Comparative Study: Mouth Brooding Fish (MBF) as a Novel Approach for Android Malware Detection

Kangle Zhou[1]*, Panpan Wang[2], Baiqing He[3]
Nanchang Institute of Technology School of Computer Information Engineering, Nanchang 330044, China[1, 2]
University of Campania Luigi Vanvitelli, Italy, Naples[3]

***Abstract*—Android Malware Detection has become increasingly prevalent, with the highest market share among all other mobile operating systems due to its open-source nature and user-friendliness. This has resulted in an uncontrolled proliferation of malicious applications targeting the Android platform. Emerging trends of Android malware are employing highly sophisticated detection and analysis evasion techniques, rendering traditional signature-based detection methods less effective in identifying modern and unknown malware. Alternative approaches, such as Machine Learning methods, have emerged as leading solutions for timely zero-day anomaly detection. Ensemble learning, a common meta-approach in machine learning, seeks to improve predictive performance by amalgamating predictions from multiple models. This paper introduces an enhanced strategy, Mouth Brooding Fish (MBF), based on ensemble learning for Android Malware Detection (AMD). The findings are further compared with the outputs of various algorithms including Support Vector Machine (SVM), AdaBoost, Multilayer Perceptron (MLP), Gaussian Kernel (GK), and Random Forest (RF). Compared to the other selected models, MBF exhibits remarkable performance with an F-score of 98.57%, precision of 99.65%, sensitivity of 97.51%, and specificity of 97.51%. Thus, the significant novelty of this work lies in the accuracy and authenticity of the selected algorithms, demonstrating their superior performance overall.**

***Keywords*—Android malware detection; ensemble learning; SVM; MLP; RF**

I. Introduction

Due to the nearness of innovation in all areas of our everyday lives, cyber security has become one of the biggest concerns to be attended to by society. In a long time, there has been a considerable number of assaults and, what is indeed more exceptional, to a wide assortment of destinations. A few later well-known cases incorporate refusal of benefit assaults such as that performed by the Mirai botnet [1] and an enormous information seizure driven by the ransomware Wannacry [2]. However, the widespread use of mobile phones has turned out to be a significant contributing factor to a sharp increase in malware attacks. Because these malicious programs are hidden within legitimate programs, it is difficult to identify and categorize them. Because they use a signature-based methodology, the current processes are unable to differentiate between hidden malware [3].

The most widely used operating system (OS) is Android, which is also continuing to increase its market share. Android is an open-source platform that allows users to download apps from the Google Play Store and third-party developers. Because of its popularity and openness, Android has drawn the attackers' attention. According to McAfee's security reports, 49 million new malware and 121 million existing malwares were discovered in 2020 [4]. Any malicious code that compromises a user's privacy, accessibility, or keenness is referred to as malware. The malicious programs seem to be real, but they carry out harmful operations behind the scenes. Malware uses a variety of techniques, such as tracking the client's region and jumbling individual data. Malicious programs (apps) try to infiltrate Android devices in order to steal personal data, place phone calls, send SMS, and do other activities. According to MacAfee's estimate, there will be 49 million and 121 million new instances of contemporary malware and cumulative malware by 2020, respectively [5]. The escalating pace of Android malware's advancement poses a significant threat to users of the Android operating system. Clients are required to determine the malicious nature of an application due to the need for data acquisition and comprehension. When acquiring an application from the Android application store, a significant number of Android users tend to overlook or neglect the examination of the terms and conditions. Regrettably, perpetrators exploit this reality and specifically target portable electronic devices [6].

Due to the increment in Android malware, physically handling malevolent tests has become troublesome. To overcome this restriction, it is vital to construct a proficient strategy for better distinguishing hazards of applications. A prior signature-based approach was utilized to distinguish proof of malware. This approach is based on coordinating the app's signature within the database. This strategy's confinement is that it cannot identify obscure malware [7]. On a customary premise, malware designers make modern malware to undermine the framework's security and its clients' protection. The chance posed by malware requires the improvement of successful strategies. This assessment helps with the arrangement of early notices concerning a particular Android app, permitting quick consideration to be paid to it in terms of apportioning assets [8].

The Android operating system has the dominant position in market share primarily as a result of its seamless functionality and an extensive array of features, which serve to captivate and entice cyber criminals [9]. Traditional Android malware detection methods, such as signature-based or battery consumption monitoring, may fail to detect recent malware. Therefore, we present a novel method for detecting malware in Android applications using MBF. The outcomes of the proposed method are compared with several algorithms, including SVM, Adaboost, MLP, GK, and RF. The following outlines the main

gaps and shortcomings of the related works in the second section. The third section illustrates the selected algorithms for the considered problem and specifies the evaluation criteria calculated for comparison. The used dataset is also explained in the fourth section. The results are discussed in the fifth section to specify the superiority of MBF over the other algorithms. The findings and suggestions for future work are presented in the sixth section.

II. Literature Review

As seen from the literature, many advances have been made regarding AMD. Grace et al. [10]proposed RiskRanker, which is an automated method designed to assess the level of risk associated with a given application. The experiments were conducted by aggregating a total of 118,318 applications sourced from various Android stores. The findings indicate that RiskRanker showed effectiveness and flexibility in regulating Android marketplaces. Idress et al. [11] introduced PIndroid, a system designed to locate and analyze malware. This system focuses on gathering learning tactics to enhance its effectiveness. This study focuses on a methodology for detecting malware that integrates the intersection and union set operations with data aggregation techniques. The aforementioned methodology was implemented on a sample size of 445 untainted and 1300 contaminated Android applications that were obtained from both third-party and official channels. The researchers reached the conclusion that the suggested approach has the potential to be used for the categorization of Android applications. In Sharma et al.'s [11] study, the malicious capabilities were categorized by analyzing notable features identified during both passive and active malware assessments, as well as the malware nomenclature used by antivirus vendors. The authors presented a methodology for addressing discrepancies in malware analysis by using fuzzy logic to evaluate the many functionalities of malicious software. In agreement with the planned FIS, it was determined that 83% of malware testing was discovered to belong to the same cluster for malware-recognizable proof. Mariconti et al. [12] presented the MAMADROID framework, which depends on static malware analysis and was successfully deployed. Malware detection employs static characteristics, like API calls and call graphs. The study included the evaluation of a dataset consisting of 3.5 million harmful applications and 8.5 million benign applications. The strategy that was suggested resulted in an F-measure of 0.99. Jang et al. [6] demonstrated Andro-Autospy, an antimalware mechanism that protects mobile devices. The findings suggested that the proposed system can detect and classify malware. In the work of Sharma et al.[13], the RNPDroid approach was offered as a means of doing risk assessments by leveraging permissions. The suggested methodology is assessed using the M0Droid dataset, including 400 Android samples with 165 characteristics. The T-test and ANOVA were employed for statistical analysis. The findings indicate that, with a significance level of 5%, the computed F value of 517.3 exceeds the critical F value of 2.61. Gandotra et al. [14]presented a novel approach using fuzzy logic to automate the calculation of the damage potential of malware programs. This technique relies on extracting characteristics via automated analysis.

Moreover, Zhu et al. [15] used SVM as the fusion classifier to learn the implicit supplementary information from the output of the ensemble members and yield the final prediction result. The creators appeared that exploratory comes about on two partitioned datasets collected by inactive investigation to demonstrate the viability of the SEDMDroid. The primary ones extricate consent, touchy API, checking framework occasion, and so on that are broadly utilized in Android malware as highlights. Sedmdroid accomplishes 89.07% precision in terms of these multi-level inactive highlights. The moment one, an open enormous dataset, extricates the touchy information stream data as the highlights, and the normal exactness was 94.92%. The promising try reveals that the proposed strategy was a successful way to recognize Android malware. Bhat et al. [16] proposed a precise dynamic analysis approach to identify several malicious attacks. The proposed strategy centered on behavioral examination of malware that requires remaking the behavior of Android malware. The energetic behavior highlights incorporate framework calls, covers, and complex Android objects (composite behavior). The strategy was utilized to evacuate unessential highlights for effective malware location and classification. For classification, the homogeneous and heterogeneous outfit machine learning calculations were utilized.

