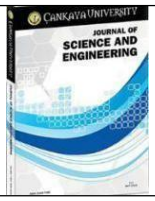
e-ISSN: 2564-7954 CUSJE 22(2): 073-089 (2025) Research Article



Çankaya University

Journal of Science and Engineering

https://dergipark.org.tr/cankujse

CUJSE

Development of an Android-Based Malware Detection Model

Ajibola Gbotosho[1], Patrick Ozoh[1*], Tosin Oyetayo[1]

[1] Department of Computer Science, Osun State University, Osogbo, Nigeria

Keywords

Androids,
Malicious Threats,
Machine Learning,
Manifest Analysis.

Abstract

The study to identify malicious threats in Android mobile phones is presented in this study. The sample Dataset was from DroidFusion-2018—Jupyter Notebook, together with Python for implementation. The techniques considered include the classifiers mentioned in the study. An ensemble of techniques was developed for the study. The Ensemble model achieved 97% accuracy, compared to 71%, 77%, and 79% attained by SVM, KNN, and RF. The study designed a model for detecting malicious Android applications that are integrated into existing malware detection platforms to improve their usage and acceptance.

1. Introduction

Mobile devices have become increasingly important for contemporary civilization over the last decade, directly contributing to its growth in establishing mobile information access. Nowadays, smartphones are frequently used to complete tasks such as making payments, bookings, and so on. The enhancement of the operating system has also supported the growth of mobile devices. During the market's volatile growth, it became its largest intelligence. Android and iOS are the dominant operating systems in the industry. Android phones continued to hold the top spot in the mobile operating system, dominating the market [1]. Google penetrated the mobile industry and later made it available to the general public.

The cost is another factor dominating the global market for Android [2]. Cost is therefore the first consideration when choosing a mobile phone. Apple cannot compete with Google in that market. They can be given to several users worldwide [3]. According to McAfee, any malicious software is malware. Cybercriminals utilize software against victims to their advantage to profit financially. The term contains malicious software like viruses, Trojans, worms, bots, backdoors, spam, spyware, ransomware, etc. [4]. Mobile malware controls a mobile device remotely, disables it, or steals personal information. It targets Android-powered devices.

Malware is disseminated via altered or blatantly harmful applications hosted on third-party mobile app stores [5]. Android malware distribution is not limited to third-party applications but also includes visiting unidentified websites. In September 2019, 172 fraudulent apps, totaling nearly 330 million installations, were discovered. This study claims that the malicious components were concealed within the system [6]. Android malware is commonly deceptive. It is one form of malware that irritates Android phone users by bombarding them with advertisements, even when using unrelated apps, for which the developers receive payment to display them [7], hence [8]. This research detects malicious Android applications with several features.

* Corresponding Author: patrick.ozoh@uniosun.edu.ng
Received: June 13, 2025, Accepted: September 9, 2025

The contributions to the knowledge are:

1) This will produce a reliable malware detection mechanism for Android
2) This study will be integrated into existing malware detection platforms to improve usage and acceptance.

2. Past Works

The reviews are relevant and include a brief description of techniques used in illegal transactions made in mobile applications [9]. The malicious software for Android devices is known as Android malware [10]. The Various characteristics of malware apps have been apportioned into various families [11].

Various critical elements are measured during analysis for malware's functionality. Vulnerability analysis, source code analysis, and behavioral analysis are examples of informal analysis [12]. There are three predominant analysis techniques [13]. These include application binaries, source code, and malicious threats [14]. The taxonomy of the view is in Fig. 1.

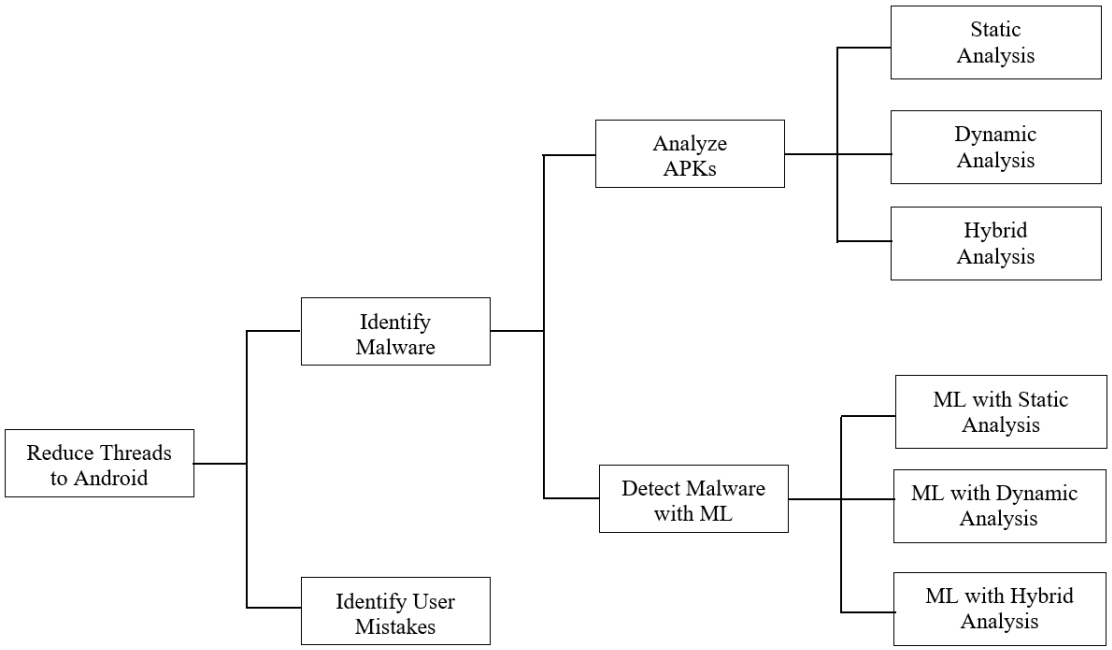


Figure1. Taxonomy of the Review

The dynamic analysis approach was developed to address limitations [15]. This paper enhances the ability to detect malicious software effectively. Android malware systems have greater features compared to traditional inflexible phone bugs. [16]. This enables the detection of variants of old malware and wholly novel malware.
[17] measures machine learning classifiers to obtain authorization [18] aimed to improve the foundation of malware detection. [19] aimed to examine the effect of authorizations in a malware detection system using established techniques. [20]-[24] proposes static detection methods. [21] aimed to provide a solution that detects malware from several families efficiently and accurately with minimum time and resources. The aim of [22] is to utilize the attributes used by Androids. The study performed a manifest analysis for permissions.

