Volume 10, Issue 5, May – 2025 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

Android Malware Classification with Feature Selection using Artificial Bee Colony Algorithm

Anthony Obogo Otiko[1]; Gabriel Akibi Inyang[2]; Etim Esu Oyo-Ita[3]; Utoda Reuben Agim[4]

[1](0009-0009-07264811)

[1, 2, 3, 4] Computer Science Department, University of Cross River State, Nigeria

Publication Date: 2025/06/5

Abstract: The proliferation of Android devices has resulted in a rise in complex malware specifically designed for these platforms, requiring higher detection techniques beyond conventional static and dynamic analyses. In this study, the Artificial Bee Colony (ABC) algorithm for feature selection is integrated with the eXtreme Gradient Boosting (XGBoost) and Random Forest (RF) classifiers to provide a novel method for Android malware detection. The ABC algorithm, which draws inspiration from honeybee foraging behavior, improves the performance of classifiers by balancing exploration and exploitation within feature subsets. Evaluation of the suggested approach on the Debrin Android malware dataset showed significant enhancements in detection accuracy and decreased false positives. The experimental findings demonstrated that both RF and XGBoost classifiers showed excellent performance, with RF slightly surpassing XGBoost in accuracy, precision, recall, and ROC-AUC metrics. The results highlight the efficacy of integrating metaheuristic feature selection with strong classifiers to enhance Android malware detection and tackle the difficulties presented by progressing threats.

Keywords: *Android Malware, Feature Selection, Machine Learning, Extreme Gradient Boosting, Random Forest.*

How to Cite: Anthony Obogo Otiko; Gabriel Akibi Inyang; Etim Esu Oyo-Ita; Utoda Reuben Agim (2025). Android Malware Classification with Feature Selection using Artificial Bee Colony Algorithm. *International Journal of Innovative Science and Research Technology*, 10(5), 3482-3490. https://doi.org/10.38124/ijisrt/25may1232

I. INTRODUCTION

The proliferation of mobile devices, especially those running on the Android platform, has basically transformed the digital environment [1]. Although Android's open-source nature has stimulated creativity and personalization, it has also made the platform an attractive target for cybercriminals [2]. Thus, the increased usage of Android devices has been accompanied by a commensurate increase in advanced malware assaults that take advantage of weaknesses in both applications and the Android operating system [3]. Malware, are software programs designed to damage, disrupt, or gain unauthorized access to computer systems, networks, or data [4]. A wide-ranging classification of malicious software, includes distinct detrimental programs specifically created to interfere with systems, pilfer confidential information, and obtain unauthorized entry. Android malware presents substantial risks because of its capacity to undermine device operating systems, pilfer sensitive data, and spread through several channels, such as malicious software and phishing attacks [5]. Malware varieties, including Trojans, ransomware, spyware, and adware, have the potential to significantly compromise the safeguarding of data security, integrity, and availability, while also causing a decline in system performance [6].

Android malware detection has traditionally depended on static and dynamic analysis methodology. Although efficient, static analysis, which evaluates code without running it, is susceptible to evasion methods such code obfuscation and polymorphism [7]. In contrast, dynamic analysis monitors the behavior of an application while it is running, but it is typically demanding in terms of resources and may not be able to identify complex threats that imitate harmless software behavior [8]. Conventional malware detection approaches, such as signature-based and rule-based solutions, have shown to be progressively inadequate in dealing with the ever-changing and dynamic characteristics of Android malware. Signature-based techniques rely on established malware patterns, making them inadequate for detecting novel or obscured malware variations [9]. In a similar vein, rule-based systems can be readily bypassed by sophisticated viruses that use advanced evasion techniques including polymorphism, encryption, and hiding [10].

The constraints of traditional malware detection methods have prompted researchers to investigate more sophisticated approaches, namely those based on machine learning (ML) and deep learning (DL) [11]. The increasing complexity and volume of threats posed by malware need the development of more adaptive methodologies, as contemporary detection methods face challenges in keeping up with their evolution.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

Machine learning is a highly promising approach for detecting malware via the identification of patterns in extensive datasets and the precise classification of previously unobserved criminal software [12]. Within the machine learning pipeline considering the size of modern dataset, feature selection is crucial as it helps improve model performance by minimizing the number of dimensions and preserving only the most pertinent characteristics. However, a dart in ML feature selection algorithm such as Analysis of variance (ANOVA) and correlation coefficients is that, although are efficient filter methods, but they do not take into account feature interactions [13]. Moreover, wrapper approaches, such as forward selection and recursive feature reduction, assess feature subsets based on model performance, but sometimes incur high computing costs [14]. Recently, complicated feature selection algorithm such metaheuristic algorithms have demonstrated exceptional performance by exploring a wide range of solutions. However, these algorithms may also be computationally demanding.

