Mesopotamian journal of Big Data
Vol. **(2025)**, 2025, **pp**. 261–277
DOI: https://doi.org/10.58496/MJBD/2025/017 , ISSN: 2958-6453
https://mesopotamian.press/journals/index.php/BigData

Research Article

A Robust Model for Android Malware Detection via ML and DL classifiers

Omar Almomani[1], *, Areen Arabiat[2], Muneera Al Tayeb[2], Mohammed Amin Almaiah[3], Mansour Obeidat[4], *, Theyazn H. H. Aldhyani[4], Rami Shehab[5], Mahmood rowad[6],

[1] *Department of Networks and Cybersecurity, Al-Ahliyya Amman University, Amman, Jordan*

[2] *Department of Communications and Computer Engineering, Faculty of Engineering, Al-Ahliyya Amman University, Amman, Jordan*

[3] *King Abdullah the II IT School, University of Jordan, Amman, 11942*

[4] *Applied College, King Faisal University, Al-Ahsa, Saudi Arabia*

[5] *Vice-Presidency for Postgraduate Studies and Scientific Research, King Faisal University, Al-Ahsa 31982, Saudi Arabia*

[6] *Quantitative method department, College of Business, King Faisal University, Al-Ahsa, 31982, Saudi Arabia*

ARTICLEINFO

Article History

Received 12 Jul 2025
Revised 14 Aug 2025
Accepted 26 Aug 2025
Published 24 Sep 2025

Keywords

Android Malware
Logistic regression
Decision Tree
Artificial Neural Networks
NATICUS-Droid dataset

ABSTRACT

The rapid growth of sophisticated Android malware (AM) threats is significant, as Android devices often store private and sensitive personal and financial information. These threats allow stealing of data, interference with device functioning, and network compromise. One of the greatest difficulties in efficient interception systems is ensuring a high level of detection accuracy for distinguishable AM variants. This study focuses on developing a robust Android malware detection model via machine learning (ML) and deep learning (DL) techniques. The model combines ML classifiers, which consist of logistic regression (LR) and decision trees (DTs), and a DL classifier, an artificial neural network (ANN). The model was implemented via an open-source data mining program called Orange. The NATICUSdroid dataset was used to train and test the model, which was measured in terms of accuracy, precision, recall, F-measure and AUC. The experimental findings revealed that the ANN performed the best (accuracy: 98.0%, precision/recall/F-measure: 98.0%, AUC: 0.997) and was better than the LR (accuracy: 96.1%, AUC: 0.989) and DT (accuracy: 96.0%, AUC: 0.971) methods. The results highlight the high potential of DL-based approaches, especially ANNs, to detect Android malware and reinforce their suitability for enhancing mobile security systems.

1. INTRODUCTION

The rapid rise of the internet and mobile applications has increased the number of cyberattacks using malicious applications such as Trojan horses and rootkits [1] [2] [3]. These can result in system attacks, remote code execution, or theft of information [4] [5] [6]. However, malware development began in the 1980s with viruses, which swiftly spread and caused hundreds of computer systems to be damaged. Every day, thousands of new pieces of malware are created that are targeted, zero-day, stealthy, and persistent. To avoid detection systems, these complicated programs use computer system flaws, including obfuscation, encryption, and encoding. [7] [8] . Malware analysis techniques are static and dynamic and detect malicious patterns without execution via feature vectors or signature-based detection. To date, researchers have combined static and dynamic methods for malware detection. [9] [10] [11]. On the other hand, malware developers have increased code complexity, making it more difficult to detect next-generation malware. These types can evade security mechanisms and persist indefinitely. Signatures, heuristics, behaviour, model verification, the cloud, mobile devices, the Internet of Things (IoT), ML, and DL are some of the detection methodologies [12] [13].

Recently, malware, considered an increasingly dangerous vector as technology has advanced, has necessitated sophisticated detection techniques to avoid possible damage. Strategies for cyber threat intelligence include data collection, profile creation, the use of intelligent algorithms [14] [15], and the development of improved detection and mitigation approaches [16] [17]. ML is a reliable and effective threat detection technique [18] [19] [20] [21] . Windows executable files, the world's most widely used operating system, are a possible threat vector [22]. Android's existing malware defense mechanism is an unsafe communication technique that requires too much technical expertise for average users to differentiate between legitimate and dangerous apps. DL, a relatively new branch of ML research, has sparked interest in artificial intelligence and

*Corresponding author. o.almomani@ammanu.edu.jo, mobaydat@kfu.edu.sa

inspired innovative applications in the recognition of speech and images [23]. However, one of the best malware detection methods is DL, which addresses a variety of legal, social, and economic concerns in enterprises [24] [25] [26]. Additionally, ML algorithms are growing in use in detecting malware because of their promising performance [27]. Additionally, because the size of each file varies, IoT malware research requires a substantial dataset and fixed-length features [28] [29] [30].

Orange is an open-source software used for data mining. [31] provides a user-friendly environment for creating, evaluating, and visualizing malware detection models, integrating data preparation, feature engineering, and model training for efficient problem space exploration via both ML and DL techniques to create a prediction model. Additionally, Orange is a visual programming tool used for data visualization and data analysis. The Orange tool was selected because of the following features, such as visualization, feature selection, and preprocessing, to evaluate the ML predictive modelling and is free and open source. This study implements robust AM detection via Orange, the model that combines LR, DT, and ANN. The Orange data mining platform has made it possible to design, assess, and visualize malware detection models thoroughly and intuitively. On the other hand, Orange's data-mining approach to AM detection is unique since it combines sophisticated analytics, ML, and an intuitive interface. By utilizing these advantages, Orange Data mining enables users to create a system for detecting malware that is efficient and flexible enough to change with the always-changing mobile threat landscape. To further increase detection rates, future studies might examine ways to improve feature extraction methods, use more sophisticated ML algorithms, and foster community collaboration [32] [33] [34].

Detecting AM is still one of the main challenges of mobile security. Malware creators continue developing new evasion techniques, obfuscating malicious code and circumventing traditional detection mechanisms. The diversity of AM, rapid emergence of new threats, and high volume of applications further complicate accurate detection, keeping end users vulnerable to data breaches and system breaks. To solve these challenges, the main goal of this study is to propose a robust detection framework for AM by integrating both the ML and DL approaches. The framework leverages the interpretability and efficiency of ML classifiers, such as LRs and DTs, alongside the representational power of DL models, specifically ANNs, to enhance the detection performance of MAs. To achieve the goal of this study, the following research objectives are threefold:

1. to develop a hybrid Android malware detection model using ML and DL classifiers.
2. to evaluate the proposed model via the NATICUSdroid dataset.
3. to compare the performances of the LR, DT, and ANN methods in terms of accuracy, precision, recall, F-measure, and ROC AUC.

The originality of this study is that both the ML and DL approaches are compared in one framework as a comprehensive study, whereas most researchers have considered the mechanism of AM detection on the basis of ML or DL models. In contrast to the uncommon former literature where single categories of algorithms are being tested or closely guarded environments are employed, this paper combines LR, DT, and an ANN in the same experimental circumstances, employing the publicly available NATICUSdroid dataset. Furthermore, the use of the Orange open-source framework guarantees transparency and reproducibility, so the desired framework will be accessible to both practitioners and researchers. The critical analysis of several evaluation metrics, such as accuracy, precision, recall, F-measure, and AUC, used in this study provides a more in-depth look at the performance of a classifier. The significance of this research is that it contributes to enhancements in mobile security with the high-performance detection method provided. The results of the experiments prove that the ANN is better than both the LR and DT classifiers and that it is appropriate as an AM detector because it is robust and effective.

This paper is arranged in the following manner: Section 2 provides an overview of studies on AM detection; Section 3 provides a glimpse into the proposed model; Section 4 presents the evaluation process of classification under the umbrella of the proposed model; Section 5 infers the results; and finally, Section 6 presents the conclusions of this study.

2. RELATED WORKS

Several AM detection techniques have been presented in the literature, and some of these techniques are discussed in this section.

A study by A. R. Nasser, A. M. Hasan, and A. J. Humaidi [35] proposed the DL-AMDet framework to detect AM applications in two steps: the CNN-BiLSTM method handles the analysis of static features such as permissions or API calls, whereas the dynamic features, that is, system calls, are analysed via deep autoencoders for anomaly detection. This two-stage approach thus spots the obfuscations, zero-day attacks, compromises, and dynamic code loading that pure static methods cannot address. DL-AMDet, trained and tested with a set of 21,000 good and malicious apps containing 3,100 static and 100 dynamic features from each sample, achieved an F score of 99.935%, thus outperforming even the highest existing competing techniques. Moreover, the embedded decision module uses dynamic feedback to improve the static model, thus strengthening the detection ability. The authors propose that future works explore lightweight versions of this architecture so that perhaps it can be implemented on the device without trading detection accuracy.

In another study by A. Alhogail and R. A. Alharbi [36], in this study, the CICMalDroid 2020 dataset uses chi-square and select-from-model techniques, which reduce the feature dimensions from 470 to 120, followed by a two-step classification procedure for actual, binary malware detection and multiclass categorization. Their framework, which uses random forest classification methods, yielded a malware detection accuracy of 97.82% and a 96.09% classification accuracy, which speaks strongly of its generalizability across existing malware types. Taken together, these studies emphasize the integration of multilevel analysis and optimization of features with sturdy classification techniques to construct Android malware detection systems that are scalable, interpretable, efficient, and suitable for real-world deployments.