2.1 Related Works

The extensive use of Android applications has made it easier for Cybercriminals to pilfer sensitive information and execute malicious attacks. The quick advances made in the use of Android-based smartphones have led to emerging threats from malware devices. As a result, Android malware has become increasingly complicated in its usage. The Malware detection model characterizes malicious behavior. Traditional techniques lack high-dimensional feature specifics, which increases computational complexity and reduces the detection accuracy.

Work [25] proposes a technique to enhance Android malware detection by utilizing a reliable feature optimization model. The study utilizes the Relief algorithm to reduce the feature space. As a result, eliminates

redundant and irrelevant features. The ensemble model uses three datasets (Derbin, Malgenome, and Prerna) across all features. The experimental results show that detection accuracy can improve with an optimal feature vector. [26] presents a widely used study of current malware detection techniques in Android-based phones. The paper examines the techniques for developing malware threats and includes a hybrid detection technique, together with real-time threat intelligence, applicable to the cyber threat landscape.

Work [27] presents a general detection framework for system calls as a privacy-preserving data source. The method allows preserving user privacy. The study utilizes the accuracy of detection in a secure environment in the real world. A framework integrating Genetic Algorithms (GA) with its level of accuracy specified at 98% [28]. The experimental results on the Android malware detection datasets indicate a success rate, increasing accuracy from 24% to 97%. The results show that current security controls do not sufficiently alleviate the impact of the proposed framework against attacks. [29] propose a malware detection technique for improved performance, proposing that the ensemble is more accurate than current techniques in a thorough manner. The study lays down a basic foundation for future research. The reviewed literature proposes widespread mobile phone threats over the past decade, specifically in the area of malware. Recent studies have leveraged artificial intelligence techniques to tackle these challenges.

2.2 Exploratory Texts on Methods

2.2.1 Support Vector Machine

[30] investigate machine learning techniques to select spam in messages using Kaggle. The study employs supporting vector machine technique, with performance evaluation. The support vector machine using these features results in better accuracy compared to previous studies.

A parallel between the techniques was carried out by [31]. An evolutionary programming technique was employed during the testing. [32] Propose a system by evaluating the model. The system shows a high accuracy in identifying spam messages. The study analyzes language patterns and features within the text and proposes a robust framework for reducing SMS spam and improving user experiences.

2.2.2 K-Nearest Neighbour (KNN)

This study uses [33]. The study estimates the sample size with the outcome having more accuracy using the Multinational Naive Bayes Algorithm, when the methods' average performance is compared to identify the more accurate technique. [34] present a unique technique by combining the K-Nearest Neighbour technique for feature extraction with the collection of classification characteristics. The outcome shows an accuracy of 98.36% and precision (99.19%). This indicates a more accurate model performance compared to current measurements. This method provides a functional SMS spam detection technique, enhancing the classifiers.

[35] proposes a framework for detecting SMS spam by developing robust and efficient spam detection models. The K-Nearest Neighbour technique oversees feature extraction in representing text data. The model can then capture important characteristics by differentiating spam from non-spam messages. The results provide a solution for enhancing SMS spam detection.

2.2.3 Random Forest (RF)

The study utilizes the Random Forest Algorithm to remove irrelevant features and proposes that accuracy can be improved through optimal feature selection. [36]. The performance of this study is assessed using Random Forest models. The datasets were evaluated on their level of accuracy. The study was also conducted in a non- English format. [37] Apply an ensemble to incorporate separate methods for detection. The techniques used indicate that the ensemble model is more accurate than individual algorithms, providing active protection against malicious SMS attacks. [38] propose a unique technique for spam detection, achieving improved accuracy than previous methods. The results present a robust and remarkable improvement in spam detection.

2.3 Comparison of Techniques

The positive and negative attributes for each technique are considered in this section, summarized in the table below to identify and discuss the reasons for choosing the methods used in this study.

Table 1. Characteristics of Methods Used in the Study

Method	Advantages	Disadvantages
Random Forest (RF)	• The performance evaluation is less complex • Reduced overfitting • Takes care of missing data • Usefull for classification and regression functions • Random Forest gives knowledge of the features that are important for predictions	• Requires much memory and needs many resources • Less interpretable than individual decision trees
K-Nearest Neighbor (KNN)	• Develops training instances for a general model • Approximations are made to the target function • These algorithms can easily adapt to new data collected over time	• A form of lazy learning • Relies on storing data • The complexity of the hypothesis can grow with the data • Each query consists of beginning the new model from scratch, leading to high classification costs • A huge memory is required to store data
Support Vector Machine	• SVM is adequate for high-dimensional spaces and image classification analysis • Handles nonlinear Relationships • Improves accuracy • SVM is adequate for text Classification	• Overfitting of models • Slow for big datasets, impacting performance • Adjusting parameters needs careful tuning • SVM is difficult with noisy datasets, reducing its effectiveness

This study compared techniques reviewed in literature with a dataset from Kaggle, an online data repository. The study compares the methods to investigate the most accurate technique. The results from the study indicate that the Random model was the most widely used technique. An Ensemble model of the three techniques will enhance the accuracy of the results. The four variables used in [39] improve the study by using a combination of these four parameters. This study proposes an ensemble model to detect malicious Android mobile applications, as the decision-making of a bigger group is superior to that of a single expert.

2.4 Description of Practical Applications

For use in real-world apps, there is a need to describe the following:

2.4.1 Real-Time Processing

Real-time performance of a system encompasses every component, including hardware, BIOS, operating system, network, and application. Real-time applications execute within a certain period, across several iterations. Real-time applications conventionally perform the following tasks: 1. Process new input. 2. Compute a computation. 3. Submit a result. These functions must be completed within a period.

[40]-[43] presents an evaluation of different models, providing insights into malware characteristics. The study outcome includes result visualization, with results displayed in real-time, presenting their prevalence for preventing related cyber threats. In [44], a hybrid mobile malware detection model that enhances detection

accuracy and makes the model more resilient to new malware is proposed. The proposed model is more accurate than classical machine learning algorithms. However, there is a need to build real-time detection components to tackle mobile malware in new connected ecosystems.

2.4.2 Memory Usage and Latency

Memory is a vital component of a computer system, focusing on the CPU and the storage system for the fast processing of data. Computer users often examine the relationship between memory speed and system performance. Work [45] presents a model that introduces data splitting as Testing and Training data, enhancing the classification process. A major challenge of this paper is its inability to deal with the prediction of noise. [46] presents the performance of machine learning classifiers using representations learned by an encoder from malware datasets. Results from the study indicate the Ensemble model performs better than the others. A drawback to the study is that the study outcome cannot provide insight into features impacting the latent space.

