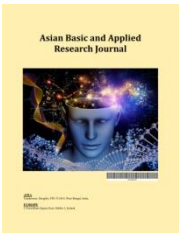

Asian Basic and Applied Research Journal

Volume 6, Issue 1, Page 132-141, 2024; Article no.ABAARJ.1491

An Android Malware Detection System Based on a Hybrid Artificial Neural Network and Decision Tree

Oforjetu Chukwudi Peter [a*] and Andrew Ishaku Wreford [a]

[a] Department of Computer Science, Federal University Wukari, Nigeria.

Authors' contributions

This work was carried out in collaboration between both authors. Both authors read and approved the final manuscript.

Article Information

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://prh.globalpresshub.com/review-history/1491

Original Research Article

Received: 29/12/2023
Accepted: 01/03/2024
Published: 16/09/2024

ABSTRACT

When it comes to sharing data, the broad use of cloud computing has been a huge boon. The development of cloud computing has brought many benefits, but it has also opened the door for criminals to steal sensitive information by taking advantage of its accessibility. This research looks at how deep learning and machine learning techniques, particularly ANN and DT algorithms, can be used to counteract malware. To construct a hybrid model, these methods are combined using a stacking methodology. The investigation was conducted using the Android Network Traffic dataset from Kaggle. We use the Information Gain algorithm for feature selection and a variety of metrics to measure the models' performance, with accuracy serving as the main indicator. On the Android network traffic dataset, the hybrid model attained a 99% accuracy score, according to the results. To improve data security in cloud-based systems and deal with malware, this study shows that integrating deep learning and machine learning approaches works

Keywords: Anomaly-based; malicious software; signature-based; android malware; heuristic-based detection.

**Corresponding author: Email: chukwudioyoko@gmail.com;*

Cite as: *Peter, Oforjetu Chukwudi, and Andrew Ishaku Wreford. 2024. "An Android Malware Detection System Based on a Hybrid Artificial Neural Network and Decision Tree". Asian Basic and Applied Research Journal 6 (1):132-41. https://jofresearch.com/index.php/ABAARJ/article/view/147.*

1. INTRODUCTION

Over the past several decades, internet usage has skyrocketed thanks to the development of Android smart devices. As a result, there has been an exponential growth in the amount of personal, sensitive, and critical data travelling across worldwide networks [1]. The rise of app shops such as Apple's App Store and Google Play has been a boon to the growth of mobile internet technologies, but it has also brought certain problems, most notably invasions of privacy and security caused by bad actors [2].

Currently, the market share and sales volumes are dominated by Android, an open-source mobile operating system that uses the Linux kernel framework [3]. The increasing use of Android devices prompts users to store personal and critical information on their mobile devices, making them targets for malicious application developers who exploit app store downloads, often requiring unnecessary permissions or internet access (Recep et al., 2021). This vulnerability is compounded by certain applications persistently demanding specially designated unnecessary permissions, undermining security mechanisms and making systems susceptible to malicious attacks.

Malware, recognized as a significant threat to modern computer systems, poses severe risks to societal activities [4]. Recent high-profile cyber-attacks on global corporations, particularly ransomware incidents, have resulted in substantial financial, operational, and reputational damages [5]. Modern anonymous payment mechanisms have made ransomware more prevalent by allowing its creators to encrypt specific files in exchange for a ransom, endangering users' privacy, authenticity, and data availability in the process. A high degree of vulnerability to disclosure, modification, or unavailability of sensitive information due to unauthorised access is posed by these assaults.

Hence, there is a growing need for efficient and effective techniques to counter the increasingly complex and advanced malicious attacks that attempt to breach networks, devices, or systems. These attacks pose a considerable threat to information security, affecting the operation of systems and compromising data confidentiality [6].

