International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Malware Detection in Android Mobile Devices by Applying Swarm Intelligence Optimization and Machine Learning for API Calls

Suribabu Naick B[1], Srinivasa Rao P[2], Prakash Bethapudi[3], Surya Prakash Rao Reddy[4]

Submitted: 10/09/2022 **Accepted:** 24/12/2022

Abstract: Attacks on mobile devices, such as smartphones and tablets, have been on the rise as their use has grown. Malware attacks are some of the most significant threats, resulting in a variety of security issues as well as financial losses. The feature space-restricted malware analysis helps to detect malware effectively. The purpose of this research is to find the most useful features of Application Programming Interface (API) calls to improve the detection accuracy of Android malware. Two Swarm Intelligence Optimization techniques, namely Bald Eagle Search (BES) & Sailfish Optimization (SFO) are evaluated with API Calls to identify the most promising features for Android Malware detection. The BES & SFO features selection techniques are assessed using machine learning classifiers such as K-Nearest Neighbour (KNN), Decision Tree (DT), Support Vector Machine (SVM), Linear Regression (LR) and Random Forest (RF). Experimentation resulted in an accuracy of 98.92% with 21 features out of 100 API call features.

Keywords: *Android Malware, API Calls, Bald Eagle Search, Sailfish Optimization, Feature Selection, Machine Learning*

1. Introduction

The transition from traditional to smart technologies has changed the economy. Smart gadgets are currently growing at an exponential rate, both for personal and business use. In 2021, around 328 million smartphones were sold, according to a Gartner report [1][3]. Based on the popularity of smartphones, stakeholders have shown a strong desire to develop proprietary mobile operating systems (OS) [2]. Android is a leading giant in the telecommunications sector and a de facto standard for numerous smart phone makers since it is an open source and extensible platform. In 2019, With over 70% of the global smartphone market share, Android is the most popular platform. [4][6]. Apart from smartphones, Android is also gaining ground on smart watches and tablets.

Android is becoming a possible target for cyber threats [5], [7] due to its open-source nature. Malware authors are primarily motivated to create intricate malware in order to take advantage of established OS flaws. Malware is a phrase that refers to a group of malicious software variations that are specifically designed (i.e., Trojans, Viruses, Adware, Ransomware, and Spyware) to cause substantial data and system harm, such as remote control, information theft, privacy breaches, and privilege escalation. Sophisticated malware is impeccably designed and has the potential to completely disrupt the industry. Furthermore, the fact that Android malware can have a significant impact on both enterprises and end-users continues to fuel malware proliferation [8][9].

Malware detection on Android smartphones has been attempted numerous times [10][12]. Traditional signature-based malware detection approaches compare the signature of an APK file to the signature of a harmful application in a malware database, excluding malware that isn't in the database. In this context, enhanced detection algorithms [11] [13] are required for effective malware detection [14]. This paper's primary contributions are as follows:

Detecting suspicious APIs to accurately classify Android apps as goodware or malware.

A hybrid classifier combining wrapper-based feature selection techniques and machine learning classifiers is designed and implemented.

By penalising the learning process, an objective function for swarm intelligence optimization is defined, allowing for the efficient identification of near-optimal solutions.

Using a range of parameters to determine the optimal algorithm for predicting Android malware.

The remaining paper is structured as follows: Section 2 elucidates the related work, Section describes the methodology, experimentation setup is detailed in Section 4, section 5 explains the performance analysis and experimental results, and Section 6 gives out the conclusion & future work.

[1]*Assistant Professor, Department of ECE, GITAM (Deemed to be University) Visakhapatnam, Andhra Pradesh- 530045. India. sbhukya@gitam.edu*
[2]*Associate Professor, Department of CSE, MVGR College of Engineering (A), Vizianagaram, Andhra Pradesh, India. psr.sri@gmail.com*
[3]*Associate professor, Department of CSE GITAM School of Technology, GITAM Deemed to be University, Visakhapatnam, Andhra Pradesh, India. prakash.vza@gmail.com*
[4]*Assistant Professor, Department of ECE, GVP College of Engineering(A), Visakhapatnam, Andhra Pradesh, India. drspreddi4u@gmail.com*

1. Related Work

In recent years, multiple strategies [15][16] for detecting Android malware utilising API calls have made substantial progress. Bibi et al. [17] elucidated Gated Recurrent Unit (GRU) based Android Malware detection system. A Deep Learning (DL) driven architecture is experimented using datasets such as AndroZoo, Android Malware Dataset (AMD), etc., and tested using multiple evaluation metrics. The results show-cased that GRU-based model outperformed other competing models with increased accuracy. W. Yuan et al. [18][19] addressed the challenge of on-device training for Android Malware detection by implementing a light-weight Android Malware detection architecture. The model includes Support Vector Machines (SVM), AdaBoost, and Deep Learning based architectures. The experimentation obtained a robust model with improved accuracy and reduced computation time.