3. Methodology

This section discusses the methodology for developing a malware detection model for Android mobile phones. The section gives details of data collection, model design, and implementation.

3.1. Data Collection

The first module is data collection. The data was sourced from the Kaggle repository and includes both malicious (S) and benign (B) categories extracted from various applications. It contains 5560 malicious applications and 9476 benign applications. Fig. 2 shows that the features represented in Table 1 have no class imbalance in the dataset. Fig 3 shows the Android malware.

RECEIVE_BOOT_COMPLETED	Manifest Permission
RESTART_PACKAGES	Manifest Permission
Ljava.lang.Class.getPackage	API call signature
chmod	Commands signature
Ljava.lang.Class.getDeclaredClasses	API call signature
android.intent.action.ACTION_POWER_DISCO	Intent
android.intent.action.PACKAGE_ADDED	Intent
PathClassLoader	API call signature
TelephonyManager.getSimSerialNumber	API call signature
Runtime.load	API call signature
TelephonyManager.getCallState	API call signature
BLUETOOTH	Manifest Permission
READ_CALENDAR	Manifest Permission
READ_CALL_LOG	Manifest Permission
SUBSCRIBED_FEEDS_WRITE	Manifest Permission
READ_EXTERNAL_STORAGE	Manifest Permission
TelephonyManager.getSimCountryIso	API call signature
sendMultipartTextMessage	API call signature
PackageInstaller	API call signature
VIBRATE	Manifest Permission
remount	Commands signature
android.intent.action.ACTION_SHUTDOWN	Intent
sendDataMessage	API call signature
ACCESS_NETWORK_STATE	Manifest Permission
chown	Commands signature
HttpPost.init	API call signature
Ljava.lang.Class.getClasses	API call signature
SUBSCRIBED_FEEDS_READ	Manifest Permission
TelephonyManager.isNetworkRoaming	API call signature
CHANGE_WIFI_MULTICAST_STATE	Manifest Permission
WRITE_CALENDAR	Manifest Permission
android.intent.action.PACKAGE_DATA_CLEAR	Intent

Figure 2. Dataset Features

Table 2. Dataset Features Distribution

S/N	Category	Feature
1	Manifest Permissions	113
2	API Call Signatures	73
3	Command Signatures	6
4	Intents	23
TOTAL		215

transact	onServiceConnected	bindService	attachInterface	ServiceConnection	android.os.Binder	SEND_SMS	Ljava.lang.Class.getCanonicalName	Ljava.lang.Class.getMethods	Ljava.lang.Clas
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	1	1	0	1	1	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	
1	0	0	1	0	1	1	1	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	0	0	1	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	1	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	0	1	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	
1	1	1	1	1	1	0	0	1	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	

Figure 3. Android Malware and Benign Data Set

3.2 Methods Used in the Study

This research utilizes Pearson's Correlation Coefficient for feature selection to select relevant variables from the original dataset. Pearson's correlation coefficient is the association between the variables of interest and selects features for a machine-learning model associated with two variables. The correlation coefficient 'r' is given by

$$r = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2\sum(y_i-\bar{y})^2}} \quad (1)$$

Where,

r: correlation coefficient

x_i: r: values of the x − variable in a sample

$\bar{x}$: mean of the values of the x − variable

y_i: r: values of the y − variable in a sample

$\bar{y}$: mean of the values of the x − variable

The data set, which contains 15036 instances, was later reduced to 15031 instances because five (5) out of the entire data set contained null values, which were removed after prepossessing, and 55 attributes out of its 215 attributes were selected by the Pearson correlation coefficient for training the model. This is given in Fig. 4.

```
START
Generate Correlation between variable and Target Class
IF correlation > 0.2
        data_corr <- ADD
ELSE
        SKIP
REPEAT
        SELECTION
        Compute Correlation
UNTIL population has converged
STOP
```

Figure 4. Pearson Correlation Coefficient Pseudocode

The input process of the data set retrieved from Kaggle.

SEND_SMS	android.telephony.SmsManager	READ_PHONE_STATE	RECEIVE_SMS	READ_SMS	android.intent.action.BOOT_COMPLETED	TelephonyManager.getLine1Number	WRITE_SMS	WRITE_HISTORY_BOOKMARKS	TelephonyManager.getSubscriberId	android.telephony.gsm.SmsManager	INSTALL_PACKAGES
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	1	1	0	1	0	1
0	0	1	0	0	1	0	0	1	0	0	0
0	0	1	0	1	1	1	1	0	1	0	0
0	0	1	0	0	1	1	0	0	1	0	0
0	0	1	0	1	1	1	1	0	1	0	1
1	1	1	1	1	1	1	1	0	0	1	0
1	0	1	0	1	1	1	1	0	1	1	0
1	1	1	1	1	1	0	1	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	1	0	0	0
1	0	1	0	0	0	1	0	0	0	1	0
1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	0	1	0	0	0	0
1	0	1	1	1	1	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	1
1	0	1	1	1	1	0	1	0	1	1	1
1	1	0	0	1	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	0	0	0	0
1	0	1	1	1	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	1	0	0	0	0	1	1
1	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
1	1	1	1	0	1	1	0	0	0	1	1
0	0	1	0	1	1	1	0	1	0	0	0
0	0	1	0	0	1	0	0	1	0	0	1
1	1	1	1	0	1	0	0	0	1	0	0
1	0	1	1	1	1	0	1	0	1	0	0
1	1	1	1	0	1	1	0	1	1	0	0
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	1	0	0	1	1	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	1	1	0	0	1	0	1
1	0	1	0	0	0	1	0	0	0	1	0
0	0	1	0	1	1	0	1	0	1	0	0
1	1	1	1	1	1	0	1	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0
1	1	1	0	0	1	1	0	0	0	0	0
1	0	1	1	0	0	1	0	1	1	1	1
0	0	1	0	0	1	1	0	0	0	0	0
0	0	1	0	0	0	1	0	0	1	0	0
0	0	1	0	1	1	1	1	0	1	0	1
1	1	1	1	0	0	0	1	0	0	0	0
1	0	1	1	0	1	0	0	0	0	0	0

Figure 5. Snapshot for Data from Kaggle A

A description of the classifiers is given below.

3.2.1.SVM Technique

The technique separates classes in the classification for identifying spam messages. SVM maximizes the margin separating the margins. The better the separation, the more accurate it is. It is important and common to search for solutions by class splitting [23]. The pseudo-code for SVM is displayed in Fig. 6.