Several approaches to malware detection have been investigated by researchers. These include rule-based methods, pattern-matching approaches, artificial intelligence (deep learning) techniques, supervised, unsupervised, and semi-supervised machine learning methods, and artificial intelligence (Ce et al., [7], Nisa et al., [8], Yeo et al., [9], Gibert et al., [10]. Android malware assaults continue to be a widespread problem, despite these measures. They cause substantial economic and IP damage. According to Wazid et al. [2], malware developers are always coming out with new variants that are hard to detect because they use obfuscation and small code changes to bypass commercial detection technologies.

In light of the ever-increasing sophistication of cyber threats, this research presents an efficient method for detecting evolving malware strains by combining deep learning with machine learning methods. The Decision Tree and Artificial Neural Network are combined to produce a hybrid model to improve scalability and improve detection speed. An Android Network Traffic dataset obtained from the Kaggle machine learning archives was used as the basis for the analysis.

2. LITERATURE REVIEW

2.1 Related Works

To find malware using a neural network, Hossain et al. [11] compared and contrasted the performance of three different neural networks: CNN, LSTM, and GRU. The authors used samples from the Portable Executables (PE) dataset, which was obtained from virusshare.com, portableapps.com, and the Windows 7 x 86 directories, and CNN outperformed the other two models with an accuracy of 83% in detecting malware, compared to LSTM's 65% and GRU's 76%.

Yuxin and Siyi [12] introduced an innovative malware detection model based on the deep belief network (DBN) structure. This model utilized two hidden layers capable of accommodating up to 200 neurons. Its main goal was to identify and classify malware instances based on their unique characteristics. To evaluate its performance, the authors conducted thorough testing using an OpCode-n-gram dataset. This dataset was thoughtfully divided into four subsets, each containing 850 samples of malware and 850 samples of benign software. This rigorous evaluation allowed for a comprehensive assessment of the model's capabilities across a diverse range of samples.

The results were impressive, with the DBN model achieving an accuracy rate of 96.5% when analyzing 200 distinct features. This high accuracy demonstrates the model's effectiveness in distinguishing between malware and benign software, highlighting its potential for real-world applications in strengthening cybersecurity.

Cai et al. [13] presented a novel Android app classification system called DroidCat. This system utilises dynamic method calls and ICC Intents to achieve robust and dynamic categorization of Android applications. The proposed method effectively managed reflection without dependence on system calls or permissions. Moreover, in comparison to other comparable state-of-the-art static and dynamic malware detection techniques, the proposed technique shows greater resilience. Furthermore, the study utilised a total of 34,343 applications obtained from diverse sources spanning nine years. Subsequently, a performance metric was derived. The authors achieved a 97% F1 score accuracy when comparing their methodology to two state-of-the-art approaches.

In 2020, Zhongru et al. created two deep learning-based end-to-end algorithms for detecting Android malware. The authors preprocess the raw bytecodes of Android application classes.dex files by resampling them. These files are then input to two deep learning models, the Dalvik Executable-Convolution Neural Network (Dex-CNN) and the Dalvik Executable-Convolution Recurrent Neural Network (Dex-CNN). Eight thousand legitimate apps and eight thousand malicious apps culled from the Google Play Store made up the dataset used to train and assess the algorithms. The author's investigations showed that Dex-CNN and Dex-CRNN models obtained detection accuracy of 93.4% and 95.8%, respectively. Their models outperformed the competition when it came to input file size, manual feature engineering, and resource consumption. They reasoned that their methods would work better on Android IoT devices and would benefit from the end-to-end learning process.

Xing et al. [14] put out a deep learning-based malware detection system that makes use of Autoencoder. For both the benign labels and the malicious malware from VirusShare, the writers consulted a dataset obtained from the Google App Store. The created model integrates an autoencoder network with a virus representation in greyscale images. After determining the autoencoder's reconstruction error, the authors examine the grey-scale image approach's viability and employ the autoencoder's dimensionality reduction features to achieve malware classification from benign software. As stated by the authors, the suggested detection model attained a 96% accuracy rate and a steady F-score of 96%.

