Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Comparative analysis of malicious android-based software detection with trending metaheuristic algorithms

Kötü amaçli android tabanli yazılım tespitinin trend meta-sezgisel algoritmalar ile karşılaştırılmalı analizi

Mehmet Şirin BEŞTAŞ[1], Özlem BATUR DİNLER[2]*

[1]Mehmet Şirin Beştaş, Siirt University, Faculty of Engineering, Department of Computer Engineering, Siirt, Turkey.
mehmetsirinbestas@gmail.com

[2]Özlem Batur Dinler, Siirt University, Faculty of Engineering, Department of Computer Engineering, Siirt, Turkey.
o.b.dinler@siirt.edu.tr

lenmemis Sürüm - Uncorrected Version

Received/Geliş Tarihi: 27.01.2024 Revision/Düzeltme Tarihi: 06.04.2024 doi: 10.5505/pajes.2024.93646
Accepted/Kabul Tarihi: 30.04.2024 Research Article/Araştırma Makalesi

Abstract

Today, Android malware threats and attacks are rapidly increasing due to their use and popularity. Therefore, the need for systems effectively detecting malware is also increasing day by day. This study proposes the use of various trending metaheuristic algorithms for optimal feature selection (FS) in the detection of Android malware. For this purpose, the ten most prominent recent metaheuristic algorithms (RMAs) for feature selection such as Artificial Bee Colony Algorithm (ABC), Firefly Algorithm (FA), Grey Wolf Optimisation (GWO), Ant Lion Optimisation (ALO), Crow Search Algorithm (CSA), Sine Cosine Algorithm (SCA), Whale Optimisation Algorithm (WOA), Salp Swarm Algorithm (SSA), Harris Hawk Optimization (HHO) and Butterfly Optimization Algorithm (BOA) were used for feature selection in this study. The efficiency of these algorithms is evaluated with five different machine learning (ML) methods on two well-known datasets of Android applications (Drebin-215 and Malgenome-215). The results obtained are also compared with five well-known and widely used conventional metaheuristic algorithms (CMAs) for solving this problem. Extensive experimental results show that incorporating RMA into Android malware detection is a valuable approach.

Anahtar kelimeler: Android, Feature selection, Machine learning, Malicious software, Recent Metaheuristic algorithms

Öz

Günümüzde Android kötü amaçlı yazılım tehdit ve saldırıları, kullanımları ve popülerlikleri nedeniyle hızla artmaktadır. Bu nedenle, kötü amaçlı yazılımları etkili bir şekilde tespit edebilecek sistemlere olan ihtiyaç da gün geçtikçe artmaktadır. Bu çalışma, Android kötü amaçlı yazılımların tespitinde optimum özellik seçimi (FS) için trend olan çeşitli meta-sezgisel algoritmaların sarmalama yöntemi ile kullanılmasını önermektedir. Bu amaçla, bu çalışmada Yapay Arı Kolonisi Algoritması (ABC), Ateş Böceği Algoritması (FA), Gri Kurt Optimizasyonu (GWO), Karınca Aslanı Optimizasyonu (ALO), Karga Arama Algoritması (CSA), Sinüs Kosinüs Algoritması (SCA), Balina Optimizasyon Algoritması (WOA), Salp Sürü Algoritması (SSA), Harris Şahin Optimizasyonu (HHO) ve Kelebek Optimizasyonu Algoritması (BOA) gibi özellik seçiminde en öne çıkan on güncel meta-sezgisel algoritma (RMA) kullanılmıştır. Bu algoritmaların verimliliği, Android uygulamalarının iyi bilinen iki veri kümesi (Drebin-215 ve Malgenome-215) üzerinde beş farklı makine öğrenmesi (ML) yöntemi ile değerlendirilmiştir. Ayrıca, elde edilen sonuçlar bu problemin çözümünde yaygın olarak kullanılan ve iyi bilinen beş geleneksel metasezgisel algoritma (CMAs) ile de karşılaştırılmıştır. Kapsamlı deneysel sonuçlar, RMA'nın Android kötü amaçlı yazılım tespitine dahil edilmesinin değerli bir yaklaşım olduğunu göstermektedir.

Keywords: Android, Özellik seçimi, Makine Öğrenmesi, Kötü amaçlı yazılım, Güncel meta-sezgisel algoritmalar

1 Introduction

Android is the most widely used operating system (OS) among mobile devices. In this respect, Android plays a key role in communities, as it accounts for a large proportion of users worldwide and has a large market share. In this respect, cyber risk and security management on Android devices is of critical importance, given the huge impact that cybercrime can bring to Android users. Especially malware is one of the most dangerous threats to the cyber management processes at the highest level [1]. Malware is malicious software (e.g. viruses, ransomware, trojan horses, and spyware) that can damage or execute harmful actions on devices [2]. Malware attacks cause devastating effects such as theft of information, corruption of files and infection of the entire device network [3]. In this regard, the detection of Android malware is among the most effective techniques used to eliminate or reduce the risks and dangers posed by Android malicious activities.

In recent years, machine learning (ML) based security solutions have been extensively used in Android malware detection [4]. However, while ML methods train their models on high-dimensional feature datasets, the fact that the dataset may contain many irrelevant and redundant features has a huge impact on the computational and time complexity and can also affect the performance of the algorithm [5]. In this case, the burden of ML methods needs to be lightened [6]. Therefore, Feature Selection (FS) can be used to minimise complexity, irrelevant and redundant data [7]. FS is the process of finding

[*]Corresponding author/Yazışılan Yazar

the smallest possible number of features describing a data set in the same way as the original features. Feature selection is a very important pre-processing step for data mining techniques as it improves the performance of the prediction process in terms of speed and accuracy and also provides a better understanding of the stored data. The success of the FS process depends on a balance between two important factors: selecting the minimum number of features and ensuring maximum accuracy in the results [8]. Feature selection methods can be categorised into two main groups: filter and wrapper methods. Filter methods are not dependent on the learning or classification algorithm. Constantly, the emphasis is placed on the overall attributes of the data. Wrapper methods interact with the classifier and invariantly contain the classification algorithm. In contrast to the filter, these methods require more computation power and yield more precise outcomes in comparison to filter methods.

Metaheuristic algorithms have been recently developed and implemented in literature to tackle FS challenges: Genetic Algorithm (GA) [9], Simulated Annealing (SA) [10], Ant Colony Optimization (ACO) [11], Differential Evolution (DE) [12], Particle Swarm Optimization (PSO) [13], ABC [14], FA [15], GWO [16], SSA [17] and so on. These algorithms are preferred in order to produce results with low costs at high accuracy and speed of the problems encountered [18].

Literature survey revealed that many researchers have used various metaheuristic algorithms for feature selection to detect Android malware. However, most researchers have investigated their analyses with a limited number of algorithms. In addition, there is no detailed research on the current metaheuristic algorithms proposed for feature selection in this study. At the same time, there is very little literature on the use of our proposed state-of-the-art metaheuristic algorithms for Android malware detection. Various methods have been proposed, mostly based on GA. Therefore, in this paper, we compare the performance of ten RMAs (ABC, FA GWO, ALO, CSA, SCA, WOA, SSA, HHO, and BOA) in detecting Android malware and the results are also compared with five well-known CMAs (GA, PSO, SA, ACO and DE) which are widely used in solving this problem [19] in the study.

The contribution of this study is summarised below:

1. In this study, the performance and effectiveness of the ten most prominent recent metaheuristic algorithms for feature selection in the literature in solving the Android malware detection problem are investigated for the first time.
2. Evaluation of the Android malware detection system providing the best Performance based on ten recent metaheuristic algorithms, two datasets (Drebin-215 and Malgoneme-215), two validation options (70:30 and 10-k cross validation), and five ML methods (DT, KNN, NB, RF and SVM) in various scenarios.
3. Determining the Android malware detection system providing the best performance.
4. A comprehensive empirical investigation of which recent metaheuristic algorithms exhibit a competitive approach to FS.

The article's structure comprises the following sections: The focus of Section 2 is on related works. The proposed model is presented in Section 3. Section 4 contains the results of all experiments are analysed and discussed in detail. Finally, Section 5 has the conclusion.

2 Literature review

In recent years, many studies have proposed various methods for Android malware detection based on RMAs for feature selection to improve performance and reduce costs. If the studies to be carried out in this field are mentioned; Beştaş and Dinler [19] used five conventional metaheuristic algorithms such as GA, PSO, SA, ACO and DE, which are the most widely used in the literature for FS, to select the features that best represent benign and malicious applications on Android. They evaluated the efficiency of these algorithms on the Drebin-215 and MalGenome-215 datasets using DT, KNN, NB, RF and SVM ML methods. According to the results obtained from the experiments, DE-based feature selection and RF method have better accuracy rates. Naic et al. [20] utilized the Bald Eagle Search and Sailfish Optimisation techniques in combination with KNN, DT, SVM, Linear Regression (LR), and RF ML models. The results demonstrated a high accuracy rate of 98.92% when applied to Application Programming Interface (API) call squence dataset. Sharma [21] proposed a hybrid methodology for the detection of Android malware. This strategy integrates the feature-important Water Drop Algorithm (FIWDA) with a ML algorithm. In order to assess the consequences of the suggested paradigm, we specifically focused on two openly accessible Android malware datasets, namely Drebin-215 and Malgoneme-215. The experimental findings exhibit significant promise. The F1 score achieved was 98%, with accuracy, recall, and precision also reaching 99%. The error rates, specifically RMSE and MAE, were as low as 0.1184 and 0.014, respectively. Varma et al. [22] proposed Bat optimization approach for wrapper-based FS on the CICInvesAndMal2019 benchmark dataset. This approach enhances precision by 1.67 percent and eliminates 87.41% of superfluous characteristics in a permission-based Android malware dataset with a high number of dimensions. Chakravarthy [23], an investigation was conducted into nature-inspired wrapper-based metaheuristic algorithms, including whale, firefly, and bat optimisation algorithms, in order to analyse various Android permission patterns that are appropriate for the detection of malware. An assortment of classification algorithms based on ML are utilized in the assessment process, including LR, SVM, KNN, DT, RF, Gradient Boosting, and Extreme Learning Machine. In an experiment utilising the high-dimensional CICInvesAndMal2019 feature dataset comprising 4115 features, the FA-based wrapper-based feature selection achieved an enhanced classification accuracy of 95.28%, surpassing the performance of alternative algorithms in this regard. Bhagwat and Gupta [24] introduced a metaheuristic FS technique that incorporates the Gravitational Search Algorithm (GSA) and the GA. Additionally, they introduced a correlation known as the Correlated Genetic GSA (CGGSA). The XGBoost and AdaBoost approaches can be utilized to optimise the characteristics for malware detection. Elkabbash et al. [25] introduced a novel detection approach that relies on the Random Vector Functional Link (RVFL) optimizer. This approach incorporates Artificial Jellyfish Search optimization and subsequently reduces the dimensionality of Android applications' attributes. JavaScript was employed to ascertain an optimal configuration of the RVFL in order to enhance the performance of the classifier. Alzubi et al. [26] investigated and evaluated a novel machine learning approach for the purpose of detecting Android malware. The method employed in this study incorporates the utilization of HHO and SVM techniques. The HHO technique is specifically designed to enhance the hyperparameter optimisation of the SVM technique. The SVM algorithm is responsible for classifying malware based on the most effective method selected, and it generates optimal solutions for the weighted features. A methodology was presented by Sulaiman et al. [27] that provided a methodology that utilized the WOA for feature selection of permission-based features in Android applications to increase their classification accuracy. The outcomes of their study exhibited enhanced precision in comparison to the most recent detection models that employed WOA without feature selection.