Inputs:Determine the various training and test data.
Outputs:Determine the calculated accuracy.
Select the optimal value of cost and gamma for SVM.
while (stopping condition is not met) **do**
    Implement SVM train step for each data point.
    Implement SVM classify for testing data points.
end while
Return *accuracy*

Figure 6. SVM Pseudocode

3.2.2.K-Nearest Neighbour (KNN)

This process includes finding the "k" closest data points to a given input. As a result, the KNN algorithm is discussed by [24] and the pseudocode is in ig. 7.

Input: the training set D, test object x, category label set C
Output: the category c_x of test object x, c_x belongs to the C

1 **begin**
2 **for** each y belongs to D do
3 calculate the distance $D(y, x)$ between y and x
4 **end for**
5 select the subset N from the data set D,
the N contains k training samples which are the k
nearest neighbors of the test sample x
6 calculate the category of x:
$c_x = \arg\max_{c \in C} \sum_{y \in N} I\left(c = class\left(y\right)\right)$
7 **end**

Figure 7. KNN Pseudocode

3.2.3. Random Forest

This is applicable for a better prediction. Individual techniques have an impact on separate data sets for classification or regression. This reduces errors and improves accuracy. It is used in research to solve the credibility problem [25]. Fig 8 illustrates the pseudo-code of the Random Forest algorithm.

1. *Select randomly M features from the feature set.*
2. *For each x in M*
   a. *calculate the Information Gain*

$$Gain(t,x) = E(t) - E(t,x)$$
$$E(t) = \sum_{i=1}^{c} - P_i \, log_2 \, P_i$$
$$E(t,x) = \sum_{c \in X} P(c)E(c)$$

Where E(t) is the entropy of the two classes, E(t,x) is the entropy of feature x.

   b. *select the node d which has the highest information gain*
   c. *split the node into sub-nodes*
   d. *repeat steps a, b and c to construct the tree until reach minimum number of samples required to split*
3. *Repeat steps 1 and 2 for N times to build forest of N trees*

Figure 8. Random Forest Pseudocode

3.2.4. Ensemble

Ensemble learning is a collection (or ensemble) of basic learners or models to improve the final prediction. The ensemble technique employed in this study is known as the Stacking Classifier. It is shown in Fig. 9.

Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$ ($\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathcal{Y}$)
Output: An ensemble classifier H

Step 1: Learn first-level classifiers
for $t \leftarrow 1$ to T **do**
    Learn a base classifier h_t based on $\mathcal{D}$
end for
Step 2: Construct new data sets from $\mathcal{D}$
for $i \leftarrow 1$ to m **do**
    Construct a new data set that contains $\{\mathbf{x}'_i, y_i\}$, where $\mathbf{x}'_i = \{h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_T(\mathbf{x}_i)\}$
end for
Step 3: Learn a second-level classifier
Learn a new classifier h' based on the newly constructed data set
return $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x}))$

Figure 9. Stacking Pseudocode

3.3 Comparison and Validation of Techniques

This section compares the proposed model with previous models. The results from this study are compared with results from [40] and [41], using their respective error values. The process of validation of results was discussed by [42]. The computation of errors was described.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (2)$$

Where,

y_i: observed (actual)value
$\hat{y}_i$: predicted value
n: number of observations

4. Results

The implementations performed are presented. It provides detailed findings of the models and metrics employed, and performance of the results.

4.1. Model Evaluation

F1 would be between 0 and 1. It is calculated using a confusion matrix with the help of;

$$F1 = 2*(precision*recall)/(precision + recall) \quad (3)$$

Fig 10 to Fig 13 indicate the evaluation metrics performed on each classifier. The figures show the respective performance evaluation against the metrics for the SVM, KNN, RF, and Ensemble models.