Another study by M. A. Habeeb and Y. L. Khaleel [37] proposed a framework to compare the following ML classifiers: ANN, logistic regression, k-nearest neighbors, gradient boosting, adaptive boosting, categorical boosting, and extremely randomized trees. According to their results, the ANN classifier performed best in all evaluation parameters, with training and testing accuracies of 0.99, an average accuracy of 0.99, a precision of 0.99, a recall of 0.98, and an F-measure of 0.99, whereas conventional ML techniques were similar, with F-measures of approximately 0.95--0.96. Owing to the excellent learning capability of ANNs, this model can recognize complicated AM patterns more effectively, thus identifying next-generation and hidden malicious applications. Despite the challenges regarding heterogeneity in datasets, evolving malware characteristics, and computational requirements, the study reinforced the adaptability, reliability, and potential of ANN-based systems in safeguarding Android environments. Further research is encouraged to increase the diversity of datasets, consider evolving malware characteristics, increase training efficiency, and enhance model interpretability to support practical deployment in any real-world scenario.

The study by Mat, Sharfah Ratibah Tuan, et al. in [38] described an AM detection method that uses permission characteristics and Bayesian classification. The method collects permission features from 10,000 samples from the AndroZoo and Drebin datasets and then selects features via information gain and chi-square algorithms to achieve a 91.1% accuracy rate. Another study by Lue [39] presented SeGDroid, a new AM detection approach that derives semantic knowledge from sensitive function call graphs (FCGs). It constructs a sensitive FCG via a graph pruning method, extracts attributes from graph nodes, and induces graph embeddings via a graph CNN algorithm. The approach has an F score of 98% for malware detection and 96% for malware family classification, indicating malicious activity in AM.

Moreover, the study by Mathur, Akshay, et al. [40] introduces a new AM detection framework called NATICUSdroid, which differentiates harmless and harmful applications using only native permissions or custom permissions. The framework employs eight ML algorithms, with the random forest classifier achieving the best accuracy with a false-positive rate of 97%. A study by Odat, Esraa, and Qussai M. Yaseen [41] described an ML strategy for detecting AM based on the coexistence of static features. The model assumes that malware requests unusual permissions and (application programming interface) APIs. The common pattern growth technique was used to build a fresh dataset of features that coexisted. The RF technique and coexistence of permission characteristics at the second combination level resulted in good accuracy, with a maximum of 98%. On the other hand, a study by Mahindru et al. [42]. A system that detects malware in Android apps via dynamic analysis and feature selection is suggested. The model is built via four ML methods and rough set analysis. Experiments on more than 5,000 Android apps revealed that the model created with DL, farthest first clustering, Y-MLP, and a nonlinear ensemble DT forest technique had the greatest detection rate of 98.8%.

In another study, a lightweight AM detection system with a two-layer structure named MCADS was developed by Ma, Runze, et al. [43]. The first layer analyses malware via an upgraded multilayer perceptron (MLP), and the second layer uses a novel lightweight convolutional neural network (CNN) for additional analysis. The method attained 98.12% accuracy, proving its usefulness as an adjunctive option.

Haq, Ikram Ul, et al [44]. This approach provides highly effective hybrid DL via multivector malware detection methods. To efficiently identify persistent malware, the proposed approach uses a CNN and BiLSTM. The suggested model has been rigorously evaluated via publicly available datasets, standard performance measures, and cutting-edge hybrid deep learning architectures and benchmark algorithms. Furthermore, the suggested framework detection accuracy reached an average of 99%.

Xiong, S., and H. Zhang [45] tested numerous methods for detecting malware of Android OS, including static, dynamic, and ML analysis. However, classic single-model ML techniques have limits in terms of generalizability. Different malware behaviors. To address this, a multimodel fusion strategy is proposed in this study. The approach uses numerous ML classifiers, such as LR, DT, and KNN, to increase the detection accuracy, which reached 88%. The fusion method is more effective than separate models in detecting AM, providing a more balanced and robust approach. This approach demonstrates how ensemble techniques can improve prediction accuracy and provide insights for future cybersecurity research.

Another study by Zhu et al. [46] presented a model called SEDMDroid to ensure individual diversity. It generates subsets via random feature subspaces and bootstrapping sample approaches and then performs PCA on each subset. The accuracy is tested by training each base learner in the MLP using the entire dataset while keeping all of the major components. The SVM is then used as the fusion classifier to extract implicit supplemental information from the ensemble members' output and produce the final prediction result. We present experimental findings from two independent datasets gathered via static analysis to demonstrate the usefulness of SEDMDroid. The first extracts permissions, sensitive APIs, monitoring system events, and other features commonly used in AM, and SEDMDroid achieves an accuracy of 89.07% for these multilevel static features. The second dataset, a large dataset, extracts sensitive data flow information as features, with an average accuracy of 94.92%. The experimental results prove that the proposed method is good at detecting AM.

A study by AlSobeh et al. [47]. A framework that uses the KronoDroid dataset to extract time-correlated features and create time-aware and time-agnostic ML models is suggested. The last modification date attribute is critical for time-based classification. Real-device detection trumps emulator-based detection. Time-correlated features improve the detection performance, resulting in a 99.98% F1 score in a time-agnostic situation. Over 12 years, the time-aware classifier outperforms typical ML detection models, achieving an average F1 score of 91% and a maximum F1 score of 99%.

Table I is a summary of all the previous studies.

TABLE I. SUMMARY OF THE PREVIOUS STUDIES

Reference	Model	Performance Metrics	Dataset	Limitations
[35]	DL-AMDet (CNN-BiLSTM, Deep Autoencoders)	F score: 99.935%	CICMalDroid 2020 And Androzoo	High Resource Requirements and Processing Time Overhead
[36]	RF Classifier	Accuracy: 97.82%,	CICMalDroid 2020	Computational Overhead and Class Imbalance Strategy Issues
[37]	ANN, Logistic Regression, KNN, Gradient Boosting, etc.	Accuracy: 0.99, Precision: 0.99, Recall: 0.98, F-Fmeasure: 0.99	Android Malware Detection	Computational Resource Requirements and Overfitting Concerns
[38]	Bayesian Classification	Accuracy: 91.1%	Drebin and AndroZoo	low Detection Accuracy and No Comparative Metrics like Recall, Precision, and F-measure
[39]	SeGDroid (Graph CNN)	F score: 98% (Malware Detection), Accuracy: 96% (Malware Classification)	CICMal2020	Complex graph extraction; computationally heavy and Limited Interpretability
[40]	Random Forest	Best Accuracy: 97%, False Positive Rate: 97%	NATICUSdroid	static permissions, lack of diverse datasets
[41]	RF,KNN,LR, J45, SVM	Accuracy: 98%	CIC_MALDROID2020, Drebin-215 ,Malgenome-215	Imbalance Handling and ability to detect sophisticated or zero-day malware
[42]	Machine Learning Framework (MLDroid)	Detection Rate: 98.8%	Drebin	Performance drops for new malware variants
[43]	Lightweight MLP, Lightweight CNN	Accuracy: 98.12%	AndroZoo	Limited Malware Varieties and Class Imbalance
[44]	Hybrid DL (CNN, BiLSTM)	Accuracy: 98.86%	Androzoo and Android Malware Dataset	higher resource usage
[45]	Multimodel Fusion (LR, DT, KNN)	Accuracy: 88%	Drebin, AndroZoo	low Detection Accuracy
[46]	SEDMDroid (Stacking Ensemble)	Accuracy: (Dataset 1): 89.07% (Dataset 2) :94.92%	Dataset 1 Dataset 2	low Detection Accuracy and Ensemble requires careful tuning
[47]	Time-aware ML Models	F1 Score: 99.98% (Time-aware), Average F1 Score: 91%	KronoDroid	time-aware models require continuous data streams and FP may increase due to Malware labels change over time

3. PROPOSED MODEL

The proposed model aims to assess the efficiency of the LR, ANN, and DT classifiers in AM detection. An Orange data mining tool was used to implement the proposed model. The overall research methodology is shown in Figure 1.