3 Proposed Model

The block diagram of the proposed method is shown in Figure 1. In the recommended method, it was aimed to design an Android malware detection system based on the wrapper-based feature selection technique with ten metaheuristic algorithms that have a recent and widespread use in literature studies. The list of these metaheuristic algorithms and their usage are provided in Figure 2. Here, a KNN classifier is used as an evaluator and Recent Metaheuristic Algorithms (RMAs) are used to obtain the optimal feature subset. The KNN classifier determines the accuracy of the features (feature subset) selected by all RMA algorithms. Here, since KNN is the most preferred classifier, we consider the KNN classifier to evaluate the accuracy of the selected feature subset. The representation of Figure 2 is the "Wrapper Feature Selection with RMAs" block in Figure 1. For this purpose, FS, training, and testing were applied to the Drebin-215 and Malgoneme-215 datasets of Android applications. Then, we shuffle the samples in the relevant dataset and use as input to the wrapper-based FS technique using RMAs. Then, the optimum feature subset obtained was given as input to five different ML methods.

ted Ve

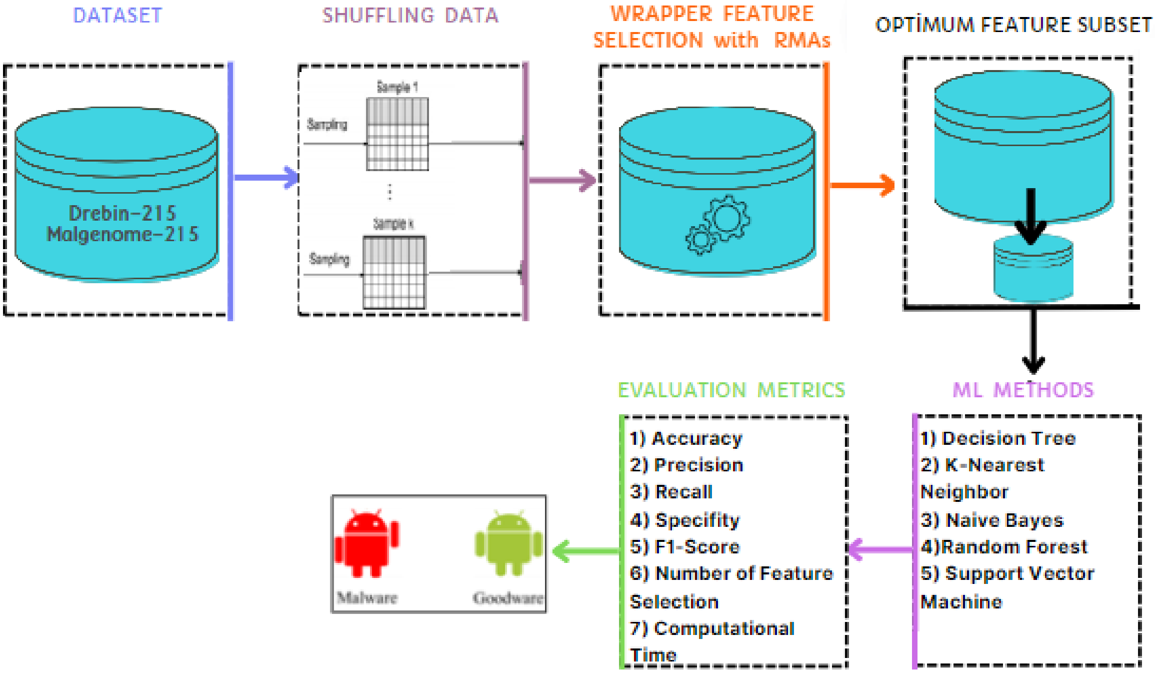


Figure 1. The architecture of the proposed model.

Düzenlen

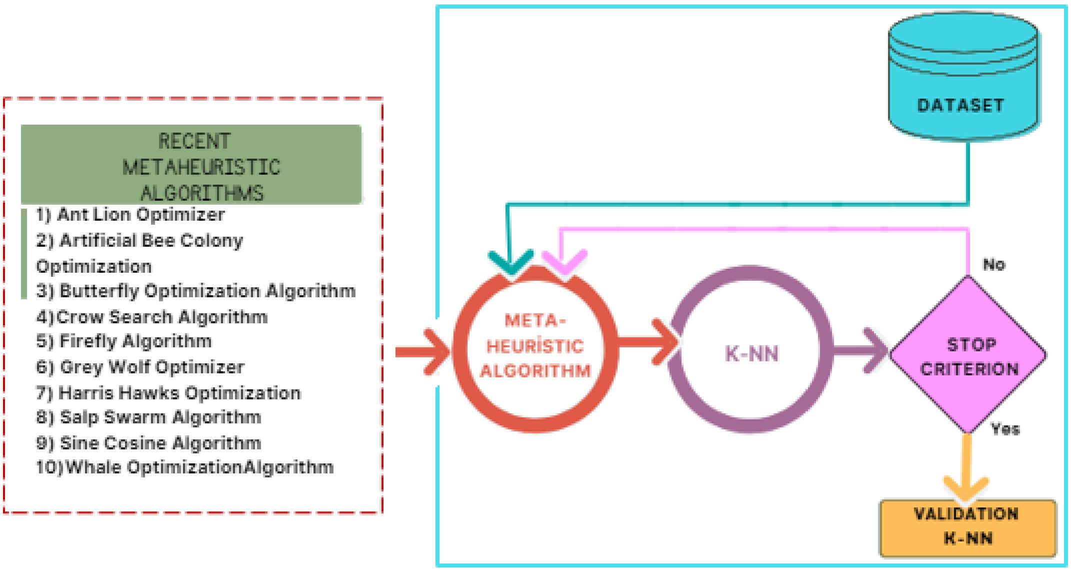


Figure 2. The wrapper feature selection approach with RMAs classifier with KNN.

3.1 Dataset

Android malware detection results of the proposed approach are tested using Drebin-215 [28, 29] and MalGenome-215 [29, 30] datasets. The Drebin-215 dataset was created in 2013, whereas the MalGenome-215 dataset was created in 2012. The Drebin-215 dataset comprises of which 5.560 are malicious and 9476 are benign samples. The MalGenome-215 dataset comprises of which 2,539 are benign and 1.560 are malicious samples. Both datasets contain 215 features with two classes benign or malware. Table 1 shows the details of each of the datasets.

Table 1. Details of datasets.

Datasets	Year	Number of Samples	Number of Bening Samples	Number of Malware Samples	Number of Features
Drebin-215[23]	2013	15036	9476	5560	215
Malgenome-215[25]	2010-2012	3799	2539	1260	215

3.2 Shuffling Data

Data shuffling is a preprocessing technique often used to improve model learning. Data shuffling was used to address potential problems arising from patterns in the sequential order of the training samples that could lead to overfitting [31].

3.3 Wrapper Feature Selection with RMAs

FS is commonly considered as an initial phase where the most optimal subset of features is identified from a pool of all available features. The RMA approaches employ a population of potential solutions. The solutions are typically expressed as a vector of values. In metaheuristic FS algorithms, solutions are typically represented using a binary encoding of a certain collection of features [32, 33]. For instance, when considering a subset of features with seven dimensions (1,0,1,1,1,0,1), the value 1 indicates that the feature is selected, whereas the value 0 indicates that the feature is not picked [32]. A candidate solution is observed in the form of its chosen features. Five out of the seven features that comprise this solution are chosen.

When making preparations for an optimisation procedure, it is imperative to give careful consideration to the objective function. Feature selection, as a wrapper technique, aims to preserve only a minimal set of features while maximizing the accuracy of the learning algorithm. This study aims to minimise both the selection ratio and the classification error rate through the use of the following objective function [32,33]:

$$\text{Objective function} = \alpha\text{ER}+\beta\left(\frac{|S|}{|O|}\right) \qquad (1)$$

where ER is the classification error, |S| is the length of the selected subset of features, and |O| is the length of all features in the original dataset. α = [0,1] and β = (1 - α).

3.3.1 Artifcial Bee Colony Optimization (ABC)

It is a natural adaptive metaheuristic algorithm inspired by the foraging and communication behaviour of honeybees [14]. The ABC comprises three groups of bees: employed bees, onlookers, and scout bees. An onlooker bee chooses a food source in accordance with the probability value P_i linked to that particular food source [34-35].

$$P_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \qquad (2)$$

where fit_i represents the fitness value of solution i; SN represents the number of food sources which is equal to the number of employed bees or onlooker bees[35].

In order to generate a candidate food position V_i = [$v_{i,1}$, $v_{i,2}$, . . . , $v_{i,D}$] from the old one X_i= [$x_{i,1}$, $x_{i,2}$, . . . , $x_{i,D}$] in memory, ABC algorithm employs the subsequent expression [35]:

$$\boldsymbol{v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj})}, \tag{3}$$

where $k \in \{1, 2, . . . , SN\}$ and $j \in \{1, 2, . . . , D\}$ are randomly selected indexes; k must be distinct from i; D is the number of variables (problem dimension); $\Phi_{i,j}$ is a random number betwen -1 and 1[35].

Food sources in the population are randomly generated and assigned to employed bees as [36]:

$$x_{i,j} = x_{min,j} + rand(0,1)(x_{max,j} - x_{min,j}). \tag{4}$$

where x_{min}, x_{max} are the lower and upper bounds of the solution vectors.

3.3.2 Ant Lion Optimizer (ALO)

It is a natural adaptive metaheuristic algorithm inspired by the hunting strategies of ant lions in nature [37]. Ants have a stochastic movement pattern in their pursuit of sustenance. The stochastic movement of ants is represented by the following model:

$$\mathrm{X(t)} = \sum_{\mathrm{i=1}}^{\mathrm{t}} 2\mathrm{r}\,(\mathrm{t_i}) - 1 \tag{5}$$

where t represents the number of iterations, and $r(t)$ is a random number within the range [0, 1].

3.3.3 Butterfly Optimization Algorithm (BOA)

It is a natural adaptive metaheuristic algorithm inspired by the movement and foraging behaviour of butterflies in nature [38]. Butteries release a scent in order to attract other butteries. The fragrance emitted by the butteries is calculated using Equation 6.

$$f_i = cI^a, \quad i = 1,2, \ldots, NP. \tag{6}$$

where the butterfly fragrance is represented by f_i, the sensory modality is represented by c, the stimulus intensity is represented by I, a is a power exponent within the range [0 to 1], and NP denotes the number of butterflies. Mathematical model of the global and local search phases of BOA is shown as follows [39]:

$$X_i^{t+1} = X_i^t + (r^2 \times X_{best}^t - X_i^t) \times f_i \tag{7}$$

$$X_i^{t+1} = X_i^t + (r^2 \times X_j^t - X_k^t) \times f_i \tag{8}$$

where X_i^t indicates the position of the i^{th} butterfly in the i^{th} iteration, X_{best}^t best indicates the global optimal individual, r ∈ (0, 1) is a random number, and X_j^t and X_k^t are the j^{th} individual and the k^{th} individual choosen randomly [39].

3.3.4 Crow Search Algorithm (CSA):

It is a natural adaptive metaheuristic algorithm inspired by the flock hunting behaviour of crows in nature [40]. Every crow in CSA is aware of its hidden food location. The term "secret location" pertains to the optimal solution that a specific crow has been able to identify thus far, represented as $m^{i,iter}$ for crow i during iteration iter. At some point, crow j may opt to observe its hiding place, i.e., $m^{j,iter}$. If crow j notices crow i, it will fly to a random position to mislead its follower[33].

$$x^{i,iter+1} = \begin{cases} x^{i,iter} + r_i x \\ fl^{i,iter} x \\ (m^{j,iter} - x^{i,iter}),\ r_j \ \geq \ AP^{j,iter} \\ \\ \text{a random position ,} \qquad \text{otherwise} \\ \end{cases} \tag{9}$$

where r_i and r_j are two random numbers within the range [0, 1], $fl^{i,iter}$ represents the flight distance of crow i, and $AP^{j,iter}$represents crow j's awareness possibility.

3.3.5 Firefly Algorithm (FA)

It is a natural adaptive metaheuristic algorithm inspired by the communication with light and attraction behaviour of fireflies in nature [15]. For two fireflies x_i and x_j, they can be updated as follows:

$$x_i^{(t+1)} = x_i^{(t)} + \beta_0 e^{-\gamma r_{i,j}^2} \left(x_i^t + x_j^t\right) + \propto \varepsilon_i^t \tag{10}$$

where $x_i^{(t+1)}$ represents the position of firefly i at iteration t+1 displacement, α is the step size, β0 is the attractiveness at r=0, γ is represented by the absorption coefficient, second part is the attraction, while the third is randomization [Guo].

3.3.6 Grey Wolf Optimizer (GWO)

It is a natural adaptive metaheuristic algorithm inspired by the behaviour of grey wolf packs in nature [16]. The behaviour of grey wolves engaging in prey gathering is characterised by the following equations:

$$\vec{X} = \left|\vec{A}.\vec{P}(t) - \vec{W}(t)\right| \tag{11}$$

$$\vec{W}(t+1) = \vec{P}(t) - \vec{B}.\vec{X}(t) \tag{12}$$

where $\vec{A}$ and $\vec{B}$ are coefficient vectors, $\vec{P}$ and $\vec{w}$ are position vectors of prey and wolves.

