A METHOD FOR DETECTING MALWARE USING MACHINE LEARNING AND DEEP LEARNING

Vivek Kumar Anand*[1], Dr. Anirban Das[2], Dr. Sanjay Kumar Bishwas[3]
Assistant Professor[1], NIIT University, Neemrana-301705*
Assistant Professor[2], NIIT University, Neemrana-301705
Associate Professor, NIIT University, Neemrana - 301705
Mail id: vivekkanand7@gmail.com[1], anirbanfuture@gmail.com[2]

Abstract
This research presents a novel approach for enhancing the incorporation of machine learning to identify malware learning and deep learning techniques. The escalating sophistication of malware poses a significant challenge to traditional detection methods, necessitating the exploration of advanced technologies. Leveraging the power of machine learning algorithms, the proposed method analyzes intricate patterns and features in large datasets to identify potential malicious activities. Furthermore, the incorporation of deep learning models, particularly neural networks, enhances the system's ability to discern subtle and evolving characteristics of malware. The research contributes to the ongoing efforts in cybersecurity by offering a robust and adaptive solution that demonstrates improved accuracy in identifying diverse forms of malware. The findings of the experiment demonstrate how well the suggested strategy works to identify dangerous entities, which supports the idea that combining machine learning and deep learning might strengthen cybersecurity measures.

Keywords: malware detection, machine learning, deep learning, cybersecurity, neural networks, pattern analysis, feature extraction.

I. INTRODUCTION
These days, a person's mobile devices are an integral part of their life. Global smartphone use is projected to be 6.4 billion, and Statista predicts that number will increase by a number of billion customers over the next years.(Müller, 2019). The largest app store in the world, Google Play Store, is expected to have 3.48 million applications accessible by the first quarter of 2021.Most everyday actions, such as online shopping, bill payment, and mobile banking, are performed using mobile apps. Credit and debit card numbers, ATM PINs, and other sensitive information make identity theft more likely to happen. Examples include tricking people into giving up their PINs by brute force, tampering with accounts, and fraudulently rerouting tracks to mobile money providers. Future cellphones will be able to run a wide range of programmes, including powered by AI medical care, portable edges technology, including innovative industrial apps.

As a result, they need to be equipped with the most cutting-edge, high-security features on the market. Upon installation, every programme requests a list of permissions from the user. If granted the appropriate permissions, a user may infer the behaviour of any programme. Users are alerted to possible danger and may exploit abnormal application activity more easily when crucial permissions for capabilities and anticipated demands for permission are identified. This is how the permission-based method notifies the user prior to installation. Before allowing the programme to use their mobile device, the user is given the opportunity to weigh the risks. Cybercriminals produce 529,48 new families of malware year, according to Statista. Therefore, it's critical to recognise malware on cellphones. Malware may not download if it is found during the setup process. We want a technique that can quickly and accurately identify flawed code in order to thwart this massive onslaught. Scaling the detection for multiple uses is a difficult undertaking, however. is a serious problem that requires immediate attention? This study offers an efficient malware attack detection technique for (MADNET) for malware classification and detection as a defense against malware for Android. The following is the main achievement of MAD-NET.:(Arora et al., 2017).

1.1 ANDROID MALWARE CATEGORIES AND FAMILIES

Mobile malware is any spyware, including Android malware, that attempts to damage the target cellphone by carrying out an illegal action. Malware may be divided into smaller groups based on characteristics that distinguish each category..(Poornima & Mahalakshmi, 2024). Malware for Android is growing, much like humans. Under each group there are several families of viruses. The enormous danger presented by, which is the root cause of many internet security problems, is an unsolved subject for scholars and cybersecurity specialists. This danger can only be removed by early identification and remediation of malicious samples. Comprehending the many categories and variants of Android malware is vital in order to do this. The list of Mobile malwares and their families is shown in the summary below.

1. File Infector: Malware that has been concealed inside an APK file is known as a file attack. The data necessary to use the Android Package Kit, or APK, is included in the programme. APK files are used for downloading malicious files. When APK files are set up, malware is used. All Android apps, include editors, video games, as well as navigational aids, are included in the APK file.

2. Software: Riskware is software that raises the possibility of security flaws in the system. Although it is a genuine software, its main objective is to monitor users' internet activities and send people to dubious online sites. It is often referred to as malware, and the security of the device is impacted by how well it works. Malware belonging to the following categories is often seentriada, skymobi, deng, jiagu, smspay, smsreg, tordow, mobilepay, wificrack, badpac, and skymobi.

3. Ransomware: This kind of virus encodes computer files and folders, making them inaccessible to users. To unlock the data, the code must be cracked. The most often seen strains of ransomware consist of lock screen, slocker, jisut, koler, congur, masnu, fusob, and smsspy.

4. Trojan: This malware poses as reliable applications. They get data from the device and search the boot mechanism. The gugi, hqwar, obtes, gluper, lotoor, rootnik, guerrilla, and hypay Among the most well-known Trojan families are the Troy clans..(Rani & Ojha, 2020).

5. Adware: Adware is harmful software with an advertising display. Customers are exposed to unwanted adverts on their displays as a result of this malicious software, especially while they are using internet services. The adware entices user's expensive products. Every time a user clicks on one of the advertisements, the maker of this annoying programme gets paid.