The stacking approach had the most excellent classification, with a precision rate of 98.08%. The thorough test of the viability and predominance of the show. In another paper [17], a total of seven feature selection methods were used in order to choose permissions, API calls, and opcodes. Subsequently, the outcomes of each feature selection process were combined to provide a novel feature set. Following this, the authors used this technique to educate the foundational learner. The researchers used logistic regression as a meta-classifier in order to extract implicit information from the output of the base learners and generate the final classification outcomes. Following the examination, the F1-score of MFDroid achieved a value of 96.0%. Ultimately, an examination was conducted on each sort of feature in order to ascertain the distinctions between dangerous and benign applications. Atacak [18] proposed the use of a fuzzy logic-based dynamic ensemble (FL-BDE) model for the purpose of detecting malware that is targeted towards the Android operating system. The findings indicated that the FL-BDE model had outstanding results compared to the ML-based models. It achieved an accuracy of 0.9933, a recall of 1.00, a specificity of 0.9867, a precision of 0.9868, and an F-measure of 0.9934.

Due to the outcomes of the previous works, it is interesting to compare Android malware detection techniques with MBF. Even though malware detection algorithms and MBF function in entirely separate fields, comparing the two might be a thought-provoking exercise. To demonstrate the possible benefits of MBF over conventional algorithms in the context of Android malware detection, the following comparison study is provided in the current work:

Adaptability and Learning: Fish that raise their young in their jaws demonstrate adaptable parental care. Likewise, MBF may represent a method that, instead of algorithms, learns from and adjusts to novel dangers more naturally. By monitoring and responding to abnormalities similar to a live creature, they could

"protect" the system and continuously adjust to new dangers without explicit programming.

Resilience to Unknown Threats: Fish that rear their young by mouth can keep their young safe from predators. Similarly, utilizing innate reflexes or pattern recognition unconstrained by preset rules or signatures, MBF may represent a theoretical system naturally resistant to dangers posed by zero-day or previously undisclosed malware.

Complexity and Interpretation: Providing MBFs with care necessitates a sophisticated comprehension of the dangers surrounding them. On the other hand, predetermined signatures or behavior patterns are frequently the basis of Android malware detection algorithms, which may miss more nuanced or sophisticated threats. A method that transcends algorithms' interpretive capabilities might be represented by MBF, which is capable of interpreting contextual signals and subtleties.

Resource Efficiency: Fish raising their young by mouthbrooding expend significant energy and resources. Comparatively speaking, MBF may represent a method that maximizes the use of computing resources for malware detection, or it may draw attention to high-risk regions of an Android system.

The adaptation and evolutionary advantage of mouth-brooding fish have developed throughout time to improve their chances of surviving and procreating. By comparison, MBF may stand for continuous evolution in malware detection, in which the system improves with time through experience-based learning and grows increasingly capable of fending off new threats. Even though this analogy is purely theoretical and symbolic, it's crucial to remember that a realistic, ethical, and computationally constrained implementation of MBF in Android malware detection would need thorough investigation and technological viability. However, taking cues from the workings of nature might occasionally result in novel concepts in cybersecurity and technology.

III. Methodology

The detection of Android malware has significant importance due to many factors. The safeguarding of personal data is a critical concern since malware often targets the theft of sensitive information, including personal particulars, financial records, and login passwords. The detection and prevention of malware on Android devices are crucial in order to protect sensitive information from potential intrusion.

The prevention of financial loss is a critical concern in the realm of cybersecurity. Certain types of malicious software, such as ransomware or banking trojans, possess the capability to target individuals' financial accounts specifically. This targeted approach may result in unlawful transactions or the coercion of monetary funds from unsuspecting users. Detection plays a crucial role in mitigating financial losses resulting from these illicit acts. The preservation of device performance is a crucial concern since malware has the potential to substantially diminish it via resource consumption, resulting in delays and the presentation of invasive advertisements. The identification and eradication of malicious software contribute to the preservation of the device's optimum functionality. Malicious software often capitalizes on weaknesses within the Android operating system or its apps in order to get illegal entry. The process of detection plays a crucial role in identifying and addressing these vulnerabilities, hence reducing the risk of possible exploitation. Besides, the preservation of user privacy is a critical concern in the realm of cybersecurity. It has been observed that some types of malicious software have the capability to seize control of cameras and microphones, as well as monitor user actions without obtaining proper approval. This unauthorized intrusion into personal devices and activities poses a significant threat to the privacy of users. The act of detection plays a pivotal role in preventing and obstructing instances of privacy infringements. Given the vast number of accessible applications, it is possible that some apps may include harmful code or exhibit undesirable behavior. The detection of malware plays a crucial role in enabling users to download and use apps securely, hence mitigating the risk of compromising their devices or data. The presence of malware may serve as a potential entry point for malicious actors seeking unauthorized access to computer networks. The identification and eradication of malware on Android devices contribute to the preservation of network security, particularly in scenarios where compromised devices serve as gateways for more extensive cyber assaults. Efficient and reliable techniques for detecting malware, such as antivirus software and security upgrades, play a crucial role in mitigating these threats and ensuring a safer and more secure Android environment for consumers. In this section, SVM, Adaboost, MLP, GK, RF, and MBF are illustrated for Android malware detection.

A. Selected Algorithms

1) Support vector machine (SVM): Support Vector Machines (SVM) is a powerful supervised learning algorithm that exhibits optimal performance when applied to datasets of smaller sizes. However, its effectiveness diminishes when confronted with complicated datasets. The Support Vector Machine (SVM), sometimes referred to as SVM, is a versatile algorithm that may be used for both regression and classification tasks. However, it is generally more effective in addressing classification problems. Support Vector Machines (SVM) is a supervised machine learning algorithm often used for both classification and regression tasks. Although the term "relapse issues" is often used, it is most appropriate for the purpose of categorization. The primary goal of the Support Vector Machine (SVM) technique is to identify the optimal hyperplane in an N-dimensional space that can effectively separate the data points into distinct classes within the given space. The hyperplane postulates that the boundary separating the nearest centroids of different classes should be maximized. The determination of the hyperplane's measurement is dependent upon the quantity of highlights. When the number of input features is two, the hyperplane may be described as a straight line that fairly separates the data points. When the number of input highlights reaches three, the hyperplane transforms into a two-dimensional plane. It gets difficult to make assumptions when the number of highlights exceeds three. There exists a multitude of potential hyperplanes that may be selected to separate the two groups of data points effectively. Our objective is to identify a plane that exhibits

optimal discrimination, namely, the greatest separation between data points belonging to different classes. The act of maximizing the elimination of edges provides a modest level of support, hence enhancing the accuracy of classifying future information.

2) Adaboost: There are numerous machine learning calculations to select from for your issue explanations. One of these calculations for prescient modeling is called AdaBoost. The AdaBoost calculation, brief for Versatile Boosting, may be a Boosting strategy utilized as a Gathering Strategy in Machine Learning. It is called Versatile Boosting, as the weights are re-assigned to each occasion, with higher weights allowed to classify occurrences inaccurately. What this calculation does is that it builds a show and gives rise to weights to all the information focuses. At that point, it allocates higher weights to wrongly classified focuses. All the higher-weight focuses are given more significance within the other demonstration. It'll keep training models until and unless a lower mistake is made. The foremost suited and thus most common calculation utilized with AdaBoost is choice trees with one level. Because these trees are so brief and, as it were, contain one decision for classification, they are often called choice stumps. An AdaBoost classifier may be a meta-estimator that starts by fitting a classifier on the initial dataset and, after that, fits extra duplicates of the classifier on the same dataset but where the weights of erroneously classified occasions are balanced such that consequent classifiers center more on troublesome cases. AdaBoost limits misfortune work related to any classification mistake and is best utilized with powerless learners. The strategy was primarily planned for twofold classification issues and can be used to boost the execution of choice trees. Slope Boosting is utilized to unravel the differentiable misfortune work issue.