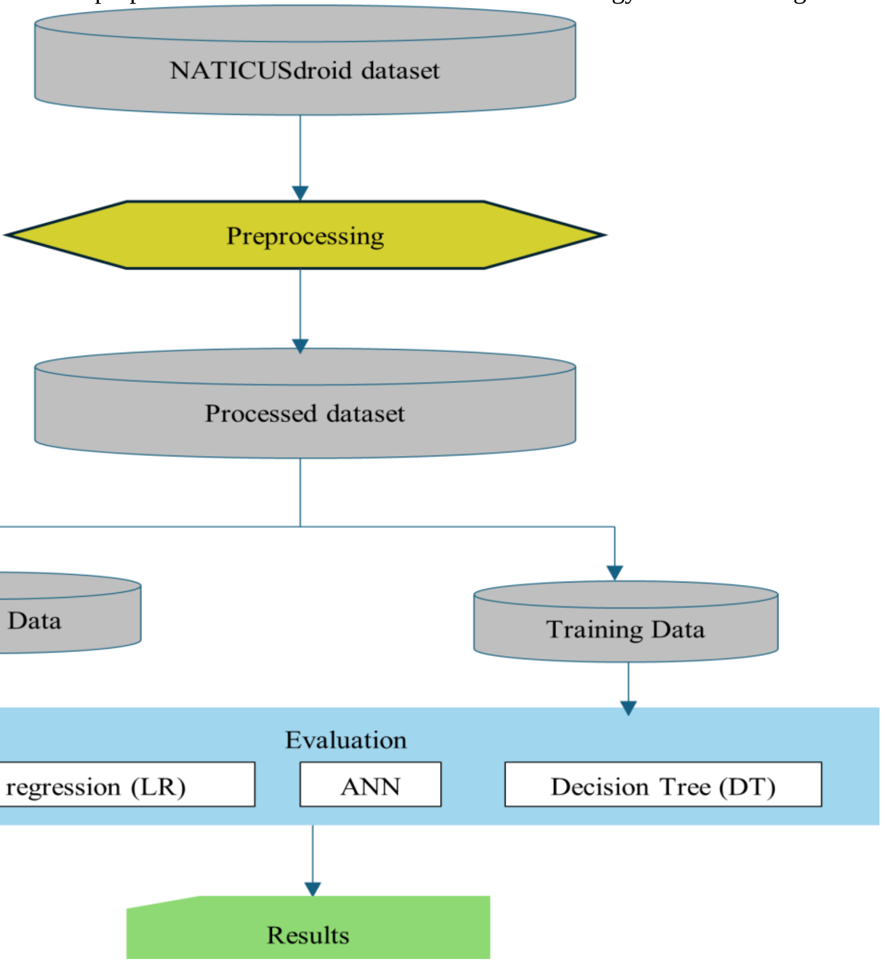


Fig. 1. Overall research methodology

Next, each stage of the proposed model architecture is discussed.

3.1 Dataset: NATICUSdroid (Android Permissions)

In this study, the NATICUSdroid dataset [40] was used, and it was balanced. It has 86 unique features, and it contains 29,332 permission samples issued between 2010 and 2019, 14,700 examples of benign permissions, and 14,632 examples of malicious permissions. The binary numbers 0 and 1 were used in the datasets to encode the data. Certain permissions are indicated by these numbers, either present or not. Table II compares the most recent android malware dataset.

TABLE II. ANDROID MALWARE DATASET

Criteria	**NATICUSdroid**	**CICMalDroid 2020**	**CIC-AAGM2017**	**CIC-AndMal2017**
Year of Release	2022	2020	2017	2017
Source/Provider	NATICUS Research Group	Canadian Institute for Cybersecurity (CIC)	Canadian Institute for Cybersecurity	Canadian Institute for Cybersecurity
Access Type	Public (upon request)	Public	Public	Public
Benign/Malicious Ratio	Balanced	unbalanced	unbalanced	unbalanced
Feature Types	Permissions, API calls, behavior events	Static + dynamic	Static + network features	Static + network features

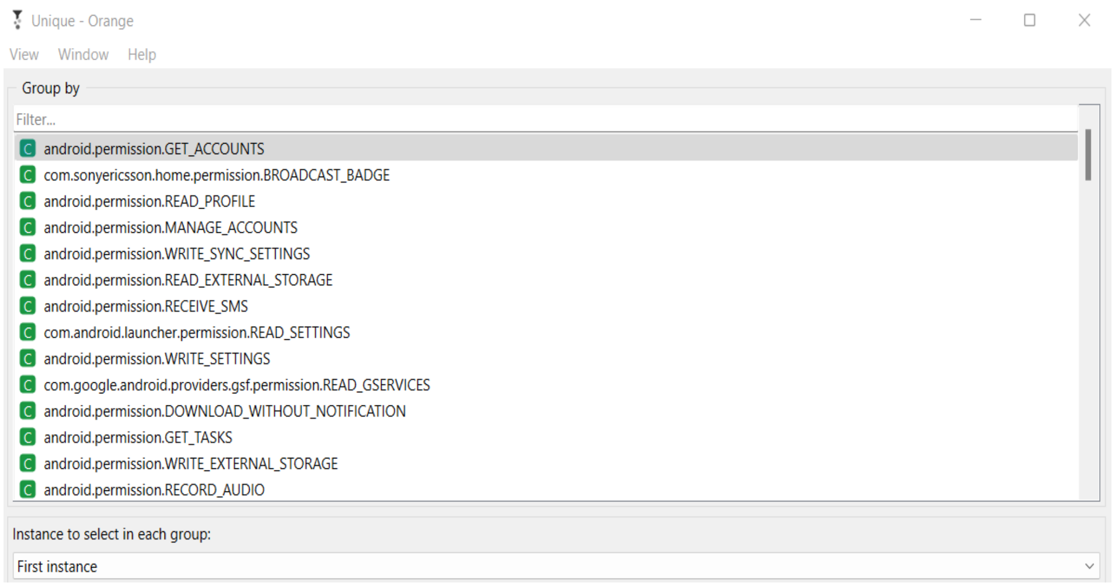
NATICUSdroid provides an up-to-date and curated balanced dataset to reflect modern threats from Android malwares. The NATICUSdroid dataset is used for direct experimental evaluation to achieve reproducibility and relevance.

3.2 Data preprocessing

Preprocessing datasets is a crucial phase in ML, in which data vectors are transformed into modified versions to preserve valuable information, eliminate problems, and improve data quality. NATICUSdroid is balanced, with 14,700 records benign and 14,632 records of malware. In ML, a balanced dataset is necessary to prevent bias towards the dominant label and enable precise generalization across all labels. To reduce overfitting, duplicate instances have been removed from larger datasets [48], and dataset permissions and APIs are thoroughly reviewed in the preprocessing stage. The preprocessing step has to take some substages to enhance the data quality and achieve a fair comparison of classifiers.

3.2.1 Duplicate removal

Once the dataset is imported, the feature types and the target label are checked. It is worth removing redundant rows to minimize overfitting among the redundant samples. Within Orange, an open-source ML and data visualization system, one can find and discard duplicate instances relying on a unique widget that removes rows depending on the properties of one set. This widget checks the replications and compares values on the various columns used and only stores the first time that value is used. This functionality can be used by researchers to clean datasets prior to analysis to eliminate redundancy and strengthen model accuracy. The unique widget allows one to use not only exact matching but also customizable similarity thresholds so that the widget can be applied to different data types with categorical and numerical features. Through incorporation of this tool into their workflow, data scientists will be able to increase the robustness of their analyses with a substantially reduced computational overhead caused by redundant data. Figure 2 displays the unique widget employed to eliminate the duplicated instances in the NATICUSdroid dataset.



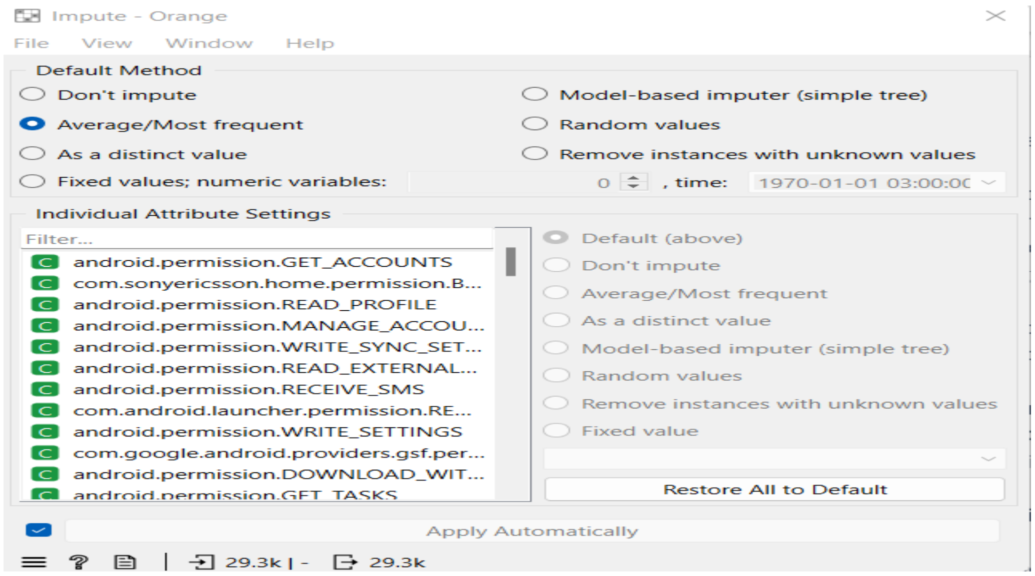
android.permission.GET_ACCOUNTS
com.sonyericsson.home.permission.BROADCAST_BADGE
android.permission.READ_PROFILE
android.permission.MANAGE_ACCOUNTS
android.permission.WRITE_SYNC_SETTINGS
android.permission.READ_EXTERNAL_STORAGE
android.permission.RECEIVE_SMS
com.android.launcher.permission.READ_SETTINGS
android.permission.WRITE_SETTINGS
com.google.android.providers.gsf.permission.READ_GSERVICES
android.permission.DOWNLOAD_WITHOUT_NOTIFICATION
android.permission.GET_TASKS
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.RECORD_AUDIO

Instance to select in each group: First instance

Fig. 2. Unique widget

3.2.2 Missing data handling

Missing data are one of the problems that negatively impacts the learning ability of a machine. Orange data mining offers powerful facilities to manage missing values by using imputation methods such that all the data are complete and an analytical metric can be accurate. The software implements several imputation strategies: mean, median, and mode imputation on a numerical and categorical value scale, in addition to more complex approaches, including k-nearest neighbors (KNN) and model-based imputation. Using the easy-to-use visual programming interface of Orange, users can easily plug-in imputation as part of their data preprocessing routine. The impute widget enables investigators to choose the most suitable method depending on the data pattern of missingness and distribution to reach the lowest level of bias and maintain statistical integrity. Moreover, orange can be useful when analysing comparisons because it allows for the determination of the effects of various imputation techniques on the results of performing modelling in a session to ensure the best data treatment. The "**Average/Most-**Frequent" method is the most common approach; therefore, it was considered in this study. This functionality contributes to the effectiveness of Orange as a tool that allows researchers to confront the missing data issue with some efficiency and, at the same time, ensures the integrity of the dataset to be used when addressing the issue of missing data by means of machine learning in the future. Figure 3 shows the imputed widget utilized to remove the missing values in the NATICUSdroid dataset.