T. Kim et al. [20] studied the multimodal deep learning models using similarity-based feature extraction method. The model outperformed traditional models w.r.t efficacy of the diverse features and feature extraction methods. P. Feng et al. [21][22] proposed a dynamic analysis-based ensemble learning method for Android Malware detection called EnDroid. The proposed method includes feature selection method to remove noisy features for learning. The model achieved better outcome in terms of classification performance with an improved efficiency in malware detection. K. Liu et al. [23] surveyed different research works pertaining to Android Malware detection in terms of processing the data, selection of features, and machine learning models, etc. Similar work comparison is presented in Table. 1.

Paper, Year	Dataset Used	Classifier	Feature Selection	Accuracy
[13], 2021	Multiple Datasets	CNN	-	94.63
[14], 2019	Own Dataset	MNN	-	98
[15], 2018	Drebin	EnDroid	Chi-Square	98.18
[17], 2019	Drebin, AMD	A3CM-DNN	Static Analysis	98
[18], 2020	FARM	RF	K-Means	98.59
[23], 2018	Multiple Datasets	FalDroid	TF-IDF	97.2

J. Qiu et al. [24][25] addressed the problem of Malware Capability Annotation (MCA) by introducing Automatic Capability Annotation for Android Malware (A3CM). A3CM extracts features automatically and applies statistical methods for feature mapping. Android Malware is identified using a multi-label classification technique. The results show-cased that experimented methodology outperformed other competitive algorithms in terms of accuracy. Q. Han et al. [26][27] developed method called Feature transformation based AndRoid Malware detector (FARM) to transform features into a new feature domain for efficient detection of Android Malware. The FARM confirmed its effectiveness by detecting two malwares in VirusTotal that had previously been missed by other approaches.

As per the literature review, several researchers worked on different state-of-art [28], [29][30] and advanced techniques [31], [32] [33] for detection of malware in Android. The paper suggests a method focusing on wrapper-based feature selection techniques to identify the influential features for the detection of Android Malware. Bald Eagle Search and Sailfish Optimization techniques are extensively investigated for feature selection. Multiple machine learning classifiers are used for the classification of good ware from malware android applications.

2. Proposed Methodology

Fig. 1 depicts the suggested wrapper-based feature selection technique for Android Malware detection using a machine learning classifier. The dataset is divided in a 7:3 ratio of train and test sets. Initially, all the features are considered and a subset of features are selected iteratively using BES and SFO algorithms. The obtained reduced feature set is tested for its accuracy in classifying good ware applications from malware applications. Finally, the feature set with reduced feature dimensionality and improved accuracy is chosen for Android Malware detection.

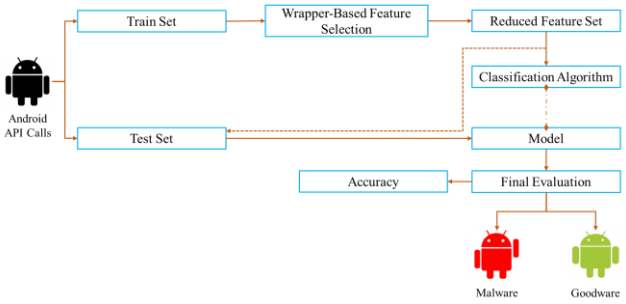

Fig. 1. Architecture of the proposed Android Malware Detection System

When picking a good feature set, feature reduction and feature selection are critical in data pre-processing. A strategy for lowering the number of features in a feature space is feature reduction, often known as dimensionality reduction. Feature reduction reduces multicollinearity in the feature population. The feature selection technique, on the other hand, determines which aspects of the ML model are the most influential. It removes characteristics that are either irrelevant or redundant, reducing the performance of the ML model. By lowering the processing cost of the model learning phase and increasing model understandability, the ML model's performance is improved. It decreases the model's training time in half and reduces overfitting.

In order to find an optimal solution with an underestimated feature set, the objective function used is crucial in converging the fitness throughout the optimization. The number of features chosen and the error achieved during the machine learning model evaluation affect the fitness of the selected feature subset, as indicated in Eq. (1).

$$f(x) = \tau * error + (1 - \tau) * \frac{u - s(l)}{u} \quad (1)$$

Where $\tau \epsilon [0,1]$ represents the penalty given to error produced during feature subset fitness calculation, and an increase in τ value implies an increase in penalty. The length of the entire feature set is denoted by u, and the length of the solution is denoted by l.

3.1 Wrapper-based Bald Eagle Search Optimized Feature Selection:

To validate the co-sequences of each step of hunting, the BES algorithm simulates the hunting behaviour of bald eagles. As a result, there are three stages to this algorithm: picking the search space, searching inside the selected search space, and swooping.