3) Multilayer perceptron (MLP): The multilayer perceptron (MLP) has the potential to enhance and strengthen the forward neural architecture. The system is composed of three distinct levels, namely the input layer, yield layer, and covered-up layer, as seen in Fig. 1. The input layer is responsible for receiving the input flag that needs to be processed. The yield layer is responsible for executing the designated task, such as prediction and categorization. The presence of several hidden layers in a multilayer perceptron (MLP) serves as a crucial computational mechanism, allowing for the transformation of input data into output predictions. Similar to a feedforward architecture in a multilayer perceptron (MLP), the flow of information in the forward direction occurs from the input layer to the output layer. The neurons of the Multilayer Perceptron (MLP) are trained using the backpropagation learning algorithm. Multilayer perceptrons (MLPs) are designed to handle continuous tasks effectively and have the ability to address problems that are not easily separable. The primary applications of multilayer Perceptron (MLP) are design categorization, pattern recognition, prediction, and estimate.

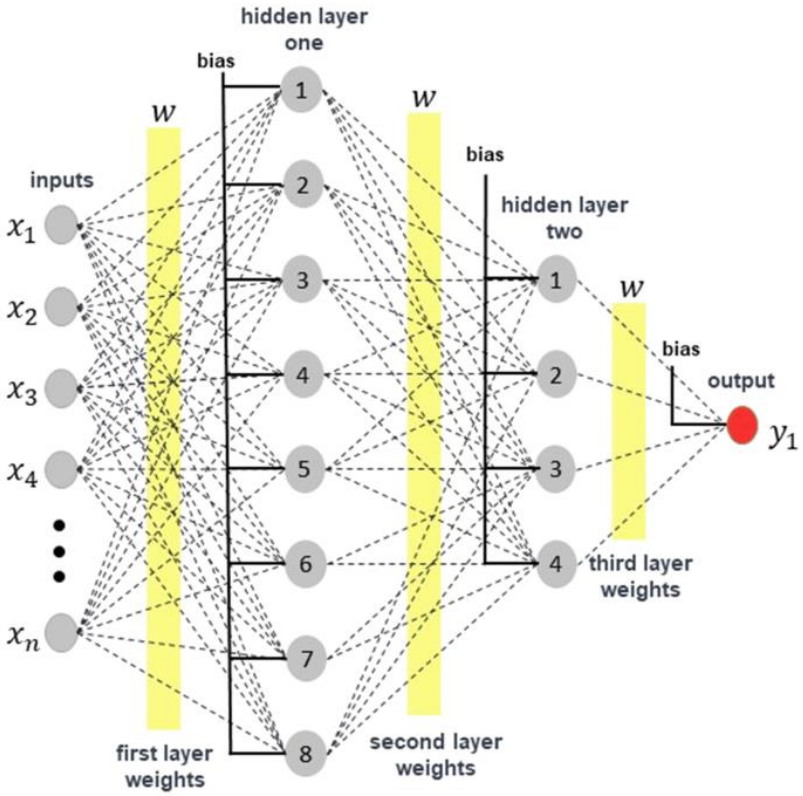


Fig. 1. System modeling utilizing an MLP neural network

4) Gaussian kernel (GK): The GK is defined as follows in one-dimensional, two-dimensional, and neuronal dimensions:

$$G_{1\ D}(x;\ \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}},\ G_{2\ D}(x, y', \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},$$
$$G_{ND}(\vec{x};\ \sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^N} e^{-\frac{|\vec{x}|^2}{2\sigma^2}} \quad (1)$$

The σ value determines the width of the Gaussian kernel. In statistics, the Gaussian probability density function is the standard deviation, while its square, σ^2 is the variance. When we discuss the Gaussian as an aperture function in observations, we will use "s" to refer to the inner scale or simply the scale. This paper's scale is limited to positive values, where σ > 0. During the observation process, s can never be reduced to zero. This implies observing through a tiny aperture, which is practically impossible. The inclusion of the factor of 2 in the exponent is merely a matter of convention. It allows us to have a more simplified formula for the diffusion equation, which we will discuss in more detail later. The convention is to include a semicolon between the spatial and scale parameters to distinguish between them clearly.

5) Random forest (RF): As shown in Fig. 2, the Random Forest (RF) classifier is a technique that involves the simultaneous training of many decision trees using bootstrapping, followed by the aggregation of their outputs by a process referred to as bagging. The process of bootstrapping entails the simultaneous training of several decision trees on different subsets of the training dataset, employing varying subsets of the available characteristics. By ensuring the uniqueness of each decision tree inside the random forest, the total variance of the RF classifier is reduced. The Random Forest classifier combines the judgments made by individual trees in order to get a final conclusion, allowing it to demonstrate strong generalization capabilities. In comparison to other classification approaches, the Random Forest (RF) classifier often achieves superior accuracy while avoiding the problem of overfitting [19].

Similar to the Decision Tree (DT) classifier, the Random Forest (RF) classifier does not need feature scaling. Nevertheless, the Random Forest (RF) classifier has superior robustness in the selection of training samples and handling noise within the training dataset compared to the Decision Tree (DT) classifier. Although the RF classifier is more difficult to read, it has the advantage of simplified hyperparameter adjustment in comparison to the DT classifier.

6) Mouth brooding fish (MBF): According to Fig. 3, Paternal mouthbrooders, often known as mouth-brooding fish, are a group of animals in which the male fish assumes the responsibility of incubating the fertilized eggs inside his oral cavity until they reach the hatching stage. The manifestation of this distinctive kind of parental care is mostly seen in certain species of cichlids, which constitute a diversified assemblage of freshwater fish distributed throughout numerous regions globally [21]. In the phenomenon of mouth brooding, after the deposition of eggs by the female, the male proceeds to fertilize them and then collects them into his oral cavity by the use of his lips. The male exhibits parental care by safeguarding the eggs inside his oral cavity, so shielding them from any threats posed by predators. Additionally, he ensures enough oxygen supply to the eggs by a recurrent process of expelling and re-ingesting them, facilitating their oxygenation [22]. During the incubation stage, which exhibits variability in length contingent upon the species under consideration, the male abstains from consuming sustenance and dedicates his efforts exclusively towards the protection and preservation of the eggs. After the eggs have hatched, it is common for the fry, which refers to the juvenile fish, to be temporarily sheltered inside the oral cavity of the male fish until they have acquired the strength to explore their surroundings independently. The observed behavior exemplifies noteworthy parental investment, which serves to enhance the likelihood of offspring survival via the provision of protection throughout the crucial first phases of development. There are variances seen across different species in terms of their mouth-brooding behaviors, including factors such as the period of incubation and the extent of parental care shown after the discharge of the fry.

In nature, marriage is a crucial mechanism that aids colonies or populations in achieving optimal outcomes by promoting convergence. However, it only sometimes yields favorable outcomes when it occurs. Mouth-brooding fish allow their best cichlids to mate. Thus, the MBF algorithm selects one pair of parents from each cichlid using a probability distribution or Roulette Wheel selection (where higher point values have a higher likelihood). Cichlids that hatch in a new position replace their parents in the population without moving [24]. Before assessing the fitness of the newly hatched fish using a fitness function, we need to ensure that the new positions for the offspring are within the boundaries of the search space.