Wolves live in four hierarchical societies: α , β, δ, and ω. The positions of other grey wolves are adjusted according to the presence of α, β, and δ wolves. The formulas used for this calculation are as follows[41]:

$$\begin{gathered} D_\alpha = |C_1 \times X_\alpha - X(t)| \\ D_\beta = \left|C_2 \times X_\beta - X(t)\right|, \\ D_\delta = |C_3 \times X_\alpha - X(t)| \end{gathered} \tag{13}$$

Düzenlenmemiş Sürüm - Uncorrected Version

$$\begin{aligned} X_1 &= X_\alpha - A_1 \times\ D_\alpha \\ X_2 &= X_\beta - A_2 \times\ D_\beta\ , \\ X_3 &= X_\delta - A_3 \times\ D_\delta \end{aligned} \tag{14}$$

and

$$X(t+1) = (X_1 +\ X_2 + X_3)/3 \tag{15}$$

3.3.7 Harris Hawk's optimization (HHO)

It is a natural adaptive metaheuristic algorithm inspired by the hunting strategies of Harris hawks observed in nature [33-42]. The hawks adopt perching locations based on the positions of other hawks and the prey (rabbit), or they perch randomly as seen below:

$$X(t+1) = \begin{cases} X_{rand}(t) - \\ r_1|X_{rand}(t) - 2r_2X(t)|, \quad q \geq 0.5 \\ \left(X_{rabbit}(t) - X_m(t)\right) - \\ r_3\left(LB + r_4(UB - LB)\right), \quad q < 0.5 \end{cases} \tag{16}$$

where $X(t)$ and $X(t{+}1)$ represents the current and next positions of the hawk, respectively. $X_{rabbit}(t)$ denotes is the current location of the rabbit, q, r_1, r_2, r_3 and r_4 denotes random numbers within the range [0,1]. LB and UB denote the boundaries of the variables, $X_{ran}(t)$ denote the current position of a randomly choosen hawk, and X_m denote the average position of the hawks [33].

The energy of the rabbit is provided as follows:

$$E = 2E_0(1 - \frac{t}{T}) \tag{17}$$

where E represents the escaping energy, T represent the number of iterations, and E_0 represents the initial energy state. At each iteration of the method, E_0 is randomly set between -1 and 1.

The hawks silently encircle the rabbit before executing the surprise attack once the prey has become exhausted. The following describes the computational model of this behaviour:

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)| \tag{18}$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \tag{19}$$

where $\Delta\ X(t)$ denotes the distance between the rabbit and the hawk at iteration t and J denote the jump strength of the rabbit.

In order to keep the positions current, we use the following formula:

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \tag{20}$$

3.3.8 Sine Cosine Algorithm (SCA):

It is a natural fit metaheuristic algorithm inspired by the properties of the trigonometric functions sine and cosine [43]. In SCA, the mathematical equations for updating positions are given Equation (21) [44]:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times sin(r_2) \times \left|r_3P_i^t - X_i^t\right|, & r_4 < 0.5 \\ X_i^t + r_1 \times cos(r_2) \times \left|r_3P_i^t - X_{ii}^t\right|, & r_4 \geq 0.5 \end{cases} \tag{21}$$

where X_i^{t+1} denotes the i^{th} dimension of the solution at t^{th} iteration, P_i denotes the target in i^{th} dimension. $sin(.)$, $cos(.)$, and $|.|$ represent the *sine, cosine,* and *absolute value*, respectively. r_1, r_2, r_3 and $r_4$variables are randomly produced.

3.3.9 Salp Swarm Algorithm (SSA)

It is a natural adaptive metaheuristic algorithm inspired by the movement and grouping behaviour of organisms called salps in the sea [17]. The mathematical model of herd behaviour exhibited by Salp chains starts by dividing the population into two groups, leaders, and followers. Salps have specific behaviors called the salp chain. This behavior is used for foraging. The location update equation for the leader salp follows [45] :

$$x_j^i = \begin{cases} F_j + c_1((ubj - lbj)c_2 + lb_j), & c_3 > 0.5 \\ F_j - c_1((ubj - lbj) + lb_j)c_2, & c_3 < 0.5 \end{cases} \tag{22}$$

where, X_j^i represents the leader salp position in the j-th dimension, F_j represents j-th dimensional target food source, c_1, c_2 and c_3 represent random numbers, ubj and lbj represent the upper and lower bounds in the j-th dimension, respectively.

The coefficient c1 is calculated as follows:

$$c_1 = 2e^{-(\frac{4m}{M})^2} \tag{23}$$

where, m represents the current step and M represents the total number of steps. c_2 and c_3 numbers are randomly generated coefficients in the range [0, 1].

The following equation is used to update the position of the salps following the leader salp.

$$x_j^i = \frac{1}{2}\left(x_j^i + x_j^{i-1}\right), \quad i \geq 2 \tag{24}$$

where, x_j^i represents the i-th follower salp location in the j-th dimension.

3.3.10 Whale Optimization Algorithm (WOA)

It is a natural adaptive metaheuristic algorithm inspired by the movement and hunting behaviour of whale pods at sea [46]. During prey encirclement, other whales try to approach the best agent and update their position using as follows[47]:

$$\vec{D} = \left|\vec{C}.\vec{X}^*(t) - \vec{X}(t)\right| \tag{25}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A}.\vec{D} \tag{26}$$

Düzenlenmemiş Sürüm - Uncorrected version

where t represents instant iteration. $\vec{X}^{*}$ represent the location of the best individual ever achieved. $\vec{X}$ represents the location vector. $\vec{A}$ and $\vec{C}$ vectors denotes the specific coefficients. The following equations are used to determine these coefficients.

$$\vec{A} = 2\vec{a}.\vec{r} - \vec{a} \tag{27}$$

$$\vec{C} = 2.\vec{r} \tag{28}$$

where $\vec{a}$ is parameter whose initial value decreases linearly from 0 to 2 during iterations. $\vec{r}$ is a random number in [0-1].

Humpback whales execute the attack with two approaches of both shrinking containment and curled updating of position[48]. These approach can be expressed as follows:

$$\vec{X}_{(t+1)} = \begin{cases} \vec{X}^{*}_{(t)} - \vec{A}.D, & p < 0.5 \\ \vec{D}e^{bt}\cos(2\pi l) + \vec{X}^{*}_{(t)}, & p \geq 0.5 \end{cases} \tag{29}$$

where b describes shape of the fixed value logarithmic curled. l gets a random numbers between -1 and 1.

During the initial stage of exploration, this update is executed in a random manner. The time model equations are formulated in the following manner:

$$\vec{D} = \left|\vec{C}.\vec{X}_{rand} - \vec{X}\right| \tag{30}$$

$$\vec{X}(t+1) = \vec{X}_{rand} - \vec{A}.\vec{D} \tag{31}$$

where $\vec{X}_{rand}$ denotes the position vector (the position of the whale) that is randomly chosen from the population.

3.4 ML Methods

In this study, each feature subset obtained based on the FS of various RMAs was used for the classification of Android malware detection using DT[49], KNN[50], NB[51], RF[52], and SVM[53] ML methods.

3.5 Evolution Metrics

The performance metrics to evaluate and compare the proposed methods include accuracy, precision, sensitivity, specificity, F1-Score, number of feature selections and computation time. The mathematical expressions for these performance measures are denoted by Equation (32-36) correspondingly[54].

$$Accuracy\ (Acc) = (TP + TN)/\ (TP + FP + FN + TN) \tag{32}$$

$$Precision\ (Pr) = TP/(TP + FP) \tag{33}$$

$$Recall\ (Rc) = TP/(TP + FN) \tag{34}$$

$$Specificity\ (Sp) = TN/(FP + TN) \tag{35}$$

$$F1 - Score\ (F) = (2*\mathrm{Pr}*\mathrm{Rc}) / (\mathrm{Pr}+\mathrm{Rc}) \tag{36}$$

where, the terms TP and TN represent true positive and true negative, respectively. Similarly, the abbreviations FP and FN denote false positive and false negative, respectively.

- TP: The number of Android apps that are actually malware and are predicted as malware,
- TN: The number of Android apps that are actually benign and are predicted as benign,
- FP: The number of Android apps that are actually benign but are predicted as malware.
- FN: The number of Android apps that are actually malware but are predicted as benign.

4 Result and Discussion

Experimental results are presented and discussed in this section. This study aimed to design an Android malware detection system based on the wrapper-based feature selection method with ten metaheuristic algorithms that have a current and widespread use in literature studies. The proposed approach is evaluated on two Android datasets (Drebin-215 & Malgoneme-215), two validation models (Model-1 (70:30) & Model-2 (k-fold = 10)) and five different ML methods (DT, KNN, NB, RF & SVM). Table 2 lists the control parameters of the RMAs used. The population size (N) for the algorithms is set to 100. The maximum number of iterations (T) is 100. Each algorithm was run 30 times and its average values were used for a fair evaluation.

Table 2. Algorithm parameter settings

RMA	Parameter	Value
ABC	Limit Parameter	max=5
ALO	Selection method	Roulette wheel
BOA	Modular modality	C=0.01
	Switch probability	P=0.8
CSA	Awareness probability	AP=0.1
	Flight length	f_1 = 1.5
FA	Alpha	α = 1
	Beta	β_0 = 1
	Gamma	γ =1
GWO	Convergence parameter	α =2
	Random variables	r_1, r_2= [0,1]
HHO	Convergence parameter	1.5
SCA	Convergence factor	2
SSA	Controlling parameter	C_1= Decreases
	Random variables	exponentially from 2 to 0 c_2, c_3= [0,1]
WOA	Convergence parameter	a = Decreases linearly from 2 to 0 b =1

4.1 Accuracy/F1- Score Performance by Algorithm

A comparison of the average accuracy and F1-Score results between the algorithms is shown in Table 3 and Table 4, respectively. According to the experimental results in Table 2, Drebin-215 and Malgoneme-215 datasets have the highest accuracy rate with the combination depending on ABC + RF + Model-2 parameters. The same conclusion was reached for the F1-Score measurements.

Düzeltilmemiş Sürüm - Uncorrected Version

Table 3. Average accuracy of all algorithms

		Drebin-215					Malgoneme-215				
	RMA	DT	KNN	NB	RF	SVM	DT	KNN	NB	RF	SVM
Model-1	ABC	0.94649	0.9791	0.93022	0.98162	0.94109	0.97428	0.98654	0.9595	**0.98838**	0.94267
	ALO	0.94513	0.98081	0.95622	0.98188	0.935	0.97272	0.98683	0.95622	0.98188	0.94709
	BOA	0.94225	0.9768	0.9768	0.98033	0.94765	0.97126	0.98341	0.9559	0.98622	0.94765
	CSA	0.94585	0.98066	0.98066	**0.9827**	0.94058	0.97252	0.98636	0.9566	0.98768	0.94058
	FA	0.9449	0.9777	0.9777	0.98226	0.94443	0.97179	0.98557	0.9566	0.98783	0.94264
	GWO	0.94497	0.97857	0.97857	0.98113	0.9588	0.96772	0.98417	0.95932	0.98378	0.9588
	HHO	0.94554	0.97917	0.92965	0.98154	0.94324	0.97112	0.97989	0.95953	0.98639	0.94644
	SCA	0.93832	0.97466	0.97466	0.97715	0.95843	0.96787	0.98127	0.95291	0.98358	0.96646
	SSA	0.94399	0.97801	0.97801	0.98092	0.94377	0.97182	0.98589	0.95563	0.98651	0.94377
	WOA	0.94362	0.98032	0.98032	0.98244	0.93929	0.97073	0.9849	0.95525	0.98642	0.94363
Model-2	ABC	0.94602	0.98003	0.93022	**0.99**	0.94938	0.97456	0.98802	0.95968	**0.98966**	0.94989
	ALO	0.94862	0.98343	0.93191	0.98506	0.94243	0.97306	0.98727	0.95802	0.98916	0.94975
	BOA	0.93656	0.97783	0.92942	0.98126	0.9525	0.97009	0.98358	0.95166	0.98614	0.95118
	CSA	0.94511	0.98073	0.93789	0.98269	0.94959	0.97291	0.98698	0.95494	0.98935	0.94879
	FA	0.944	0.97913	0.92376	0.98184	0.9545	0.97531	0.98521	0.95496	0.98904	0.95044
	GWO	0.94119	0.97955	0.92441	0.98078	0.96318	0.97078	0.98603	0.95611	0.98677	0.96571
	HHO	0.94479	0.98129	0.92506	0.98287	0.94687	0.97243	0.98465	0.95279	0.98733	0.94803
	SCA	0.93836	0.97749	0.92361	0.97908	0.96003	0.96813	0.9815	0.94642	0.98483	0.97042
	SSA	0.94488	0.97816	0.93221	0.98243	0.94971	0.97338	0.98452	0.95434	0.98866	0.94958
	WOA	0.94634	0.98202	0.92944	0.98426	0.94423	0.97415	0.98581	0.9508	0.98894	0.95044