This, cutting-edge feature selection methods, including the Artificial Bee Colony (ABC), Particle Swarm Optimization (ABC), Ant Colony (ACO), algorithms, have shown promise in maximizing feature subsets for classification problems [15].[13],[16]. One of the notable meta-heuristic feature selection approaches is the ABC algorithm. ABC is an optimization method based on swarm intelligence, which draws inspiration from the foraging behavior of honeybees and has demonstrated successful use in several environments [17]. Notably, the efficiency with which it explores and exploits feature areas makes it a potential contender for enhancing malware detection systems. The integration of ABC for feature selection with robust classifiers such as eXtreme Gradient Boosting (XGBoost) and Random Forest (RF) is a highly promising method for improving detection accuracy and reducing false positives. In classification tasks, XGBoost is renowned for its scalability and capacity to handle huge, complicated datasets. On the other hand, Random Forest, a commonly used ensemble learning technique, is appreciated for its robustness against overfitting and its consistent classification results.

Hence, an innovative Android malware detection method is proposed in this paper, which combines the ABC algorithm for feature selection with XGBoost and Random Forest classifiers. The suggested method seeks to achieve high detection accuracy while minimizing computing costs and false positives by utilizing the optimization skills of ABC and the classification strengths of XGBoost and Random Forest models. A system of this nature has the capacity to greatly enhance the identification of both familiar and new Android malware risks, therefore tackling the difficulties presented by the ever-changing field of mobile security.

II. LITERATURE REVIEW

[18] developed a Time-Aware Machine Learning (TAML) framework, specifically designed for the purpose of detecting Android malware. This framework utilizes time-correlated characteristics extracted from the KronoDroid dataset. Their methodology combines time-aware and time-agnostic models, culminating in an impressive 99.98% F1 score in a time-agnostic environment and a steady 91% F1 score in time-aware configurations. The present study emphasizes the enhanced efficacy of time-aware approaches in comparison to traditional machine learning models and emulator-based detection systems. The author's research underscores the need of integrating temporal characteristics into data detection for malware. [19] have demonstrated the efficacy of combining static behavior analysis with machine learning methods for detecting Android ransomware. Their study employed a static feature selection approach, utilizing ML techniques including Decision Tree (DT), Extra Trees (ET), Light Gradient Boosting Machine (LGBM), and Random Forest (RF) to classify ransomware applications. They evaluated their method using a Kaggle dataset with 331 app permissions, including 199 ransomware instances. Their approach achieved a notable detection rate of 98.05%, highlighting its significant potential for enhancing malware identification and forensic analysis. The influence of filter-based feature selection methodology on Android malware detection was also investigated by [20]. Their research utilizes Information Gain, Chi-Square Test, and Fisher's Score to examine Android permission patterns and evaluates the efficacy of these techniques with Decision Trees, K-Nearest Neighbors, Random Forest, Support Vector Machine, and Logistic Regression machine learning classifiers. The results indicate that Information Gain and Fisher's Score demonstrate exceptional performance in reducing features, attaining classification accuracies of 91.53% and 91.22%, respectively, on the CICInvesAndMal2019 research dataset. The present study highlights the efficacy of filter-based approaches in enhancing the efficiency and precision of Android malware detection. [21] introduced a a lightweight MLP-CNN unified Android Malware Detection System (MCADS) which is a compact deep learning framework specifically developed for Android malware detection. This framework aims to overcome the constraints of conventional, resource-intensive methods. MCADS incorporates a dual-layer structure: an enhanced Multilayer Perceptron (MLP) for thorough malware analysis at the first stage, and a unique version of Convolutional Neural Network (CNN) for improving detection accuracy in challenging scenarios. In malware identification, the framework exhibits a remarkable accuracy rate of 98.12%, outperforming alternative approaches.

III. RESEARCH METHODOLOGY

In this study, the machine learning models for android malware detection were developed using a systematic three-phase methodological framework. At first Phase, the dataset was obtained using the Pandas library, which provides several interfaces for importing datasets from many file formats, including comma-separated values (CSV), the specific format employed in this study. The second phase, tagged a data preprocessing and feature selection phase, entails, a data preprocessing procedure conducted to eliminate extraneous characteristics, standardizing, and encoding dataset attributes. A feature selection procedure was also employed utilizing the Artificial Bee Colony (ABC) algorithm to discover and optimize the most pertinent subset features. Finally, after undergoing filtration, scaling, encoding, and feature selection, the processed and selected features were fed into the specified machine learning algorithms, namely XGBoost and RF on a training and test scale of 70 and 30% ration split. Here, the training dataset was used to train the models, while the test

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

dataset was used to assess their overall performance. This phase also entails the application of performance evaluation metrics to determine the models that exhibited higher effectiveness. Figure 1 depicts the detailed sequential methodological approach.