B. Evaluation Criteria

The primary factors for comparing the results are F-score, accuracy, specificity, sensitivity, and precision [25]. Precision refers to a slight variation between two or more measurements, whereas accuracy represents the disparity between a result and its actual value. The end outcomes should align well, as indicated by precision. The F1 score is the weighted average of precision and recall, including false positives and negatives. Specificity is the test's ability to identify unstick people correctly. Mathematically, a test with high specificity that produces a positive result can confirm a disease because it rarely produces positive results in healthy people. A test's sensitivity determines whether it detects a disease. High-sensitivity tests have few false negatives, reducing disease cases missed. The specificity of a test refers to its capability to correctly identify someone who does not have a disease as being negative. To put it differently, specificity refers to the percentage of individuals who do not have Disease X and receive a damaging result on their blood test. A particular test ensures that all healthy individuals are accurately recognized as healthy, meaning no incorrect positive results exist.

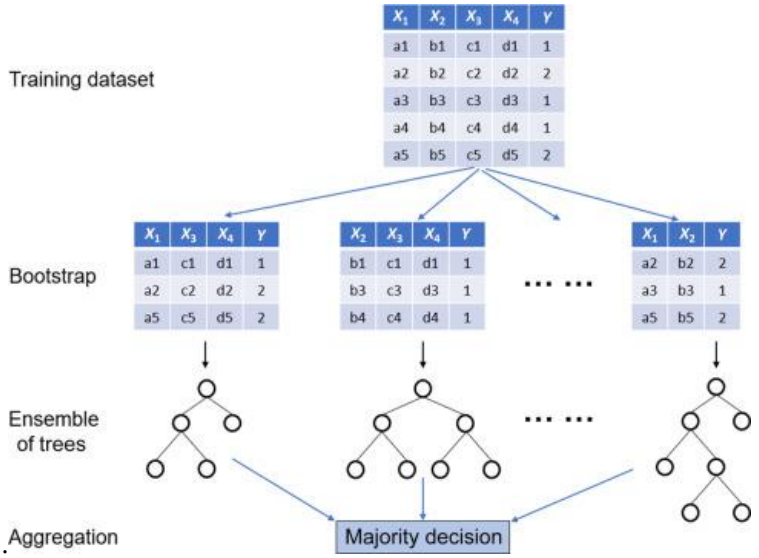


Fig. 2. A dataset with two classes (Y = 1) and four features (X1, X2, X3, and X4) is employed to build a Random Forest (RF) classifier. The RF classifier is an ensemble method that uses bootstrapping and aggregation to train multiple decision trees. Each tree is trained on unique subsets of training samples and features [20]

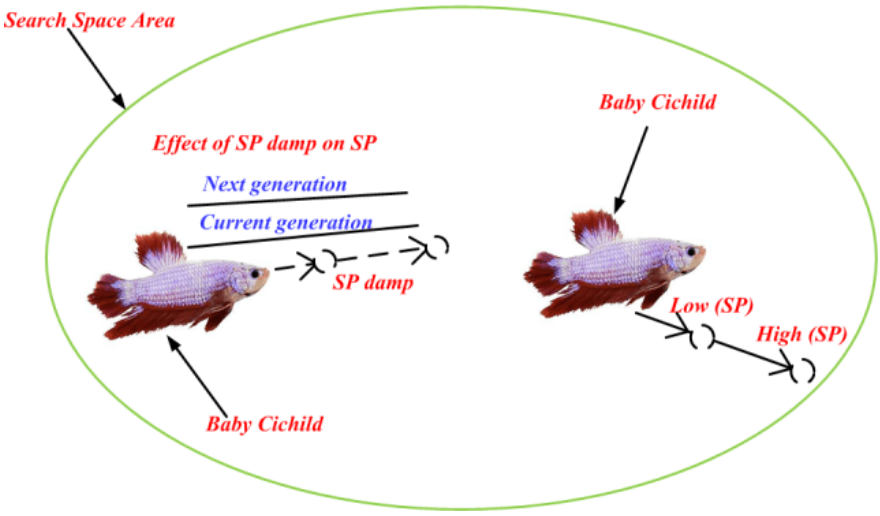


Fig. 3. Mouth Brooding Fish Algorithm [23]

The term "True Negative," sometimes abbreviated as "TN," refers to the outcome that accurately identifies the number of negative instances that have been properly classified. Likewise, the acronym "TP" denotes True Positive, indicating the ratio of accurately recognized positive instances. The term "FP" is used to denote the occurrence of false positives, which refers to the number of cases that are negative but are incorrectly classified as positive. On the other hand, the word "FN" is used to denote the False Negative value, which refers to the count of real positive cases that have been misclassified as negative. The metric of accuracy is often used for the classification of data. The correctness of a model may be determined by using a confusion matrix, which is calculated using the following equation [26].

$$Accuracy = \frac{TN+TP}{TN+FP+FN+TP} \quad (2)$$

Moreover, precision (P), sensitivity (Sn), also known as true positive rate (TPR), specificity (Sp), and F-score values considered for the calculations based on the values of the confusion matrix are as follows [26]:

$$P = \frac{TP}{FP+TP} \quad (3)$$

$$Sn = \frac{TP}{FN+TP} \quad (4)$$

$$Sp = \frac{TN}{FP+TN} \quad (5)$$

$$F - score = 2 \times \frac{P \times Sn}{P+Sn} \quad (6)$$

IV. Dataset

Malware is a pernicious computer software that poses a significant threat to the security integrity of computer systems. Malicious software instructions are concealed among a substantial amount of data, hence rendering conventional protection mechanisms often ineffective in preventing malware attacks. Malicious attacks, such as viruses, worms, and Trojans,

have the potential to inflict damage on a wide range of internet-connected devices [27]. The structure of malware attacks may vary. However, they may be identified by their nature due to the crucial use of online information. The presence of malware on websites poses a significant threat to both individual customers and enterprises. Malware continues to pose a significant cyber danger, as shown by the observation of over 357 million varieties of malware in 2016 [28]. According to AVTEST, a total of ninety-five million websites were found to be infected with malware in 2017 [29]. The distinguishing characteristics of malware may be discerned from site content and browser history or data. The data obtained from malware may provide insights into the characteristics of the virus itself, but it does not often reveal the interrelationships between key data points. Moreover, such data is generally insufficient to identify behavior that can be classified as 'suspicious.' In all instances, perpetrators use several strategies in their endeavor to breach a target's system.

The use of the Android Malware Detection dataset in this simulation is seen as both innovative and suitable. The simulation incorporates many pre-processing techniques, including the conversion of non-numerical variables into numerical representations and the removal of missing values. These operations are necessary due to the categorical and textual nature of some features. Furthermore, each input data point undergoes a translation process to be represented inside the 0-1 interval and then normalized. The probability of misplacing a device remains higher than the probability of contracting malware. Implementing robust encryption measures significantly enhances the security of electronic devices, making them very resistant to unauthorized access and data theft. It is important to establish a robust password for both the device and the SIM card. The dataset known as TUNADROMD has a total of 4465 instances and encompasses 241 distinct attributes. The classification target attribute may consist of a binary categorization, distinguishing between malware and good ware. (Note: The following text is the pre-processed form of TUANDROMD).