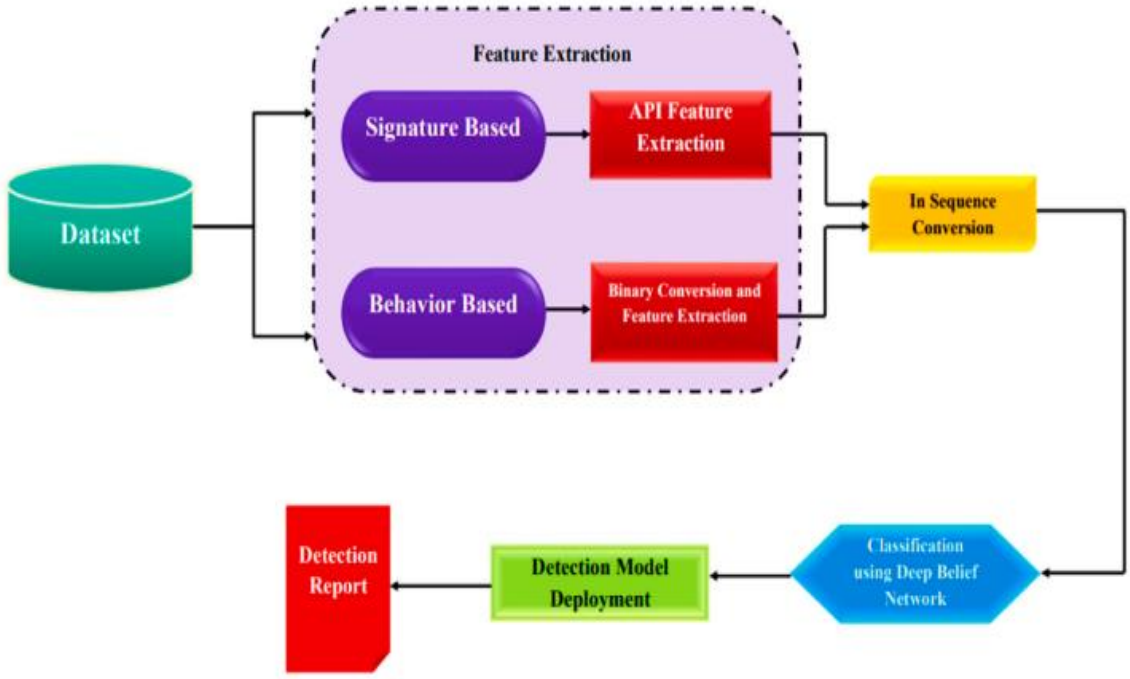


Fig. 1. Malware classification and detection approach.

6.Backdoor: Through the back doors, smartphones may be accessed in discreetly. Put differently, backdoors let attackers to bypass authentication and get extra rights, so giving them unrestricted access to devices. With the use of backdoors, one may remotely attack a target device without needing to be there in person. The families of Android Trojan horses that are often seen involve Droidkung.fu, Mobby, Kapuser, Hidad, Dendroid, Levida, Fobus, Moavt, androrat, kmin, and Pyls. (Kessedjian et al., 2010).

7. Scareware: Scareware a tactic used to trick people into downloading or buying dangerous software by instilling fear in them. Three prominent providers of malware are Fakeapp, Mobwin, & Avpass.

8. Spyware: Spyware If spyware is installed on a machine; it might be able to steal personal data. Companies, other parties, or adverts may be able to access the data that spyware collects. Afterwards, this information is misused. Spy note, qqspy, spy dealer, smithies, spy agent, smszombie, and smforw are examples of a typical spyware family.

9. PUA: Potentially unwanted apps, or PUAs, are legitimate, free software that include PUA. They are sometimes known as potentially undesirable programmes, or PUPs. For Android mobile devices, some of the most well-known PUA malware families include Apptrack, Secapk, Wiyun, Youmi, Scamapp, Utchi, Cauly, and Umpay..(Allison et al., 2016).

II LITERATURE REVIEW

R. J. Mangialardo et.al (2015) This study addresses the challenges in malware analysis by proposing a unified approach that combines both static and dynamic techniques(Mangialardo & Duarte, 2015). By merging these methods, the research aims to mitigate evasion techniques used by malicious codes, enhancing the overall effectiveness of identification and classification. The integration of C5.0 and Random Forest machine learning algorithms within the FAMA framework demonstrates promising results, achieving an impressive 95.75% accuracy for binary classification and 93.02% for multiple categorization. Notably, the unified analysis consistently outperforms isolated static and dynamic analyses, showcasing its potential as a robust solution for comprehensive malware detection. **Routa Moussaileb et.al (2018)** This paper addresses the persistent threat of ransomware, highlighting its evolution from Reveton's attack in 2012 to more recent incidents like WannaCry and Petya(Moussaileb et al., 2018). Introducing a novel approach, the study presents a graph-based countermeasure for ransomware detection, diverging from conventional metrics such as Shannon's entropy. Unlike existing methods, this mechanism relies on per-thread file system traversal, proving effective in identifying malicious behaviors. The research stands out as the first to explore this specific area, conducting experiments with over 700 active ransomware examples in a bare-metal sandbox environment, achieving accurate detection and providing a promising defense against evolving ransomware threats**. S. Poornima et.al (2023)** Cyber threats have escalated in response to the growing popularity of mobile devices, especially malware assaults targeting the Android platform(Poornima et al., 2023). A unique Malware Attack Detection system called MAD-NET is suggested as a solution to this problem. The method combines feature extraction and data classification into signature-based and behavior-based classes using the CICAndMal2017 datasets. When it comes to classification, When compared Generative Adversarial Networks and Long Short-Term Memory, a type Deep Belief Networks exhibit superior performance. Networks with an astounding 99.83% accuracy rate, MAD-NET proves its effectiveness in identifying and thwarting malware attacks on Android smartphones. **Ahmed R. Nasser et.al (2023)** Given the broad usage and open-source nature of the Android operating system, this article tackles the increasing difficulty of identifying malware in the system. With two primary detection models—deep Autoencoders for dynamic analysis and CNN-BiLSTM for static analysis—the suggested DL-AMDet makes use of a deep learning architecture. DL-AMDet's efficacy is shown by its remarkable 99.935% accuracy in combined static and dynamic analysis, which is obtained via the evaluation of its performance using two datasets. The research emphasises CNN-BiLSTM and Deep Autoencoders' importance in surpassing current methods and their usefulness in improving Android malware detection capabilities(Nasser et al., 2024).

Alejandro Guerra-Manzanares(2023) Despite the apparent success in detection, this paper challenges the notion that the problem is solved. Through an extensive review of existing literature, it identifies five unresolved challenges, ranging from methodological flaws to dataset limitations. The research highlights the misconception that hinders further exploration in this field. By exposing these overlooked challenges, the paper motivates future research directions, emphasizing the need for a more nuanced approach to achieve effective and long-term solutions in the Android malware detection domain.

This comprehensive analysis encourages a reevaluation of the current state of the art and prompts further investigation into these persisting issues(Guerra-manzanares et al., 2023)**.** **Muhammad Azeem et.al (2023)** Because of the (IoT) and ongoing technical breakthroughs, the Internet has become essential to communication and information in the digital era. However, cybersecurity faces difficulties due to the growing danger of malware. This research addresses malware identification and classification using state-of-the-art machine learning techniques, such as K-Nearest Neighbours, Extra Tree, Multilayer Perceptron. For the best feature selection, the study uses Term Frequency-Inverse Document Frequency and feature encoding using the UNSWNB15 dataset(Azeem et al., 2023). The research reveals that Random Forest is the most accurate model examined (97.68%), highlighting its effectiveness in improving internet security. **Pascal Maniriho et.al (2023)** Here is a new deep learning-based framework for Windows malware detection called API-MalDetect. With a hybrid feature extractor that combines CNNs and BiGRU, as well as an NLP-based encoder for API requests, the framework is very effective at detecting malware assaults that go undetected while causing the least amount of ROC across a range of datasets, API-MalDetect outperforms current methods by addressing temporal and spatial biases during training and testing. The technology not only efficiently separates malicious activity from benign activity, but it also identifies critical API calls for cybersecurity professionals(Maniriho et al., 2022). Furthermore, the dataset that is made available improves joint research projects in the field of cybersecurity**.** **Wadha Al-Khater et.al (2023)** Addressing the rising concern of malware detection and cybersecurity breaches, this study emphasizes the limitations of conventional antivirus software in identifying novel malware swiftly. The proposed solution involves countering imbalanced and insufficient malware datasets using the (FABEMD) technique. Employing 3D architectures, namely 3D VGG-16 and 3D Resnet-18, on the Malimg and MaleVis datasets, the experiment achieves impressive accuracy rates. Notablythe Res net-18 infrastructure demonstrates its effectiveness in identifying unknown malware in the early phases of implementation, with excellent performance of 99.64% and 99.46% for the Malimg and MaleVis datasets, accordingly. (Al-Khater & Al-Madeed, 2024)**.**