Impute - Orange

Default Method
- Don't impute
- [✓ Average/Most frequent]
- As a distinct value
- Fixed values; numeric variables: 0 , time: 1970-01-01 03:00:0C
- Model-based imputer (simple tree)
- Random values
- Remove instances with unknown values

Individual Attribute Settings

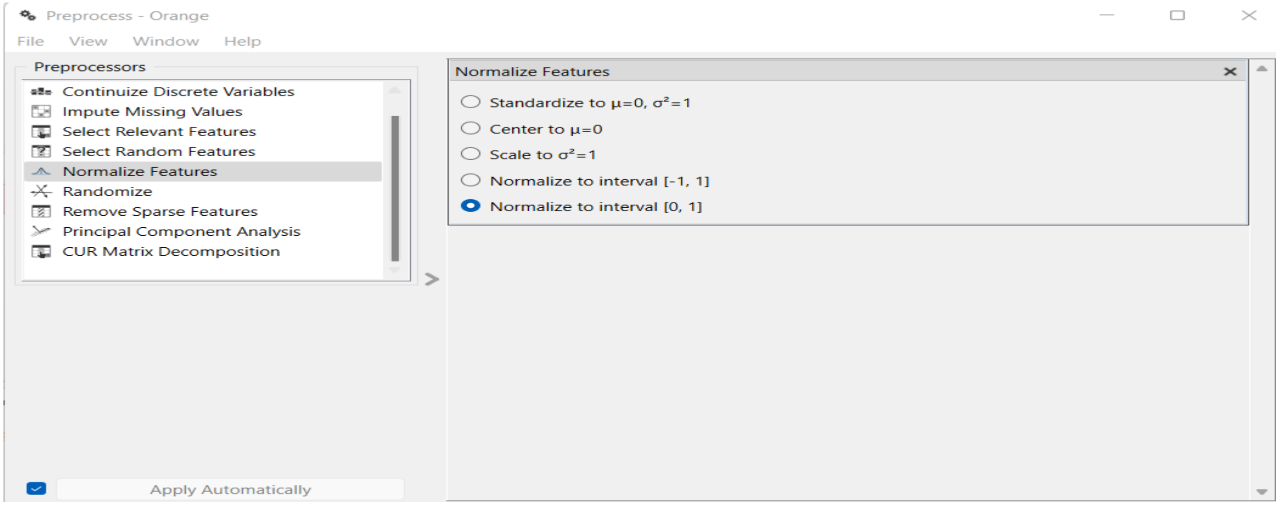
- android.permission.GET_ACCOUNTS
- com.sonyericsson.home.permission.B...
- android.permission.READ_PROFILE
- android.permission.MANAGE_ACCOU...
- android.permission.WRITE_SYNC_SET...
- android.permission.READ_EXTERNAL...
- android.permission.RECEIVE_SMS
- com.android.launcher.permission.RE...
- android.permission.WRITE_SETTINGS
- com.google.android.providers.gsf.per...
- android.permission.DOWNLOAD_WIT...
- android.permission.GET_TASKS

- [✓ Default (above)]
- Don't impute
- Average/Most frequent
- As a distinct value
- Model-based imputer (simple tree)
- Random values
- Remove instances with unknown values
- Fixed value

Fig. 3. Impute widget

3.2.3 Normalization

Normalization is a necessary method of preprocessing data in data mining practices that determines the features that make a similar contribution to analytical models by scaling those features to others of a standardized range. The min–max normalization method transforms numerical features to a user-specified range via a linear transformation, usually [0, 1]. The strategy retains the original distribution of the data but neutralizes the effect of different scales; hence, it is beneficial to algorithms whose performance depends on the magnitude of features. In Orange, one can use min–max normalization via the preprocess widget, which has a more intuitive interface, to set the scaling parameters. This approach standardizes features into a range of [0, 1], which increases the accuracy of the model, results in less bias toward highly scaled variables and accelerates the convergence rate of gradient-based optimization methods. Nevertheless, Min–Max can be easily affected by outliers, as the extreme values are capable of overly compressing the transformed data. In this way, the outliers should be treated before normalization to achieve the best possible results. Figure 4 shows the preprocessing widget utilized to scale the values in the NATICUSdroid dataset.


Preprocess - Orange

Preprocessors
- Continuize Discrete Variables
- Impute Missing Values
- Select Relevant Features
- Select Random Features
- Normalize Features
- Randomize
- Remove Sparse Features
- Principal Component Analysis
- CUR Matrix Decomposition

Normalize Features
- Standardize to $\mu=0$, $\sigma^2=1$
- Center to $\mu=0$
- Scale to $\sigma^2=1$
- Normalize to interval [-1, 1]
- [✓ Normalize to interval [0, 1]]

Fig. 4. Preprocessing widget

After the preprocessing steps of the NATICUSdroid dataset are complete. The preprocessed data are then partitioned into 70% training and 30% testing sets, and three distinct classifiers are then employed—LR, ANN, and DT—to detect the AM.

3.3 Orange3 Data mining tool

Orange 3 [31] is a data mining application that employs Python scripting and visual programming to analyse data and evaluate ML and DL model validity. The primary goal of Orange 3 is to enable users to create simple Python scripts that extend C++ implementations of computationally intensive activities. Orange 3 is intended for both experienced users and coders. Figure 5 shows the full orange implementation of the model to detect the AM via LR, ANN, and DT.

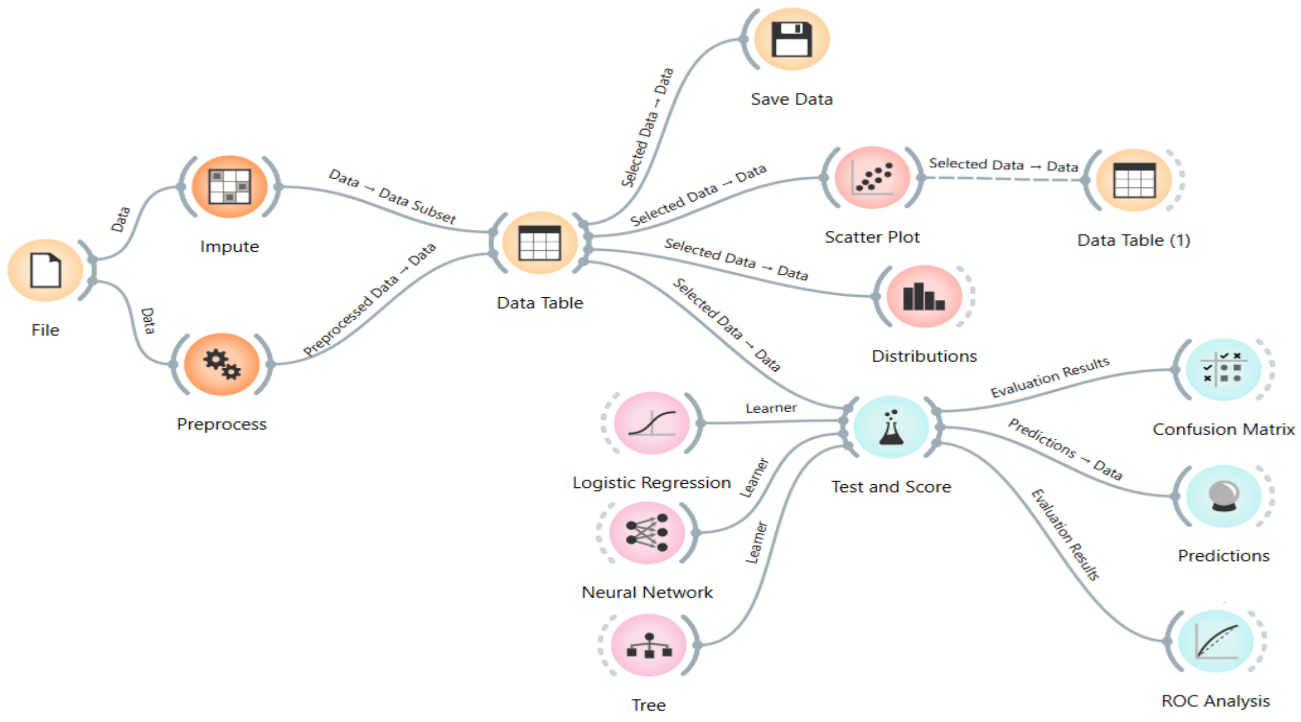


Fig. 5. Model Implementation using Orange

3.4 Classifier

ML and DL are types of AI that focus on developing computer algorithms that evaluate and learn data via intrinsic patterns, gradually improving their accuracy rate [49] [50]. ML can be classified into four categories: supervised, unsupervised, evolutionary, and reinforcement learning [51] [52]. However, AI, ML, and DL are frequently used interchangeably to explain the development of application ideas, referring to machines that are built to learn and identify the best options [53] [54]. Figure 6 shows the relationships among AI, ML, and DL. On the other hand, data classification requires developing and testing a classifier that can categorize unknown classes. The proposed model is validated via split valuation techniques (30% testing and 70% training). Classification is performed with the LR, DT, and ANN classifiers. The LR, DT, and ANN classifiers were chosen to evaluate the proposed model because these classifiers have been used in previous studies to detect the AM.