Variables:

1-214: Permission-based features

215-241: API-based features

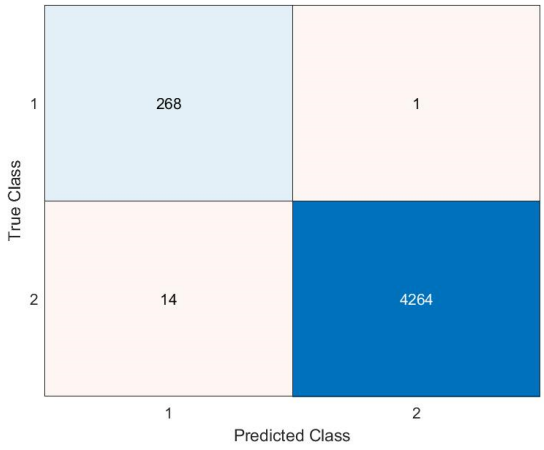
Class Labels

Class: 1) Malware 2) Goodware

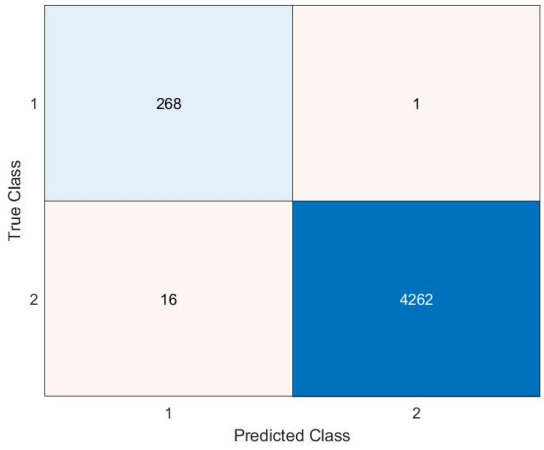
In this study, we utilized the dataset available at https://www.kaggle.com/datasets/subhajournal/android-malware-detection, which serves as a comprehensive resource for Android malware detection research. This dataset comprises a diverse collection of samples, including both malicious applications and benign ones, providing a robust foundation for evaluating the efficacy of different detection methods. The dataset offers detailed information about each sample, such as permissions requested, API calls made, and other relevant features, enabling a thorough analysis of malware behavior and characteristics. By leveraging this dataset, we were able to conduct rigorous experiments to compare the performance of MBF with other established algorithms for Android malware detection, using standard evaluation metrics such as precision, recall, and F1-score. This dataset served as a crucial component in ensuring the validity and reliability of our findings, contributing to the advancement of research in this critical domain of cybersecurity.

V. Results and Discussion

This section provides a discussion of the main findings derived from the study. Furthermore, the efficacy of the suggested algorithm in the field of data categorization is substantiated by an examination of the relevant literature. The evaluation of a classification model's performance in statistics and machine learning may be conducted via the use of a confusion matrix, as seen in Fig. 4. The provided information offers a comprehensive summary of the categorization results, including the quantities of true positive, true negative, false positive, and false negative estimates. According to the data shown in Fig. 4, the MBF algorithm exhibits superior performance compared to the other algorithms. Confusion matrices are an often used evaluation measure in the context of classification problem-solving. The use of this approach may be advantageous for both binary and multiclass classification problems. Confusion matrices provide a tabular representation of the observed and predicted values, displaying the counts for each combination.



MBF CM



Kernel CM

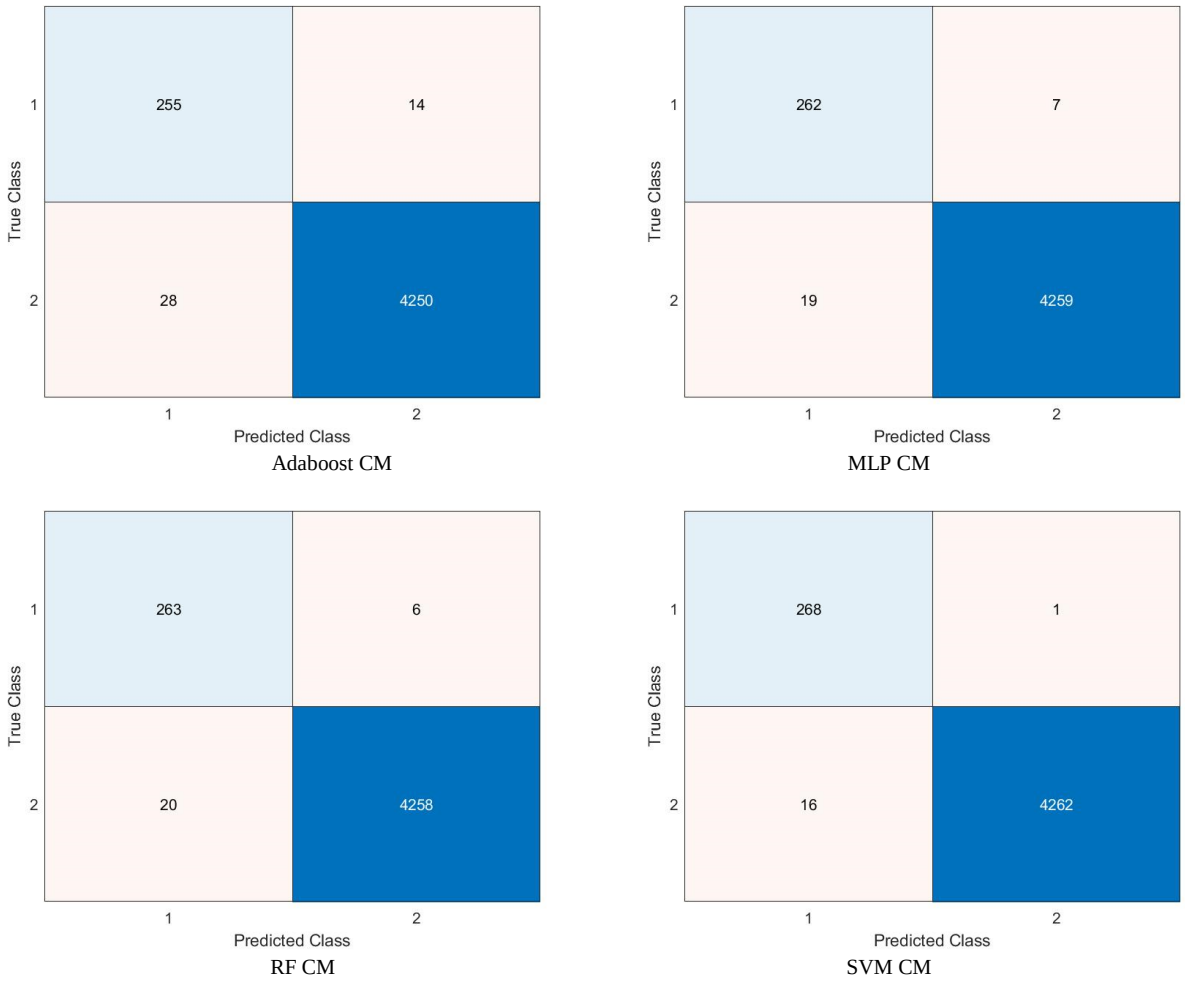


Fig. 4. Confusion matrix for the selected algorithms

Fig. 5 illustrates that MBF has superior sensitivity, indicating a noteworthy proportion of genuine positive cases that the model correctly identified or classified as positive. When it comes to TPR, SVM has the lowest performance. Additionally, based on the data shown in Fig. 6, the accuracy of MBF is satisfactory. The weighted combination of each machine learning model's outputs is the foundation for the working ensemble model's structure. The MBF method seeks to determine the most optimum weighted sum of probability values computed by each model for each issue class. The MBF algorithm's objective function is also the classification's final accuracy value. Thus, after adding up the weighted probability values of each class, they are determined for each class sample by the MBF algorithm, and the accuracy value is determined by comparing the labels assigned by the algorithms. The MBF algorithm is associated with the expected labels. Also, the machine learning models were compared with the proposed ensemble's primary method by calculating the classification's evaluation criteria.

Fig. 7 to 11 demonstrate the values of F-score, accuracy, specificity, and sensitivity obtained for the various selected models. MBF is superior in terms of the criteria values obtained in the work. The Adaboost does not have acceptable performance in data classification. Accordingly, SVM can be an excellent alternative to MBF as it has the highest values of F-score, accuracy, specificity, and sensitivity after that. The results reported in Table I match those in Fig. 7 to 11. MBF, with a value of about 99.67%, is slightly different from Adaboost, which has an accuracy of 99.56%. Compared to the other selected models, the F-score, precision, sensitivity, and specificity values obtained for MBF are remarkable, with 98.57%, 99.65%, 97.51%, and 97.51%, respectively.