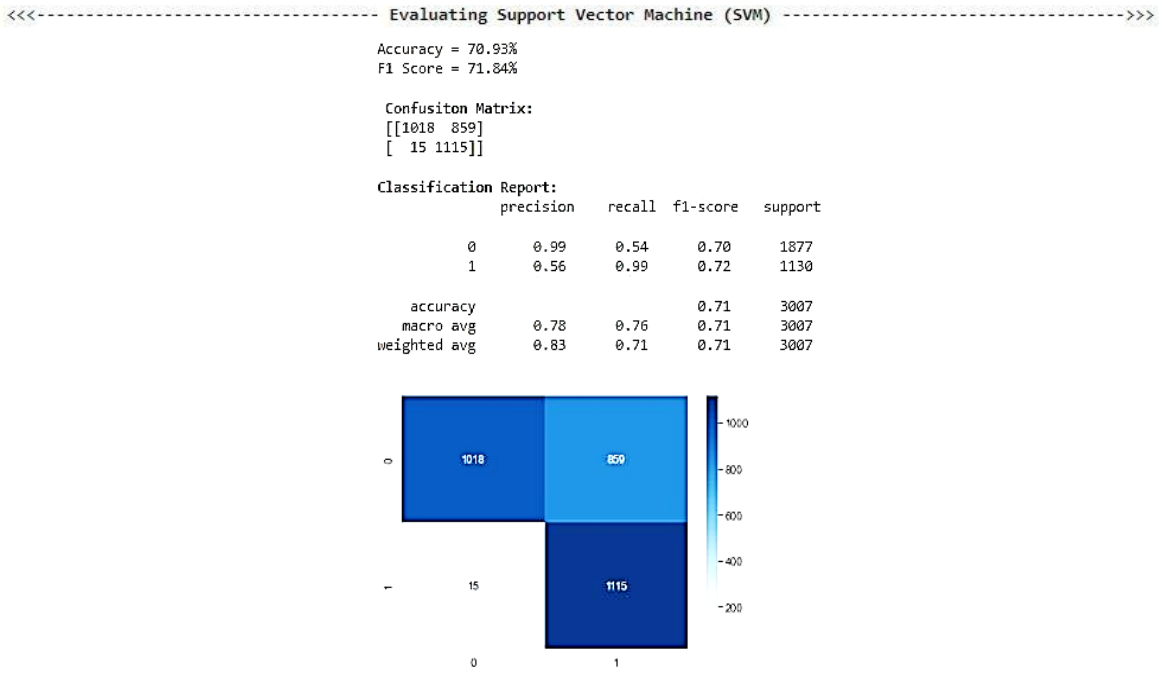


Figure 10. Result for SVM

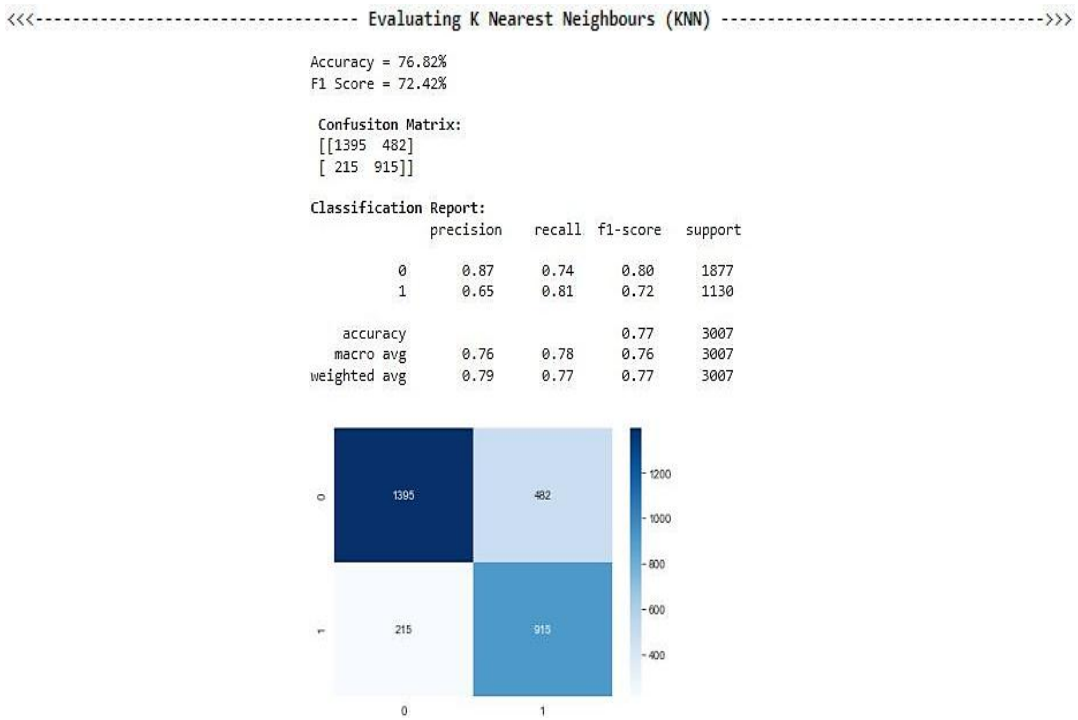


Figure 11. Result for KNN

<<<---------- Evaluating Random Forest Classifier (RF) ---------->

Accuracy = 79.17999999999999%
F1 Score = 75.83%

Confusiton Matrix:
[[1399 478]
[148 982]]

Classification Report:

	precision	recall	f1-score	support
0	0.90	0.75	0.82	1877
1	0.67	0.87	0.76	1130
accuracy			0.79	3007
macro avg	0.79	0.81	0.79	3007
weighted avg	0.82	0.79	0.80	3007

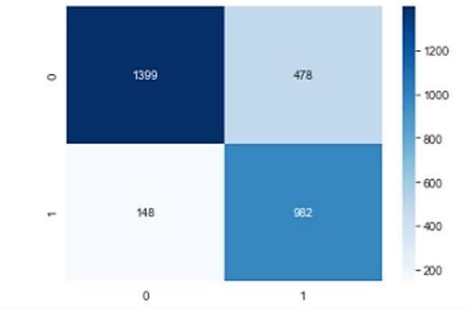


Figure 12. Result for Random Forest

<<<---------- Evaluating Stacking Classifier ---------->>>

Accuracy = 97.41%
F1 Score = 96.50999999999999%

Confusiton Matrix:
[[1852 25]
[53 1077]]

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	1877
1	0.98	0.95	0.97	1130
accuracy			0.97	3007
macro avg	0.97	0.97	0.97	3007
weighted avg	0.97	0.97	0.97	3007

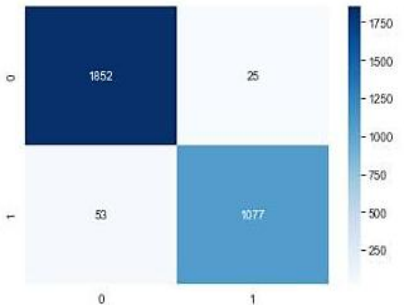


Figure 13. Ensemble Result

Table 3 presents the performance evaluation of the Ensemble model with SVM, KNN, RF, and Ensemble models. The performance metric chart is given in Fig. 14.

Table 3. Performance Evaluation Result

Classifiers	Accuracy	Precision	Recall	F1-Score	Time(sec)
Support Vector Machine (SVM)	70.93	56.48	98.67	71.84	48.37
K-Nearest Neighbor (KNN)	76.82	65.50	80.97	72.42	34.91
Random Forest (RF)	79.18	67.26	86.90	75.83	31.29
Ensemble	97.41	97.73	95.31	96.51	46.64

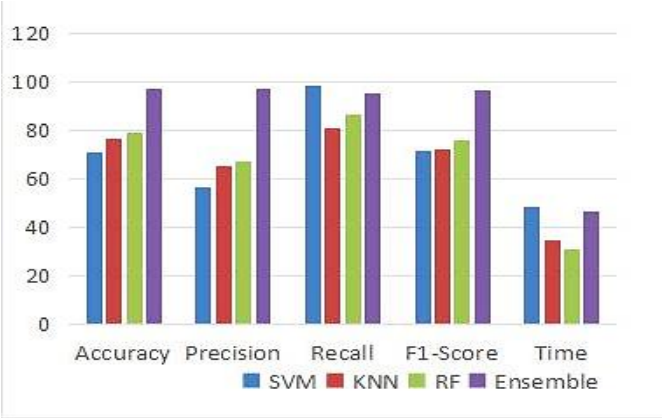


Figure 14. Performance Evaluation Chart

4.2.Model Deployment

The Python software, Streamlit, and Heroku are used for the model. After creating the back-end with Python and Streamlit, Heroku was used to deploy the model to be accessed via the web. This is shown in Fig. 15.