Selection: During the select stage, bald eagles identify and select the optimal place (in terms of food volume) inside the designated search space where they can pursue prey. Eq. (2) is a mathematical representation of this behaviour.

$$P_{new,i} = P_{best} + \alpha * r\ (P_{mean} - P_i) \quad (2)$$

where, $\alpha \epsilon [1.5, 2]$, controls the change in position. $r \epsilon [0, 1]$ is a random number, the current best selected search space is represented using P_{best}, the eagles that have used all the space previously is represented using P_{mean}.

Search: In the search stage, bald eagles look for prey inside a predetermined search zone and go in distinct ways within a spiral territory to speed up their search. Eq. (3) expresses the optimal swoop position mathematically.

$$P_{i,new} = P_i + y(\mathrm{i}) * (P_i - P_{i+1}) + x(\mathrm{i}) * (P_i - P_{mean}) \quad (3)$$

$$x(\mathrm{i}) = \frac{xr(\mathrm{i})}{max(|xr|)},\ y(\mathrm{i}) = \frac{yr(\mathrm{i})}{max(|yr|)} \quad (4)$$

$$xr(\mathrm{i}) = r(\mathrm{i}) * sin\big(\theta(\mathrm{i})\big),\ yr(\mathrm{i}) = r(\mathrm{i}) * cos\big(\theta(\mathrm{i})\big) \quad (5)$$

$$\theta(\mathrm{i}) = a * \pi * rand,\ r(\mathrm{i}) = \theta(\mathrm{i}) + R * rand \quad (6)$$

where, $a \epsilon [5, 10]$ is used to determine corner between point search in center point, no. of cycles is determined with $R \epsilon [0.5, 2]$, $rand \epsilon [0, 1]$ is a random number.

Algorithm 1: WBESOFS

Initialization of population (Pi)
Fitness calculation of the population f(pi)
While termination condition not met
    For every point i in population
        $P_{new,i} = P_{best} + \alpha * r\ (P_{mean} - P_i)$
        If $P_{new} < f(P_i)$ then
            $P_i = P_{new}$
            If $f(P_{new}) < f(P_{new})$ then
                $P_{best} = P_{new}$
            End If
        End If
    End For
    For every point i in population
        $P_{i,new} = P_i + y(\mathrm{i}) * (P_i - P_{i+1}) + x(\mathrm{i}) * (P_i - P_{mean})$
        If $P_{new} < f(P_i)$ then
            $P_i = P_{new}$
            If $f(P_{new}) < f(P_{new})$ then
                $P_{best} = P_{new}$
            End If
        End If
    End For
    For every point i in population
        $P_{i,new} = rand * P_{best} + x_1(\mathrm{i}) * (P_i - C_1 * P_{mean}) + y_1(\mathrm{i}) * (P_i - C_2 * P_{mean})$
        If $P_{new} < f(P_i)$ then
            $P_i = P_{new}$
            If $f(P_{new}) < f(P_{new})$ then
                $P_{best} = P_{new}$
            End If
        End If
    End For
    $K = K + 1$
End While

Swoop: During the swooping stage, bald eagles swoop from the prime location in the solution space to their targeted prey. All points are also moving in the direction of the best point. This behaviour is quantitatively illustrated in Eq. (7).

$$P_{i,new} = rand * P_{best} + x_1(i) * (P_i - C_1 * P_{mean}) + y_1(i) * (P_i - C_2 * P_{mean}) \quad (7)$$

$$x_1(i) = \frac{xr(i)}{m(|xr|)},\ y_1(i) = \frac{yr(i)}{m(|yr|)} \quad (8)$$

$$xr(i) = r(i) * sinh[\big(\theta(i)\big)],\ yr(i) = r(i) * cosh[\big(\theta(i)\big)] \quad (9)$$

$$\theta(i) = a * \pi * rand,\ r(i) = \theta(i) \quad (10)$$

here, $C_1, C_2 \epsilon [1, 2]$.

3.2 Wrapper-based Sailfish Optimized Feature Selection:

The SFO is a metaheuristic algorithm based on population. The sailfish are assumed to be candidate solutions in this technique, and the issue variables are the position of the sailfish in the search space. As a result, the population of the solution space is produced at random. With their changeable location vectors, sailfish can hunt in one, two, three, or hyper-dimensional space.

