closest elements in the training dataset.

Support Vector Machine (SVM) is a widely used supervised learning model for pattern recognition and data analysis. It generates a non-probabilistic binary linear classification model that classifies new data based on which class it belongs to.

LightGBM is a scalable and efficient tree-based learning framework. It has been utilized for various reasons, including faster training speed, higher performance, lower

memory usage, improved accuracy, support for parallel and GPU-based learning, and the ability to process large-scale data.

The results showed a clear advantage for XGBOOTS compared to the rest of the classifications used, which achieved an accuracy of 96.6%, followed by LightGBM with an accuracy of 93.3%, and was the least accurate for Random Forest, achieving only 72.25% (Table 4).

Table 4. Model accuracy

Model	Accuracy
XGBoost	96.697310
LightGBM	93.309499
KNN	87.742996
Decision Tree	87.420192
SVM	75.188203
Random Forest	72.259148

Since the data was not balanced, we worked to increase the model's Accuracy by working to control the data balance. SMOTE was used to obtain balanced data and thus better Accuracy. The Accuracy of XGBOOTS has reached 99%, and the Accuracy of the rest of the classifiers has also increased, except for SVM, where it was not affected, and the results of its Accuracy as it appeared in the first experiment remained 75.1% (Table 5).

Table 5. The accuracy results after SMOTE

Model	Accuracy
XGBoost	99.061967
LightGBM	93.348493
KNN	89.906080
Decision Tree	89.226396
Random Forest	77.948591
SVM	75.642610

5 Conclusion

Malware detection is one of the critical topics that the world cares about today. The great advances in machine learning and artificial intelligence have accelerated and greatly

advanced malware detection. The types of malicious programs and applications are detected in two ways, static and dynamic. In this research, a set of machine learning algorithms are applied to data extracted from applications that have been implemented and features extracted from them in a virtual work environment, and six classifiers were used: DT, random forest, LightGBM, KNN, SVM, and XGBOOST were applied. The highest Accuracy for XGBOOST was 97%, followed by LightGBM at 93.309499%, then KNN at 87.742996%, and the lowest was Random Forest at 72.259148%. To increase the Accuracy of the classifier, the data was converted into balanced data because it was not unbalanced, and it was re-model to increase the Accuracy of the data.

References

1. Ye, Y., Li, T., Adjeroh, D., Iyengar, S.S.: A survey on malware detection using data mining techniques. ACM Comput. Surv. **50**(3), 1–40 (2017)
2. Jerlin, M.A., Marimuthu, K.: A new malware detection system using machine learning techniques for API call sequences. J. Appl. Secur. Res. **13**(1), 45–62 (2018)
3. Biondi, F., Given-Wilson, T., Legay, A., Puodzius, C., Quilbeuf, J.: Tutorial: an overview of malware detection and evasion techniques. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11244, pp. 565–586. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03418-4_34
4. Poudyal, S., Subedi, K.P., Dasgupta, D.: A framework for analyzing ransomware using machine learning. In: Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence SSCI, January 2018, pp. 1692–1699 (2019)
5. Vurdelja, I., Blažić, I., Drašković, D., Nikolić, B.: Detection of Linux Malware Using System Tracers – An Overview of Solutions, pp. 1–6 (2020)
6. Niveditha, V.R., Ananthan, T.V., Amudha, S., Sam, D., Srinidhi, S.: Detect and classify zero day malware efficiently in big data platform. Int. J. Adv. Sci. Technol. **29**(4) Special Issue, 1947–1954 (2020)
7. Rabadi, D., Teo, S.G.: Advanced windows methods on malware detection and classification, pp. 54–68 (2020)
8. Singh, J., Singh, J.: Assessment of supervised machine learning algorithms using dynamic API calls for malware detection. Int. J. Comput. Appl. 1–8 (2020)
9. Kumar, R., Alenezi, M., Ansari, M., Gupta, B., Agrawal, A., Khan, R.: Evaluating the impact of malware analysis techniques for securing web applications through a decision-making framework under fuzzy environment. Int. J. Intell. Eng. Syst. **13**(6), 94–109 (2020)
10. Zhang, H., Xiao, X., Mercaldo, F., Ni, S., Martinelli, F., Sangaiah, A.K.: Classification of ransomware families with machine learning based on N-gram of opcodes. Future Gener. Comput. Syst. **90**, 211–221 (2019)
11. Singh, J., Singh, J.: Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms. Inf. Softw. Technol. **121**, 106273 (2020)
12. Alzaylaee, M.K., Yerima, S.Y., Sezer, S.: Emulator vs real phone: android malware detection using machine learning. In: IWSPA 2017 – Proceedings of the 3rd ACM International Workshop on Security and Privacy Analytics co-located with CODASPY 2017, pp. 65–72 (2017)
13. Kilgallon, S., De La Rosa, L., Cavazos, J.: Improving the effectiveness and efficiency of dynamic malware analysis with machine learning. In: Proceedings of the - 2017 Resilience Week, RWS 2017, pp. 30–36 (2017)