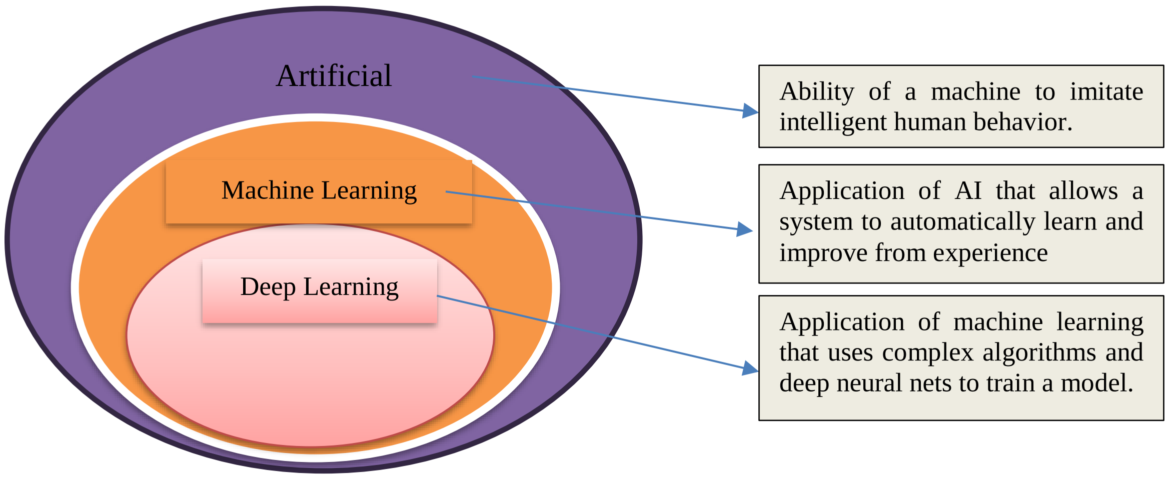

Artificial
Machine Learning
Deep Learning
Ability of a machine to imitate intelligent human behavior.
Application of AI that allows a system to automatically learn and improve from experience
Application of machine learning that uses complex algorithms and deep neural nets to train a model.

Fig. 6. Relationships among AI, ML, and DL [51].

- Logistic Regression (LR)

LR is a supervised learning method used to predict dependent values in datasets and is categorized via regression models [55]. LR can be considered a classification rather than a forecasting technique for constant variables. It is also used to assess discrete values such as 0/1 from a collection of independent variables.

Using the sigmoid function, LR is applied to discontinuous possibilities, which converts quantitative outputs into likelihood assertions. Libraries are imported, the datasets are input and visualized, null/missing values are managed, outliers are assessed, dependent and independent variables are established, and ensemble and boosting methods are employed to increase accuracy. LR can be represented mathematically by equation (1):

$$y = \frac{e(b0+b1x)}{1} + e(b0 + b1x) \tag{1}$$

In the equation, 'x' represents the input value, 'y' represents the expected outcome, 'b0' is the bias or intercept term, and 'b1' is the input coefficient.

- Decision Tree (DT)

DT is used for regression [56], data analysis, and classification. It is made up of decision nodes, edges, and leaf nodes, with each attribute serving as a node. The tree is built from a single node to meet parameters and decisions, finally reaching a terminal node that forecasts the result [57]. Figure 7 shows the DT algorithm structure.

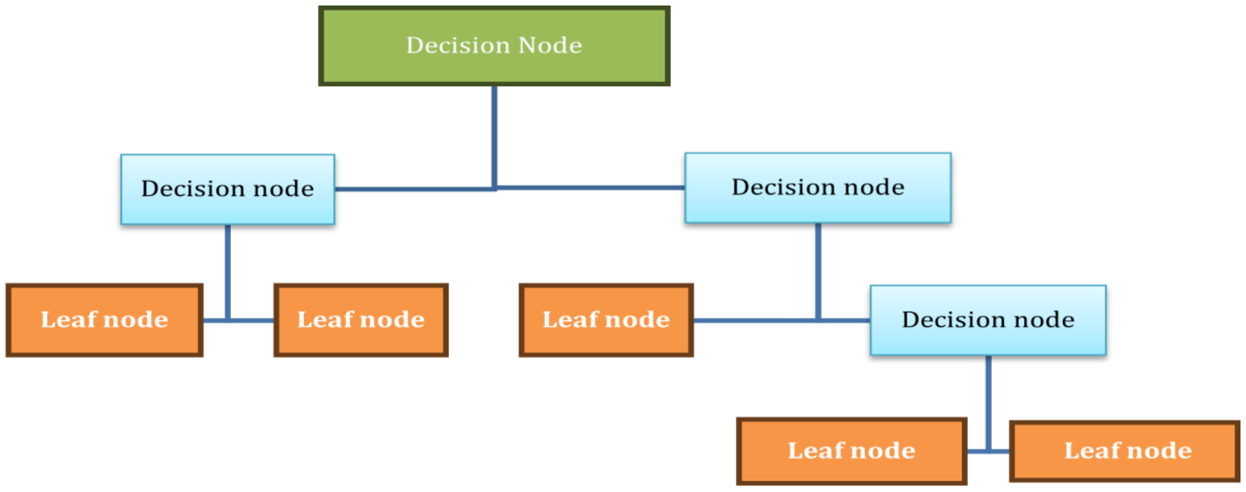


Fig. 7. DT structure

- Artificial Neural Network (ANN)

ANNs are made up of neurons linked by connections that can be changed by learning processes. They calculate activity by combining data from connections, including bias nodes in each layer [58, 59]. However, an ANN is made up of several layers, each containing several neurons. The input is the first layer, while the output is the final layer. Hidden layers are those that exist between layers. Figure 8 shows the structure of an ANN.

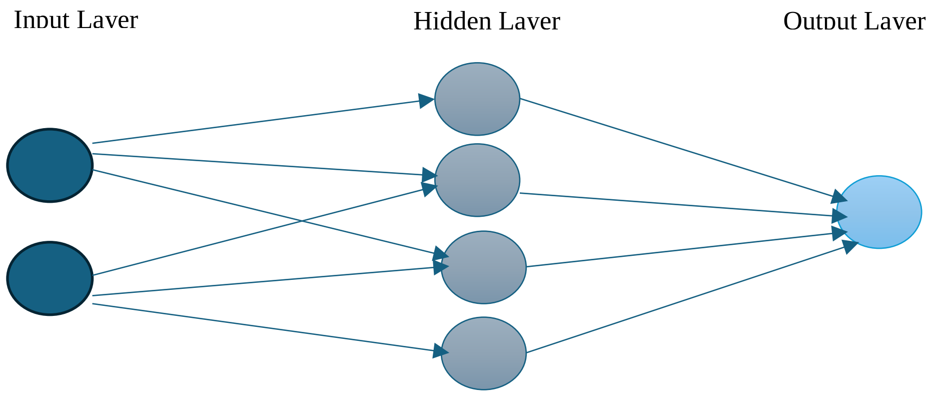


Fig. 8. Structure of ANN [60]

Table III presents the ANN setting parameters that were used in the experiment. The pseudocode of the ANN is shown in Algorithm 1.

TABLE III. ANN SETTING PARAMETERS

Neural in hidden layers	100
Activation Rule	ReLu
Solver	Adam
Regularization Alpha	0.05
Error Threshold	0.0001
Number of iterations	200

Algorithm 1 : Pseudocode of ANN

```
Require: Training dataset $X$, Target labels $Y$, Learning rate $\eta$, Maximum iterations $T_{\max}$, Error threshold $\epsilon$
Ensure: Trained network parameters $W, b$
1: Initialize weights $W$ and biases $b$ randomly
2: for $t = 1$ to $T_{\max}$ do
3:     Feedforward: Compute network output $\hat{Y}$ from $X$ using activation function (e.g., ReLU)
4:     Error Computation: Calculate loss $L(Y, \hat{Y})$
5:     Backpropagation: Compute gradients $\nabla W, \nabla b$ using chain rule
6:     Parameter Update:
           $W \leftarrow W - \eta \cdot \nabla W$
           $b \leftarrow b - \eta \cdot \nabla b$
7:     if $L < \epsilon$ then
8:         break
9:     end if
10: end for
11: return $W, b$
```

4. PERFORMANCE EVALUATION

The confusion matrix (CM) is used to determine the efficacy of the ML and DL detection algorithms [61] [62]. It provides a summary of forecasts per class, with True (1) and False (0) representing actual values and expected values, respectively. To assess the possibilities of the classification model, the CM contains true positive (TP), true negative (TN), false positive (FP), and false negative (FN) entries. The CM's components are the FP, TN, TP, and FN, which assist in evaluating the model's accuracy while dealing with several classes in a dataset. Table IV. General binary CM.