Sailfish, in fact, attack the prey school when none of their peers are attacking. The position of sailfish $X^i_{new\_SF}$ at i^{th} iteration is updated using Eq. (11)

$$X^i_{new\_SF} = X^i_{elite\_SF} - \lambda_i * \left(r(0,1) * \frac{X^i_{elite\_SF} + X^i_{injured\_S}}{2} - X^i_{old\_SF}\right) \quad (11)$$

where, $X^i_{elite\_SF}$ represents the elite sailfish position, $X^i_{injured\_S}$ indicates the best injured sardine. The $X^i_{old\_SF}$ represents current sailfish position, $r \epsilon [0, 1]$, λ_i represents i^{th} iteration coefficient obtained using Eq. (XX)

$$\lambda_i = 2 * r(0,1) * PD - PD \quad (12)$$

Here, PD indicates total preys at each iteration. PD is calculated using Eq. (xx)

$$PD = 1 - \left(\frac{N_{SF}}{N_{SF} + N_S}\right) \quad (13)$$

In each cycle, total sailfish and total sardines are represented using N_{SF} & N_S. Each sardine is required to update its position in relation to the current best position of the sailfish and the power of its attack at each iteration in order to imitate the haunting and catching process. The position of new sardine $X^i_{new\_S}$ is updated using Eq. (xx).

$$X^i_{new\_S} = r * (X^i_{elite\_SF} - X^i_{old\_S}) + AP \quad (14)$$

Here, AP represents the attack power of sailfish, which is generated using Eq. (xx).

$$AP = A * (1 - (2 * Itr * \varepsilon)) \quad (15)$$

To linearly decrease power attack value from A to 0, A & ε are considered as coefficients. Furthermore, the sailfish attack strength decides the number of sardines that adjust their locations and the

amount of dispersion they cause (AP). You can use the parameter AP to calculate the number of sardines (α) and the number of variables (β) to update their position as follows:

$$\alpha = N_S * AP \\ \beta = d_i * AP \quad (16)$$

Here, d_i indicates the total variables at i^{th} iteration. Finally, for increasing the chances of hunting new prey, sailfish position is updated with available best sardine hunted using Eq. (17).

$$X_{SF}^i = X_S^i \; if \; f(S_i) < f(SF_i) \quad (17)$$

where, X_S^i indicates the sardine's current position at i^{th} iteration and X_{SF}^i represents the sailfish current position at i^{th} iteration.

Algorithm 1: WSFOFS

Sailfish and Sardine population initialization

Fitness calculation for sailfish and sardines

Obtain best sailfish and sardine and consider as elite

While end criterion not obtained

For every sailfish

Calculate λ_i using Eq. (12)

Sailfish position updation using Eq. (11)

End For

$Attack\ Power$ Calculation using Eq. (15)

If $Attack\ Power < 0.5$

α is calculated using Eq. (16)

β is calculated using Eq. (16)

Sardine position updated using selected α and β with Eq.(14)

Else

All sardine positions are updated using Eq. (14)

End If

If superior solution for sardine is found

Sailfish is used to replace injured sardines with Eq. (17)

Hunted sardine is removed

Best sailfish and sardine updation

End If

End While

4. Experimental Setup

All experiments are run on a 64-bit Windows 10 operating system with a 2.30 GHz Intel® CoreTM i5 processor, 8 GB RAM, and 2TB hard drive as well as a Jupyter platform that supports machine learning and matplotlib packages. Python 3.7 is the programming language utilised.

Table. 2. Description of API Call Sequence dataset

Dataset	No. of Features	No. of Samples		Dataset Size
API Call	100	43,876		
Sequence Data		Goodware	Malware	17.1 MB
		1,079	42,797	

IEEE Dataport provided API call sequence data used in the investigation. The data contains 43,876 API call sequences, with 42,797 being malware API call sequences and 1,079 being goodware API call sequences. The Cuckoo Sandbox environment was used to collect the experimented data, which was then confirmed using Virus Total [24]. The data for API call sequences is described in Table. 2.

5. Performance Analysis and Experimentation Results:

To confirm classification accuracy, the suggested system for android malware detection employs various classification assessment metrics such as MSE, RMSE, Precision, Recall, F-Score, and Accuracy:

$$Precision = \frac{T_{pos}}{F_{pos} + T_{pos}} \quad (18)$$

$$Recall = \frac{T_{pos}}{F_{neg} + T_{pos}} \quad (19)$$

$$F1 - Score = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall} \quad (20)$$

$$Accuracy = \frac{T_{pos} + T_{neg}}{T_{pos} + F_{pos} + T_{neg} + F_{neg}} \quad (21)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \quad (22)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \quad (23)$$

Here, T_{pos} denotes the samples correctly classified as goodware, T_{neg} denotes the samples correctly classified as malware, F_{pos} denotes the samples incorrectly classified as goodware, and F_{neg} denotes the samples incorrectly classified as malware. $\hat{Y}_i$ denotes the predicted output, Y_i denotes the actual output, and n represents the no. of samples.