Table 4. Average F1-score of all algorithms

		Drebin-215					Malgoneme-215				
	RMA	DT	KNN	NB	RF	SVM	DT	KNN	NB	RF	SVM
Model-1	ABC	0.92715	0.97165	0.90778	0.97485	0.91356	0.9611	0.97971	0.93993	**0.98235**	0.90521
	ALO	0.92535	0.974	0.97525	0.97525	0.90365	0.95896	0.98011	0.93494	0.97525	0.91322
	BOA	0.92044	0.96854	0.90389	0.9731	0.92404	0.95669	0.97488	0.93423	0.97905	0.92404
	CSA	0.92619	0.97381	0.91047	**0.97635**	0.91271	0.95851	0.97936	0.93535	0.98128	0.91271
	FA	0.92453	0.96983	0.90825	0.97577	0.91888	0.95754	0.97819	0.93535	0.98155	0.90528
	GWO	0.92408	0.9709	0.89935	0.97421	0.94119	0.92408	0.97609	0.93933	0.97538	0.94119
	HHO	0.92539	0.9718	0.907	0.97478	0.91704	0.95642	0.96962	0.94006	0.97931	0.912
	SCA	0.91457	0.96562	0.89479	0.9687	0.94072	0.95125	0.97177	0.92903	0.97498	0.94671
	SSA	0.92347	0.97025	0.90822	0.97392	0.91783	0.95752	0.9787	0.93404	0.97947	0.91783
	WOA	0.92272	0.97333	0.90872	0.976	0.91059	0.95571	0.97722	0.93337	0.97933	0.907
Model-2	ABC	0.92596	0.97295	0.90778	**0.98481**	0.92662	0.96044	0.98195	0.93982	**0.9843**	0.91825
	ALO	0.92986	0.97755	0.90994	0.97963	0.91559	0.95936	0.98078	0.93804	0.98351	0.91795
	BOA	0.91385	0.96998	0.90613	0.97438	0.93155	0.95465	0.97524	0.92818	0.97892	0.92058
	CSA	0.92465	0.97393	0.91178	0.97636	0.92697	0.95907	0.98035	0.93295	0.98382	0.91636
	FA	0.92374	0.97176	0.89958	0.97519	0.93462	0.96288	0.97765	0.93327	0.98336	0.91926
	GWO	0.91948	0.97228	0.89752	0.9737	0.94781	0.95582	0.97895	0.93435	0.97992	0.94549
	HHO	0.92452	0.97465	0.90191	0.97661	0.9226	0.9584	0.97682	0.92986	0.98076	0.9149
	SCA	0.91477	0.9695	0.89646	0.97135	0.94308	0.95185	0.97206	0.91977	0.97696	0.95344
	SSA	0.92449	0.97052	0.9099	0.97598	0.92711	0.95986	0.97663	0.93202	0.98277	0.91772
	WOA	0.92686	0.97564	0.90677	0.9785	0.91835	0.96085	0.97861	0.92685	0.98317	0.91917

4.2 Precision/ Specificity Performance by Algorithm

Table 5 and Table 6 shows the comparison of the average precision and specifity results between the algorithms. Experimental results show that the combinations of CSA+SVM+Model-1 and ABC+SVM+Model-1 parameters provide the best results on the Drebin-215 and Malgoneme-215 datasets, respectively. The SVM ML method obtained the best average results in two datasets and two models. The same conclusion was reached for the specificity measurements in Table 5.

Table 5. Average precision of all algorithms

		Drebin-215					Malgoneme-215				
	RMA	DT	KNN	NB	RF	SVM	DT	KNN	NB	RF	SVM
Model-1	ABC	0.93375	0.97497	0.88853	0.98681	0.99819	0.96396	0.97998	0.92609	0.99009	**0.99989**
	ALO	0.93161	0.97616	0.98541	0.98541	0.9988	0.95772	0.9823	0.92249	0.98541	0.9992
	BOA	0.93799	0.97126	0.88739	0.98441	0.99622	0.95717	0.97936	0.92444	0.97143	0.99622
	CSA	0.93424	0.97526	0.89464	0.98698	**0.99883**	0.96103	0.98378	0.92576	0.98893	0.99883
	FA	0.93773	0.97067	0.89211	0.98583	0.99756	0.95697	0.9809	0.92576	0.9875	0.99979
	GWO	0.943	0.97515	0.9019	0.9846	0.99545	0.9551	0.97756	0.93058	0.983	0.99545
	HHO	0.93783	0.97329	0.88759	0.9845	0.99747	0.95802	0.97113	0.92623	0.9869	0.99875
	SCA	0.9361	0.9689	0.89895	0.98132	0.99441	0.95794	0.97189	0.9287	0.98432	0.99334
	SSA	0.93403	0.97079	0.89674	0.98462	0.99761	0.95827	0.98066	0.92256	0.98882	0.99761
	WOA	0.93586	0.9756	0.88684	0.98654	0.99778	0.95952	0.97872	0.92286	0.98828	0.99938
Model-2	ABC	0.93948	0.97454	0.88853	0.99176	0.99811	0.99139	0.98196	0.93046	0.99147	0.99965
	ALO	0.9389	0.97932	0.89125	0.98786	0.99862	0.96065	0.98243	0.92123	0.99193	0.99933
	BOA	0.91822	0.97126	0.89189	0.98508	0.99698	0.96045	0.97468	0.9153	0.98754	0.99925
	CSA	0.93875	0.97461	0.89775	0.98653	0.99772	0.96197	0.98193	0.92083	0.99143	0.9994
	FA	0.93039	0.9726	0.87717	0.9856	0.99694	0.96029	0.97977	0.91769	0.99006	**0.99966**
	GWO	0.93014	0.9747	0.89999	0.98507	0.99546	0.96009	0.97827	0.92812	0.98612	0.99723
	HHO	0.93522	0.97682	0.87581	0.98598	**0.99863**	0.9595	0.97844	0.91741	0.98805	0.99957
	SCA	0.93468	0.97167	0.89781	0.98391	0.99516	0.95445	0.97353	0.91629	0.98406	0.99292
	SSA	0.93674	0.97119	0.8953	0.98655	0.99826	0.96085	0.97775	0.92128	0.98984	0.9994
	WOA	0.93467	0.97753	0.88727	0.9881	0.99821	0.96526	0.97882	0.91453	0.992	0.99958

Table 6. Average specifity of all algorithms

		Drebin-215					Malgoneme-215				
	RMA	DT	KNN	NB	RF	SVM	DT	KNN	NB	RF	SVM
Model-1	ABC	0.96145	0.9854	0.93139	0.99243	0.9991	0.98213	0.99001	0.96202	0.99514	**0.99996**
	ALO	0.96011	0.98606	0.99161	0.99161	0.99941	0.9788	0.99124	0.96023	0.99161	0.99965
	BOA	0.9645	0.98321	0.93106	0.99105	0.99808	0.97867	0.98979	0.96145	0.99356	0.99808
	CSA	0.96179	0.98551	0.93569	0.99252	**0.99942**	0.98068	0.99198	0.96202	0.99457	0.99942
	FA	0.96424	0.98278	0.93414	0.99185	0.99877	0.97849	0.99054	0.96202	0.99382	0.99991
	GWO	0.96761	0.98553	0.94258	0.99114	0.99758	0.97774	0.98883	0.9647	0.99165	0.99758
	HHO	0.96424	0.98435	0.93094	0.99108	0.99873	0.97911	0.98568	0.96189	0.99356	0.99947
	SCA	0.96378	0.98185	0.94093	0.9893	0.99704	0.97919	0.98598	0.96435	0.99235	0.99689
	SSA	0.9619	0.98286	0.9377	0.99116	0.9988	0.97919	0.99041	0.96036	0.99452	0.9988
	WOA	0.96283	0.98574	0.92991	0.99226	0.99888	0.97994	0.9894	0.96053	0.99426	0.99974
Model-2	ABC	0.96529	0.9851	0.93139	0.99596	0.99904	0.99598	0.99103	0.96471	0.99583	0.99985
	ALO	0.96479	0.98791	0.93323	0.99299	**0.9993**	0.98041	0.9913	0.95903	0.99606	0.99971
	BOA	0.95184	0.98318	0.93427	0.99143	0.99844	0.98052	0.98801	0.9566	0.99391	0.99968
	CSA	0.96483	0.98511	0.93789	0.99225	0.99884	0.98117	0.99105	0.95954	0.99581	0.99975
	FA	0.95974	0.98394	0.92406	0.99173	0.99842	0.98015	0.99	0.95764	0.99513	**0.99986**
	GWO	0.96	0.98523	0.94148	0.99144	0.99757	0.98022	0.98919	0.96359	0.9932	0.99875
	HHO	0.96265	0.98645	0.92188	0.99193	**0.9993**	0.97989	0.98931	0.95766	0.99414	0.99982
	SCA	0.96309	0.98344	0.93995	0.9908	0.99743	0.97744	0.98688	0.95764	0.99219	0.99667
	SSA	0.96361	0.98295	0.93632	0.99228	0.99911	0.98053	0.98897	0.95986	0.99502	0.99975
	WOA	0.9622	0.98686	0.93056	0.99315	0.99909	0.98291	0.98947	0.95632	0.9961	0.99982

Düzenlenmemiş Sürüm - Uncorrected Version

4.3 Recall Performance by Algorithm

A comparison of the average recall results between the algorithms is shown in Table 7. Experimental results show that the combinations based on ABC+RF+Model-2 and ABC+KNN+Model-2 parameters provide the best results on the Drebin-215 and Malgoneme-215 datasets, respectively. The ABC metaheuristic algorithm obtained the best average results on two datasets for Model-2.

Table 7. Average recall of all algorithms

		Drebin-215					Malgoneme-215				
	RMA	DT	KNN	NB	RF	SVM	DT	KNN	NB	RF	SVM
Model-1	ABC	0.921	0.96837	0.92822	0.96319	0.84227	0.95847	**0.97954**	0.95441	0.97478	0.82734
	ALO	0.9196	0.97187	0.96531	0.96531	0.82524	0.96049	0.97795	0.94815	0.96531	0.84127
	BOA	0.90432	0.96586	0.92153	0.96207	0.86172	0.95635	0.97055	0.94471	0.97143	0.86172
	CSA	0.91868	**0.97239**	0.92714	0.96597	0.84032	0.95612	0.97504	0.94568	0.97381	0.84032
	FA	0.91195	0.96904	0.9253	0.96593	0.85186	0.95829	0.97557	0.94568	0.97575	0.82734
	GWO	0.9064	0.96673	0.89709	0.96406	0.89272	0.94756	0.97478	0.9485	0.96794	0.89272
	HHO	0.91367	0.97034	0.92746	0.96528	0.84868	0.95503	0.96825	0.95476	0.97196	0.83968
	SCA	0.89494	0.96241	0.89173	0.95645	0.89265	0.94506	0.97178	0.92989	0.96592	0.9052
	SSA	0.91349	0.96974	0.92016	0.96346	0.85001	0.95697	0.97681	0.94612	0.97037	0.85001
	WOA	0.91088	0.97108	0.93211	0.96571	0.83774	0.9522	0.97584	0.94462	0.97063	0.83069
Model-2	ABC	0.91318	0.97137	0.92822	**0.97798**	0.86474	0.9314	**0.98196**	0.94955	0.97725	0.84921
	ALO	0.92107	0.97579	0.92968	0.97155	0.8455	0.95823	0.97915	0.95599	0.97525	0.84906
	BOA	0.91052	0.96871	0.92117	0.96392	0.87421	0.94906	0.97468	0.9417	0.97049	0.85346
	CSA	0.91149	0.97327	0.92652	0.9664	0.86565	0.95628	0.97879	0.95494	0.97633	0.8461
	FA	0.91718	0.97094	0.92324	0.965	0.87964	0.96556	0.97556	0.94957	0.97677	0.85087
	GWO	0.90914	0.96987	0.89532	0.9626	0.90457	0.95176	0.97965	0.94105	0.97381	0.89913
	HHO	0.91437	0.97249	0.93049	0.96742	0.85751	0.95741	0.97524	0.94296	0.9736	0.84368
	SCA	0.89622	0.96736	0.89577	0.9591	0.89629	0.94935	0.97063	0.92381	0.96999	0.91753
	SSA	0.91295	0.97	0.92521	0.96565	0.8655	0.95899	0.97554	0.94322	0.97583	0.84848
	WOA	0.91932	0.97378	0.92754	0.9691	0.85072	0.95651	0.97843	0.93968	0.97453	0.85094

4.4 Feature Selection Performance by Algorithm

Table 8 shows the average number of features selected by the algorithms. Experimental results showed that the GWO RMA achieved the best results in two models (Model-1 and Model-2) for the Drebin-215 dataset and only in Model-1 for the Malgoneme-215 dataset. The average number of features selected for the Drebin-215 dataset is 59.6471 while it is 35.6471 for the Malgoneme-215 dataset. The reason for this result is the small number of samples in the Malgoneme-215 dataset.