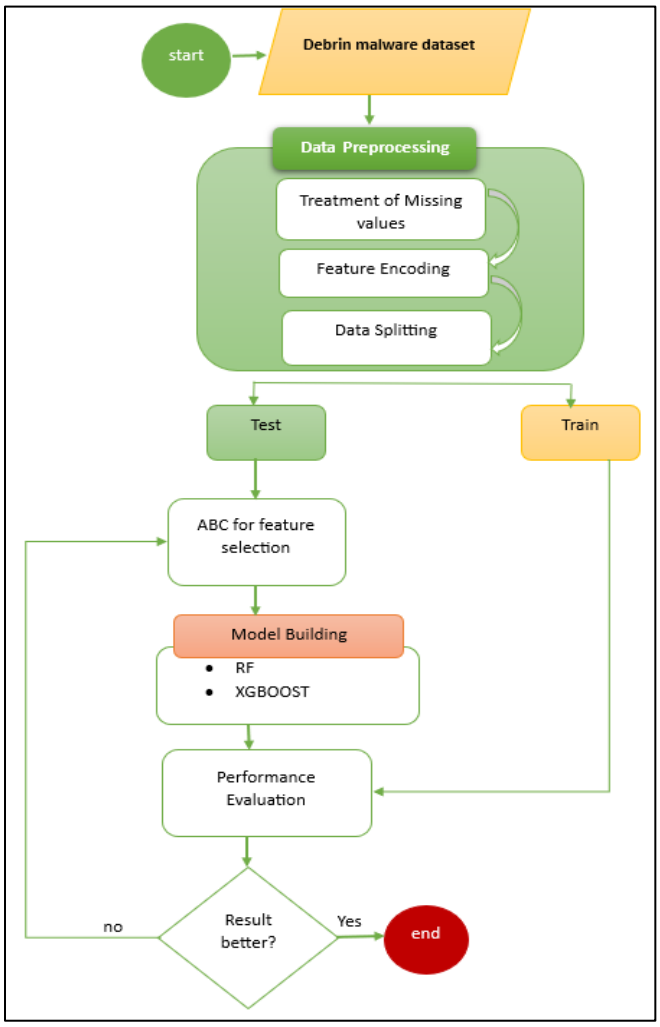

Fig 1 Research Method Framework

➢ *Dataset Description*

The dataset employed in this study is the Android Debrin dataset sourced from the Kaggle machine learning repository. The dataset has 216 features taken from a collection of 15,036 Android applications, including 5,560 malware apps obtained from the Drebin project and 9,476 benign apps [22]. The purpose of this dataset is to facilitate the creation and assessment of machine learning models for the classification of Android malware.

➢ *Feature Selection using ABC*

In the selection of the best features, the research applied the ABC algorithm. The Artificial Bee Colony (ABC) algorithm is a nature-inspired optimization technique introduced by Karaboga and Basturk, designed to imitate the foraging behavior of honey bees [13]. ABC comprises of three distinct categories of bees: employed bees, observer bees, and scout bees [23]. The algorithms have unique functions in the exploration and exploitation of the search space to identify the most efficient solutions. According to [24], the optimization process step in ABC involves:

- *Initialization Phase:*

Here, the algorithm randomly determines the position of the Food source an optimal subset feature via the equation (1):

$$x_{ij} = x_{minj} + rand[0,1]x(x_{max\,j} - x_{min\,j}) \quad \ldots(Eq.1)$$

- *Employee Bee:*

At this phase, the employee bee explores the neighborhood of food sources related to the employee bee. The equation (2) of the neighborhood exploration is as follows:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - v_{kj}) \quad \ldots(Eq.2)$$

After v_{ij} is generated, the fitness value is generated using the following equation (3):

$$fitness_i = \begin{cases} \dfrac{1}{1+f_i}, & if\ f_i \geq 0 \\ 1 + abs(f_i), & if\ f_i < 0 \end{cases} \quad \ldots(Eq.3)$$

Once the employed bee has completed its search, it communicates information about the food sources to the onlooker bee. The probability value helps the onlooker bee to choose a food source to be explored next with the following equation (4):

$$p_i = \frac{fitness_i}{\sum_{n=1}^{F} fitness_i} \ldots(Eq.4)$$

- *Onlooker Bee Phase*:

In this phase, onlooker bees utilize a roulette wheel selection mechanism to choose the food sources with the highest probability values. These selected sources are then assigned to the role of employed bees, facilitating further neighborhood exploration of the promising solutions identified in the employed bee phase.

- *Scout Bee Phase*:

Here, the scout bee decides whether to renew the food/solution source by evaluating the LIMIT variable. This variable is updated after each exploration. If the LIMIT value is more than MAX LIMIT, the scout bee will randomly update the food source. In the event that LIMIT is less than MAX LIMIT, the food/solution source will not be replenished.

This research as aforementioned applied the ABC method for feature selection by specifying parameters such as the number of features, bee count, iteration limitations, and also an arbitrary threshold number which defines the point at which a bee transitions into a scout. After randomly initializing the population with binary vectors that represent possible feature subsets, the initial fitness is determined by calculating the accuracy of a Random Forest classifier. Each binary vector represents specific features that are assessed for the accuracy of the classifier in order to establish its fitness. As the algorithm progresses through its primary stages, employed bees produce novel solutions by flipping random features. Provided that these solutions enhance fitness, they supersede prior ones. Onlooker bees subsequently choose alternatives by considering their fitness probability and apply

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

modifications, replacing the previous solutions if enhancements are achieved. The scout bees, then investigates novel random solutions in order to identify potentially superior feature subsets. Finally, the optimal feature subset is determined by evaluating the fitness score with the highest value. This strategy efficiently achieves a balance between exploration and exploitation, resulting in the identification of optimal feature subsets that improve the performance of the model.

➢ *Classification Algorithm*

To classify malware from the selected features using ABC, the research explored the performance of the RF and XG-Boost algorithm.