On the API calls sequence dataset, the BES and SFO algorithms wrapped with LR, DT, SVM, KNN, and RF are tested for their performance. All of the experiments are conducted for a total of 10 iterations with a total of 10 agents. The analysis findings are listed in Table. 3. with a graphical representation in Fig. 2. The SFO optimizer when wrapped with RF classifier stemmed better results with 79% reduced feature space and an accuracy of 98.92%.

The experimental results show that utilising a wrapper-based sailfish optimised feature selection approach, the dimensionality of feature space was lowered while classification accuracy was maintained using an RF Classifier. Fig. 3 shows the RF Classifier's evaluation metrics using the WSFOFS algorithm. Table. 4. shows the list of 21 features selected by the wrapper-based firefly feature selection technique utilising the RF Classifier.

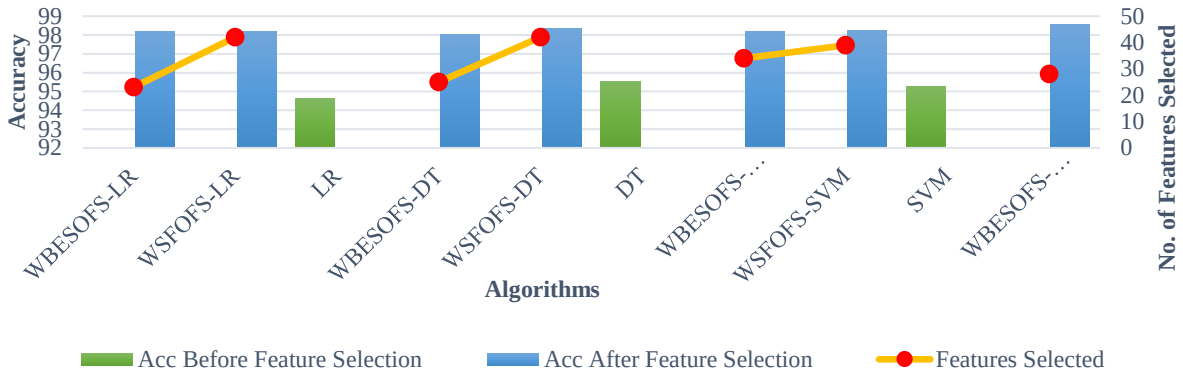

Fig. 2. Performance comparison of BES & SFO wrapped with LR, DT, SVM, KNN & RF

Table. 3. Accuracy comparison of BES & SFO wrapped with LR, DT, SVM, KNN & RF

S. No	Classifier	Accuracy Before Feature Selection	Feature Selection Method	Accuracy After Feature Selection	% Change in Accuracy	Features Selected	% Decrease in Features
1	LR	94.6311	BES	98.1654	3.7348	23	77%
			SFO	98.211	3.783	42	58%
2	DT	95.5485	BES	98.0173	2.5838	25	75%
			SFO	98.3705	2.9534	42	58%
3	SVM	95.2654	BES	98.1882	3.068	34	66%
			SFO	98.2547	3.1378	39	61%
4	KNN	95.3586	BES	98.5525	3.3493	28	72%
			SFO	98.4529	3.2449	42	58%
5	**RF**	**95.6485**	BES	98.9061	3.4058	37	63%
			SFO	**98.9236**	**3.4241**	**21**	**79%**

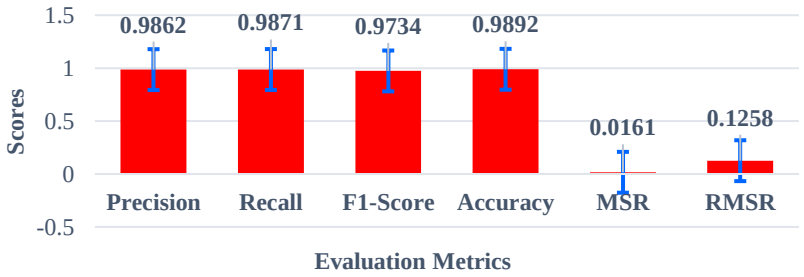

Fig. 3. Evaluation Metrics of WSFOFS with RF

Table. 4. Features Selected by WSFOFS wrapped with RF

S. No	API No.	API Description
1	2	ExitWindowsEx
2	3	FindResourceW
	5	CreateRemoteThreadEx
4	6	MessageBoxTimeoutW
5	7	InternetCrackUrlW
6	8	StartServiceW
7	9	GetFileSize
8	11	GetFileInformationByHandle
9	14	SetWindowsHookExA
10	15	RegSetValueExW
11	16	LookupAccountSidW
12	17	SetUnhandledExceptionFilter
13	19	GetComputerNameW
14	20	RegEnumValueA
15	24	Recv
16	25	GetFileSizeEx
17	27	SetInformationJobObject
18	29	CryptDecrypt
19	31	InternetOpenW
20	32	CoInitializeEx
21	34	GetAsyncKeyState

The AUC_ROC curves for the SFO method wrapped with machine learning Classifiers were created because SFO outperformed the comparison algorithm. When compared to the area under the complete feature set, the area under the AUC_ROC curve of the SFO classifier embedded with RF is smaller when utilising a reduced feature set. Fig. 4-8. show the AUC_ROC graphs for all of the machine learning classifiers.