Elliot Mbunge et.al (2023) This review addresses the growing threat of cyberattacks on Android applications due to the widespread use of smartphones. Despite the surge in attacks, the application of (DL) models for detecting emerging malware in Android remains underexplored. Utilizing the PRISMA guidelines, the study identifies convolutional neural networks, gated recurrent neural networks, and other DL models as prominent in malware detection. While DL models prove effective in real-time detection, challenges persist in monitoring evolving malware behavior. The study underscores the importance of user training, sharing updated malware datasets, and pre-download detection to enhance Android smartphone security(Mbunge & Batani, 2023). **Mumtaz Ahmed et.al (2022)** This study highlights the crucial significance of machine learning in network security research and tackles the everyday creation of new variants and the dissemination of malware for criminal reasons. By modelling malware fingerprints as 2D pictures and using deep learning algorithms for classification on the BIG15 dataset, the research offers a fresh approach. Long Short Term Memory, Comparative comparison examples include Artificial Neural Network, , Transfer Learning on CNN, and Logistic Regression.

Classification accuracy on the examination dataset was 98.76%, while on the training dataset, it was an astounding 99.6%. transfer learning approach using InceptionV3 performs better than the competition, demonstrating its effectiveness in malware classification(Mumtaz et al., 2023)**. Hani AlOmari et.al (2023)** In light of the growing danger of malware for Android mobile devices, this article discusses the difficulties in developing effective detection systems. It assesses how well different machine learning algorithms perform and uses methods like Principal Component Analysis, feature normalisation, and synthetic minority oversampling to increase accuracy. A Light Gradient Boosting Model is included in the research to detect and categorise Android malware into five groups. By making use of a large and up-to-date dataset of 11,598 APKs that were obtained from various sources and made available by the Canadian Institute of Cybersecurity, the research advances efforts to address the persistent unresolved issue of Android malware categorization(AlOmari et al., 2023). **Saddam Hussain Khan et.al (2023)** This article discusses security risks in the context of the Internet of Things (IoT), highlighting the need of early identification to protect real-time devices managed by open-source Android smartphones. In order to identify complex malware assaults, the proposed (DSBEL) framework combines ensemble learning with a new (SB-BR-STM) convolutional (CNN). With the use of boosting methods and multi-path dilated convolutional processes, the SB-BR-STM CNN performs well. By demonstrating its remarkable accuracy (98.50%), F1-Score (97.12%), MCC (91.91%), Recall (95.97%), and Precision (98.42%), DSBEL's evaluation on the IOT_Malware dataset highlights its resilience and efficacy in timely malware identification for improved network security(Khan et al., 2023).

Ali Muzaffar et.al (2022) With approximately 70% of mobile users using Android devices, the Android operating system faces a significant threat from malware attacks. Traditional signature-based detection struggles with the vast user and application diversity. This paper addresses the challenge by critically reviewing past research utilizing. Examining supervised, unsupervised, deep learning, and online the review categorizes them based on their utilization of static, dynamic, or hybrid features. Machine learning emerges as a promising solution, capable of addressing the evolving nature of Android malware and offering resilience against zero-day attacks without relying on predefined malicious signatures(Ali et al., 2022). **Hamid Bostani et.al (2023)** Researchers have long studied evasion attack vulnerabilities such assaults in actual situations has been questioned. This work presents Evade Droid, an adversarial problem-space approach intended to successfully evade black-box Android malware detection. Evade Droid builds transformations from benign donors using an n-gram-based method, obtaining high evasion rates (80%–95%) against different detectors with few queries. Notably, this method challenges conventional wisdom in adversarial attacks on Android malware detectors by maintaining stealthiness against well-known commercial antiviruses, proving its viability and efficacy in real-world scenarios where attackers have little knowledge of target classifiers(Bostani & Moonsamy, 2024).

III. METHODOLOGY

PROPOSED SYSTEM ARCHITECTURE

The following are the main elements of the proposed malware classification and detection scheme: (ifeatures extraction; (ii) feature refinement/selection; (iv) classification; (vi) pre-processing; and (iii) detection. Fig. depicts the recommended method's structure.

Fig. 2 Architecture of the proposed malware detection system
Source: (Chowdhury et al., 2018)

PRE- PROCESSING

The files that have been obtained are binary code-containing raw executable archives that are kept in the file system. To meet our demands, they went through preparation. First, we run executables on a virtual machine (VM) that is safe. We utilise the PEid utility for extracting the encoded programmes.

FEATURE EXTRACTION

Static and dynamic analysis of executable files is the method used to extract features. In this study, we extract two kinds of characteristics from the malicious and cleaning software's downloadable files: N-gram's functionality and connections to Microsoft API(Chowdhury et al., 2018).

N-gram Features

Programme file segments with a length of n bytes are the substring sequences with n-gram characteristics. One advantage of the n-gram approach is the capacity to express the amount in words of n-gram lengths. We empirically discover that the best accurate results are obtained with n-grams of size 5, where each sequence has precisely 5 bytes. Using the n-gram feature acquisition approach, we are able to get the n-gram properties.

Windows API Calls

Malware behaviour may be revealed via API call information. The API list may be extracted thanks to the PE format of the executable files. A program's resource allocation and its handling by the operating system are described in the Portable Executable (PE) header. We disassemble the binary file using Interactive Disassembler Pro (IDA Pro), the most reliable disassembly tool, in order to inspect and take out the calls made by the Windows API. Any kind of file, playable and non-executable (such as ELF, EXE, PE, and so on), may be disassembled using IDA Pro. It provides user-defined plugin hooks and automatically detects API calls for different compilers, resulting in an extremely powerful solution with adjustable management and evaluation levels.(Syeda, 2024).