3. METHODOLOGY

For feasible and effective implementation of a malware detection model. Data preprocessing, model application, and performance evaluation are the three main ways that this study suggests as part of a methodology. At the outset, there is data preprocessing, which comprises cleaning up the acquired dataset of any extraneous characters and symbols (such as missing values, noise, etc.). In addition, before data encoding and scaling, a filtering method called information gain is suggested for use in data preprocessing. This method assesses and chooses all potential combinations of attributes that substantially impact the malware classification as benign or malignant. In the context of machine learning tasks, feature selection essentially means reducing overfitting by picking the relevant features, which in turn allows for faster algorithm training, simpler models, higher prediction power, and easier interpretation. The second step, known as "model application," involves feeding the two chosen models—the Artificial Neural Network and the Decision Tree—the preprocessed and chosen input dataset that has been encoded and divided into test and training sets. The result is a trained model that can be evaluated and tested. The last stage is performance evaluation, which involves testing the trained model's ability to distinguish between benign and malicious malware using test data. The model is then assessed using various machine learning classification metrics, including accuracy score, precision metric, and others. Fig 1 illustrates the three procedures that were discussed for the methodology.

3.1 Data Source

The dataset utilised by this study is the Android network traffic dataset also sourced from the Kaggle machine learning repository [15]. This dataset is based on another dataset (DroidCollector) that gets all the network traffic in pcap (package capture) files. Furthermore, the

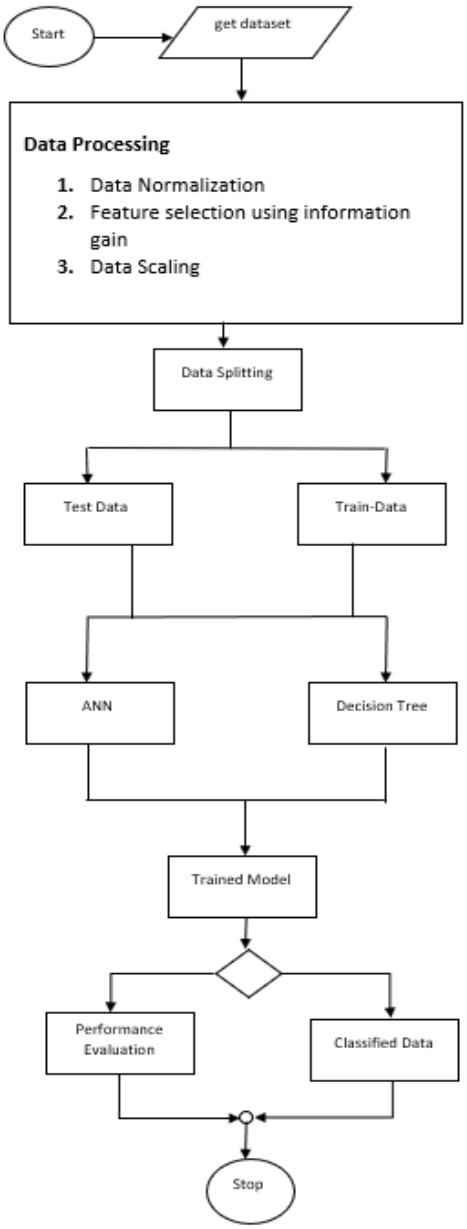

Fig. 1. Research methodology framework

Android network traffic dataset has a total of 4,704 benign records and a total of 3,141 malignant malware records.