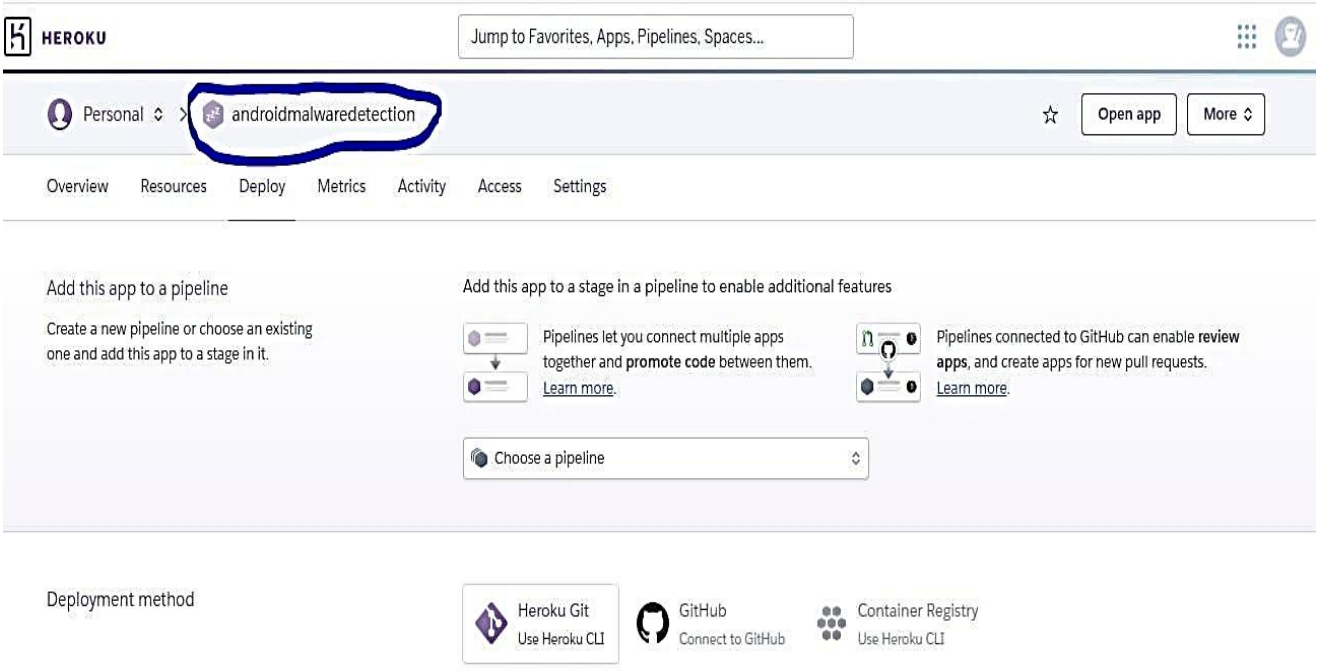


Figure 15. Heroku's View when Deploying the Web Application

4.3. Results for Comparison of Techniques

The results are computed based on models described in Section 3.3. From Table 4, the most accurate technique is seen as the Ensemble model. This is compared with previous studies [40] and [41].

Table 4. Comparisons

Ahmadi et al. (2025)		Shen et al. (2025)		EnsembleModel	
Actual data	**Estimates**	**Actual data**	**Estimates**	**Actual data**	**Estimates**
41.59	48.11	41.59	45.56	41.59	46.18
45.60	50.43	45.60	48.77	45.60	49.91
42.74	46.67	42.74	45.14	42.74	45.94
44.75	50.95	44.75	48.32	44.75	49.01
41.18	88.21	41.18	45.34	41.18	46.22
36.79	42.66	36.79	39.21	36.79	40.54
41.68	47.79	41.68	45.11	41.68	46.32
42.84	46.41	42.84	44.32	42.84	44.99
44.03	47.76	44.03	45.01	44.03	45.87
43.78	49.46	43.78	45.32	43.78	46.58
44.94	48.42	44.94	45.11	44.94	46.36
42.12	47.68	42.12	43.92	42.12	44.93
37.91	42.43	37.91	39.88	37.91	40.03
42.95	47.74	42.95	44.43	42.95	45.46
43.56	49.32	43.56	46.47	43.56	47.84
44.34	47.46	44.34	45.13	44.34	45.79
44.64	48.34	44.64	45.73	44.64	46.74
44.15	48.85	44.15	45.58	44.15	46.36
38.90	42.85	38.90	39.41	38.90	40.58
37.76	41.47	37.76	38.74	37.76	39.74
43.58	47.27	43.58	44.82	43.58	45.43
45.15	48.11	44.15	45.15	44.15	46.85

The values for results are indicated in Table 5. The results show RMSE and MAPE given by [40] and [41].

Table 5. Evaluation of Results

	Ahmadi et al. (2025)	**Ensemble model**	**Shen et al. (2025)**
RMSE	0.873	0.596	0.704
MAPE (%)	1.957	0.921	1.199

The results show the Ensemble to be the best technique. A study of past literature presented in Section 2 indicates that the techniques combined to form the Ensemble are efficient machine learning techniques. The strengths and weaknesses of these techniques are listed. The combination of the three techniques considered will improve the accuracy of the results. This study proposes an Ensemble model to detect malicious Android mobile applications, as the results from a set are more accurate than those of a single model.

4.4. Results for Split Validation

The split validation technique is applied to methods, ensuring a robust performance estimation (k = 5) for malware detection and malware category classification. The validation procedure divides the dataset into five folds. For each

iteration, the training technique validates the model.

The results in Table 5 clearly show the validity of the Ensemble model.

Table 6. Results for 5-Fold Cross Validation

	Ensemble model	RF	KNN	SVM
Malware detection	96.41	78.19	71.21	70.23
Malware category classification	86.10	84.28	83.22	83.15

5. Conclusion

This study proposed an ensemble for identifying bugs compared with the individual results of the aforementioned machine learning. The research was evaluated using a collection comprising 15036 instances (benign and malware) from the Kaggle repository. The dataset was preprocessed, and 55 features were identified in the model. During the experiment, the ensemble model achieved 97% accuracy compared with 71%, 77%, and 79% in SVM, KNN, and RF, respectively.

Heroku is a natural language processing-based model that is quick and easy to use when the spam detection model is deployed. The application takes natural text from users and predicts if it's legitimate text. The model works by inputting the message and checking whether it's legitimate text. The limitations of Heroku in real- world use include cost constraints, inability to adapt and grow, and handling a growing amount of work. These limitations have led users to turn to alternatives. In the deployment of a Heroku application, trust is an issue because it involves critical and sensitive data about businesses' customers.
The Ensemble model will analyze any form of data, including complex, cutting-edge data, adaptable in detecting diverse malware types, next-generation malware, and identifying malicious applications. It can perform a comprehensive analysis with real-time detection capabilities. The model is designed for detecting malicious applications that are integrated into existing malware detection platforms. As a result, enhancing their usage and acceptance. The model will perform similarly on different datasets.