To investigate the relevant application area, IDA Pro imports the selected file into memory as well as builds an IDA database. IDA Pro disassembles and searches through the file's binary to aggregate the many IDA database files into a single IDB file (.idb). By providing its internal resources via Users may create plugins for IDA Pro to use by using its API. The disassembling components are launched by the Ida python system, which is used to construct a.idb dataset. To enhance binaries evaluation, use.idb systems may be exported into MySQL databases using the ida2sql plugin. (.db).

With each binary executable, the ida2sql plugin creates sixteen tables (PI system calls along with the length (beginning and ending locations) of any function names that are difficult to understand. Every operation code (OP), together with its addresses and block discusses, are included in the directions table.(Tian, 2011).

FEATURE SELECTION/REFINEMENT

Following the n-gram feature extraction process, the chosen, and the top feature is evaluated by calculating the precision of a classifier in relation to the number of attributes chosen using various methods for selecting features. Principal Components Analysis (PCA) is the method used in this study to choose features. Because the PCA may reduce complexity, it is utilised to speed up operations. Its foundation is the discovery of just a few of orthogonal linear connections between the initial factors with the highest variance, which reduces a large number of factors into a fewer amount of uncorrelated variables. To improve the efficacy of detection and extraction of relevant API calls for every malware category, we further use the Class-wise document repetition (DCFS) dependent choice of features technique. identification.

MALWARE CLASSIFICATION AND DETECTION

Training and testing are the two phases of the procedure for categorization. The computer is given a set of benign and harmful files to work with throughout the training stage. Classifiers may be trained by an approach of training. The classification tagged data sets. In the evaluation phase, a classifier receives an array of fresh harmful and benign files to be identified as either malware or clean ware. We provide a novel hybrid malware classification framework in this study, shown in Fig. 2, that combines a multilayer perceptron (MLP) neural network with a binary associative memory (BAM). In order to speed up and improve the efficiency of categorization, the BAM reduces the feature matrix's dimensions. To identify and classify malware, the MLP with backpropagation algorithm is trained using a selection of features. The BAM networks' input layer receives the chosen malware dataset features, which provide reduced features as an output. The output of the BAM networks is then fed into the MLP neural network. There are exactly as many malwares in the dataset to be classified as there are nodes in the MLP output layer. This experiment has fifty thousand epochs with a minimal error margin of 0.002.

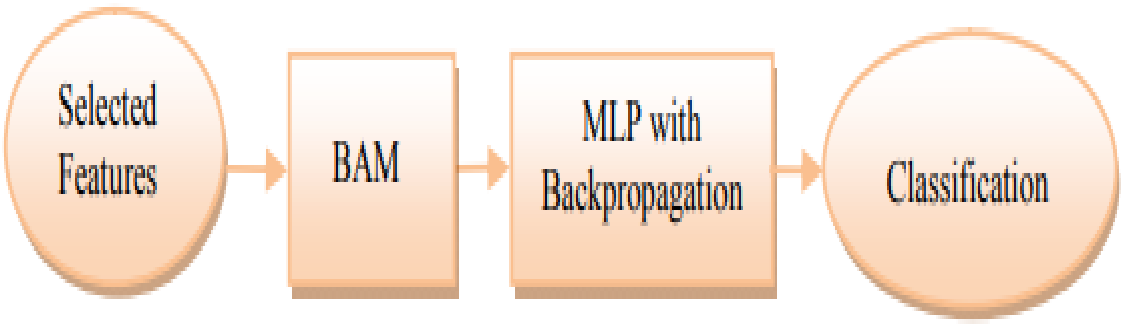


Fig. 2: The proposed hybrid classification approach.

IV DATA SET

This dataset includes 942 examples of good software (good ware) and 582 instances of ransomware, for a total of 1524 samples from dynamic analysis. By late February 2016, the dataset was obtained and examined using Cuckoo Sandbox. The publication has further information about the dataset (see below).

The relationship between the SHA1 and MD5 of the programme under analysis (both ransomware and good ware) and the local IDS that we utilise in our dataset is included in the file IDS.txt. The following is a description of the header in that file:

- ID: A local identifier used in our dataset.
- SHA1: The software's hash identity.
- MD5: This is the software's MD5 hash.
- Ransomware: 1 for ransomware, 0 for good ware.
- Ransomware Family: numeric identification for the ransomware family (same coding as described above).

Performance Measure:

The performance measures for the Logistic Regression and Random Forest Classifier models are crucial indicators of their effectiveness in predicting classes for the given dataset. For Logistic Regression, with a precision of 0.84 for class 0, it accurately predicts class 0 instances 84% of the time. With a recall of 0.94, it can accurately identify 94% of real class 0 instances. Class 0's F1-score, which is a harmonic mean of recall and accuracy, is 0.89. Class 1 has comparable metrics: F1-score of 0.83, recall of 0.77, and accuracy of 0.91. The percentage of accurately identified occurrences out of the total is shown by the overall accuracy of 0.87 for Logistic Regression. The macro average F1-score is 0.86, giving equal weight to each class, while the weighted average F1-score, at 0.87, considers class imbalances. Comparatively, the Random Forest Classifier demonstrates exceptional performance. It achieves a precision of 1.00 for both classes, indicating perfect predictions without false positives. The recall for class 0 is 1.00, implying it identifies all actual class 0 instances. Class 1 recall is still very good, although it is significantly lower at 0.99. The F1-scores for both classes are 1.00, showcasing a harmony between recollection and precision. The precision of 1.00 indicates flawless classification across the dataset. The macro and weighted average F1-scores are also 1.00, underlining the model's consistent and high-quality predictions. These results highlight the Random Forest Classifier's robust performance, making it a strong candidate for this classification task.

Algorithms Used:

Two potent algorithms are used in the machine learning and deep learning approach to malware detection: Random Forest Classifier and Logistic Regression. Because it excels at binary classification, Logistic Regression is a great tool for separating dangerous software from benign software. With parameters like precision, recall, and F1-score—all important for accurate malware detection—it offers exact predictions. However, the Random Forest Classifier, with its ensemble of decision trees, is particularly good at handling complicated datasets and provides very good accuracy. By combining their powers to examine file properties and behavior patterns, these algorithms create an effective weaponry against malware that strengthens systems against possible cyberattacks.

Novelty

- Data Import: You start by importing necessary libraries like pandas and numpy for data manipulation and handling.
- Data Loading: You load a CSV file named "data_file.csv" into a pandas Data Frame df.
- Data Preprocessing: You preprocess the data by dropping unnecessary columns ('md5Hash' and 'Filename') from the Data Frame using the drop () method.
- Data Splitting: You split the dataset into features (X) and the target variable (y). The target variable, 'Benign', seems to indicate whether the files are benign or not.
- Train-Test Split: You further split the data into training and testing sets using train_test_split () from scikit-learn.
- Feature Scaling: You standardize the features using Standard Scaler () to ensure that each feature contributes equally to the distance computations in machine learning models.
- Model Selection and Training: You choose logistic regression as your classification model, instantiate it using Logistic Regression (), and then fit the model to the training data using fit ().