Table 8. Average the number of selection of all algorithms

		Drebin-215					Malgoneme-215				
	RMA	DT	KNN	NB	RF	SVM	DT	KNN	NB	RF	SVM
Model-1	ABC	111.40	111.50	111.27	119.00	111.50	105.96	105.40	102.36	103.76	106.00
	ALO	124.86	126.15	108.78	108.78	124.86	87.53	91.00	91.50	108.78	87.90
	BOA	106.29	105.58	106.29	103.83	106.29	92.66	95.40	93.56	93.46	106.29
	CSA	109.41	109.41	109.41	110.13	109.41	98.88	95.03	104.43	96.43	109.41
	FA	113.25	107.52	109.96	112.52	108.46	106.56	106.83	104.43	106.36	105.80
	GWO	**59.64**	61.36	59.64	61.05	59.64	36.88	37.60	98.20	**35.64**	59.64
	HHO	110.23	110.23	120.18	114.54	106.86	116.10	107.60	101.30	100.20	97.70
	SCA	75.41	75.41	75.41	73.47	75.41	53.00	54.66	46.46	47.35	48.23
	SSA	111.47	111.47	109.16	108.52	111.47	104.26	104.00	103.63	104.53	111.47
	WOA	130.94	135.15	132.40	132.52	130.94	101.70	107.06	96.26	104.73	95.23
Model-2	ABC	111.06	110.40	111.27	106.75	108.71	102.46	103.46	102.80	103.10	103.37
	ALO	149.25	160.20	124.87	152.60	144.00	105.81	105.36	125.36	105.63	108.18
	BOA	108.62	97.60	106.56	105.20	106.40	100.00	98.72	89.00	97.00	94.36
	CSA	111.37	111.00	109.46	111.40	110.60	103.45	106.18	100.00	105.54	104.00
	FA	108.00	111.00	108.80	105.00	106.00	106.80	104.60	107.54	105.90.	107.09
	GWO	80.40	69.40	**59.93**	69.76	68.80	61.00	56.54	52.09	58.00	57.90
	HHO	110.2	110.23	144.25	104.75	117.50	109.53	103.93	104.43	97.23	112.53
	SCA	95.20	81.60	72.56	77.60	87.40	74.54	**50.63**	55.00	61.27	54.54
	SSA	110.20	112.70	110.56	114.60	111.60	104.26	103.72	98.90	103.18	104.09
	WOA	142.80	156.10	123.30	142.50	137.00	106.60	113.45	113.09	99.45	98. 81

Düzenlenmemiş sürüm - uncorrected version

4.5 Computational Time Performance by Algorithm

Algorithms were implemented using the MATLAB R2018a (MathWorks, Natick, MA 01760-2098, USA) on PC i7-8565U, GeForce 2GB and 16GB RAM. Table 9 summarizes the average computational times of the implemented algorithms. In this respect, it is possible to state that: computational complexity of the combinations of Drebin-215+Model-1+GWO+DT and Malgoneme-215+ Model-1+ABC+DT have the lowest computation time and are the fastest compared to the others. The average computation time for the Drebin-215 and Malgoneme-215 data sets are 0.045736 sec and 0.013936 sec, respectively. The reason for this result is the small number of samples in the Malgoneme-215 dataset.

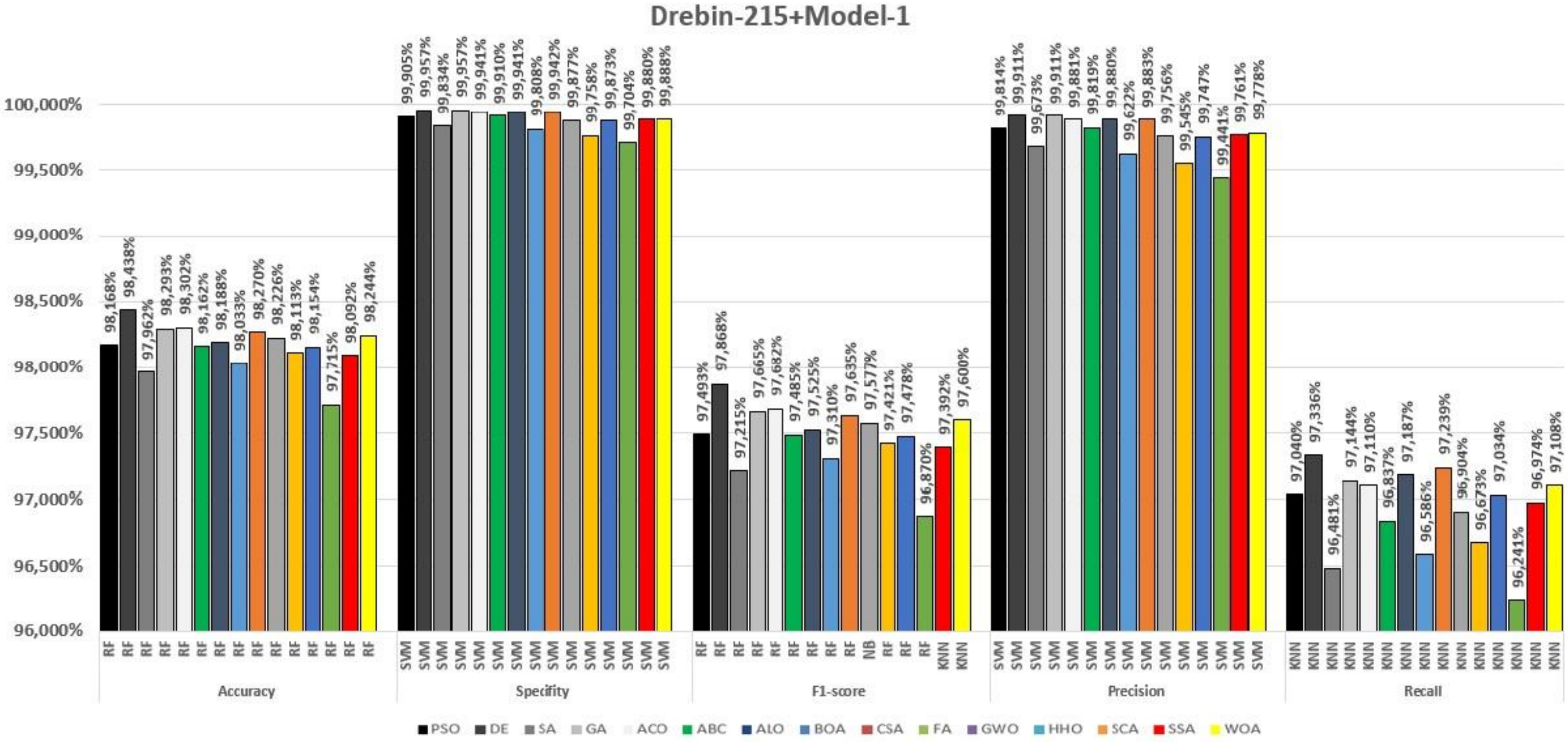
Table 9. Average the computational times (in sec) of all algorithms

		Drebin-215					Malgoneme-215				
	RMA	DT	KNN	NB	RF	SVM	DT	KNN	NB	RF	SVM
Model-1	ABC	0.117	0.116	0.378	0.624	1.110	**0.013**	0.025	0.020	0.195	0.424
	ALO	0.057	2.912	0.623	0.623	4.654	0.029	0.053	0.048	0.623	2.630
	BOA	0.061	0.816	0.131	0.581	4.364	0.029	0.051	0.047	0.157	2.364
	CSA	0.066	0.800	0.128	0.554	4.587	0.044	0.045	0.060	0.147	2.628
	FA	0.402	0.614	0.056	0.728	50.564	0.046	0.053	0.060	0.206	2.444
	GWO	**0.045**	0.408	0.107	0.425	3.534	0.033	0.023	0.063	0.110	1.846
	HHO	0.054	0.452	0.133	0.626	2.042	0.046	0.056	0.112	0.179	0.180
	SCA	0.051	0.370	0.114	0.507	3.580	0.024	0.026	0.025	0.222	1.642
	SSA	0.065	0.874	0.059	0.592	4.505	0.027	0.048	0.072	0.173	2.377
	WOA	0.069	0.800	0.063	0.718	4.858	0.027	0.062	0.032	0.168	2.479
Model-2	ABC	0.521	1.962	0.378	1.697	1.949	7.235	0.269	0.148	1.869	2.482
	ALO	0.808	5.092	**0.072**	6.680	40.564	0.127	0.233	0.127	1.672	2.514
	BOA	0.499	2.868	0.376	6.194	25.237	0.120	0.219	0.146	1.567	2.487
	CSA	0.517	3.480	0.376	6.267	26.757	0.138	**0.045**	0.150	1.557	2.624
	FA	0.489	3.516	0.498	5.794	24.440	0.132	0.224	0.157	1.563	2.721
	GWO	0.325	1.945	0.266	5.303	17.615	0.097	0.131	0.131	1.362	1.796
	HHO	0.412	2.413	0.628	6.484	34.014	0.142	0.239	0.178	1.707	2.882
	SCA	0.409	2.327	0.289	5.573	19.556	0.102	0.123	0.126	1.396	1.664
	SSA	0.513	3.432	0.365	6.201	26.218	0.123	0.218	0.151	1.563	2.523
	WOA	0.740	5.011	0.405	6.833	37.701	0.128	0.238	0.158	1.589	2.649

Figures 3-10 reveal the comparasion of best performing average results of the evaluation performance metrics for the Drebin-215 and Malgoneme-215 dataset based Model-1 and Model-2 with the CMAs and RMAs approaches.

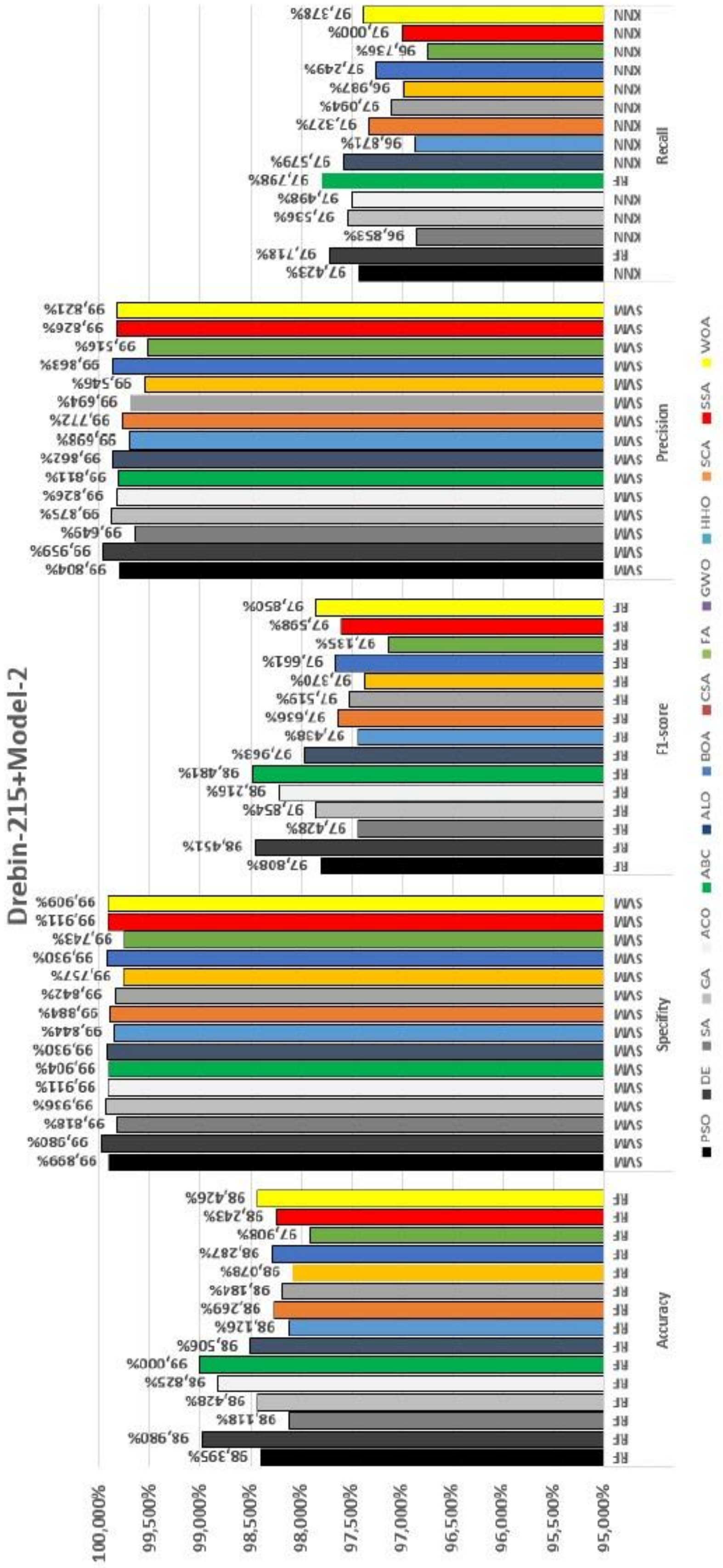
Düzenlenmemiş Sürüm - Uncorrected Version

Düzenler



Version

Figure 3. Comparasion of best performing result with the Drebin-215 + Model-1 based on CMAs [19] (black-white bars) and RMAs (colored bars).