3.2 Feature Selection

Using a threshold value and an ordered ranking of all characteristics, information gain is a univariate filter approach that picks attributes. More specifically, the information gain method calculates a meaningful information metric that measures the amount of data obtained from a dataset's class prediction depending on the presence or absence of malware. Therefore, the whole spectrum of malware characteristics is divided into two parts according to the established threshold. The characteristics above the threshold are defined in the first part, while the qualities below the threshold are defined in the second. So, the feature that is higher than the threshold is used as the data attribute that is supplied to the model. Information gain can be mathematically expressed as:

$$IG_z = \sum_{k=0}^{1}\sum_{m=1}^{2}\frac{i_{km}}{N}\log_2\frac{i_{km}}{n_{vm}} - \sum_{k=0}^{1}\frac{i_{k1}+i_{k2}}{N}\log_2\frac{i_{k1}+i_{k2}}{N} \ldots\ldots \quad (1)$$

where IG defines the information gain, n_{vm} is the number of instances of class k in region m, and i_{km} is the number of expression levels of malware z in region m.

Algorithm 3.1: Information Gain (IG)

Input: *Dataset D with features F(x1...xn)*

Θ Threshold value

Parameters: *N - Quantitative amount of bootstrap samples*

M - Total number of features

m – number of selected features

Step 1: *perform attribute filtration on Dataset D*

Step 2: *Compute Information Gain using equation 3.1*

Step 3: ***if*** *(IG > Θ)*

Step 4: ***Select*** *subset x with the highest IG*

Step 5: *Create N bootstrap samples from the selected subset x.*

Step 6: ***increment*** *m where m<M.*

Step 7: end

3.3 Classification Algorithms

3.3.1 Artificial neural network

To analyse data thoroughly, computational models can use Artificial Neural Networks (ANNs), which mimic the functioning of the brain's neural circuits. Dendrites stand in for inputs in artificial neural networks (ANN), nodes are cell nuclei, synapses are weights, and axons are the outputs. It is crucial to construct the ANN model with W-weight values that minimise prediction error for the suggested ANN algorithm. To do this, the "backpropagation algorithm" turns ANN into a learning system that can improve itself based on its previous errors. The "gradient descent" method is suggested as an optimisation strategy. The prediction errors are quantified using gradient descent. We try out different values for W and see how they affect prediction errors to discover the sweet spot. Lastly, those W values are considered optimum because adjusting W further does not result in a decrease in errors.

3.3.2 Decision tree

The versatility of decision trees makes them a go-to for many machine learning applications, particularly those involving classification and regression. Decision trees, as its name suggests, make judgements based on data and how it behaves. Since it doesn't rely on a linear classifier, its performance is unaffected by the data's linearity; instead, it uses a tree-like structure to make decisions based on conditional statements about whether or not an event has occurred. At each stage, or node, of a decision tree, the study attempts to form a condition on the features to fully separate the two-class labels (Benign or Malignant) contained in the dataset as pure as possible by decreasing the entropy threshold, which is a measure of the impurity or randomness in the dataset. This is how a broad decision tree is visualised.

3.4 Performance Metrics

To assess the performance of the Artificial Neural Network and Decision Tree on the Android Network Traffic malware dataset obtained from the Kaggle Machine Learning repository. This study used the viability of the following evaluation metrics.

Precision: Measures the classifier’s accuracy. It is the percentage of the number of correctly predicted positive instances divided by the total number of predicted positive instances. Equation 3.2 depicts the mathematical expression for the precision metrics.

$$precision = \frac{TP}{TP + FP} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2$$

Where True Positives (TP) is a situation where the actual class from a data record is true and thus predicted true by the model and False Positives (FP) is an instance where the actual data record class is false but the model predicted true.

Recall: One measure of a classifier's thoroughness is its recall, which is sometimes called its sensitivity or true positive rate. The accuracy rate is determined by dividing the number of positive cases that were accurately predicted by the total number of positive examples in the dataset. Recall can be mathematically represented as shown in equation 3:

$$Recall = \frac{TP}{TP + FN} \ldots\ldots\ldots \quad 3$$

Where False Negatives (FN) is an instance where an actual data record point is true but the model predicted false and

F-measure (or F-score): defines the harmonic mean of precision and recall. It combines recall and precision metrics to obtain a score. Equation 3.4 shows the mathematical formula for the F-score.