The findings of this study underscore the remarkable performance of MBF as a novel approach for Android malware detection and better than previous ones [30]. Across all evaluated metrics including accuracy, F-score, precision, sensitivity, and specificity, MBF consistently outperforms the other algorithms tested, including SVM, Adaboost, MLP, Gaussian Kernel, and RF. With an accuracy of 99.67% and an F-score of 98.57%, MBF demonstrates exceptional accuracy and robustness in identifying both known and unknown malware threats. Additionally, MBF achieves high precision and sensitivity, indicating a low false positive rate and a high true positive rate, respectively, which are crucial for effective malware detection in real-world scenarios.

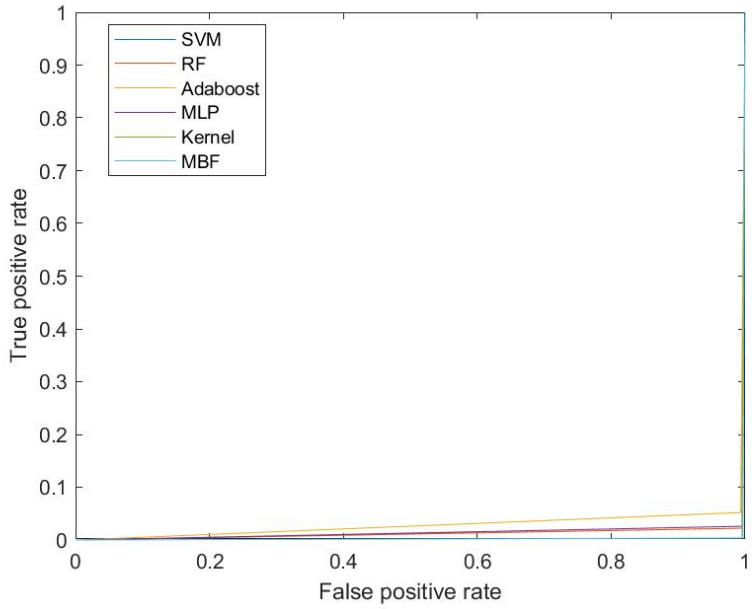


Fig. 5. The true positive rate for the selected models

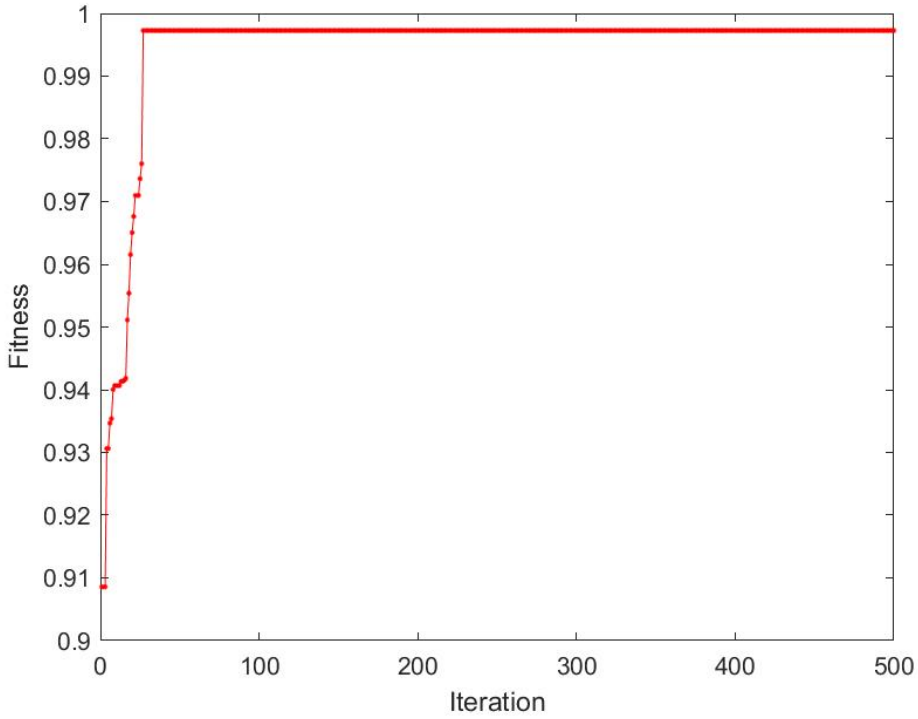


Fig. 6. The accuracy of the proposed method based on iteration and fitness

The superiority of MBF over the previous algorithms [5,8,17] lies in its utilization of ensemble learning techniques, which leverage the strengths of multiple models to enhance predictive performance. By combining the predictions from various base models, MBF achieves a synergistic effect that effectively mitigates the limitations of individual algorithms. Furthermore, the utilization of ensemble learning allows MBF to adapt and evolve over time, enabling it to effectively detect new and evolving malware threats. These findings not only highlight the efficacy of MBF in Android malware detection but also underscore the importance of exploring innovative approaches, such as ensemble learning, to address the escalating challenges posed by malicious actors in the mobile ecosystem.

In Fig. 7, the F-score values illustrate the balance between precision and recall achieved by each model. Fig. 8 showcases

the accuracy values, indicating the overall correctness of the predictions made by the models. Specificity values, depicted in Fig. 9, represent the true negative rate, indicating how well the models distinguish benign samples from malicious ones. Fig. 10 displays the sensitivity values, reflecting the true positive rate or the models' ability to correctly identify malicious samples. Lastly, Fig. 11 presents the precision values, indicating the proportion of correctly identified positive cases among all cases identified as positive by the models.

These figures provide a comprehensive visual representation of the performance of each model across different evaluation metrics, offering insights into their relative strengths and weaknesses in Android malware detection. They serve as valuable tools for understanding and interpreting the results of your study, facilitating comparisons and highlighting the superiority of certain models, such as MBF, over others.