Düzenle
Version

Figure 4. Comparasion of best performing result with the Drebin-215 + Model-2 based on CMAs [19] (black-white bars) and RMAs (colored bars).

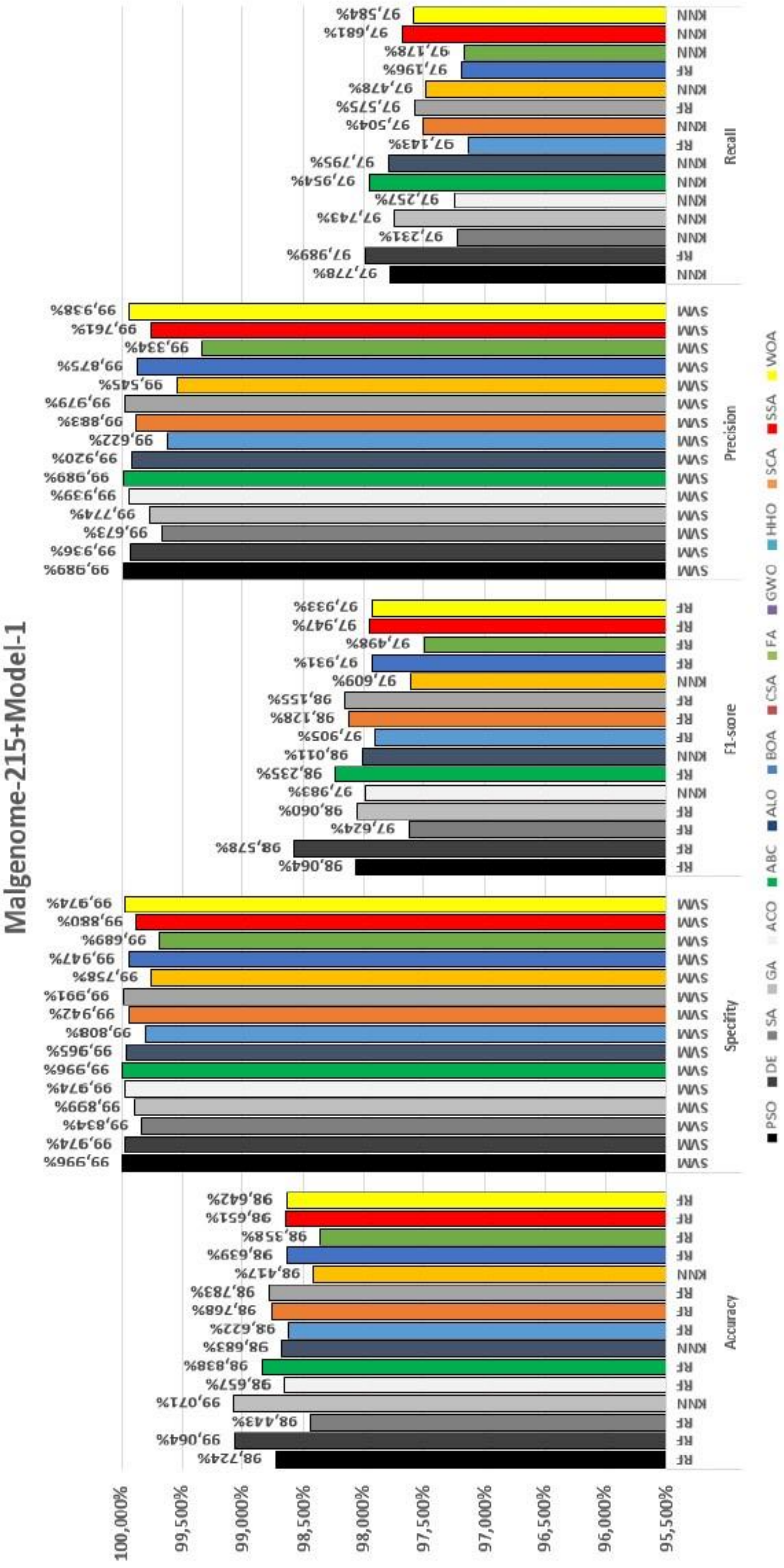
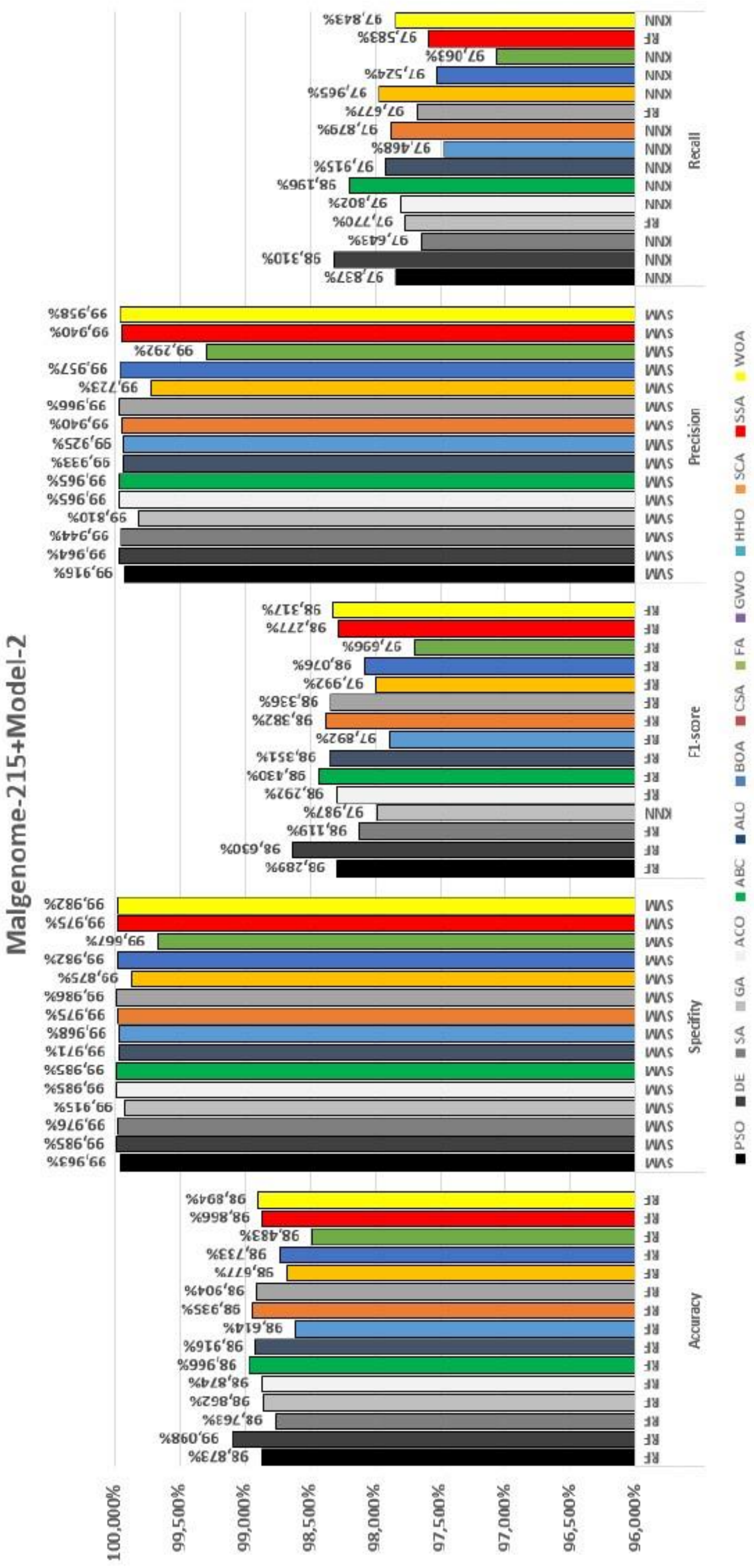
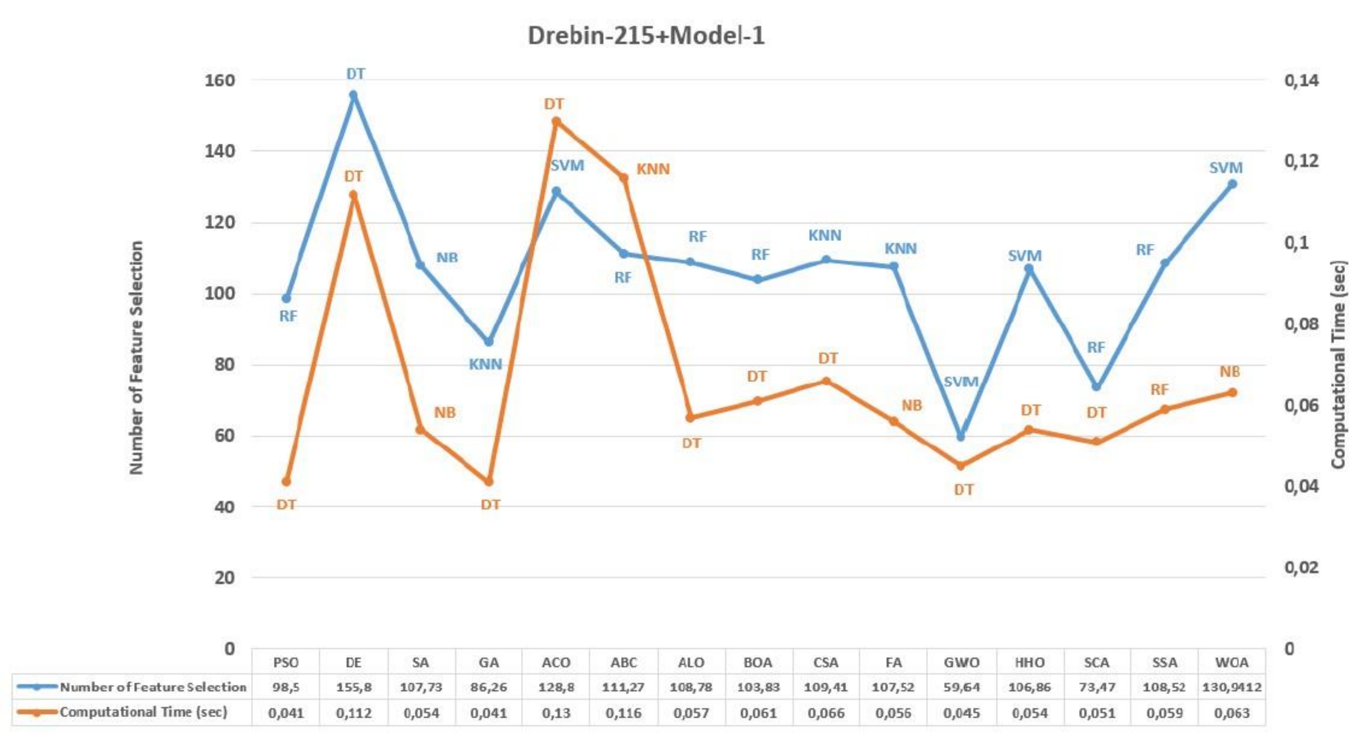


Figure 5. Comparasion of best performing result with the Malgoneme-215 + Model-1 based on CMAs [19] (black-white bars) and RMAs (colored bars).