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \ldots\ldots\ldots 4$$

Accuracy: one way to quantify accuracy is by considering the ratio of accurately predicted occurrences to the total number of instances in the dataset. According to equation 5, the accuracy is determined by the proportion of inputs in the test set that the classifier correctly labels:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \ldots\ldots\ldots\ldots\ldots\ldots 5$$

Where True Negatives (TN) is a scenario where a data record was false and hence predicted false by the model.

4. RESULTS AND DISCUSSION

4.1 Environmental Setup

The Android malware detection models were implemented using Python programming language via the Anaconda programming environment on a Windows operating system with a dual-core Intel Core I5 processor and 4GB RAM. The model integrated includes the Artificial Neural Network, the Decision Tree algorithm, and a hybrid of both. The Python packages utilized include the usage of Tensor API (for deep learning model developments), NumPy modules for numerical and multi-dimensional vector operation, pandas for reading both the ClaMP malware and the Android network traffic datasets, and seaborn and Matplotlib (which are some of the machine learning algorithms targeted at visualizing the graphical behaviour of the implemented models).

4.2 Parameter Setting

The parameters utilized for the ANN, DT, and hybrid model are presented in Table 1. From the parameters, the batch size defines the number of the dataset input instances passed to the model per unit layer. The batch size was set to 32 units with the max epochs set to 10 epochs, a learning approach of 0.0001, with Adam set as the optimizer and drop out of 0.2 for each of the layers of the model (Essentially, it is important to note that the values for the defined parameters were tweaked for several ranging values, the presented values are the ones that gave the best performance based on the experiment conducted).

Table 1. Parameter setting

Parameter	Value
Batch Size	32
Max epochs	10
Initial learn rate	0.001
Optimizer	Adam
Drop	0.2
Loss	Sparse Category Cross Entropy

4.3 Feature Selection

As aforementioned the study incorporates a feature selection technique using the information gain algorithm. The essence of the feature selection is to reduce the dataset dimensionality and further eradicate less correlated features in the detection and classification of Android malware. The algorithm 'information gain' allows the specification of the number of features to be selected using a value called the k-threshold. The value of k as a threshold was set to 10 during the experiment.

4.4 Results Presentation

The results of the Android malware detection study present a comprehensive evaluation of three different algorithms: Artificial Neural Network (ANN), Decision Tree (DT), and the Hybrid approach as shown in Table 3. These algorithms were assessed based on several performance metrics, including accuracy, precision, recall, and F1-Score. The Artificial Neural Network (ANN) achieved an accuracy rate of 71%. In terms of precision for classifying malware (1), it exhibited a precision of 68 and 83% (benign and malignant), indicating that it correctly identified malicious software a significant portion of the time. However, the recall rate for malware (1) detection was relatively high at 95 and 34%, suggesting that the ANN tended to identify most of the actual malware instances. Consequently, the F1-Score for malware detection stood at 79 and 48%, reflecting a relatively balanced performance between precision and recall. On the other hand, the Decision Tree (DT) algorithm outperformed the ANN in terms of accuracy, achieving an impressive accuracy rate of 94%. This indicates that the DT model made fewer misclassifications overall. Furthermore, the precision for malware detection was notably high at 97 and 89%, demonstrating a strong ability to correctly identify malicious applications. The recall rate for malware detection was also commendable at 92 and 96%, indicating that the DT model managed to capture a substantial portion of actual malware instances. Consequently, the F1-Score for malware detection was a robust 95 and 92%, suggesting a well-balanced performance between precision and recall.

The Hybrid approach emerged as the top performer among the three algorithms, boasting an impressive accuracy rate of 99%. This signifies an exceptional ability to classify applications accurately. Notably, the precision for malware detection in the Hybrid approach was an astounding 99%, indicating an almost perfect precision in identifying malicious software. The recall rate for malware detection was also impressive at 100 and 98%, signifying the model's capability to capture nearly all actual malware instances. Consequently, the F1-Score for malware detection reached an impressive 99%, underscoring the Hybrid approach's outstanding balance between precision and recall.