System Overview

Using the processed feature sets as inputs, machine learning algorithms including neural networks classify the unknown malware (from the test dataset) to form one of the recognised malware families. Our DNN model is constructed using the Keras framework from Tensor Flow. To speed up the training process, we employed the CUDA 9 platform and NVIDIA's TITAN V GPU in our setup.

Table 1 Evaluation metrics for different machine learning algorithms

Algorithms	precision	recall	f1-score
Logistic Regression	0.84	0.94	0.89
	0.91	0.77	0.83
Random Forest Classifier	1.00	1.00	1.00
	1.00	0.99	0.99

The table presents evaluation metrics for two machine learning algorithms: Logistic Regression and Random Forest Classifier. For Logistic Regression, the precision, recall, and F1-score are 0.84, 0.94, and 0.89 respectively in the first row, and 0.91, 0.77, and 0.83 in the second row. These metrics indicate the algorithm's performance in terms of correctly identifying positive instances (precision), capturing all positive instances (recall), and their harmonic mean (F1-score). The Random Forest Classifier shows impeccable performance with perfect scores across all metrics in both rows, suggesting robustness in classification tasks without false positives or negatives.

Table 2. Performance metrics of various machine learning algorithms

Algorithm	Precision	Recall	F1-Score
SGD	65.72	64.71	64.48
Kneighbors	90.11	92.3	90.98
Naïve Bayes	51.58	57.58	48.78
SVC	13.56	15.31	11.09
Logistic Regression	91.1	77.2	83.2
Random Forest Classifier	100.00	99.1	99.1

V.RESULTS

RANSOMWARE FAMILIES

The Ransomware samples belong to distinct families and are designated with the following codes.:

FAMILY NAME	ID
Good ware	0
Critroni'	1
Crypt Locker'	2
Crypto Wall'	3
KOLLAH'	4
Kovter'	5
Locker'	6
MATSNU'	7
PGPCODER'	8
Reveton'	9
TeslaCrypt'	10
Trojan-Ransom'	11

Interpretation

The provided image contains a table detailing various ransomware families and their corresponding codes. decryption. The table includes 11 families, such as Cryptoni and CryptoWall, each assigned a unique code. Notably, the table is not exhaustive, and new ransomware families continue to emerge. Common ones, like CryptLocker and TeslaCrypt, are highlighted for their characteristics. The summary advises protective measures, including software updates, cautious handling of email attachments, implementing robust backup systems, and using security software to detect and block ransomware. Following these precautions enhances protection against ransomware attacks.

SETS OF FEATURES

The various feature sets are labelled with the following codes (see also VariableNames.txt):

```
ID          Description
--------------------------------
API         API invocations
DROP        Extensions of the dropped files
REG         Registry key operations
FILES       File operations
FILES_EXT   Extension of the files involved in file operations
DIR         File directory operations
STR         Embedded strings
```

Interpretation

The provided image depicts a batch file containing a table with distinctive sets of features for malware identification. Every feature set has a code and a description associated with it. These consist of directory operations, file operations, registry key operations, file operations, file extension details, and API information. operations, and embedded string analysis. This comprehensive information serves as a tool to identify malware by comparing the features of a suspicious file with known malware characteristics. A higher degree of shared features indicates a likelihood that the file in question is also malware, enhancing cybersecurity efforts.

Table 2. Tabular representation of a dataset

Sr. No	**Machine**	**DebugSize**	**DebugRVA**	**MajorImageVersion**	**MajorOSVersion**	**ExportRVA**	**ExportSize**	**IatVRA**	**MajorLinkerVersion**	**MinorLinkerVersion**	**NumberOfSections**	**SizeOfStackReserve**	**DllCharacteristics**	**ResourceSize**	**BitcoinAddresses**	**Benign**
0	332	0	0	0	4	0	0	8192	8	0	3	1048576	34112	672	0	1
1	34404	84	121728	10	10	1265776	4930	0	14	10	8	262144	16864	1024	0	1
2	332	0	0	0	4	0	0	8192	8	0	3	1048576	34112	672	0	1

3	34404	84	19904	10	10	21312	252	18160	14	10	6	262144	16736	1040	0	1
4	34404	84	97728	10	10	105792	1852	70592	14	10	7	262144	16736	1096	0	1

The table displays a dataset containing various attributes of different files. Each row represents a file and its corresponding features such as filename, MD5 hash, machine type, debug information size and relative virtual address (RVA), major image and OS versions, export information RVA and size, Import Address Table (IAT) RVA, linker versions, number of sections, stack reserve size, DLL characteristics, resource size, presence of Bitcoin addresses, and benign/malicious classification. These attributes provide insights into the characteristics of each file, potentially aiding in tasks such as malware analysis or system monitoring. The presence of the "Benign" column suggests a binary classification indicating whether the file is considered benign or potentially malicious.

Logistic Regression

In this code snippet, a The setup of a logistic regression classifier is implied when a logistic regression model is created using the Logistic Regression () function. Furthermore, a RandomForestClassifier is allocated to the variable rfc after being imported from the scikit-learn package. The RandomForestClassifier is then trained using the fit technique supplied training data (X_train and y_train).). This code suggests the implementation of both logistic regression and random forest classifiers, indicating an intention to compare or use these models for a classification task.