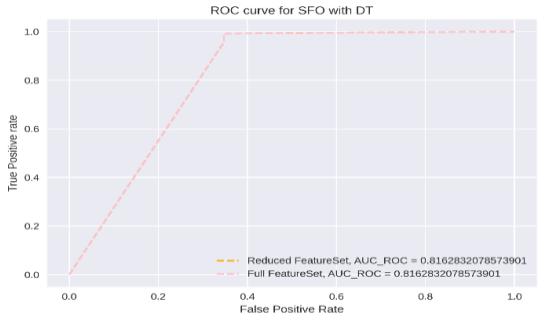

Fig. 4. AUC_ROC Curve of SFO with DT

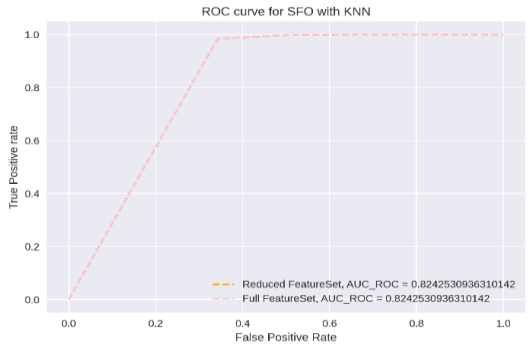

Fig. 5. AUC_ROC Curve of SFO with SVM

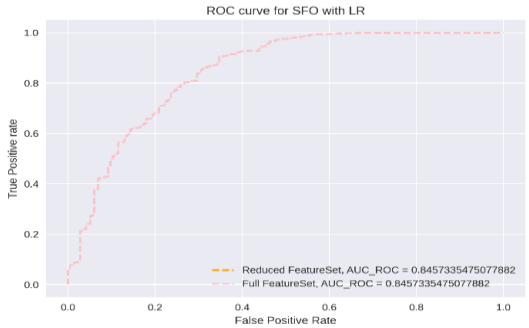

Fig. 6. AUC_ROC Curve of SFO with LR

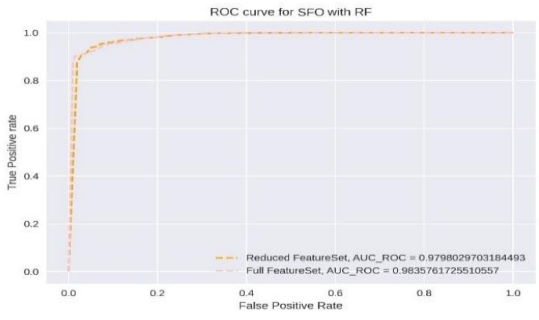

Fig. 7. AUC_ROC Curve of SFO with RF

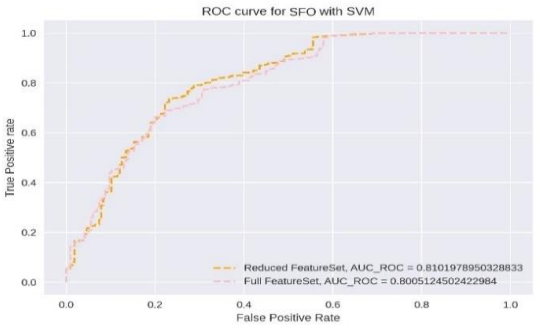

Fig. 8. AUC_ROC Curve of SFO with SVM

There is no feature selection approach that uses wrapper-based sailfish optimization on API call sequence data, according to the literature review. As a result, Table. 5. shows a comparison of accuracy based on relevant work.

Table. 5. Related Work Accuracy Comparison

Paper, Year	Classifier	Feature Selection	Accuracy
[13], 2021	CNN	-	94.63%
[14], 2019	MNN	-	98.00%
[15], 2018	EnDroid	Chi-Square	98.18%
[17], 2019	A3CM-DNN	Static Analysis	98.00%
[18], 2020	RF	K-Means	98.59%
[23], 2018	FalDroid	TF-IDF	97.20%
This Paper	**RF**	**WSFOFS**	**98.92%**

6. Conclusion & Future Work

On Android, malware threats are increasing, and evasion methods are becoming more intricate. Android mobile systems and applications are widely used in smart cities and industries. One of the most powerful and effective techniques for maintaining Android system security, particularly for smart cities and industrial platforms, is malware detection. Malware detection research based on machine learning has recently received a lot of interest. However, the bulk of accessible solutions need feature analysis and selection, which is a time-consuming process known as feature engineering that is based on simulated experience. As a result, feature selection and detection performance must be constantly improved.