TABLE IV. CONFUSION MATRIX

Actual	**Predicted**	
	(TP)	(FN)
	(FP)	(TN)

4.1 Performance metrics

To evaluate the efficiency of the proposed model in detecting AM, five metrics are used in the evaluation of classification models, such as accuracy, precision, recall (sensitivity), F1 score, and area under the curve (AUC). The measures of each are defined below.

1) Accuracy

The accuracy defines how many of the instances are true compared with the number of all the instances. The accuracy was measured via Formula 2.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.

2) Precision

Precision measures how many of those that were predicted to be malware are malware. The greater the precision is, the fewer false alarms there are. Precision is calculated via formula 3.

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

3) Recall (sensitivity)

The quantity of correctly classified instances of actual malware depending on the totality of the malware present is termed the recall or sensitivity. Good recall avoids several malware detections. Recall is measured by formula (4).

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

4) F1 score

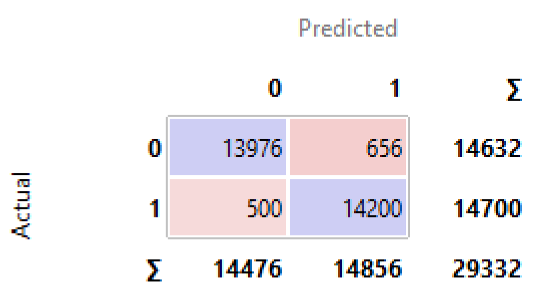
The harmonic mean of the precision and recall gives an equal value of the F1 score, which gives a balanced metric when both precision and recall matter. The F1 score is computed via formula 5.

$$\mathrm{F1-Score} = \frac{2*Precision*\mathrm{Recall}}{Precision+\mathrm{Recall}} \tag{5}$$

5. RESULTS AND DISCUSSION

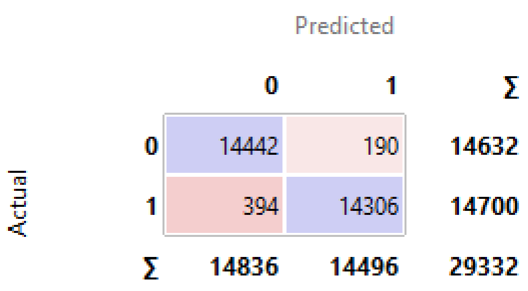
In this section, an informative assessment of the proposed model and a critical contrast with other classifiers and earlier literature are given. The goal is to qualitatively review the efficiency of the ANN-empowered method to counter malware in Android and to define its overall advantages. Different metric-based performance evaluations are performed with the application of the AUC, accuracy, precision, recall, and F1 score, which represents a balanced and reliable assessment. Every metric was selected to outline the predictive power, robustness, and suitability for AM detection.

Figure 9 depicts the confusion matrices obtained, which can provide the first insight into the performance of each classifier in the detection of AM. The ANN classifier results in the fewest misclassifications, with only a few benign apps falsely detected as malware applications and few malwares detected as benign. Conversely, the numbers of false positives and false negatives generated by the LR and DT methods are greater, yet the values of the LR method are superior to those of the DT method. These findings indicate the strong ability of the ANN to discriminate between benign and malicious applications.



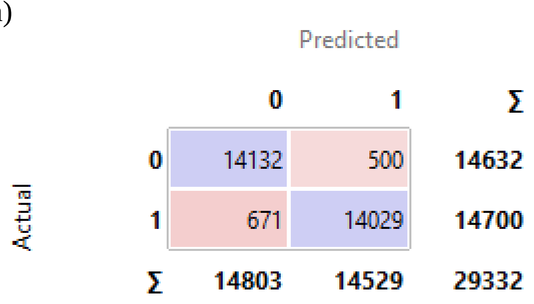
		Predicted		
		0	1	Σ
Actual	0	13976	656	14632
	1	500	14200	14700
	Σ	14476	14856	29332

(a)



		Predicted		
		0	1	Σ
Actual	0	14442	190	14632
	1	394	14306	14700
	Σ	14836	14496	29332

(b)

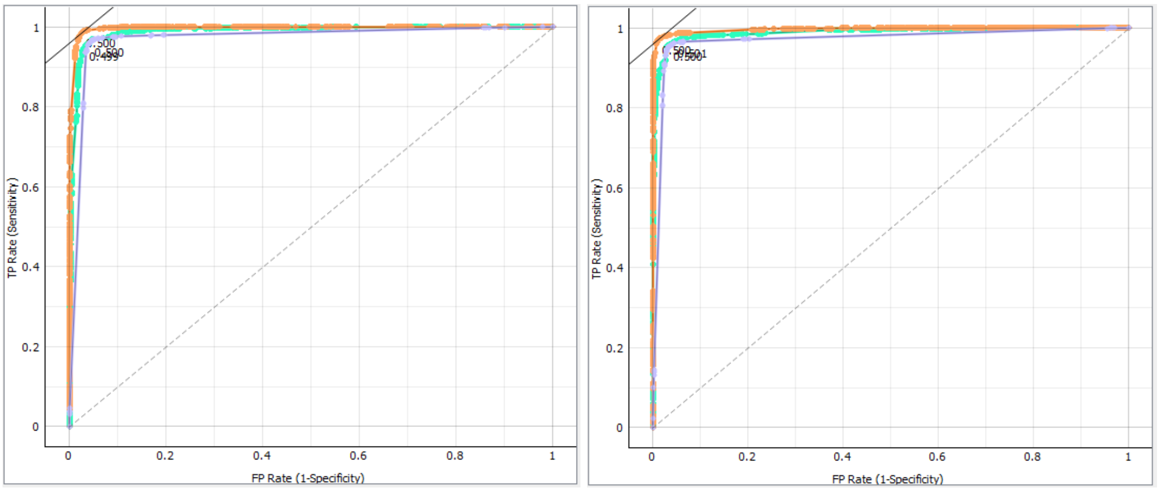


		Predicted		
		0	1	Σ
Actual	0	14132	500	14632
	1	671	14029	14700
	Σ	14803	14529	29332

(c)

Fig. 9. Obtained Confusion matrix (a) LR, (b) ANN, (c) DT.

To further substantiate the performance of the classifiers, receiver operating characteristic (ROC) curves were plotted with each model, as shown in Figure 10. The ANN had an AUC = 0.997, which is near the zero value of complete separability of the ANN model. The superiority of the ANN over the LR model was also revealed, with an AUC of 0.989, indicating moderate performance and less effectiveness than the ANN. The DT performed reasonably well, with an AUC of 0.971, except that overlapping feature spaces were slightly cumbersome. The results verify that the ANN is the best at identifying AMs.



(a) (b)

Fig. 10. ROC curves of (a) Class 0 (no malware was detected) and (b) Class 1 (malware was detected)

This finding is also further supported by the accuracy results. The ANN was very good, with an accuracy of 98.0%, whereas the accuracies of the LR and DT were 96.1% and 96.0%, respectively. The margin of almost 2% indicates that the ANN is better since it correctly classifies malware as well as benign. This increase is especially significant in security systems, where a slight increase in accuracy can have a large effect on the mitigation of risks.

In terms of precision, the ANN achieves a precision of 0.980 first, followed by the LR (0.961) and DT (0.960) methods. Specifically, most of the interrupted applications detected as malware were indeed malicious and thereby reduced false positive surveillance. This is required for use in real life; since there can be many false positives, it may saturate the users or system administrators with alerts that are not needed.

Recall or sensitivity is also important, as it indicates the accuracy of the model in identifying malware without falsely identifying threats. The ANN achieved a recall of 0.980 in that almost all malicious samples were recognized correctly. LR and DT (0.961) were good and were likely to miss some malware samples. With cybersecurity, superior recall is especially worthwhile because unnoticed malware can be catastrophic.

Finally, the F1 score, which guarantees precision and recall, is superior to the ANN. The ANN returned an F1 score of 0.980, and the LRs and DTs were 0.961 and 0.960, respectively. The fact that all the values consistently decreased with ANN further demonstrates that the latter technique not only reduces false positives but also guarantees high coverage with respect to AM detection. A performance matrix comparison with classifiers is shown in Figure 11.