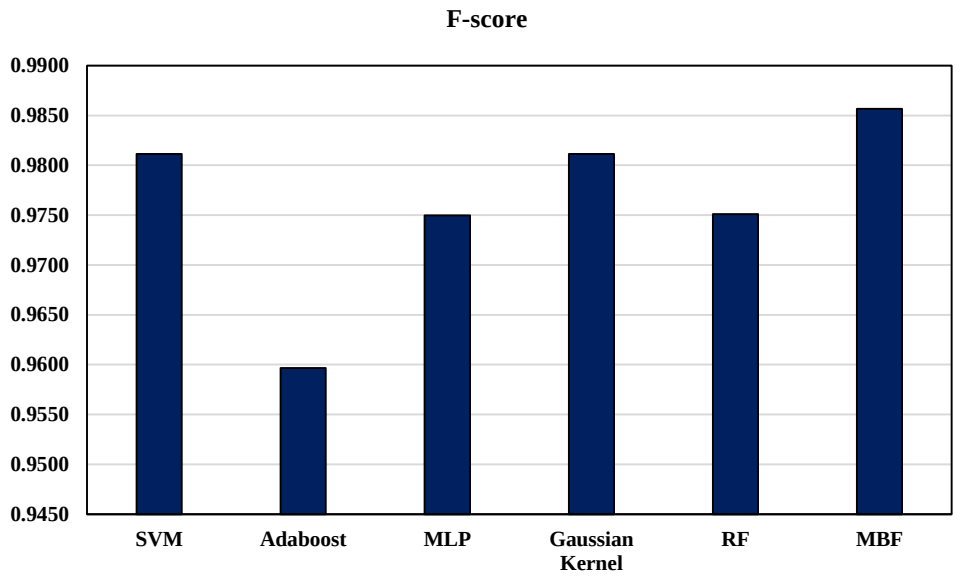


Fig. 7. F-score values of the selected models

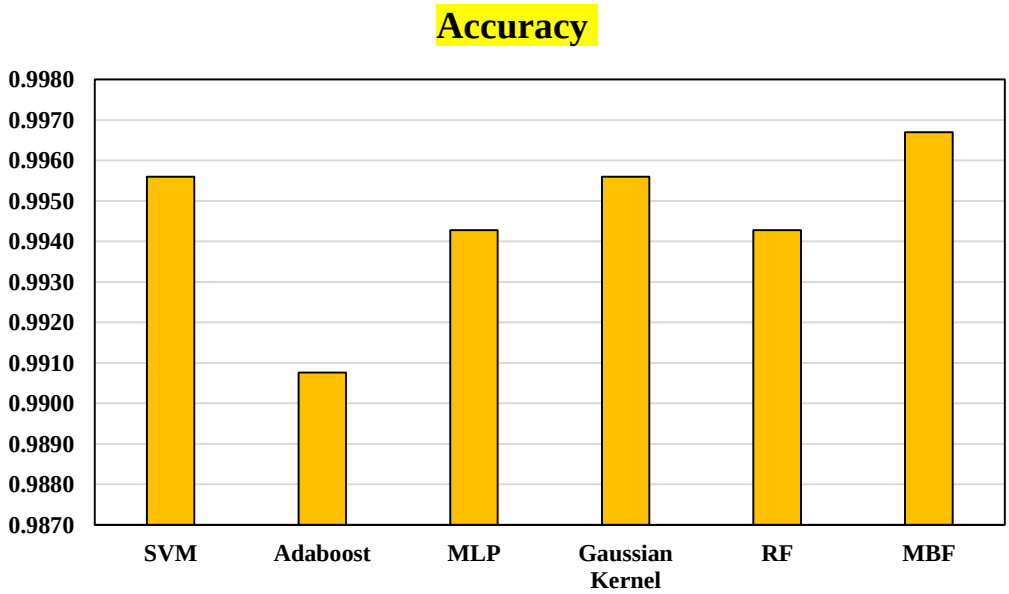


Fig. 8. Accuracy values of the selected models

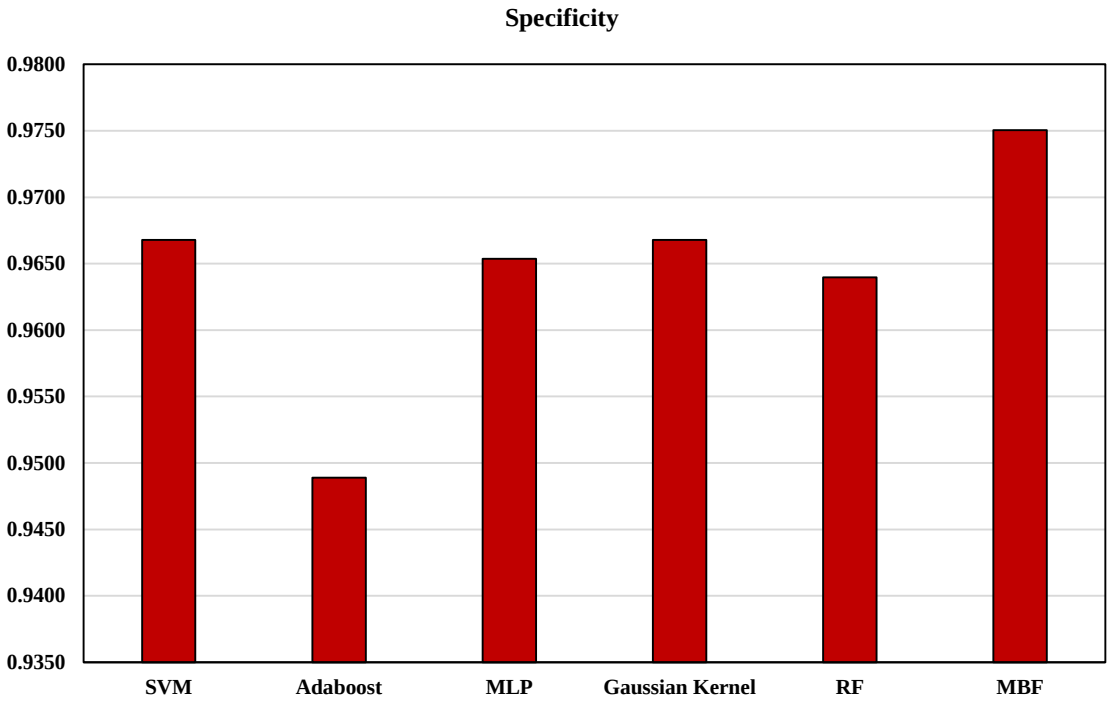


Fig. 9. Specificity values of the selected models

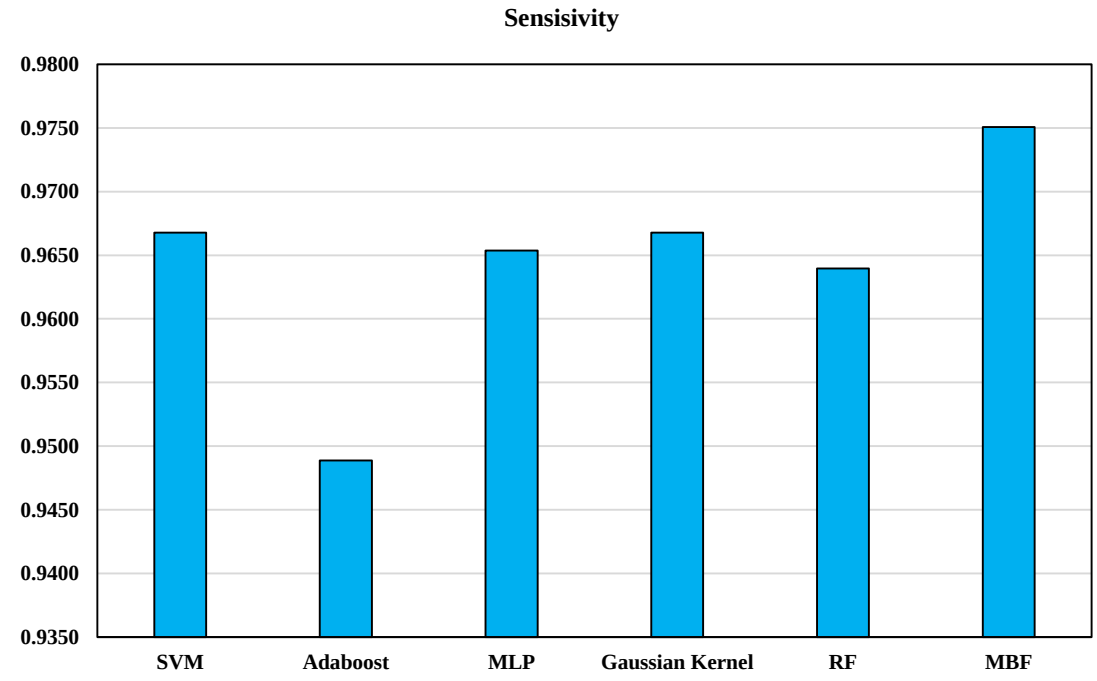


Fig. 10. Sensitivity values of the selected models

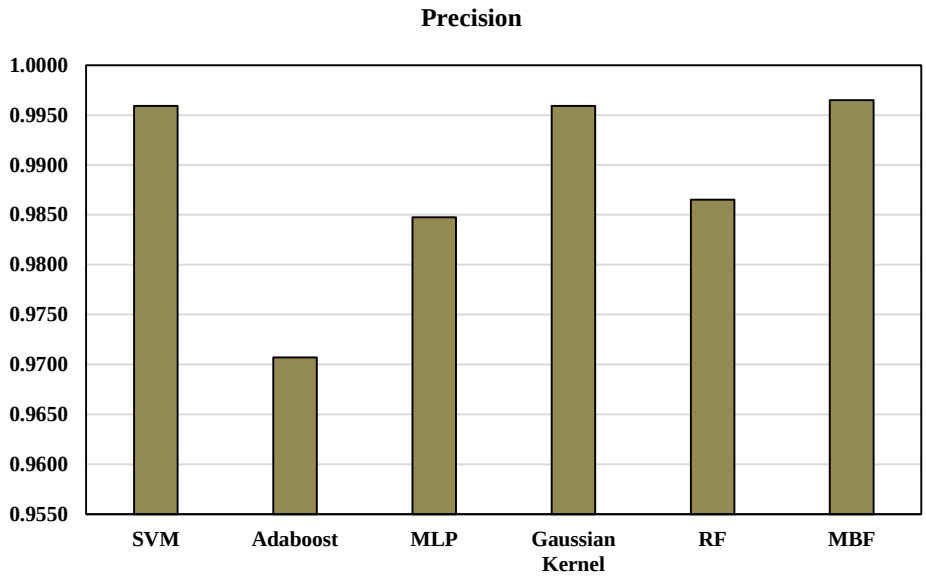


Fig. 11. Precision values of the selected models

TABLE I. COMPARISON BETWEEN THE SELECTED METHODS BASED ON THE STATISTICAL RESULTS

Criteria	SVM	Adaboost	MLP	Gaussian Kernel	RF	MBF
Accuracy	0.9956	0.9908	0.9943	0.9956	0.9943	0.9967
F_score	0.9811	0.9597	0.9750	0.9811	0.9751	0.9857
Precision	0.9959	0.9707	0.9848	0.9959	0.9865	0.9965
Sensisivity	0.9668	0.9489	0.9654	0.9668	0.9640	0.9751
Specificity	0.9668	0.9489	0.9654	0.9668	0.9640	0.9751

VI. CONCLUSION

In summary, a new ensemble model is developed for classification problems in the current study. The dataset considered in this simulation is related to Android malware detection, which is considered a new and suitable dataset. Due to the categorical and textual nature of some features, several pre-processing steps, including coding non-numerical variables into numbers and removing missing values, have been performed in the simulation. Also, all input data are mapped and normalized to intervals of 0 and 1. The structure of the working ensemble model is based on the weighted combination of the outputs of each of the used machine learning models. Finding the most optimal weighted sum of probability values calculated by each model for each class of the problem is the goal of the MBF algorithm. The F-score, accuracy, specificity, and sensitivity values for the chosen models are shown in Fig. 7 to 11. When it comes to the criterion values that were found during the job, MBF is better. In terms of data categorization, the Adaboost's performance is unacceptable. Because SVM has the greatest values of F-score, accuracy, specificity, and sensitivity after MBF, it can be a great substitute for MBF. The outcomes shown in Fig. 7 to 11 correspond with those in Table I. With an accuracy of 99.56%, Adaboost and MBF differ somewhat, with MBF having a value of around 99.67%. The F-score, accuracy, sensitivities, and specificities for MBF are impressive compared to the other chosen models; they are 98.57%, 99.65%, 97.51%, and 97.51%, respectively. Further research on the use of deep learning and insider threat identification issues is warranted. Further attempts could prove quite beneficial to the literature.

FUNDING

This work was supported by the Science and Technology Projects of Jiangxi Provincial Department of Education (No. GJJ2202711，No. GJJ2202722).

REFERENCES

[1] M. Antonakakis et al., “Understanding the mirai botnet,” in 26th USENIX security symposium (USENIX Security 17), 2017, pp. 1093–1110.

[2] N. Scaife, P. Traynor, and K. Butler, “Making sense of the ransomware mess (and planning a sensible path forward),” IEEE Potentials, vol. 36, no. 6, pp. 28–31, 2017.

[3] M. Dhalaria and E. Gandotra, “Android malware detection using chi-square feature selection and ensemble learning method,” in 2020 Sixth international conference on parallel, distributed and grid computing (PDGC), IEEE, 2020, pp. 36–41.

[4] T. Rains and T. Y. CISSP, Cybersecurity Threats, Malware Trends, and Strategies: Discover risk mitigation strategies for modern threats to your organization. Packt Publishing Ltd, 2023.