In summary, the results show that the Hybrid approach significantly outperforms both the ANN and DT algorithms in terms of accuracy, precision, recall, and F1-Score. While the ANN and DT models exhibit respectable performance, the Hybrid approach's exceptional precision and recall make it a promising candidate for Android malware detection, with the potential to provide highly accurate and reliable results in real-world applications. Fig. 2 shows the graphical result of the algorithms used [16].

Table 2. Shows the selected features for the dataset

S/N	Andriod Network Traffic
1	Name
2	TCP-Packets
3	Dist-Port-TCP
4	External IPs
5	DNS-Query-times
6	UDP-packets,
7	Source App Packets
8	Remote App Packets
9	Source App bytes
10	Remote App Bytes

4.5 Receiver Operating Characteristics

To visualise the performance of the developed model on the applied Android network malware datasets, the receiver operating characteristic was applied. The ROC curve is a graph that shows the performance of a classification model at all the classification thresholds. The curve normally plots two parameters namely the true positive and false positive rate. The ROC for the models is shown in Figs. 3-5 [17,18].

Table 3. Android network traffic dataset result

Algorithm	Accuracy (%)	Precision (0/1) (%)	Recall (0/1) (%)	F1-Score (0/1) (%)
ANN	71	68/83	95/34	79/48
DT	94	97/89	92/96	95/92
Hybrid	99	99/99	1.0/98	99/99

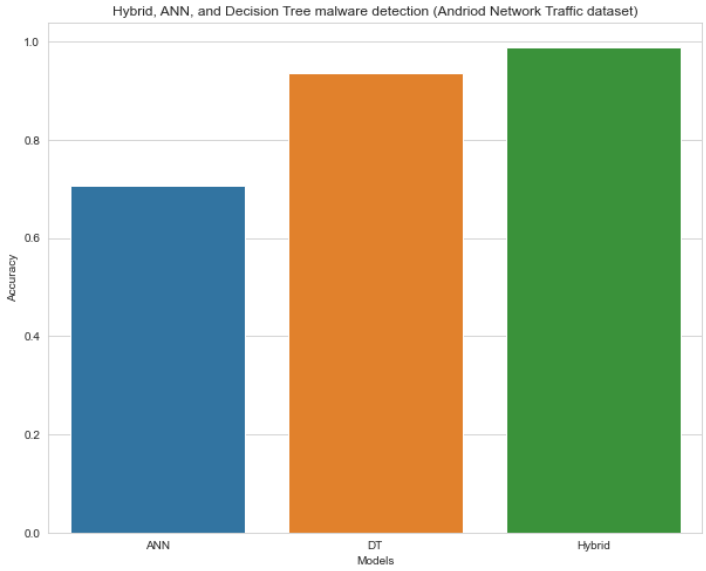

Fig. 2. Result comparison visualization

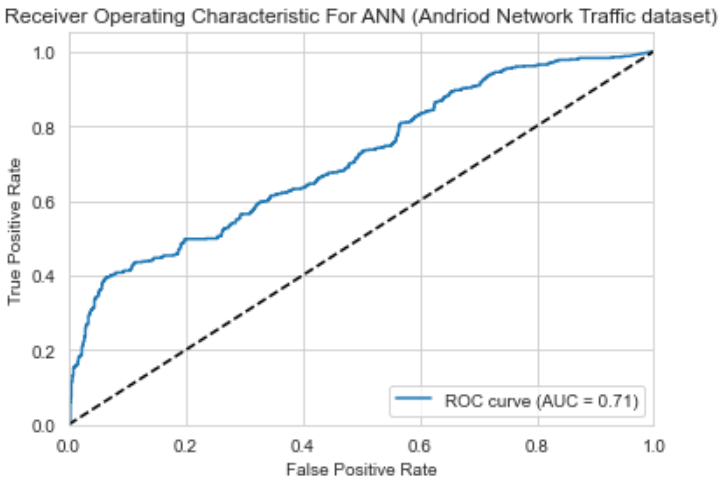

Fig. 3. ANN malware model (ROC and AUC)