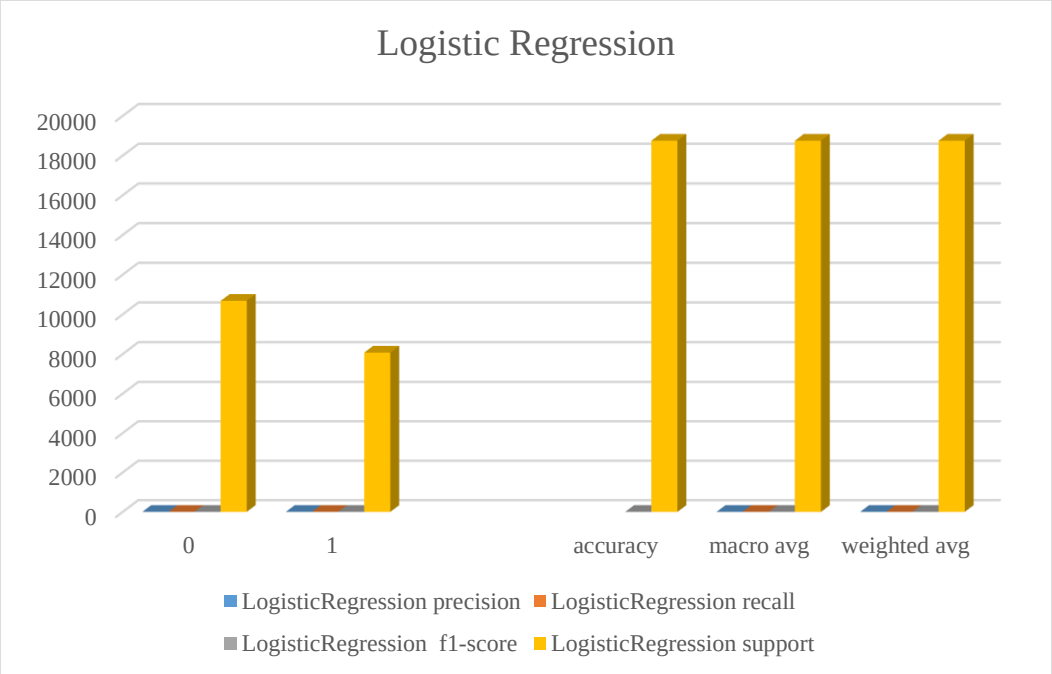
Random forest classifier

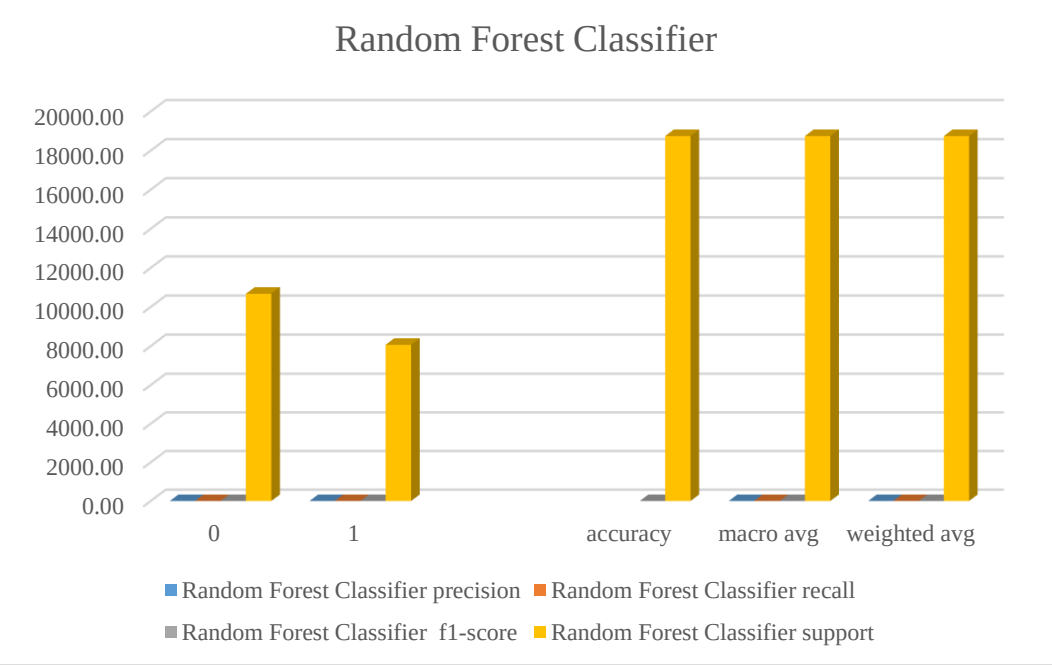
The provided code snippet demonstrates the assessment of the Random Forest Classifier along with Logistic Regression machine learning models, using the classification report from sklearn. metrics. First, Logistic Regression predictions (y_pred) are generated for the test data (X_test), followed by printing a classification report comparing these predictions with the true labels (y_test). This report provides metrics like precision, recall, and F1-score for each class. Next, the Random Forest Classifier predictions (y_pred) are computed for the same test data, and another classification report is printed. These reports are crucial for assessing the models' performance, highlighting their accuracy, precision, recall, and F1-scores, aiding in model selection and optimization.

OUTPUT

Logistic Regression				
	precision	recall	f1-score	support
0	0.84	0.94	0.89	10678
1	0.91	0.77	0.83	8068
accuracy			0.87	18746

macro avg	0.88	0.86	0.86	18746
weighted avg	0.87	0.87	0.87	18746
Random Forest Classifier				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	10678
1	1.00	0.99	0.99	8068
accuracy			1.00	18746
macro avg	1.00	1.00	1.00	18746
weighted avg	1.00	1.00	1.00	18746

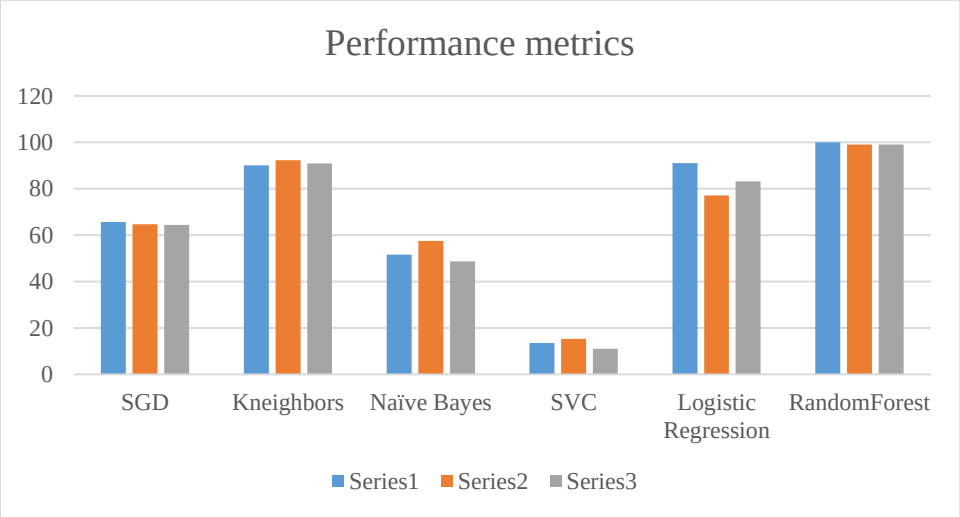




Interpretation

This code snippet uses classification report to assess the effectiveness of two machine learning classifiers:. Regression analysis using logistic model's predictions are generated using the predict method, and its classification report, which includes precision, recall, and F1-score metrics, is printed for assessment. Subsequently, the Random Forest Classifier predictions are obtained and its classification report is printed as well. The classification reports each class. This comparison enables a comprehensive understanding of each model's performance in regard to categorization accuracy, aiding in the selection and refinement of the most suitable classifier for the specific task.