14. Kumar, R., Sethi, K., Prajapati, N., Rout, R.R., Bera, P.: Machine learning based malware detection in cloud environment using clustering approach. In: 2020 11th International Conference on Computing, Communication and Networking Technologies ICCCNT 2020 (2020)
15. Krüger, F.: Activity, context, and plan recognition with computational causal behaviour models. ResearchGate (2018)
16. Al-Shatnwai, A.M., Faris, M.: Predicting customer retention using XGBoost and balancing methods. Int. J. Adv. Comput. Sci. Appl. **11**(7), 704–712 (2020)
17. Vafeiadis, T., Diamantaras, K.I., Sarigiannidis, G., Chatzisavvas, K.C.: A comparison of machine learning techniques for customer churn prediction. Simul. Model. Pract. Theor. **55**, 1–9 (2015)
18. Gul, F., et al.: A centralized strategy for multi-agent exploration. IEEE Access **10**, 126871–126884 (2022)
19. Abualigah, L., Elaziz, M.A., Khodadadi, N., Forestiero, A., Jia, H., Gandomi, A.H. Aquila optimizer based pso swarm intelligence for IoT task scheduling application in cloud computing. In: Houssein, E.H., Abd Elaziz, M., Oliva, D., Abualigah, L. (eds.) Integrating Meta-Heuristics and Machine Learning for Real-World Optimization Problems. Studies in Computational Intelligence, vol. 1038, pp. 481–497. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-99079-4_19
20. Abualigah, L., Forestiero, A., Elaziz, M.A.: Bio-inspired agents for a distributed NLP-based clustering in smart environments. In: Abraham, A., et al. (eds.) SoCPaR 2021. LNNS, vol. 417, pp. 678–687. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-96302-6_64
21. Alzu'bi, D., et al.: Kidney tumor detection and classification based on deep learning approaches: a new dataset in CT scans. J. Healthc. Eng. (2022)
22. Khazalah, A., et al.: Image processing identification for sapodilla using convolution neural network (cnn) and transfer learning techniques. In: Abualigah, L. (eds.) Classification Applications with Deep Learning and Machine Learning Technologies. Studies in Computational Intelligence, vol. 1071, pp. 107–127. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-17576-3_5
23. Melhem, M.K.B., Abualigah, L., Zitar, R.A., Hussien, A.G., Oliva, D.: Comparative study on Arabic text classification: challenges and opportunities. In: Abualigah, L. (eds.) Classification Applications with Deep Learning and Machine Learning Technologies. Studies in Computational Intelligence, vol. 1071, pp. 217–224. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-17576-3_10
24. Anuar, N.A., et al.: Rambutan image classification using various deep learning approaches. In: Abualigah, L. (eds.) Classification Applications with Deep Learning and Machine Learning Technologies. Studies in Computational Intelligence, vol. 1071, pp. 23–43. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-17576-3_2
25. Ke, C., et al.: Mango varieties classification-based optimization with transfer learning and deep learning approaches. In: Abualigah, L. (eds.) Classification Applications with Deep Learning and Machine Learning Technologies. Studies in Computational Intelligence, vol. 1071, pp. 45–65. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-17576-3_3