There are several machine learning techniques; it is impossible to exhaust all of these methods. This study investigates these techniques because they were widely used in previous studies. Furthermore, this study is limited to static analysis. This technique enables understanding of the application and ensures it is compliant and safe. This work is far-reaching for Android users in detecting malware. The result shows that the ensemble has better accuracy. Future research will involve the impact of failure cases and misclassifications on the spam detection model.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Authorship Contribution Statement

Gbotosho Ajibola: Data Preparation, Editing
Ozoh Patrick: Writing, Methodology
Oyetayo Tosin: Data Preparation, Editing

References

[1] J. Hightower, W. B. Glisson, R. Benton, and J. T. McDonald, “Classifying Android Applications Via System Stats,” in IEEE International Conference on Big Data (Big Data), virtual, 2021 pp. 5388-5394, doi:10.1109/BigData52589.2021.9671999.

[2] J. Wallen, "Why is Android more popular globally, while iOS rules the US," 2021, www. techrepublic.com/article/why-is-android-more-popular-globally-while-ios-rules-the-us.

[3] D. O. Sahin, S. Akleylek, and E. Kilic, "LinRegDroid: Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers," IEEE Access, vol. 10, pp. 14246–14259, Jan.2022, doi:10.1109/ACCESS.2022.3146363.

[4] D. Gibert, M. Carles, and P., "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," J. Netw. Comput. Appl., vol. 153, pp. 102526, Mar. 2020, doi:10.1016/j.jnca.2019.102526.

[5] J. Vijayan, "Android Malware Hijacks Phone Shutdown Routine," Security Intelligence, 2021, securityintelligence.com/news/android-malware-hijacks-phone-shutdown-routine/.

[6] R. Jusoh, A. Firdaus, S. Anwar, M. Z. Osman, M. F. Darmawan, and M. F. A. Razak, "Malware Detection Using Static Analysis in Android: a review of FeCO (Features, Classification, and Obfuscation)," PeerJ Comput. Sci., vol. 7, pp. 1–54, Jun. 2021, doi:10.7717/peerj-cs.522.

[7] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri. "Android mobile malware detection using machine learning: A systematic review", Electronics, vol. 10, no. 13, 1606, 2021, doi:10.3390/electronics10131606.

[8] O. Yildiz, and I. A. Doğru, "Permission-based Android Malware Detection System Using Feature Selection with Genetic Algorithm," Int. J. Softw. Eng. Knowl. Eng., vol. 29, no. 2, pp. 245–262, 2019, doi:10.1142/S0218194019500116.

[9] H. A. Alatwi, "Android Malware Detection Using Category-Based Machine Learning Classifiers," 2016, www. scholarworks.rit.edu/theses.

[10] F. Tchakounte, "A Malware Detection System for Android Malware Detection based on Android Permissions View project IoT security," 2016, www.researchgate.net/publication/282866516.

[11] M. S. Alhebsi, "Android Malware Detection using Machine Learning Techniques," 2022, www.scholarworks.rit.edu/theses.

[12] E. Masabo, "A Feature Engineering Approach for Classification and Detection of Polymorphic Malware using Machine Learning," Ph.D. dissertation, Depart. Comp. Networks, Sch. Computing and Inform. Tech., Makerere Uni., Kampala, 2019.

[13] V. Kouliaridis, and G. Kambourakis, "A comprehensive survey on machine learning techniques for android malware detection," Information 2021, vol. 12, 185, Apr. 2021, doi:10.3390/info12050185.

[14] F. Akbar, M. Hussain, R. Mumtaz, Q. Riaz, A. Wahab, and K. H. Jung, "Permissions-Based Detection of Android Malware Using Machine Learning," Symmetry, vol. 14, no. 4, pp. 718, Apr. 2022, doi:10.3390/sym14040718.

[15] Y. Kamalrul Bin Mohamed Yunus, and S. bin Ngah, "Review of Hybrid Analysis Technique for Malware Detection," IOP Conf. Ser.: Mater. Sci. Eng., vol. 769, no. 1, 012075, Jun. 2020, doi:10.1088/1757-899X/769/1/012075.

[16] A. Muzaffar, H. Ragab Hassen, M. A. Lones, and H. Zantout, "An in-depth review of machine learning based Android malware detection," Comput. Secur., vol. 121, 102833,Jul. 2022, doi:10.1016/j.cose.2022.102833.

[17] E. Amer, S. E. Mohamed, M. Ashaf, A. Ehab, O. Shereef, H. Metwaie, and A. Mohammed, "Using Machine Learning to Identify Android Malware Relying on API calling sequences and Permissions," J. Comput. Commun., vol. 1, no. 1, pp. 38-47, Feb. 2022, doi: 10.21608/jocc.2022.218454.

[18] A. S. Shatnawi, Q. Yassen, and A. Yateem, "An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms," Procedia Comput. Sci., vol. 201, pp. 653–658, Nov. 2022, doi:10.1016/j.procs.2022.03.086.

[19] D. O. Sahin, S. Akleylek, and E. Kilic, "LinRegDroid: Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers," IEEE Access, vol. 10, pp. 14246–14259, Jan. 2022, doi:10.1109/ACCESS.2022.3146363.

[20] H. Li, H. Zhang, X. Chen, D. Liao, and M. Zhang, "Android Malware Detection Based on Sensitive Patterns," Research Square, 2022, doi : 10.21203/rs.3.rs-1592245/v1.

[21] N. Sarah, F. Y. Rifat, M. S. Hossain, and H. S. Narman, "An Efficient Android Malware Prediction Using Ensemble machine learning algorithm," Procedia Comput. Sci., vol. 191, no.1, pp. 184–191, Jan. 2021, doi:10.1016/j.procs.2021.07.023.

[22] D. O. Şahin, O. E. Kural, S. Akleylek, and E. Kılıç, "A novel permission-based Android malware detection system using feature selection based on linear regression," Neural Comput. Appl., vol.35, no.5, pp. 1-16, Mar. 2021, doi:10.1007/s00521-021- 05875-1.

[23] E. H. Houssein, M. E. Hosney, M. Elhoseny, D. Oliva, W. M. Mohamed, and M. Hassaballah, "Hybrid Harris hawks optimization with cuckoo search for drug design and discovery in chemoinformatics," Scientific Reports, vol. 10, no. 1, 14439, Sept. 2020, doi:10.1038/s41598-020-71502-z.

[24] Q. Cao, L. La, H. Liu and S. Han, “Mixed weighted KNN for imbalanced datasets,” Int. J. Performability Eng., vol. 14, no. 7, pp. 1391-1400, 2018. doi: 10.23940/ijpe.18.07.p2.13911400.

[25] H. K. Almulla, H. J. Mohammed, N. Clarke, A. A. Hadi,and M. A. Mohammed, “An Effective Feature Optimization Model for Android Malware Detection,’’ Mesop. J. CyberSecur., vol. 5, no. 2, pp. 563-576, 2025.doi: 10.58496/MJCS/2025/034.