The Random Forest algorithm is an ensemble learning technique that enhances the accuracy of classification by constructing several decision trees and combining their predictions [25]. A bagging technique is used to train each tree on a randomly selected portion of the data and features [26]. In order to mitigate overfitting, this method minimizes variance by aggregating the predictions of individual trees. Random Forest exhibits robustness, excels in handling extensive datasets, and offers feature importance metrics, thereby proving to be highly efficient for both classification and feature selection tasks with modest tuning needs.

The Extreme Gradient Boosting (XG-Boost) improves predictive accuracy by using gradient boosting, a technique that constructs trees in a sequential manner to rectify the mistakes made by earlier trees [27]. It employs regularization to prevent overfitting and iteratively adds trees to minimize a loss function. XG-Boost is renowned for its rapidity and effectiveness resulting from optimization methods and computational parallelism. Furthermore, it has the capability to handle missing values and provides functionalities for assessing the significance of features, thereby proving to be quite efficient for intricate classification problems.

➢ *Performance Metrics*

To evaluate the performance of the ABC algorithm on the XG-Boost and RF algorithm, the below metrics are employed:

- *Precision:*

Focuses on the positive class and assesses how many of the predicted positives were actually true. Equation 5 depicts the mathematical expression for the precision metrics.

$$precision = \frac{TP}{TP + FP} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (Eq.5)$$

Recall: also known as sensitivity or true positive rate, evaluates how many of the actual positives were correctly predicted. The formula is shown in equation 6:

$$Recall = \frac{TP}{TP + FN} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (Eq.6)$$

F-measure (or F-score): is the harmonic mean of precision and recall. It provides a balanced evaluation of a classification model, taking into account both false positives and false negatives. Equation 7 shows the mathematical formular for the F-score.

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \dots (Eq.7)$$

Accuracy: measures the proportion of correctly classified instances (both true positives and true negatives) out of the total number of instances as shown in equation 8:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \dots\dots\dots (Eq.8)$$

IV. EXPERIMENTAL SETUP

For this study, we conducted analyses and made predictions on the Debrin android malware dataset using a 64-bit Windows operating system on a computer equipped with an Intel(R) Corel Trade Mark (TM) i5-2560QM CPU @2.40GHZ and 8.00 GB of RAM (Random Access Memory). The programme code was implemented using the Anaconda environment with the Python 3.11 software development kit. The application programming interfaces utilized were Sklearn, Pandas, Matplotlib, Seaborn, and NumPy.

➢ *Feature Selection Result*

As aforementioned, the research applied the ABC algorithm to select the most optimal subset features. The parameters used for the ABC algorithm are shown on Table 1:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

Table 1 ABC Parameters

Parameter	Description	Value	Role in Algorithm
n_features	Number of features in the dataset	216	Defines the size of each solution vector (binary vector indicating selected features).
n_bees	Number of bees (solutions) in the population	20	Determines the number of candidate solutions considered in each iteration.
max_iter	Maximum number of iterations for the algorithm	100	Sets the maximum number of iterations to search for the optimal feature subset.
limit	Limit for scout bees (condition to determine when to replace a solution)	10	Controls the number of iterations a solution can remain without improvement before being replaced by a scout bee.
fitness_function	Function to evaluate the quality of a solution	Random Forest	Assesses the performance of each solution based on the accuracy of a Random Forest classifier.
population	Initial set of solutions (binary vectors indicating feature selection)	Randomly initialized	Represents the starting point of the algorithm with different feature subsets.
fitness	Array to store the fitness values of solutions	Initialized to zero	Stores the performance score of each solution to guide the search process.