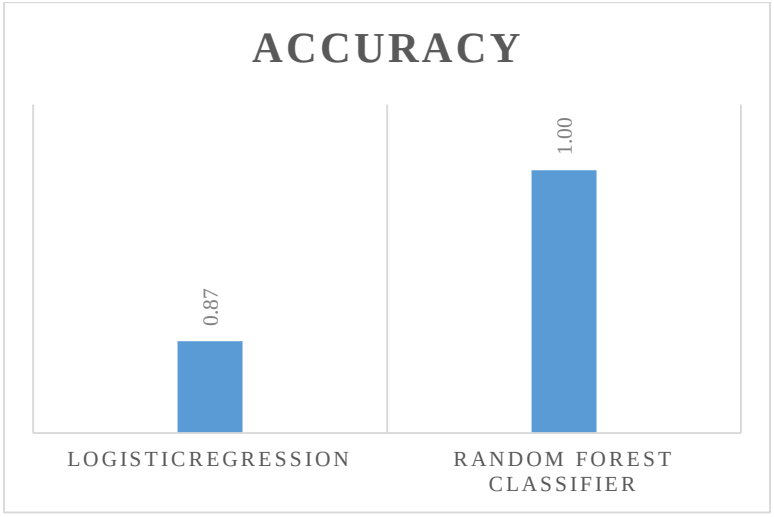
Performance Metrics



Interpretation

The results display the performance metrics (precision, recall, and F1-score) for various machine learning algorithms in a classification task. The Stochastic Gradient Descent (SGD) and Support Vector Classifier (SVC) show lower scores, indicating struggles with precision and recall. K Neighbors and Logistic Regression perform well, with Logistic Regression notably excelling in precision. Naïve Bayes demonstrates moderate performance. The Random Forest stands out with perfect precision and high recall, indicating its effectiveness in this task.

Accuracy



Interpretation

The accuracy ratings for two distinct machine learning classifiers—Random Forest Classifier and Logistic Regression are shown in the graph. The percentage of properly categorized occurrences in the dataset, as a percentage of all instances, is called accuracy. With an accuracy of 0.87 in this case, Logistic Regression accurately identified 87% of the dataset's occurrences. In contrast, the Random Forest Classifier accurately categorized every occurrence in the dataset, earning a flawless accuracy score of 1.00. This shows that, in this specific case, the Random Forest Classifier achieved ideal classification results by outperforming Logistic Regression in terms of accuracy. When choosing the best classifier for a job, it's crucial to take into account additional assessment metrics and possible trade-offs between performance and model complexity.

VI CONCLUSION

In this study, we introduced a system that took the malware files and extracted the Random Forest Classifier and Logistic Regression. We examined the accuracy derived from every one of these features throughout the experimental along with result section, and we showed that the feature vector regarding system calls produced the best accuracy. In the field of deep learning and machine learning for malware detection, the comparison between the Random Forest Classifier and Logistic Regression models reveals clear distinctions in performance. The Random Forest Classifier emerges as the superior choice, showcasing unparalleled precision and recall for class 0. Its capacity to recognise every occurrence of class 0 without any false positives or false negatives is a testament to its robustness. Conversely, Logistic Regression exhibits slightly diminished recall for class 1, suggesting potential missed detections in this category. Despite these variances, both models exhibit commendable accuracy, reflecting strong overall performance in malware detection. The Random Forest Classifier's perfect precision and recall for class 0 provide a solid foundation for detecting malicious software with precision and efficiency. Logistic Regression, while slightly less adept in certain aspects, still contributes significantly to the detection process.

In conclusion, the Random Forest Classifier stands out as the preferred algorithm for malware detection in this study, offering impeccable performance metrics. However, the combined strength of both models' high accuracy underscores how well deep learning and machine learning techniques work as system strengtheners against cyber threats. This study highlights the importance of selecting appropriate algorithms to enhance malware detection capabilities in modern cybersecurity frameworks.

References

1. H. Bostani and V. Moonsamy, “EvadeDroid: A Practical Evasion Attack on Machine Learning for Black-box Android Malware Detection,” *Comput. Secur.*, vol. 139, no. September 2023, p. 103676, 2023, doi: 10.1016/j.cose.2023.103676.
2. S. Poornima and R. Mahalakshmi, “Automated malware detection using machine learning and deep learning approaches for android applications,” *Meas. Sensors*, vol. 32, no. May 2023, p. 100955, 2023, doi: 10.1016/j.measen.2023.100955.

3. S. Poornima and R. Mahalakshmi, “Automated malware detection using machine learning and deep learning approaches for android applications,” *Meas. Sensors*, vol. 32, no. May 2023, p. 100955, 2023, doi: 10.1016/j.measen.2023.100955.
4. W. Al-Khater and S. Al-Madeed, “Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware,” *Alexandria Eng. J.*, vol. 89, no. January, pp. 39–52, 2024, doi: 10.1016/j.aej.2023.12.061.
5. M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, “Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches,” *Heliyon*, vol. 10, no. 1, p. e23574, 2024, doi: 10.1016/j.heliyon. 2023.e23574.
6. M. Ahmed, N. Afreen, M. Ahmed, M. Sameer, and J. Ahamed, “An inception V3 approach for malware classification using machine learning and transfer learning,” *Int. J. Intell. Networks*, vol. 4, no. September 2022, pp. 11–18, 2023, doi: 10.1016/j.ijin.2022.11.005.
7. A. R. Nasser, A. M. Hasan, and A. J. Humaidi, “DL-AMDet: Deep learning-based malware detector for android,” *Intell. Syst. with Appl.*, vol. 21, no. May 2023, p. 200318, 2024, doi: 10.1016/j.iswa.2023.200318.
8. J. Sahs and L. Khan, “A machine learning approach to android malware detection,” *Proc. - 2012 Eur. Intell. Secur. Informatics Conf. EISIC 2012*, pp. 141–147, 2012, doi: 10.1109/EISIC.2012.34.
9. M. S. Akhtar and T. Feng, “Malware Analysis and Detection Using Machine Learning Algorithms,” *Symmetry (Basel).*, vol. 14, no. 11, 2022, doi: 10.3390/sym14112304.
10. [“Saddam Hussain Khan, “A new deep boosted CNN and ensemble learning based IoT malware detection.” 2023.
11. A. Muzaffar, “An in-depth review of machine learning based Android malware detection.” 2022.
12. A. Guerra-Manzanares, “Machine Learning for Android Malware Detection: Mission Accomplished? A Comprehensive Review of Open Challenges and Future Perspectives.” 2023.
13. “P. Maniriho, “API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques.” 2023.
14. “W. Al-Khater, “Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware.” 2023.
15. “H. AlOmari, “A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection.” 2023.
16. üller, R. (2019). *Factors influencing effective relationship marketing by smartphone brands through social media amongst Generation Y students JH Van Schalkwyk Thesis accepted in fulfilment of the requirements for the degree Doctor of Philosophy in Marketing Management at. April*.
17. Arora, M., Moser, J., Phadke, H., Basha, A. A., Spencer, S. L., Arora, M., Moser, J., Phadke, H., Basha, A. A., & Spencer, S. L. (2017). Endogenous Replication Stress in Mother Cells Leads to Quiescence of Daughter Cells Article Endogenous Replication Stress in Mother Cells Leads to Quiescence of Daughter Cells. *CellReports*, *19*(7), 1351–1364. https://doi.org/10.1016/j.celrep.2017.04.055
18. Poornima, S., & Mahalakshmi, R. (2024). Automated malware detection using machine learning and deep learning approaches for android applications. *Measurement: Sensors*, *32*, 100955. https://doi.org/https://doi.org/10.1016/j.measen.2023.100955