In this scenario, the wrapper-based feature selection techniques WBESOFS & WSFOFS are explored in this paper. Initially, complete feature set is passed onto the wrapper-based feature selection methods. The obtained reduced feature set is then used to classify the good ware from malware android malware applications. Among the BES & SFO, the SFO when wrapped with RF classifier achieved superior results in minimizing the dimensionality of the feature space to 79% with an improved accuracy of 98.92%.

Designing and implementing a hybrid architecture that incorporates advanced deep learning techniques to improve the efficiency of Android malware detection and classification, as well as other optimizations for systematic feature reduction using a high-dimensional feature space as considered as future work.

References:

[1] https://www.gartner.com/en/newsroom/press-releases/2021-09-01-2q21-smartphone-market-share. [Online – Accessed 21 Feb. 22].

[2] C. Zhang, P. Patras and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2224-2287, third quarter 2019, doi: 10.1109/COMST.2019.2904897.

[3] Pradeep Bheemavarapu, P S Latha Kalyampudi and T V Madhusudhana Rao, "An Efficient Method for Coronavirus Detection Through X-rays using deep Neural Network", Journal of Current Medical Imaging, [online Available] Vol.18, No. 6, with ISSN: 1875-6603,2022.

[4] Statista. Mobile OS market share 2021 - https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ [Online – Accessed 21 Feb. 22].

[5] Y. Zhang et al., "Familial Clustering for Weakly-Labeled Android Malware Using Hybrid Representation Learning," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3401-3414, 2020, doi: 10.1109/TIFS.2019.2947861.

[6] S Satyanarayana, "Privacy Preserving Data Publishing Based On Sensitivity in Context of Big Data Using Hive", Journal of Bigdata (Springer), Volume:5, Issue:20, ISSN: 2196-1115, July 2018.

[7] K. Xu, Y. Li and R. H. Deng, "ICCDetector: ICC-Based Malware Detection on Android," in IEEE Transactions on Information Forensics and Security, vol. 11, no. 6, pp. 1252-1264, June 2016, doi: 10.1109/TIFS.2016.2523912.

[8] G. Meng, M. Patrick, Y. Xue, Y. Liu and J. Zhang, "Securing Android App Markets via Modeling and Predicting Malware Spread Between Markets," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 7, pp. 1944-1959, July 2019, doi: 10.1109/TIFS.2018.2889924.

[9] P.Mahesh Kumar,P. Srinivasa Rao, "Frequent Pattern Retrieval on Data Streams by using Sliding Window", EAI Endorsed Transactions on Energy web,Volume:5,issue:35,2021.

[10] K. Tian, D. Yao, B. G. Ryder, G. Tan and G. Peng, "Detection of Repackaged Android Malware with Code-Heterogeneity Features," in IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 1, pp. 64-77, 1 Jan.-Feb. 2020, doi: 10.1109/TDSC.2017.2745575.

[11] H. Zhu, Y. Li, R. Li, J. Li, Z. You and H. Song, "SEDMDroid: An Enhanced Stacking Ensemble Framework for Android Malware Detection," in IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 984-994, 1 April-June 2021, doi: 10.1109/TNSE.2020.2996379.

[12] T.V. Madhusudhana Rao, Suresh Kurumalla, Bethapudi Prakash, "Matrix Factorization Based Recommendation System using Hybrid Optimization Technique, EAI Endorsed Transactions on Energy Web, Volume:5, issue:35, 2021.

[13] Demontis, Ambra et al. "Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection." IEEE Transactions on Dependable and Secure Computing, 16 (2019): 711-724.

[14] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," in IEEE Access, vol. 6, pp. 4321-4339, 2018, doi: 10.1109/ACCESS.2018.2792941.

[15] T.V. Madhusudhana Rao, P.S. Latha Kalyampudi, "Iridology based Vital Organs Malfunctioning identification using Machine learning Techniques", International Journal of Advanced Science and Technology, Volume: 29, No. 5,PP: 5544 – 5554,2020.

[16] P. Faruki et al., "Android Security: A Survey of Issues, Malware Penetration, and Defenses," in IEEE Communications Surveys & Tutorials, vol. 17, no. 2, pp. 998-1022, Secondquarter 2015, doi: 10.1109/COMST.2014.2386139.

[17] Bibi, A. Akhunzada, J. Malik, J. Iqbal, A. Musaddiq and S. Kim, "A Dynamic DL-Driven Architecture to Combat Sophisticated Android Malware," in IEEE Access, vol. 8, pp. 129600-129612, 2020, doi: 10.1109/ACCESS.2020.3009819.