To optimize the selection of features for a predictive model, the ABC method for feature selection utilizes several crucial criteria. The n_features parameter determines the dimensions of the binary solution vectors, with each vector representing a separate possible subset of features. The number of features here, matches the size of the dataset features, which is 216. The parameter n_bees govern the quantity of potential solutions within the population, therefore exerting an impact on the extent of the search space. Expanding the number of bees enables the exploration of a wider range of solutions, but it may necessitate increased computational resources. The max_iter parameter determines the upper limit of iterations for the algorithm, therefore specifying the duration after which the search will be terminated. A larger value enables a more thorough search but also results in longer processing time. The limit parameter governs the frequency at which solutions are substituted by scout bees, therefore promoting the search of novel solutions in cases when currently existing ones reach a state of stagnation. The fitness function assesses the quality of each feature subset by considering the performance of the model, particularly its accuracy in this instance, therefore directing the optimization procedure. The population and fitness arrays play a vital role in monitoring and comparing various proposed solutions during the search process. Hence, the result of the selected features using ABC is shown on the code snippets from figure 2.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

```
# Print column names of selected features
selected_feature_names = get_selected_feature_names(selected_features, X_train)
print("Selected feature names:", selected_feature_names)
```

```
Selected feature names: ['onServiceConnected', 'ServiceConnection', 'SEND_SMS', 'Ljava.lang.Class.getCanonicalName', 'Ljava.lan
g.Class.getMethods', 'Ljava.lang.Class.cast', 'getBinder', 'Ljava.lang.Class.getField', 'Ljava.lang.Class.getDeclaredField', 'R
EAD_SMS', 'getCallingUid', 'Ljavax.crypto.spec.SecretKeySpec', 'android.intent.action.BOOT_COMPLETED', 'USE_CREDENTIALS', 'andr
oid.content.pm.PackageInfo', 'KeySpec', 'TelephonyManager.getLine1Number', 'HttpGet.init', 'SecretKey', 'Ljava.lang.Class.getMe
thod', 'Ljavax.crypto.Cipher', 'AUTHENTICATE_ACCOUNTS', 'android.telephony.gsm.SmsManager', 'WRITE_HISTORY_BOOKMARKS', 'Telepho
nyManager.getSubscriberId', 'INSTALL_PACKAGES', 'Runtime.getRuntime', 'WRITE_SYNC_SETTINGS', 'READ_HISTORY_BOOKMARKS', 'Ljava.l
ang.Class.forName', 'android.intent.action.PACKAGE_REPLACED', 'RECORD_AUDIO', 'android.os.IBinder', 'createSubprocess', 'NFC',
'WRITE_APN_SETTINGS', 'abortBroadcast', 'BIND_REMOTEVIEWS', 'android.intent.action.TIME_SET', 'READ_PROFILE', 'getCallingPid',
'BROADCAST_STICKY', 'android.intent.action.PACKAGE_REMOVED', 'RECEIVE_BOOT_COMPLETED', 'RESTART_PACKAGES', 'chmod', 'Ljava.lang
.Class.getDeclaredClasses', 'PathClassLoader', 'TelephonyManager.getSimSerialNumber', 'Runtime.load', 'TelephonyManager.getCall
State', 'READ_CALENDAR', 'sendMultipartTextMessage', 'PackageInstaller', 'VIBRATE', 'android.intent.action.ACTION_SHUTDOWN', 'A
CCESS_NETWORK_STATE', 'HttpPost.init', 'TelephonyManager.isNetworkRoaming', 'android.intent.action.PACKAGE_DATA_CLEARED', 'Http
UriRequest', 'UPDATE_DEVICE_STATS', 'DELETE_PACKAGES', 'GET_TASKS', 'GLOBAL_SEARCH', 'DELETE_CACHE_FILES', 'android.intent.acti
on.PACKAGE_CHANGED', 'REORDER_TASKS', 'WRITE_PROFILE', 'SET_WALLPAPER', 'READ_USER_DICTIONARY', 'Runtime.exec', 'BIND_WALLPAPER
', 'RECEIVE_WAP_PUSH', 'DUMP', 'android.intent.action.SENDTO', 'WRITE_SOCIAL_STREAM', 'WRITE_SETTINGS', 'REBOOT', 'TelephonyMan
ager.getNetworkOperator', '/system/bin', 'WRITE_GSERVICES', 'KILL_BACKGROUND_PROCESSES', 'ACCOUNT_MANAGER', 'android.intent.act
ion.CALL', 'STATUS_BAR', 'PERSISTENT_ACTIVITY', 'onBind', 'SET_TIME_ZONE', 'BIND_ACCESSIBILITY_SERVICE', 'ADD_VOICEMAIL', 'READ
_LOGS', 'Ljava.lang.Class.getResource', 'android.intent.action.PACKAGE_RESTARTED', 'INSTALL_LOCATION_PROVIDER', 'CHANGE_CONFIGU
RATION', 'CLEAR_APP_USER_DATA', 'CHANGE_WIFI_STATE', 'READ_FRAME_BUFFER', 'ACCESS_SURFACE_FLINGER', 'Runtime.loadLibrary', 'EXP
AND_STATUS_BAR', 'android.intent.action.BATTERY_LOW', 'SET_ACTIVITY_WATCHER', 'android.intent.action.ACTION_POWER_CONNECTED', '
ACCESS_MOCK_LOCATION', 'GET_PACKAGE_SIZE', 'MODIFY_PHONE_STATE', 'CHANGE_COMPONENT_ENABLED_STATE', 'SET_ORIENTATION', 'WRITE_EX
TERNAL_STORAGE', 'ACCESS_FINE_LOCATION', 'WRITE_SECURE_SETTINGS']
```

Fig 2 Selected Features Sample Snippet

➢ *Result Presentation*

After applying ABC for feature selections, the selected features are passed to RF and XG-Boost algorithm. Both the RF and XG-Boost models exhibit a very high performance on the classification of android malware, indicating their efficacy in managing the Debrin malware dataset. In terms of accuracy as shown on Table 4.2, RF attained a training accuracy of 0.9966 and a test accuracy of 0.9877. In contrast, XGBoost reported somewhat lower values of 0.9929 for training and 0.9844 for test accuracy. The analysis reveals that both models exhibit outstanding performance on both the training and test datasets. The RF model exhibits a marginal superiority over XGBoost, since it consistently achieves superior accuracy in both scenarios. Nevertheless, the disparities in test accuracy are insignificant, indicating that both models exhibit strong and reliable categorization abilities.

Further analysis of the classification reports demonstrates the respective capabilities of each model in effectively managing the two classes. RF models consistently outperform XG-Boost in terms of precision, recall, and F1-scores. The RF algorithm attains a precision of 0.99 and a recall of 1.00 for class 0, and a precision of 0.99 and a recall of 0.98 for class 1. At class 0, XG-Boost achieves a precision of 0.98 and a recall of 0.99. For class 1, it achieves a precision of 0.99 and a recall of 0.97.