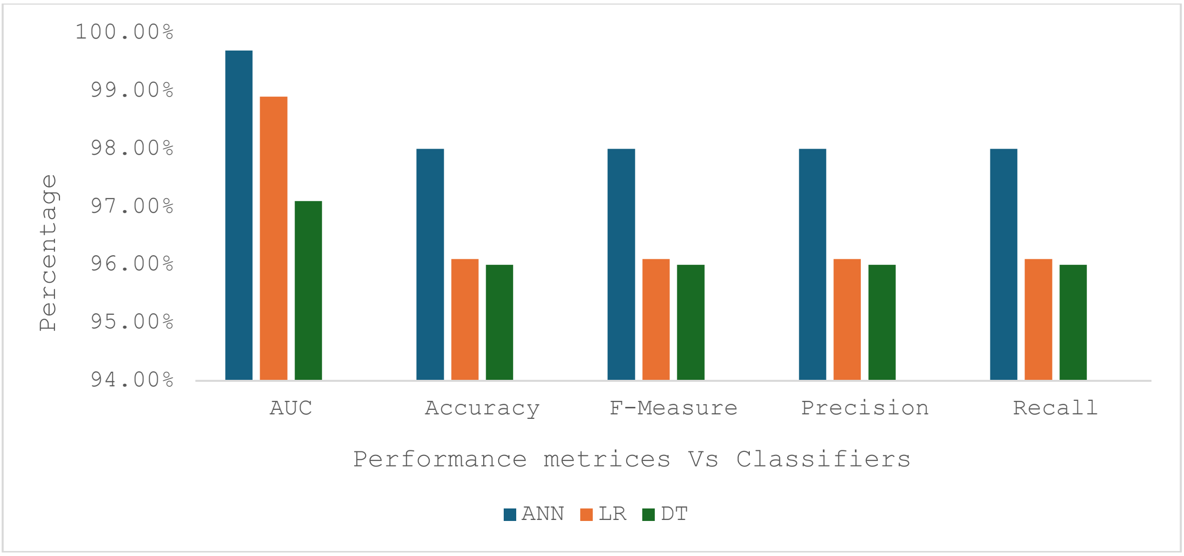


Fig. 11. Performance matrix comparison versus classifiers

Table V presents a comparative analysis of the study results with those of previous studies. Accuracies with previous works were between 91.1% [34] and 98.8% [38]. Conversely, the accuracy of the proposed ANN-based model was 98.0 and almost perfect, with an AUC value of 0.997, which is considerably higher and stronger than those of the majority of the previous methodologies.

TABLE V. COMPARISON OF THE PROPOSED MODEL WITH PREVIOUS MODELS

Study\Year	Methods\Model	Accuracy	AUC	Notes
[38]/2022	chi-square algorithms	91.1	N/A	Lower detection
[42]/2021	Deep learning algorithms, farthest first clustering, Y-MLP, and nonlinear ensemble decision tree forest approach	98.8	N/A	High accuracy, but less robust under imbalanced data
[43]/2024	Multilayer Perceptron and Convolutional Neural Network	98.12	N/A	Strong performance, but limited scalability
Proposed model	DT, LR, and ANN	98.0	97.7	Balanced performance across all metrics, robust and scalable

6. CONCLUSION

The present research demonstrates the importance of performing a comparative analysis between the ML and DL methods for AM detection on a systematic basis. In addition to raw performance, the results yield insightful data: ANN is a better choice for modelling AM patterns, LR is more accurate and computationally efficient for resource-constrained devices, and DT is more readable and can aid the goal of explainable security. This finding indicates that the selection of a classifier must involve a tradeoff between precision, interpretability, and resource limits, according to implementation requirements. The major advantage of this work is that a coherent assessment of ML and DL identifiers can provide practical recommendations for implementing flexible and effective mobile safety systems. The implication is further realized in the form of enhancing the privacy of users and their resistance to malware threats that continue evolving. The proposed model showed high performance, but because it was tested on only one dataset, its ability to generalize the results to broader real-life situations is limited. In addition, the use of static features and the complexity of the general ANN model pose issues of flexibility, explainability, and execution on devices with limited resources. To resolve these problems, future work will include several datasets, combinations of both dynamic and static features, and lightweight, explainable deep learning models that are applicable in mobile settings.

Conflicts of interest

"The authors declare no conflicts of interest".

Funding

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia (Grant No. KFU 251369).

References

[1] S. Otoom, "Risk auditing for Digital Twins in cyber physical systems: A systematic review," *Journal of Cyber Security and Risk Auditing*, vol. 2025, no. 1, pp. 22-35, 2025.

[2] E. Alotaibi, R. B. Sulaiman, and M. Almaiah, "Assessment of cybersecurity threats and defense mechanisms in wireless sensor networks," *ournal of Cyber Security Risk Auditing*, vol. 2025, no. 1, pp. 47-59, 2025.

[3] A. Alsaaidah, O. Almomani, A. A. Abu-Shareha, M. M. Abualhaj, and A. Achuthan, "ARP Spoofing Attack Detection Model in IoT Network using Machine Learning: Complexity vs. Accuracy," *Journal of Applied Data Sciences*, vol. 5, no. 4, pp. 1850-1860, 2024.

[4] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE access*, vol. 8, pp. 6249-6271, 2020.

[5] S. Ang, M. Ho, S. Huy, and M. Janarthanan, "Utilizing IDS and IPS to improve cybersecurity monitoring process," *Journal of Cyber Security and Risk Auditing*, vol. 2025, no. 3, pp. 77-88, 2025.

[6] B. Almelehy, M. Ahmad, G. Nassreddine, M. Maayah, and A. Achanta, "Analytical analysis of cyber threats and defense mechanisms for web application security," *Journal of Cyber Security and Risk Auditing*, vol. 2025, no. 3, pp. 57-76, 2025.

[7] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, vol. 112, p. 101861, 2021.

[8] R. S. Mousa and R. Shehab, "Applying risk analysis for determining threats and countermeasures in workstation domain," *Journal of Cyber Security and Risk Auditing*, vol. 2025, no. 1, pp. 12-21, 2025.

[9] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Engineering Applications of Artificial Intelligence*, vol. 122, p. 106030, 2023.

[10] M. M. Abualhaj, M. Al-Zyoud, A. Alsaaidah, A. Abu-Shareha, and S. Al-Khatib, "Enhancing Malware Detection through Self-Union Feature Selection Using Firefly Algorithm with Random Forest Classification," *International Journal of Intelligent Engineering Systems*, vol. 17, no. 4, 2024.

[11] S. Alsahaim and M. Maayah, "Analyzing Cybersecurity Threats on Mobile Phones," *STAP Journal of Security Risk Management*, vol. 1, pp. 3-19, 2023.

[12] M. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Computer Science Review*, vol. 47, p. 100529, 2023.

[13] O. Almomani, A. Alsaaidah, M. A. Almaiah, A. Alzaqebah, M. M. Abualhaj, and W. J. Alzyadat, "Evaluating Machine Learning Classifiers for Detecting Distributed Denial of Service Attacks," in *2025 12th International Conference on Information Technology (ICIT)*, 2025: IEEE, pp. 134-140.

[14] M. A. Al-Shareeda, A. A. Obaid, and A. A. H. Almajid, "The Role of Artificial Intelligence in Bodybuilding: A Systematic Review of Applications, Challenges, and Future Prospects," *Jordanian Journal of Informatics and Computing*, vol. 2025, no. 1, pp. 16-26, 2025.

[15] O. Almomani, A. Alsaaidah, M. M. Abualhaj, M. A. Almaiah, A. Almomani, and S. Memon, "URL Spam Detection Using Machine Learning Classifiers," in *2025 1st International Conference on Computational Intelligence Approaches and Applications (ICCIAA)*, 2025: IEEE, pp. 1-6.

[16] A. A. Almuqren, "Cybersecurity threats, countermeasures and mitigation techniques on the IoT: Future research directions," *Journal of Cyber Security and Risk Auditing,* vol. 1, no. 1, pp. 1-11, 2025.

[17] T. Alsalem and M. Amin, "Towards Trustworthy IoT Systems: Cybersecurity Threats, Frameworks, and Future Directions," *Journal of Cyber Security and Risk Auditing,* vol. 2023, no. 1, pp. 3-18, 2023.

[18] M. M. Abualhaj, M. O. Hiari, A. Alsaaidah, M. Al-Zyoud, and S. J. J. o. A. D. S. Al-Khatib, "Spam Feature Selection Using Firefly Metaheuristic Algorithm," *Journal of Applied Data Sciences,* vol. 5, no. 4, pp. 1692-1700, 2024.

[19] R. Almanasir, D. Al-solomon, S. Indrawes, M. Almaiah, U. Islam, and M. Alshar'e, "Classification of threats and countermeasures of cloud computing," *Journal of Cyber Security and Risk Auditing,* vol. 2025, no. 2, pp. 27-42, 2025.

[20] O. Aljumaiah, W. Jiang, S. R. Addula, and M. A. Almaiah, "Analyzing cybersecurity risks and threats in IT infrastructure based on NIST framework," *Journal of Cyber Security and Risk Auditing,* vol. 2025, no. 2, pp. 12-26, 2025.

[21] W. A. H. Salman and C. H. Yong, "Overview of the CICIoT2023 Dataset for Internet of Things Intrusion Detection Systems," *Mesopotamian Journal of Big Data,* vol. 2025, pp. 50-60, 2025.

[22] S. J. Kattamuri, R. K. V. Penmatsa, S. Chakravarty, and V. S. P. Madabathula, "Swarm optimization and machine learning applied to PE malware detection towards cyber threat intelligence," *Electronics,* vol. 12, no. 2, p. 342, 2023.

[23] D. A. Kadhim and M. A. Mohammed, "Advanced machine learning models for accurate kidney cancer classification using CT images," *Mesopotamian Journal of Big Data,* vol. 2025, pp. 1-25, 2025.

[24] E. S. Alomari *et al.*, "Malware detection using deep learning and correlation-based feature selection," *Symmetry,* vol. 15, no. 1, p. 123, 2023.