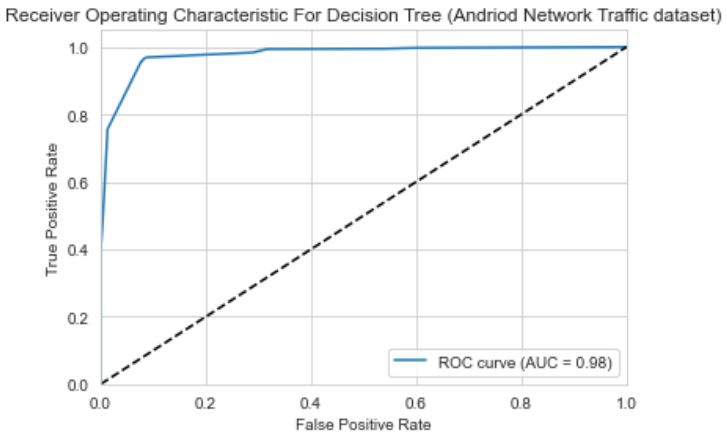

Fig. 4. DT malware model (ROC and AUC)

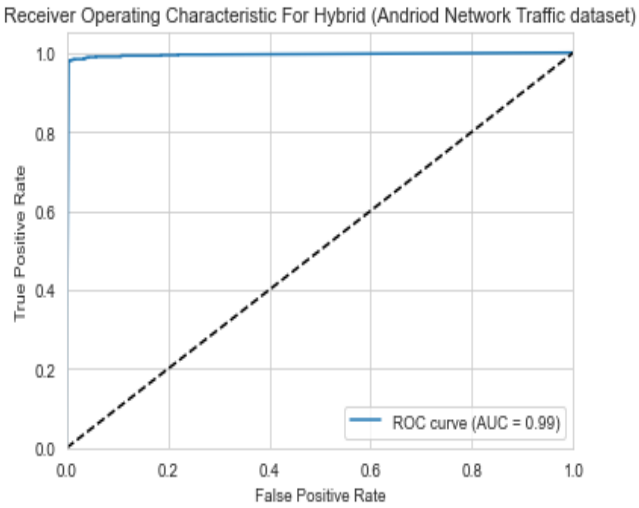

Fig. 5. Hybrid malware model (ROC and AUC)

5. CONCLUSION

This study after an extensive survey of the malware detection techniques applied machine and deep learning algorithms namely the ANN and Decision Tree algorithm while hybridization of the two algorithms via a stacking technique. Furthermore, the performance of the developed models after applying information gain as a feature selection technique was evaluated based on some metrics with the accuracy score metrics as the major indicators. The comparison of the implemented models revealed an accuracy of 97%, 94%, and 99%. To further scrutinise the performance of the best performing, the best model performance was compared with some state-of-the-algorithms. The result of the comparison revealed the hybrid model was developed to outperform the state of the algorithms with an accuracy score of 99%. Hence, in a later study, researchers can apply the viabilities of AI-based approaches such as the Jaya algorithm for feature selection.