Table 2 Accuracy and Classification Result Presentation

Metric	Random Forest	XG-Boost
Training Accuracy	0.9966	0.9929
Test Accuracy	0.9877	0.9844
Class 0 Precision	0.99	0.98
Class 0 Recall	1.00	0.99
Class 0 F1-Score	0.99	0.99
Class 1 Precision	0.99	0.99
Class 1 Recall	0.98	0.97
Class 1 F1-Score	0.98	0.98

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

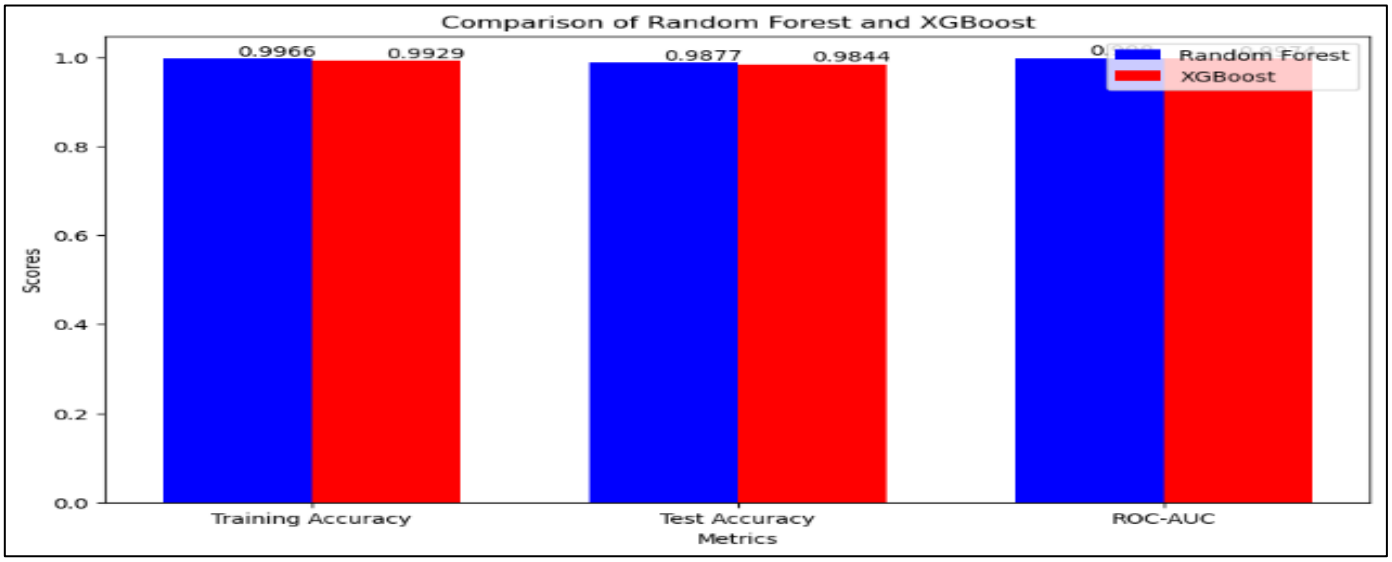

Fig 3 RF and XGBoost Training and Test Accuracy

Analytical, these results from table 4.2 and figure 3 indicate that RF has somewhat superior performance in terms of both precision and recall, resulting in somewhat higher F1-scores. This suggests that Random Forest Model may be more efficient in detecting and accurately categorizing occurrences belonging to both classes, especially class 1 (malignant).

To rigorously evaluate the models' performances, the Receiver Operating Curve and Area Under Curve (ROC-AUC) was considered. The ROC-AUC scores provide additional evidence of the excellent performance of both models. A ROC-AUC of 0.9980 was attained by RF, whereas XGBoost achieved a value of 0.9974 as shown on Figure 4. The scores are quite similar, with RF having a tiny advantage.