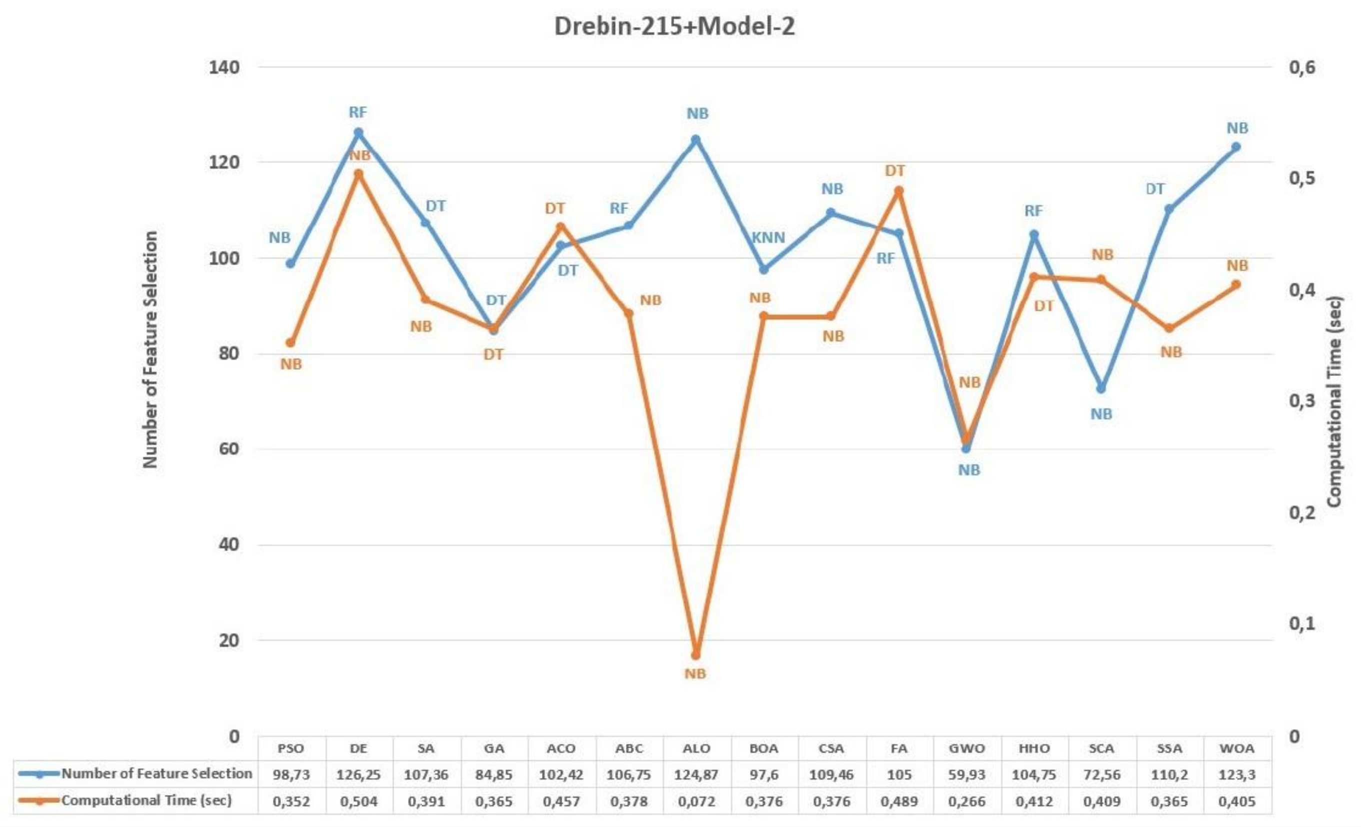
Düzenl
version

Figure 6. Comparasion of best performing result with the Malgoneme-215 + Model-2 based on CMAs [19] (black-white bars) and RMAs (colored bars).



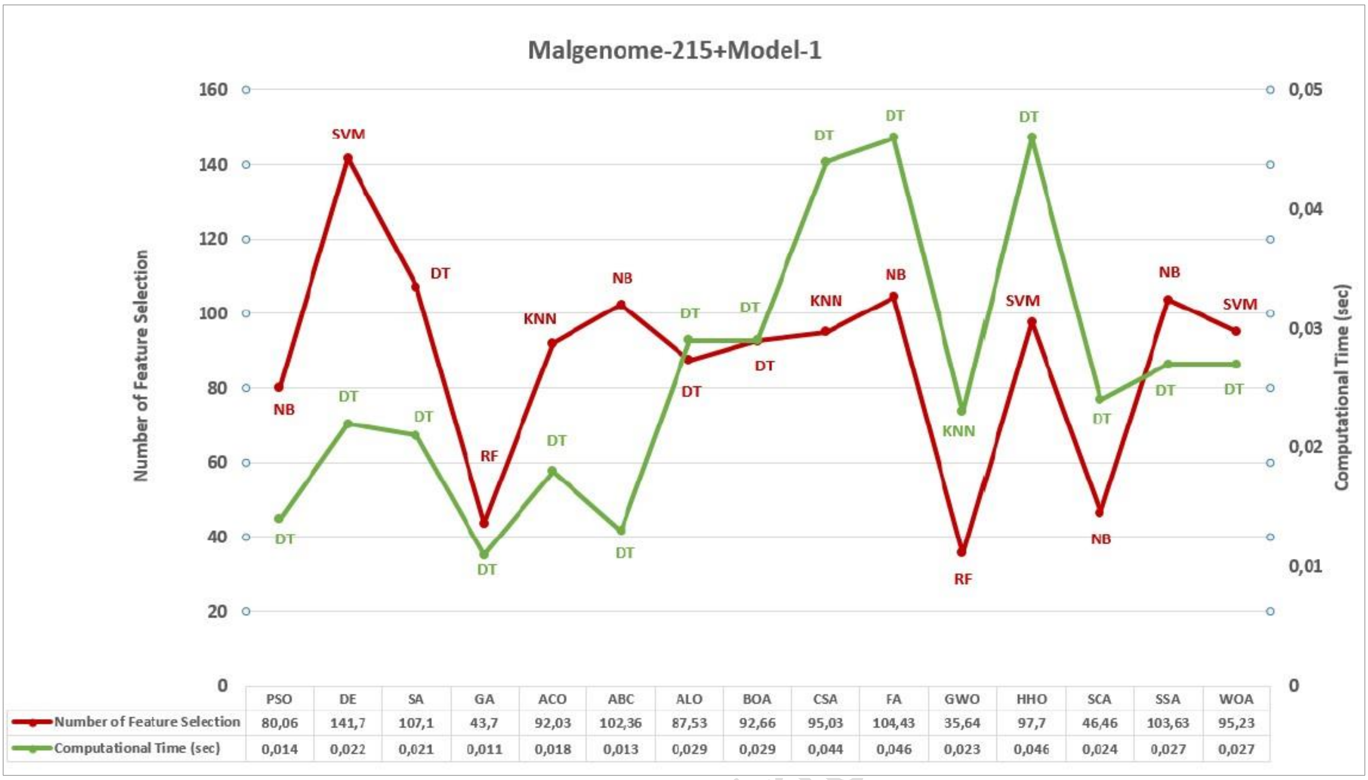
	PSO	DE	SA	GA	ACO	ABC	ALO	BOA	CSA	FA	GWO	HHO	SCA	SSA	WOA
Number of Feature Selection	98,5	155,8	107,73	86,26	128,8	111,27	108,78	103,83	109,41	107,52	59,64	106,86	73,47	108,52	130,9412
Computational Time (sec)	0,041	0,112	0,054	0,041	0,13	0,116	0,057	0,061	0,066	0,056	0,045	0,054	0,051	0,059	0,063

Figure 7. Comparasion of best performing Number of feature selection and Computational times with the Drebin-215 + Model-1 based on CMAs [19] and RMAs.



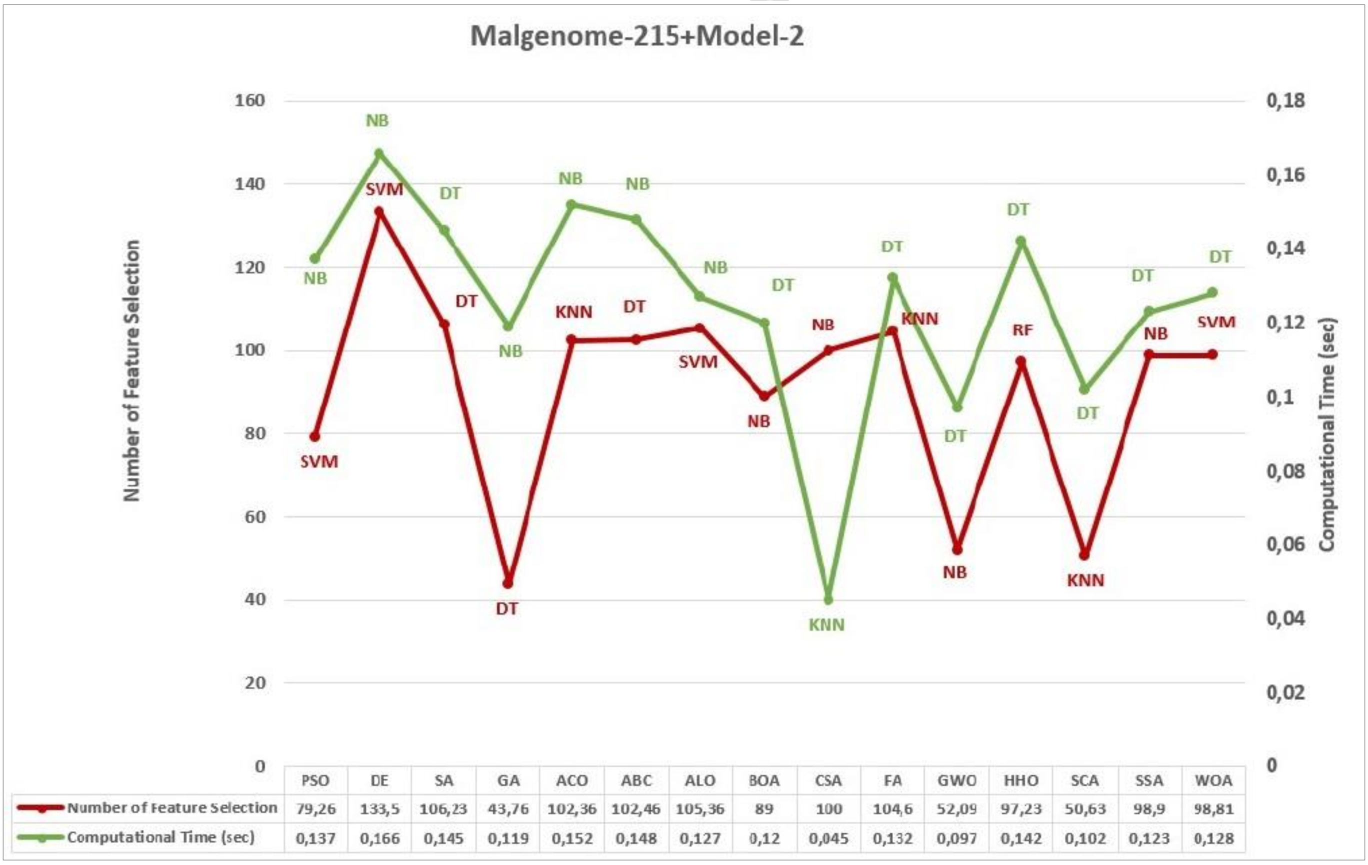
	PSO	DE	SA	GA	ACO	ABC	ALO	BOA	CSA	FA	GWO	HHO	SCA	SSA	WOA
Number of Feature Selection	98,73	126,25	107,36	84,85	102,42	106,75	124,87	97,6	109,46	105	59,93	104,75	72,56	110,2	123,3
Computational Time (sec)	0,352	0,504	0,391	0,365	0,457	0,378	0,072	0,376	0,376	0,489	0,266	0,412	0,409	0,365	0,405

Figure 8. Comparasion of best performing Number of feature selection and Computational times with the Drebin-215 + Model-2 based on CMAs [19] and RMAs.



	PSO	DE	SA	GA	ACO	ABC	ALO	BOA	CSA	FA	GWO	HHO	SCA	SSA	WOA
Number of Feature Selection	80,06	141,7	107,1	43,7	92,03	102,36	87,53	92,66	95,03	104,43	35,64	97,7	46,46	103,63	95,23
Computational Time (sec)	0,014	0,022	0,021	0,011	0,018	0,013	0,029	0,029	0,044	0,046	0,023	0,046	0,024	0,027	0,027

Figure 9. Comparasion of best performing Number of feature selection and Computational times with the Malgoneme-215 + Model-1 based on CMAs [19] and RMAs.



	PSO	DE	SA	GA	ACO	ABC	ALO	BOA	CSA	FA	GWO	HHO	SCA	SSA	WOA
Number of Feature Selection	79,26	133,5	106,23	43,76	102,36	102,46	105,36	89	100	104,6	52,09	97,23	50,63	98,9	98,81
Computational Time (sec)	0,137	0,166	0,145	0,119	0,152	0,148	0,127	0,12	0,045	0,132	0,097	0,142	0,102	0,123	0,128

Figure 10. Comparasion of best performing Number of feature selection and Computational times with the Malgoneme-215 + Model-2 based on CMAs [19] and RMAs.

Table 10. A comparison between some related work.

Literature	Year	Dataset (s)	Feature Selection Method	Classifier	# of samples	Accuracy
Our Previous Study [19]	2023	Drebin -215 Malgoneme- 215	DE	RF	15036 3799	98.98 % 99.09 %
Naic et al. [20]	2022	Drebin-215	Sailfish Optimization	RF	15036	98. 92 %
Sharma [21]	2022	Drebin -215 Malgoneme- 215	Feature-important Water Drop Algorithm	RF	15036	98.00 %
				RF / MLP	3799	99.00 %
Varma et al.[22]	2021	CICInvesAndMal2019	Bat Optimization Algorithm	KNN	1594	94.78 %
Chakravarthy [23]	2021	CICInvesAndMal2019	FA	ELM	1594	95.28 %
Lee et al.[55]	2021	Andro-AutoPsy	GA	Multilayer Perceptron	7500	98.1 %
This Paper	2024	Drebin -215 Malgoneme- 215	ABC	RF	15036 3799	99.00 % 98.96%

The proposed approach was compared with those of recent studies.