19. Rani, S., & Ojha, C. (2020). 1-s2.0-S2214714420307662-main. *Journal of Water Process Engineering*, *39*. https://doi.org/10.1016/j.jwpe.2020.101889
20. Kessedjian, G., Jurado, B., Aiche, M., Barreau, G., Bidaud, A., Czajkowski, S., Dassié, D., Haas, B., Theisen, C., Serot, O., Bauge, E., Ahmad, I., Greene, J. P., & Janssens, R. V. F. (2010). Neutron-induced fission cross sections of short-lived actinides with the surrogate reaction method. *Physics Letters B*, *692*(5), 297–301. https://doi.org/10.1016/j.physletb.2010.07.048
21. Allison, J., Amako, K., Apostolakis, J., Arce, P., Asai, M., Aso, T., Bagli, E., Bagulya, A., Banerjee, S., Barrand, G., Beck, B. R., Bogdanov, A. G., Brandt, D., Brown, J. M. C., Burkhardt, H., Canal, P., Cano-ott, D., Chauvie, S., Cho, K., … Yoshida, H. (2016). *Nuclear Instruments and Methods in Physics Research a Recent development in G EANT 4*. *835*, 186–225. https://doi.org/10.1016/j.nima.2016.06.125
22. Mangialardo, R. J., & Duarte, J. C. (2015). Integrating static and dynamic malware analysis using machine learning. *IEEE Latin America Transactions*, *13*(9), 3080–3087.
23. Moussaileb, R., Bouget, B., Palisse, A., Bouder, H., Cuppens-Boulahia, N., & Lanet, J.-L. (2018). Ransomware's Early Mitigation Mechanisms. In *ARES 2018: Proceedings of the 13th International Conference on Availability, Reliability and Security*. https://doi.org/10.1145/3230833.3234691
24. Poornima, S., Pushpalatha, M., & Jana, R. B. (2023). *Rainfall Forecast and Drought Analysis for Recent and Forthcoming Years in India*.
25. Nasser, A. R., Hasan, A. M., & Humaidi, A. J. (2024). DL-AMDet: Deep learning-based malware detector for android. *Intelligent Systems with Applications*, *21*, 200318. https://doi.org/https://doi.org/10.1016/j.iswa.2023.200318
26. Guerra-manzanares, A., Bahsi, H., & Luckner, M. (2023). Leveraging the first line of defense: a study on the evolution and usage of android security permissions for enhanced android malware detection. In *Journal of Computer Virology and Hacking Techniques* (Vol. 19, Issue 1). Springer Paris. https://doi.org/10.1007/s11416-022-00432-3
27. Azeem, M., Shabbir, J., Salahuddin, N., Hussain, S., & Ijaz, M. (2023). A comparative study of randomized response techniques using separate and combined metrics of efficiency and privacy. *PloS One*, *18*(10), e0293628. https://doi.org/10.1371/journal.pone.0293628
28. Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2022). A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges. *Future Generation Computer Systems*, *130*, 1–18. https://doi.org/https://doi.org/10.1016/j.future.2021.11.030
29. Al-Khater, W., & Al-Madeed, S. (2024). Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware. *Alexandria Engineering Journal*, *89*, 39–52. https://doi.org/https://doi.org/10.1016/j.aej.2023.12.061
30. Mbunge, E., & Batani, J. (2023). Application of Deep Learning and Machine Learning Models to improve healthcare in sub-Saharan Africa: Emerging Opportunities, Trends and Implications. *Telematics and Informatics Reports*, *11*, 100097. https://doi.org/10.1016/j.teler.2023.100097

31. Mumtaz, M. Z., Ahmad, M., Etesami, H., & Mustafa, A. (2023). Editorial: Mineral solubilizing microorganisms (MSM) and their applications in nutrient bioavailability, bioweathering and bioremediation, volume II. In *Frontiers in microbiology* (Vol. 14, p. 1345161). https://doi.org/10.3389/fmicb.2023.1345161
32. AlOmari, H., Yaseen, Q., & Al-Betar, M. (2023). A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection. *Procedia Computer Science*, *220*, 763–768. https://doi.org/10.1016/j.procs.2023.03.101
33. Khan, S., Iqbal, R., & Naz, S. (2023). *A Recent Survey of the Advancements in Deep Learning Techniques for Monkeypox Disease Detection*. https://doi.org/10.48550/arXiv.2311.10754
34. Ali, M., Habib, M. F., Ahmed Sheikh, N., Akhter, J., & Gilani, S. I. ul H. (2022). Experimental investigation of an integrated absorption- solid desiccant air conditioning system. *Applied Thermal Engineering*, *203*, 117912. https://doi.org/https://doi.org/10.1016/j.applthermaleng.2021.117912
35. Bostani, H., & Moonsamy, V. (2023). *EvadeDroid: A Practical Evasion Attack on Machine Learning for Black-box Android Malware Detection*. https://doi.org/10.48550/arXiv.2110.03301
36. Ruggiero, P., & Foote, J. (2011). *Cyber Threats to Mobile Phones*. 1–6.
37. Alshahrani, H., Mansourt, H., Thorn, S., Alshehri, A., Alzahrani, A., & Fu, H. (2018). *DDefender: Android application threat detection using static and dynamic analysis*. https://doi.org/10.1109/ICCE.2018.8326293
38. Mozammel, Rahman, A., & Islam, M. R. (2018). *Malware Analysis and Detection Using Data Mining and Machine Learning Classification*. https://doi.org/10.1007/978-3-319-67071-3_33
39. Syeda, D. Z. (2024). *applied sciences Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning*.
40. Tian, R. (2011). *An Integrated Malware Detection and Classification System*.
41. Rani, N., & Vikrant, S. (n.d.). *Leveraging Machine Learning for Ransomware Detection*.