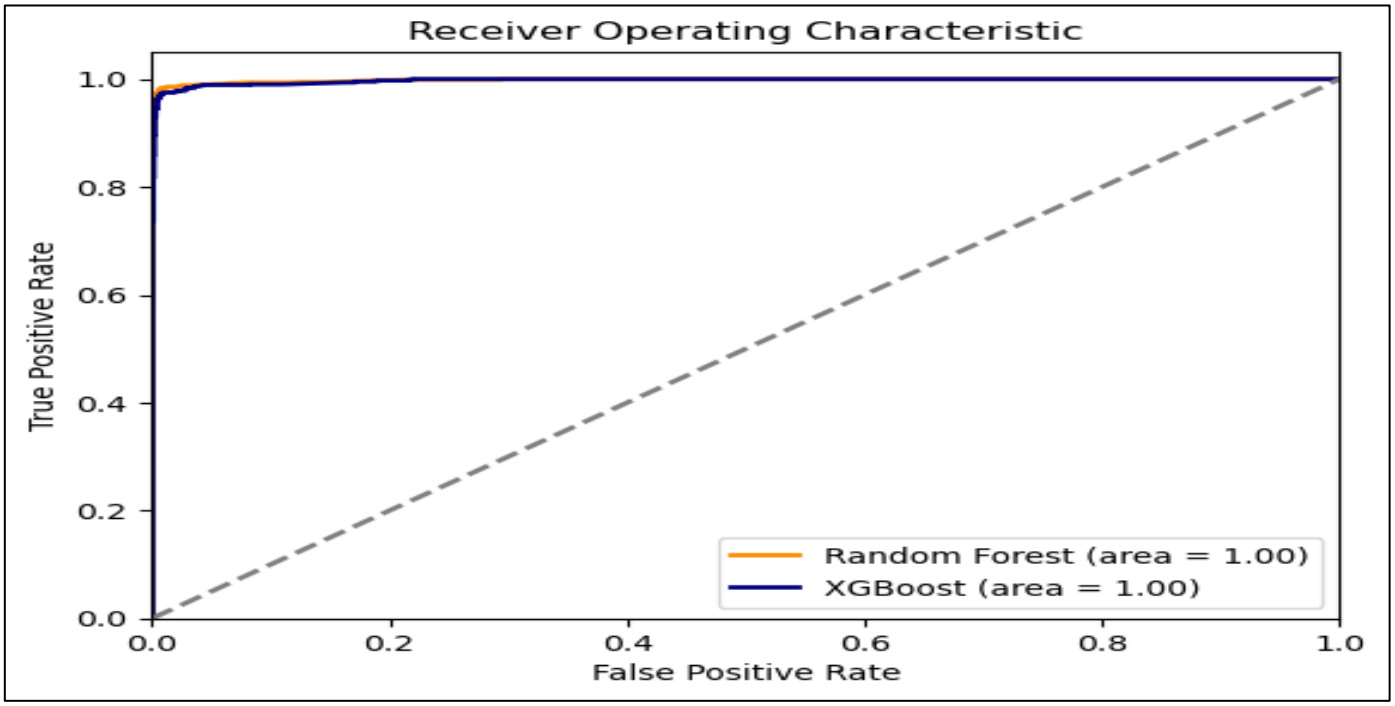

Fig 4 RF and XGBoost ROC-AUC

The ROC-AUC metric quantifies the capacity of a model to differentiate between different classes. Both values above 0.99 indicate that both models exhibit outstanding performance in this aspect. The somewhat higher ROC-AUC for Random Forest indicates a reasonably superior balance between the rates of true positives and false positives.

Conclusively, although both the RF and XG-Boost exhibit outstanding performance, RF emerges as the superior choice in terms of accuracy, classification metrics, and ROC-AUC. These findings indicate that it could be the most suitable option for this specific assignment. However, the disparities are insignificant, and each model would be a formidable contender based on particular criteria such as the duration of training, the ability to interpret, or limitations about deployment. Additional experimentation or refinement could perhaps further improve these results or result in the identification of an alternate model that may more effectively meet the requirements of the particular application.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

V. CONCLUSION

This study introduced an innovative framework for detecting Android malware by integrating the Artificial Bee Colony (ABC) algorithm for features selection with the RF and XGBoost classifiers. Through the optimization of feature subsets, the incorporation of ABC greatly improves the detection accuracy and model efficiency, which is essential considering the ever-changing nature of Android malware. The findings from the Debrin dataset demonstrate that both RF and XGBoost models exhibit strong classification performance. Among them, RF demonstrates somewhat better results in terms of accuracy, precision, recall, and ROC-AUC evaluations. The present results underscore the capacity of metaheuristic algorithms such as ABC to enhance the resilience and efficacy of malware detection systems. Notwithstanding the slight advantage of Random Forest (RF), both classifiers exhibit robust capabilities, indicating that either one might be used based on particular deployment needs. Potential future research might priorities the further enhancement of these models or the investigation of supplementary metaheuristic algorithms to improve the detection capabilities and tackle newly identified malware risks.

REFERENCES

[1]. **Sharma, Y., & Arora, A.** (2024). A comprehensive review on permissions-based Android malware detection. *International Journal of Information Security*, 23(3), 1877–1912. https://doi.org/10.1007/s10207-024-00822-2:contentReference[oaicite:3]{index=3}

[2]. **Redhu, A., Choudhary, P., Srinivasan, K., & Das, T. K.** (2024). Deep learning-powered malware detection in cyberspace: A contemporary review. *Frontiers in Physics*, 12, 1349463. https://doi.org/10.3389/fphy.2024.1349463:contentReference[oaicite:7]{index=7}

[3]. **Long, D. A.** (2024). Countering Russian cybergang ransomware attacks against critical infrastructure in the United States. *Doctoral dissertation*, National American University.

[4]. **Malik, M. S.** (2024). IoT malware: A comprehensive survey of threats, vulnerabilities, and mitigation strategies. *International Journal for Electronic Crime Investigation*, 8(1), 57–66. https://doi.org/10.54692/ijeci.2024.0801187:contentReference[oaicite:15]{index=15}

[5]. **Jhanjhi, N. Z., & Shah, I. A.** (2024). Navigating cyber threats and cybersecurity in the logistics industry. *IGI Global*.