5 Conclusions

Today, Android applications are widely used individually and corporately. In this context, since the security of Android systems covers a wide audience, the development of strong and effective security measures is extremely important. In particular, malware detection is one of the most effective techniques to provide protection for Android users. In this context, in this study, ten metaheuristic algorithms used for FS (ABC, ALO, BOA, CSA, FA, GWO, HHO, SCA, SSA, and WOA); two datasets (Drebin-215 and Malgoneme-215); two validation options (Model-1 and Model-2) and five ML (DT, KNN, NB, RF, and SVM) methods are selected as in various scenarios to perform an effective Android malware analysis. First, each metaheuristic algorithm is transferred to the wrapper-based FS method, generating the optimal feature subset from the full feature set. Then, this optimal feature subset was analyzed with different models and ML methods, and as a result of the comparisons made for the relevant data set, the combination that gave the best technique was determined. Thus, at the end of this study, the effect of the techniques used in the FS method based on a series of meta-heuristic algorithms, which are most commonly used, on the success of detecting Android malware was observed. The results show the effectiveness and efficiency of using the most prominent metaheuristic algorithm in the detection of Android malware. The results obtained show that RF and Model-2 achieve the highest accuracy. With this study, it is aimed to contribute to the development of effective defence systems against Android malware, whose area of influence is constantly increasing. In future studies, the hybrid architecture of the metaheuristic algorithms showing the highest accuracy value for Drebin-215 and Malgoneme-215 will be designed and the performance results of its use with different deep learning (DL) algorithms will be investigated.

6 Author contributions

Author 1 formation of the idea, design, data analysis, and literature review; Author 2 contributed to the evaluation and analysis of the results obtained.

7 Approval from the ethics committee and statement of conflict of interest

There is no need to obtain ethics committee permission for the article prepared. There is no conflict of interest with any person/institution in the prepared article.

8 References

[1] Tahtaci B, Canbay B. "Android malware detection using machine learning*". In 2020 Innovations in Intelligent Systems and Applications Conference (ASYU),* Istanbul, Turkey, 15-17 October 2020.

[2] Kalash M, Rochan M, Mohammed N, Bruce ND, Wang Y, Iqbal F. "Malware classification with deep convolutional neural networks". *In 2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*, Paris, France, 26-28 February 2018.

[3] Masum M, Shahriar H. "Droid-NNet: Deep learning neural network for android malware detection". *In 2019 IEEE International Conference on Big Data(Big Data)*, Los Angeles, CA, USA, 9-12 December 2019.

[4] Lee J, Jang H, Ha S, Yoon Y. "Android malware detection using machine learning with feature selection based on the genetic algorithm". *Mathematics*, 9(21), 2813, 117334-117352, 2021.

[5] Wang L, Gao Y, Gao S, Yong X. "A new feature selection method based on a self-variant genetic algorithm applied to android malware detection". *Symmetry*, 13(7), 1290, 2021.

[6] Ay Ş, Ekinci E, Garip Z. "A comparative analysis of meta-heuristic optimization algorithms for feature selection on ML-based classification of heart-related diseases". *The Journal of Supercomputing*, 1-30, 2023.

[7] Şahin CB, Diri B. "Robust feature selection with LSTM recurrent neural networks for artificial immune recognition system". *IEEE Access*, (7), 24165-24178, 2019,

[8] Şahin CB. "Learning optimized patterns of software vulnerabilities with the clock-work memory mechanism". *Avrupa Bilim ve Teknoloji Dergisi*, (41), 156-165, 2022.

[9] Goldberg DE, Holland, JH. "Machine Learning". *Machine Learning*, 3(23), 95-99, 1988.

[10] Van Laarhoven, PJ, Aarts EH. "Simulated Annealing". *Springer Netherlands*, 7-15, 1987.

Uncorrected Version

[11] Dorigo M, Birattari M, Stutzle T. “Ant colony optimization”. *IEEE Computational Intelligence Magazine*, 1 (4), 28-39, 2006.

[12] Storn R, Price K. “Differential Evolution–a simple and efficient heuristic for global optimization over continuous spaces”. *Journal of Global Optimization*, (11), 341-359, 1997.

[13] Kennedy J, Eberhart R. “Particle Swarm Optimization”. *In Proceedings of ICNN'95- International Conference on Neural Networks*, IEEE, Perth, WA, Australia 1942-1948, 27-01 December 1995.

[14] Karaboga D. “An idea based on honey bee swarm for numerical optimization”. *Technical report-tr06, Erciyes university, engineering faculty, computer engineering department*, (200), 1-10, 2005.

[15] Yang XS. “Firefly algorithm, stochastic test functions and design optimisation”. International journal of bio-inspired computation, 2(2), 78-84, 2010.

[16] Mirjalili S, Mirjalili SM, Lewis A. “Grey wolf optimizer”. *Advances in engineering software*, 69, 46-61, 2014.

[17] Mirjalili S, Gandomi A H, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. “Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems”. *Advances in engineering software*, 114, 163-191, 2017.

[18] Akalın F, Yumusak N. “Classification of acute leukaemias with a hybrid use of feature selection algorithms and deep learning-based architectures”. *Pamukkale University Journal of Engineering Sciences*, 29(3), 256-263, 2022.

[19] Beştaş MŞ, Dinler ÖB. “Detection of Android Based Applications with Traditional Metaheuristic Algorithms”. *International Journal of Pure and Applied Sciences*, 9(2), 381-392, 2023.

[20] Naick S, Bethapudi P, Reddy SPR. "Malware detection in Android mobile devices by applying Swarm Intelligence Optimization and machine learning for API Calls", *Int J Intell Syst Appl Eng,* 10 (3), 67-74, 2022.

[21] Sharma RM. “AMD-FIWDA: Android malware detection using feature importance Water Drop Algorithm”. NeuroQuantology, 20(15), 5005-5018 2022.

[22] Varma PRK, Mallidi SKR, Jhansi SJ, Dinne PL. “Bat optimization algorithm for wrapper-based feature selection and performance improvement of android malware detection”. *IET Netw*, 10, 131–140, 2021.

[23] Chakravarthy SJ. “Wrapper-based metaheuristic optimization algorithms for android malware detection: a correlative analysis of firefly, bat & whale optimization”. *J Hunan Univ*, 48 (10), 2021.

[24] Bhagwat S, Gupta GP. “Android malware detection using hybrid meta-heuristic feature selection and ensemble learning techniques”. In International Conference on Advances in Computing and Data Sciences, 6th International Conference, ICACDS 2022, Kurnool, India, 145–156, April 2022.

[25] Elkabbash ET, Mostafa RR, Barakat SI. “Android malware classification based on random vector functional link and artificial Jellyfish Search optimizer”. *PLoS ONE, 16 (11),* 1-22, 2021.

[26] Alzubi OA, Alzubi JA, Al-Zoubi AM, Hassonah MA, Kose U. “An efficient malware detection approach with feature weighting based on Harris Hawks optimization”. *Cluster Computing*, 25, 2369-2387, 2022.

[27] Sulaimon SA, Adebayo OS, Bashir SA, Ismaila I. “Android Malware Classification using Whale Optimization Algorithm*”. i-manager’s Journal on Mobile Applications and Technologies*, 5.2: 37, 1-16, 2018.

[28] Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K, Siemens CERT. “Drebin: Effective and explainable detection of android malware in your pocket”. *In Ndss*, (14), 23-26, 2014.

[29] HCRL. Hacking and Countermeasure Research Lab. https://ocslab.hksecurity.net/andro-autopsy (11.05. 2021)

[30] Zhou Y, Jiang X. “Dissecting android malware: characterization and evolution”. *In 2012 IEEE symposium on security and privacy, IEEE*, 95- 109, 2012. https://ocslab.hksecurity.net/andro-autopsy (11.05. 2021)

[31] Deci. “Lesson 1.4 Data Shuffling”. https://deci.ai/course/data-shuffling/(11.05.2021).

[32] Yildiz O, Doğru IA. “Permission-based android malware detection system using feature selection with genetic algorithm”. *International Journal of Software Engineering and Knowledge Engineering*, 29(02), 245-262, 2019.

[33] Dokeroglu T, Deniz A, Kiziloz HE. “A comprehensive survey on recent metaheuristics for feature selection”. *Neurocomputing*, 494, 269-296, 2022.

[34] Gao W, Liu S, Huang L. “A global best artificial bee colony algorithm for global optimization”. Journal of Computational and Applied Mathematics, 236, 2741-2753, 2011.

[35] Singh NSP, Nair NK. “Artificial bee colony algorithm for inverter complex wave reduction under line-load variations”. Transactions of the Institute of Measurement and Control,40(5), 1593-1607, 2018.

[36] Jin Y, Sun Y, Ma H. “A developed artificial bee colony algorithm based on cloud model“. ***Mathematics***, 6(4):61, 1-18, 2018.

[37] Mirjalili S. “The ant lion optimizer”. Advances in engineering software, 83, 80-98, 2015.

[38] Zhou H, Cheng HY, Wei ZL, Zhao X, Tang AD, Xie L. “A hybrid butterfly optimization algorithm for numerical optimization problems”. Computational Intelligence and Neuroscience, 2021,1– 4, 2021.

[39] Arora S, Singh S. “Butterfly optimization algorithm: a novel approach for global optimization”. *Soft Computing*, 23, 715-734, 2019.

[40] Çerçevik AE, Avşar Ö. “Doğrusal sismik izolasyon parametrelerinin karga arama algoritması ile optimizasyonu”. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi. 26(3), 440-447, 2020.

[41] Şahin CB. “Optimization of Software Vulnerabilities patterns with the Meta-Heuristic Algorithms”. *Türk Doğa ve Fen Dergisi*, 11(4), 117-125, 2022.

[42] Bairathi D, Gopalani D. “A novel swarm intelligence based optimization method: Harris’ hawk optimization”. In Intelligent Systems Design and Applications: 18th International Conference on Intelligent Systems Design and Applications (ISDA 2018) held in Vellore, India, December 6-8, 2018, Volume 2 (pp. 832-842). Springer International Publishing, 2020.

[43] Mirjalili S. “SCA: a sine cosine algorithm for solving optimization problems”. *Knowledge-based systems*, 96, 120-133, 2016.

[44] Abualigah, L., Diabat, A. "Advances in Sine Cosine Algorithm: A comprehensive survey". *Artif Intell Rev* **54**, 2567–2608, 2021.

[45] Ibrahim, R.A., Ewees, A.A., Oliva, D. *et al.* "Improved salp swarm algorithm based on particle swarm optimization for feature selection". *J Ambient Intell Human Comput* **10**, 3155–3169, 2019.

[46] Mirjalili S, Lewis A. "The whale optimization algorithm". *Advances in engineering software*, 95, 51-67, 2016.

[47] Nadimi-Shahraki, M., Zamani, H., Asghari Varzaneh, Z. *et al.* "A systematic review of the whale optimization algorithm: theoretical foundation, improvements, and hybridizations". *Arch Computat Methods Eng* **30**, 4113–4159, 2023.

[48] Alızada, B. "Improved whale optimization algorithm based on π number". International Scientific and Vocational Studies Journal, 4(1), 21-30, 2020.

[49] Cihan P, Kalıpsız O, Gökçe E. "Computeraided diagnosis in neonatal lambs". *Pamukkale Üniversitesi Mühendislik Dergisi*, 26 (2), 385-391, 2020.

[50] Ullah A, Şahin CB, Dinler Ö.B, Khan MH, Aznaoui H. " Heart disease prediction using various machines learning approach". *Journal of Cardiovaskular Disease Research*, 3(12), 379-391, 2021.

[51] Koşan MA, Yıldız O, Karacan H. "Kimlik avı web sitelerinin tespitinde makine öğrenmesi algoritmalarının karşılaştırmalı analizi". *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*. 24(2), 276-282,2018.

[52] Kalaycı TE. "Kimlik hırsızı web sitelerinin sınıflandırılması için makine öğrenmesi yöntemlerinin karşılaştırılması". *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*. 24(5), 870-878, 2018.

[53] Khan SN, Khan SU, Aznaoui H, Şahin, CB, Dinler, Ö.B. "Generalization of linear and nonlinear support vector machine in multiple fields: a review". *Computer Science and Information Technologies*, 3(4), 226-239, 2023.

[54] Cihan, P. " The machine learning approach for predicting the number of intensive car, intubated patients and death: The COVID-19 pandemic in Turkey". Sigma Journal of Engineering and Natural Sciences, (40) 1, 85-94, 2021.

[55] Lee, J., Jang, H., Ha, S. and Yoon, Y. "Android malware detection using machine learning with feature selection based on the Genetic algorithm". Mathematics, (9), 2813, 1-20, 2021.

Düzenlenmemis Sürüm - Uncorrected