DISCLAIMER (ARTIFICIAL INTELLIGENCE)

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc) and text-to-image generators have been used during writing or editing of this manuscript.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Kim YK, Lee JJ, Go MH, Kang HY, Lee K. A systematic overview of the machine learning methods for mobile malware detection. Security and Communication Networks. 2022;2(4).
2. Wazid M, Das AK, Rodrigues JJPC, Shetty S, Park Y. IoMT Malware Detection Approaches Analysis and Research Challenges, in IEEE Access. 2019;7. 182459-182476, 2019, DOI: 10.1109/ACCESS.2019.2960412.
3. Recep SA, Ibrahim AD, Necaattin B. Permission-Based malware detection system for andriod using machine learning techniques. International Journal of Software Engineering and Knowledge Engineering. 2021; 3(2):1-10.
4. Afianian A, Niksefat S, Sadeghiyan B, Baptiste D. Malware dynamic analysis evasion techniques: A survey. ACM Computing Survey; 2019. Available:https://doi.org/10.1145/3365001.
5. Chen Q, Bridges RA. Automated behavioural analysis of malware: a case study of wannacry ransomware. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). 2017;454–600. Available:https://doi.org/10.1109/ICMLA.2017.0-119.
6. Daniel G, Carles M, Jordi P. The rise of machine learning for detection and classification of malware: Research developments, trends, and challenges. Journal of Network and Computer

Applications. 2020;153:102526, ISSN 1084-8045,
Available:https://doi.org/10.1016/j.jnca.2019.102526.

7. Ce L, Qiujian L, Ning L, Yan W, Degang S, Yuanyuan Q. A novel deep framework for dynamic malware detection based on API sequence intrinsic features. Journal of Computers & Security. 2022;116. ISSN 0167-4048,
Available:https://doi.org/10.1016/j.cose.2022.102686.
8. Nisa M, Shah JH, Kanwal S, Raza M, Khan MA, Damaševi˘cius R, Blažauskas T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. Appl. Sci. 2020;10:49-66.
9. Yeo M, Koo Y, Yoon Y, Hwang T, Ryu J, Song J, Park C. Flow-based malware detection using a convolutional neural network. International Conference on Information Networking (ICOIN). 2018; 910-913,
DOI: 10.1109/ICOIN.2018.8343255
10. Gibert D, Bejar J, Mateu C, Planes J, Solis D, Vicens R. Convolutional Neural Networks for Classification of Malware Assembly Code; 2017.
Available:https://www.cynet.com/malware/4-malware-detection-techniques-and-their-use-in-epp-and-edr/
11. Hossain H, Kayum SI, Paul A, Rohan AA, Tasnim N, Hossain MI. Malware detection using neural networks. IEEE. 202;11-6.
12. Yuxin D, Siyi Z. Malware detection based on deep learning algorithm, Neural Computing & Applications. 2017;31(2): 461–472.
13. Cai H, Meng N, Ryder B, Yao D. Droidcat: effective android malware detection and categorization via app-level profiling, IEEE Transactions on Information Forensics and Security. 2018;14(6):1455–1470.
14. Xing X, Jin X, Elahi H, Jiang H, Wang G. A malware detection approach using autoencoder in deep learning. IEEE Access. 2022;10:25696-25706.
15. López CCU, Villarreal JSD, Belalcazar AFP, Cadavid AN, Cely JGD. Features to detect android malware. IEEE. 2018;1-6.
16. Zhongru R, Haomin W, Qian N, Iftikhar H, Bingcai C. End-to-end malware detection for Android IoT devices using deep learning. Ad Hoc Networks. 2020;101:102-118. ISSN 1570-8705.
Available:https://doi.org/10.1016/j.adhoc.2020.102098.
17. Lu Y, Pan Z, Liu J, Shen Y. Android malware detection technology based on improved Bayesian classification. In Instrumentation, Measurement, Computer, Communication and Control (IMCCC), Third International Conference on. 2014; 1338–1341.
DOI: 10.1109/IMCCC.2013.297
18. Akshay M, Laxmi MP, Keyur K, Quamar N, Ahmad YJ. NATICUSdroid: A malware detection framework for Android using native and custom permissions. Journal of Information Security and Applications. 2021;58:102-106. ISSN 2214-2126.
Available:https://doi.org/10.1016/j.jisa.2020.102696.

Peer-review history:
The peer review history for this paper can be accessed here:
https://prh.globalpresshub.com/review-history/1491