[6]. **Jimmy, F. N. U.** (2024). Cybersecurity vulnerabilities and remediation through cloud security tools. *Journal of Artificial Intelligence General Science (JAIGS)*, 2(1), 196–233. https://doi.org/10.60087/jaigs.vol03.issue01.p233:contentReference[oaicite:23]{index=23}

[7]. **Geng, J., Wang, J., Fang, Z., Zhou, Y., Wu, D., & Ge, W.** (2024). A survey of strategy-driven evasion methods for PE malware: Transformation, concealment, and attack. *Computers & Security*, 137, 103595. https://doi.org/10.1016/j.cose.2023.103595:contentReference[oaicite:27]{index=27}

[8]. **Yunmar, R. A., Kusumawardani, S. S., & Mohsen, F.** (2024). Hybrid Android malware detection: A review of heuristic-based approach. *IEEE Access*, 12, 41255–41286.

[9]. **Hamid, R. A. I., & Riad, K.** (2024). Advancing malware artifact detection and analysis through memory forensics: A comprehensive literature review. *Journal of Theoretical and Applied Information Technology*, 102(4).

[10]. **Chenet, C. P., Savino, A., & Di Carlo, S.** (2024). A survey on hardware-based malware detection approaches. *IEEE Access*.

[11]. **Ispahany, J., Islam, M. R., Islam, M. Z., & Khan, M. A.** (2024). Ransomware detection using machine learning: A review, research limitations, and future directions. *IEEE Access*.

[12]. **Ullah, S., Li, J., Ullah, F., Chen, J., Ali, I., Khan, S., & Leung, V. C.** (2024). The revolution and vision of explainable AI for Android malware detection and protection. *Internet of Things*, 101320.

[13]. **Kiliçarslan, S., & Dönmez, E.** (2024). Improved multi-layer hybrid adaptive particle swarm optimization based artificial bee colony for optimizing feature selection and classification of microarray data. *Multimedia Tools and Applications*, 83(26), 67259–67281.

[14]. **Idris, N. F., Ismail, M. A., Jaya, M. I. M., Ibrahim, A. O., Abulfaraj, A. W., & Binzagr, F.** (2024). Stacking with recursive feature elimination-isolation forest for classification of diabetes mellitus. *PLOS One*, 19(5), e0302595.

[15]. **Arık, O. A., Köse, E., & Canbulut, G.** (2024). Metaheuristic algorithms to forecast future carbon dioxide emissions of Turkey. *Turkish Journal of Forecasting*, 8(1), 23–39. https://doi.org/10.34110/forecasting.1388906:contentReference[oaicite:59]{index=59}

[16]. **Widians, J. A., Wardoyo, R., & Hartati, S.** (2024). Feature selection based on chi-square and ant colony optimization for multi-label classification. *International Journal of Electrical & Computer Engineering*, 14(3).

[17]. **Li, X., Zhang, S., & Shao, P.** (2024). Discrete artificial bee colony algorithm with fixed neighborhood search for traveling salesman problem. *Engineering Applications of Artificial Intelligence*, 131, 107816.

[18]. **AlSobeh, A. M., Gaber, K., Hammad, M. M., Nuser, M., & Shatnawi, A.** (2024). Android malware detection using time-aware machine learning approach. *Cluster Computing*, 1–22.

[19]. **Kirubavathi, G., & Anne, W. R.** (2024). Behavioral based detection of android ransomware using machine learning techniques. *International Journal of System Assurance Engineering and Management*, 1–22.

[20]. **Kumar, K. R., & Gurrala, J.** (2024). Enhancing android malware detection through filter-based feature selection and machine learning classification. *Journal of Electrical Systems*, 20(3), 2801–2809.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25may1232

[21]. **Ma, R., Yin, S., Feng, X., Zhu, H., & Sheng, V. S.** (2024). A lightweight deep learning-based android malware detection framework. *Expert Systems with Applications*, 255, 124633.

[22]. **Yerima, S.** (2024). Android malware dataset for machine learning 2. *figshare*. https://doi.org/10.6084/m9.figshare.5854653.v1:contentReference[oaicite:87]{index=87}

[23]. **Andrew, I. W., Isah, C. S., Umar, M. M., & Bolaji, L. A.** (2024). Methodological approaches used for the Google machine reassignment problem. *Aligarh Journal of Statistics*, 44, 25–50.

[24]. **Efendi, R., Yusa, M., & Hallatu, S. T.** (2024). Classification of land suitability for soybean crops using the CART method and feature selection using an algorithm ABC. *IT Journal Research and Development*, 9(1), 1–13.

[25]. **Sun, Z., Wang, G., Li, P., Wang, H., Zhang, M., & Liang, X.** (2024). An improved random forest based on the classification accuracy and correlation measurement of decision trees. *Expert Systems with Applications*, 237, 121549.

[26]. **Lee, G. T., Lim, H. G., Wang, T., Zhang, G., & Jeong, M. K.** (2024). Double bagging trees with weighted sampling for predictive maintenance and management of etching equipment. *Journal of Process Control*, 135, 103175.

[27]. **Sekhar, J. C., Roy, T. L., Sridharan, K., Saravanan, K. A., & Taloba, A. I.** (2024). Explainable artificial intelligence method for identifying cardiovascular disease with a combination CNN-XG-Boost framework. *International Journal of Advanced Computer Science & Applications*, 15(5).