[18] S.Vidya sagar Appaji, P. V. Lakshmi, "Maximizing Joint Probability in Visual Question Answering Models", International Journal of Advanced Science and Technology Vol. 29, No. 3, pp. 3914 – 3923,2020.

[19] W. Yuan, Y. Jiang, H. Li and M. Cai, "A Lightweight On-Device Detection Method for Android Malware," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 9, pp. 5600-5611, Sept. 2021, doi: 10.1109/TSMC.2019.2958382.

[20] T. Kim, B. Kang, M. Rho, S. Sezer and E. G. Im, "A Multimodal Deep Learning Method for Android Malware Detection Using Various Features," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 773-788, March 2019, doi: 10.1109/TIFS.2018.2866319.

[21] Vidya sagar Appaji setti ,P Srinivasa Rao , "A Novel Scheme For Red Eye Removal With Image Matching", Journal of Advanced Research in Dynamical & Control Systems, Vol. 10, 13-Special Issue, 2018.

[22] P. Feng, J. Ma, C. Sun, X. Xu and Y. Ma, "A Novel Dynamic Android Malware Detection System with Ensemble Learning," in IEEE Access, vol. 6, pp. 30996-31011, 2018, doi: 10.1109/ACCESS.2018.2844349.

[23] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun and H. Liu, "A Review of Android Malware Detection Approaches Based on Machine Learning," in IEEE Access, vol. 8, pp. 124579-124607, 2020, doi: 10.1109/ACCESS.2020.3006143.

[24] P Srinivasa Rao, Krishna Prasad, P.E.S.N, "A Secure and Efficient Temporal Features Based Framework for Cloud Using MapReduce", springer, 17th International Conference on Intelligent Systems Design and Applications.

[25] J. Qiu et al., "A3CM: Automatic Capability Annotation for Android Malware," in IEEE Access, vol. 7, pp. 147156-147168, 2019, doi: 10.1109/ACCESS.2019.2946392.

[26] Q. Han, V. S. Subrahmanian and Y. Xiong, "Android Malware Detection via (Somewhat) Robust Irreversible Feature Transformations," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3511-3525, 2020, doi: 10.1109/TIFS.2020.2975932.

[27] Madhusudhana Rao, T.V., Srinivas, Y, "A Secure Framework For Cloud Using Map Reduce", Journal Of Advanced Research In Dynamical.

[28] T. Chakraborty, F. Pierazzi and V. S. Subrahmanian, "EC2: Ensemble Clustering and Classification for Predicting Android Malware Families," in IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 2, pp. 262-277, 1 March-April 2020, doi: 10.1109/TDSC.2017.2739145.

[29] H. Bai, N. Xie, X. Di and Q. Ye, "FAMD: A Fast Multifeature Android Malware Detection Framework, Design, and Implementation," in IEEE Access, vol. 8, pp. 194729-194740, 2020, doi: 10.1109/ACCESS.2020.3033026.

[30] P Srinivasa Rao, Sushma Rani N, "An Efficient Statistical Computation Technique for Health Care Big Data using R", Scopus, IOP Conference Series: Materials Science and Engineering, Volume: 225, ISSN:1757-8981, ISSUE NO :012159,2017.

[31] J. Singh, D. Thakur, T. Gera, B. Shah, T. Abuhmed and F. Ali, "Classification and Analysis of Android Malware Images Using Feature Fusion Technique," in IEEE Access, vol. 9, pp. 90102-90117, 2021, doi: 10.1109/ACCESS.2021.3090998.

[32] Z. Yuan, Y. Lu and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," in Tsinghua Science and Technology, vol. 21, no. 1, pp. 114-123, Feb. 2016, doi: 10.1109/TST.2016.7399288.

[33] Krishna Prasad, M.H.M., Thammi Reddy, K, "A Efficient Data Integration Framework in Hadoop Using MapReduce" Published in Computational Intelligence Techniques for Comparative Genomics

[34] M. Fan et al., "Android Malware Familial Classification and Representative Sample Selection via Frequent Subgraph Analysis," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 8, pp. 1890-1905, Aug. 2018, doi: 10.1109/TIFS.2018.2806891.

[35] Angelo Oliveira.Malware Analysis Datasets: API Call Sequences. 2019.doi: 10.21227/tqqm-aq14.url: http://dx.doi.org/10.21227/tqqm-aq14.

[36] Nagesh Vadaparthi, Srinivas Yarramalle, "A Novel clustering approach using Hadoop Distributed Environment", Springer, (Applied Science and Technology), ISSN:2191-530X, Volume:9, pp:113-119, October 2014