[25] M. A. Almedires, A. Elkhalil, and M. Amin, "Adversarial attack detection in industrial control systems using LSTM-based intrusion detection and black-box defense strategies," *Journal of Cyber Security and Risk Auditing,* vol. 2025, no. 3, pp. 4-22, 2025.

[26] S. R. Addula, S. Norozpour, and M. Amin, "Risk Assessment for Identifying Threats, vulnerabilities and countermeasures in Cloud Computing," *Jordanian Journal of Informatics and Computing,* vol. 2025, no. 1, pp. 37-48, 2025.

[27] A. Alshuaibi, M. Almaayah, and A. Ali, "Machine learning for cybersecurity issues: A systematic review," *Journal of Cyber Security and Risk Auditing,* vol. 2025, no. 1, pp. 36-46, 2025.

[28] H. Lee, S. Kim, D. Baek, D. Kim, and D. Hwang, "Robust IoT malware detection and classification using opcode category features on machine learning," *IEEE Access,* vol. 11, pp. 18855-18867, 2023.

[29] M. M. Abualhaj, S. N. Alkhatib, A. A. Abu-Shareha, A. M. Alsaaidah, and M. Anbar, "Enhancing spam detection using Harris Hawks optimization algorithm," *TELKOMNIKA,* vol. 23, no. 2, pp. 447-454, 2025.

[30] H. Albinhamad, A. Alotibi, A. Alagnam, M. Almaiah, and S. Salloum, "Vehicular Ad-hoc Networks (VANETs): A Key Enabler for Smart Transportation Systems and Challenges," *Jordanian Journal of Informatics and Computing,* vol. 2025, no. 1, pp. 4-15, 2025.

[31] J. Demšar *et al.*, "Orange: data mining toolbox in Python," *Journal of machine Learning research,* vol. 14, no. 1, pp. 2349-2353, 2013.

[32] A. Ali, "Adaptive and Context-Aware Authentication Framework Using Edge AI and Blockchain in Future Vehicular Networks," *STAP Journal of Security Risk Management,* vol. 1, pp. 45-56, 2024.

[33] M. Al-Shareeda, L. Najm, A. Hassan, S. Mushtaq, and H. Ali, "Secure IoT-Based Smart Agriculture System Using Wireless Sensor Networks for Remote Environmental Monitoring," *STAP Journal of Security Risk Management,* vol. 1, pp. 56-66, 2024.

[34] M. Almaayah and R. Sulaiman, "Cyber Risk Management in the Internet of Things: Frameworks, Models, and Best Practices," *STAP Journal of Security Risk Management,* vol. 1, pp. 3-23, 2024.

[35] A. R. Nasser, A. M. Hasan, and A. J. Humaidi, "DL-AMDet: Deep learning-based malware detector for android," *Intelligent Systems with Applications,* vol. 21, p. 200318, 2024.

[36] A. Alhogail and R. A. Alharbi, "Effective ML-Based Android Malware Detection and Categorization," *Electronics,* vol. 14, no. 8, p. 1486, 2025.

[37] M. A. Habeeb and Y. L. Khaleel, "Enhanced Android Malware Detection through Artificial Neural Networks Technique," *Mesopotamian Journal of CyberSecurity,* vol. 5, no. 1, pp. 62-77, 2025.

[38] S. R. T. Mat, M. F. Ab Razak, M. N. M. Kahar, J. M. Arif, and A. Firdaus, "A Bayesian probability model for Android malware detection," *ICT Express,* vol. 8, no. 3, pp. 424-431, 2022.

[39] Z. Liu, R. Wang, N. Japkowicz, H. M. Gomes, B. Peng, and W. Zhang, "SeGDroid: An Android malware detection method based on sensitive function call graph learning," *Expert Systems with Applications,* vol. 235, p. 121125, 2024.

[40] A. Mathur, L. M. Podila, K. Kulkarni, Q. Niyaz, and A. Y. Javaid, "NATICUSdroid: A malware detection framework for Android using native and custom permissions," *Journal of Information Security Applications,* vol. 58, p. 102696, 2021.

[41] E. Odat and Q. M. Yaseen, "A novel machine learning approach for android malware detection based on the co-existence of features," *IEEE Access,* vol. 11, pp. 15471-15484, 2023.

[42] A. Mahindru and A. L. Sangal, "MLDroid—framework for Android malware detection using machine learning techniques," *Neural Computing Applications,* vol. 33, no. 10, pp. 5183-5240, 2021.

[43] R. Ma, S. Yin, X. Feng, H. Zhu, and V. S. Sheng, "A lightweight deep learning-based android malware detection framework," *Expert Systems with Applications,* vol. 255, p. 124633, 2024.

[44] I. U. Haq, T. A. Khan, and A. Akhunzada, "A dynamic robust DL-based model for android malware detection," *IEEE Access,* vol. 9, pp. 74510-74521, 2021.

[45] S. Xiong and H. Zhang, "A Multi-model Fusion Strategy for Android Malware Detection Based on Machine Learning Algorithms," *Journal of Computer Science Research,* vol. 6, no. 2, pp. 1-11, 2024.

[46] H. Zhu, Y. Li, R. Li, J. Li, Z. You, and H. Song, "SEDMDroid: An enhanced stacking ensemble framework for Android malware detection," *IEEE Transactions on Network Science and Engineering,* vol. 8, no. 2, pp. 984-994, 2020.

[47] A. M. AlSobeh, K. Gaber, M. M. Hammad, M. Nuser, and A. Shatnawi, "Android malware detection using time-aware machine learning approach," *Cluster Computing,* pp. 1-22, 2024.

[48] P. B. Pio, A. Rivolli, A. C. d. Carvalho, and L. P. Garcia, "A review on preprocessing algorithm selection with meta-learning," *Knowledge Information Systems,* vol. 66, no. 1, pp. 1-28, 2024.

[49] A. Arabiat and M. Altayeb, "Assessing the effectiveness of data mining tools in classifying and predicting road traffic congestion," *Indonesian Journal of Electrical Engineering Computer Science,* vol. 34, no. 2, pp. 1295-1303, 2024.

[50] M. Madi, F. Jarghon, Y. Fazea, O. Almomani, and A. Saaidah, "Comparative analysis of classification techniques for network fault management," *Turkish Journal of Electrical Engineering Computer Sciences,* vol. 28, no. 3, pp. 1442-1457, 2020.

[51] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Machine learning and deep learning approaches for cybersecurity: A review," *IEEE Access,* vol. 10, pp. 19572-19585, 2022.

[52] A. Arabiat, M. Hassan, and O. Al Momani, "Traffic congestion prediction using machine learning: Amman City case study," in *International Conference on Medical Imaging, Electronic Imaging, Information Technologies, and Sensors (MIEITS 2024),* 2024, vol. 13188: SPIE, pp. 38-45.

[53] M. Altayeb and A. Arabiat, "Crack detection based on mel-frequency cepstral coefficients features using multiple classifiers," *International Journal of Electrical Computer Engineering,* vol. 14, no. 3, 2024.

[54] O. Almomani, A. Alsaaidah, A. A. A. Shareha, A. Alzaqebah, M. Almomani, and Applications, "Performance Evaluation of Machine Learning Classifiers for Predicting Denial-of-Service Attack in Internet of Things," *International Journal of Advanced Computer Science,* vol. 15, no. 1, 2024.

[55] J. Arunkumar, S. Velmurugan, B. Chinnaiah, G. Charulatha, M. R. Prabhu, and A. P. Chakkaravarthy, "Logistic Regression with Elliptical Curve Cryptography to Establish Secure IoT," *Computer Systems Science Engineering,* vol. 46, no. 1, 2023.

[56] A. Almomani *et al.*, "Ensemble-based approach for efficient intrusion detection in network traffic," *Intelligent Automation & Soft Computing,* vol. 37, no. 2, 2023.

[57] M. M. Abualhaj, A. S. Al-Shamayleh, A. Munther, S. N. Alkhatib, M. O. Hiari, and M. Anbar, "Enhancing spyware detection by utilizing decision trees with hyperparameter optimization," *Bulletin of Electrical Engineering Informatics,* vol. 13, no. 5, pp. 3653-3662, 2024.

[58] G. Tzougas and K. Kutzkov, "Enhancing logistic regression using neural networks for classification in actuarial learning," *Algorithms,* vol. 16, no. 2, p. 99, 2023.

[59] M. Alshinwan, A. G. Memon, M. C. Ghanem, and M. Almaayah, "Unsupervised text feature selection approach based on improved Prairie dog algorithm for the text clustering," *Jordanian Journal of Informatics and Computing,* vol. 2025, no. 1, pp. 27-36, 2025.

[60] A. Shafiq, A. B. Çolak, S. A. Lone, T. N. Sindhu, and T. Muhammad, "Reliability modeling and analysis of mixture of exponential distributions using artificial neural network," *Mathematical Methods in the Applied Sciences,* vol. 47, no. 5, pp. 3308-3328, 2024.

[61] D. Božić, B. Runje, D. Lisjak, and D. Kolar, "Metrics related to confusion matrix as tools for conformity assessment decisions," *Applied Sciences*, vol. 13, no. 14, p. 8187, 2023.

[62] A. H. Mohammad, T. Alwada'n, O. Almomani, S. Smadi, and N. ElOmari, "Bio-inspired hybrid feature selection model for intrusion detection," *Computers, Materials and Continua*, vol. 73, no. 1, pp. 133-150, 2022.