[26] R.Verma, “Review of Malware Detection from Android based Smart Mobile for Cyber Security”, 2025.

[27] A. K. Nair, and D. Gupta, “AndroIDS: Android-based Intrusion Detection System using Federated Learning,” arXiv preprint arXiv:2506.17349. Accessed: June 2025. [Online] Available: https://www.arxiv.org/pdf/2506.17349.

[28] F. Nawshin, D. Unal, M. Hammoudeh, and P. N. Suganthan, “A Novel Genetic Algorithm Optimized Adversarial Attack in Federated Learning for Android-Based Mobile Systems”, IEEE Trans. Consum. Electron., vol. 99, pp.1-1, Jun. 2025, doi: 10.1109/TCE.2025.3577905.

[29] S. Nethala, P. Chopra, K. Kamaluddin, S. Alam, S. Alharbi, and M. Alsaffar, “A Deep Learning-Based ensemble framework for robust Android malware detection”, IEEE Access, vol. 13, pp. 46673-46696, Mar. 2025, doi: 10.1109/ACCESS.2025.3551152.

[30] I. Nawaz, S. N. Khosa, R. Fatima, M. Saeed, and M. S. A. Hashmi, “Smart Filters For Sms Spam: A Machine Learning Approach to Sms Classification,” Spectr. Eng. Sci. , vol. 3, no. 5, pp. 71-98, May 2025, doi: 10.5281/zenodo.15333801.

[31] N. Hafidi, Z. Khoudi, M. Nachaoui, and S. Lyaqini, “Enhanced SMS spam classification using machine learning with optimized hyperparameters,” Indonesian J. Electr. Eng. Comput. Sci, vol. 37, no. 1, pp. 356- 364, Jan. 2025. doi: 10.11591/ijeecs.v37.i1.pp356-364.

[32] N. Durshatti, and O. Sraani, “Spam Message Detection with Multiple Algorithms,” SSRN. Accessed: 16 May. [Online] Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5183187.

[33] J. K. Prasad, and S. Christy, “SMS spam detection using multinational Naive Bayes algorithm compared with K-nearest neighbor algorithm,” in AIP Conference Proceedings, vol. 3270, no. 1, pp. 020090, AIP Publishing LLC, 2025.

[34] A. B. Ahmed, and K. Haruna, “Enhanced Sms Spam Detection Using Bernoulli Naive Bayes With Tf-Idf,’’ FUDJSE, vol. 9, no. 1, pp. 393-399, Apr. 2025, doi:10.33003/fjs-2025-0901-3226.

[35] P. Ozoh, M. Ibrahim, R. Ojo, A. Sunmade, and T. Oyetayo, “SMS Spam Detection Using Machine Learning Approach,” International STEM Journal, vol. 6, no. 1, pp. 10-27, Jun. 2025.

[36] M. F. Johari, K. L. Chiew, A. R. Hosen, K. S. Yong, A. S. Khan, I. A. Abbasi,D. Grzonka, “Key insights into recommended SMS spam detection datasets,” Sci. Rep., vol. 15, 8162, Mar. 2025, doi:10.1038/s41598-025-92223-1.

[37] H. Xu, A. Qadir, and S. Sadiq, “Malicious SMS detection using ensemble learning and SMOTE to improve mobile cybersecurity,” Comput. Secur., vol. 154, 104443, Mar. 2025, doi:10.1016/j.cose.2025.104443.

[38] M. A. Bouke, O. I. Alramli, and A. Abdullah, “XAIRF-WFP: a novel XAI-based random forest classifier for advanced email spam detection”, Int. J. Inf. Secur., vol. 24, no. 1, pp.1-5, Oct. 2025, doi:10.1007/s10207-024-00920-1.

[39] A. Madhulatha, A. K. Das, S. C. Bhan, M. Mohapatra, D.S. Pai, D. R. Pattanaik, and P. Mukhopadhyay, “Feasibility of model output statistics (MOS) for improving the quantitative precipitation forecasts of IMD GFS model,” J. Hydrol., vol. 649, 132454, Mar. 2025, doi: 10.1016/j.jhydrol.2024.132454.

[40] M. Ahmadi, M. Khajavi, A.Varmaghani, A. Ala, K., Danesh, and D. Javaheri, ”Leveraging large language models for cybersecurity: enhancing sms spam detection with robust and context-aware text classification,” arXiv preprint arXiv:2502.11014, 2025.

[41] L. Shen, Y. Wang, Z. Li, and W. Ma, “SMS Spam Detection Using BERT and Multi-Graph Convolutional Networks,” Int. J. Intell. Netw.., vol.6, pp. 79-88, 2025, doi: 10.1016/j.ijin.2025.06.002.

[42] A. Langenbucher, N. Szentmáry, J. Wendelstein, A. Cayless, P. Hoffmann, and D. Gatinel, “Performance evaluation of a simple strategy to optimize formula constants for zero mean or minimal standard deviation or root-mean-squared prediction error in intraocular lens power calculation,” Am. J. Ophthalmol., vol. 269, pp. 282-292, Jan. 2025, doi: 10.1016/j.ajo.2024.08.043.

[43] A. R. Elkilany and Y. B. Chu, “Elucidation on the performance of various machine learning models for real- time malware detection, malware classification and network packet screening,” ML Comput. Sci. Eng., vol 1, 9, Jan. 2025, doi: 10.1007/s44379-024-00010-y.

[44] R. N. Al Ogaili, O. A. Raheem, M. H. G. Abdkhaleq, Z. A. A. Alyasseri, S. A. A. A Alsaidi, A. H. Alsaeedi and S. Manickam, “AntDroidNet Cybersecurity Model: A Hybrid Integration of Ant Colony Optimization and Deep Neural Networks for Android Malware Detection,” Mesop. J. CyberSecur., vol. 5, no. 1, pp. 104-120, Feb. 2025, doi: 10.58496/MJCS/2025/008.

[45] C. Devika, C. C. Chowdary, D. Ramji, B. Tharun, and C. Nalini, “A framework to detect malware using a mobile edge computing system with minimal latency,” in Hybrid and Advanced Technologies, S. Prasad Jones Christydass, Nurhayati Nurhayati, S. Kannadhasan, Eds., CRC Press, vol.2, 2025, pp. 473-478.

[46] B. Ajayi, B. Barakat, and K. McGarry, “Leveraging VAE-Derived Latent Spaces for Enhanced Malware Detection with Machine Learning Classifiers,” arXiv preprint arXiv:2503.20803. Accessed: 30 April 2025. [Online] Available: https://arxiv.org/pdf/2503.20803.