[5] M. Fire, R. Goldschmidt, and Y. Elovici, “Online social networks: threats and solutions,” IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 2019–2036, 2014.

[6] M. Dhalaria and E. Gandotra, “Risk Detection of Android Applications Using Static Permissions,” in Advances in Data Computing, Communication and Security: Proceedings of I3CS2021, Springer, 2022, pp. 591–600.

[7] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, “A review of android malware detection approaches based on machine learning,” IEEE Access, vol. 8, pp. 124579–124607, 2020.

[8] M. Dhalaria, E. Gandotra, and S. Saha, “Comparative analysis of ensemble methods for classification of android malicious applications,” in Advances in Computing and Data Sciences: Third International Conference, ICACDS 2019, Ghaziabad, India, April 12–13, 2019, Revised Selected Papers, Part I 3, Springer, 2019, pp. 370–380.

[9] O. N. Elayan and A. M. Mustafa, “Android malware detection using deep learning,” Procedia Comput Sci, vol. 184, pp. 847–852, 2021.

[10] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, “Riskranker: scalable and accurate zero-day android malware detection,” in Proceedings of the 10th international conference on Mobile systems, applications, and services, 2012, pp. 281–294.

[11] F. Idrees, M. Rajarajan, M. Conti, T. M. Chen, and Y. Rahulamathavan, “PIndroid: A novel Android malware detection system using ensemble learning methods,” Comput Secur, vol. 68, pp. 36–46, 2017.

[12] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, “Mamadroid: Detecting android malware by building markov chains of behavioral models,” arXiv preprint arXiv:1612.04433, 2016.

[13] K. Sharma and B. B. Gupta, “Mitigation and risk factor analysis of android applications,” Computers & Electrical Engineering, vol. 71, pp. 416–430, 2018.

[14] E. Gandotra, D. Bansal, and S. Sofat, “Malware threat assessment using fuzzy logic paradigm,” Cybern Syst, vol. 48, no. 1, pp. 29–48, 2017.

[15] H. Zhu, Y. Li, R. Li, J. Li, Z. You, and H. Song, “SEDMDroid: An enhanced stacking ensemble framework for Android malware detection,” IEEE Trans Netw Sci Eng, vol. 8, no. 2, pp. 984–994, 2020.

[16] P. Bhat, S. Behal, and K. Dutta, “A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning,” Comput Secur, vol. 130, p. 103277, 2023.

[17] X. Wang, L. Zhang, K. Zhao, X. Ding, and M. Yu, “MFDroid: A stacking ensemble learning framework for Android malware detection,” Sensors, vol. 22, no. 7, p. 2597, 2022.

[18] İ. Atacak, “An Ensemble Approach Based on Fuzzy Logic Using Machine Learning Classifiers for Android Malware Detection,” Applied Sciences, vol. 13, no. 3, p. 1484, 2023.

[19] Y. M. Abd Algani, M. Ritonga, B. K. Bala, M. S. Al Ansari, M. Badr, and A. I. Taloba, “Machine learning in health condition check-up: An approach using Breiman’s random forest algorithm,” Measurement: Sensors, vol. 23, p. 100406, 2022.

[20] J. Heaton, “An empirical analysis of feature engineering for predictive modeling,” in SoutheastCon 2016, IEEE, 2016, pp. 1–6.

[21] D. S. Shayegan, A. Lork, and S. A. H. Hashemi, “Mouth brooding fish algorithm for cost optimization of reinforced concrete one-way ribbed slabs,” Int. J. Optim. Civil Eng, vol. 9, no. 3, pp. 411–422, 2019.

[22] E. Jahani and M. Chizari, “Tackling global optimization problems with a novel algorithm–Mouth Brooding Fish algorithm,” Appl Soft Comput, vol. 62, pp. 987–1002, 2018.

[23] K. Ota, M. Aibara, M. Morita, S. Awata, M. Hori, and M. Kohda, “Alternative reproductive tactics in the shell-brooding Lake Tanganyika cichlid Neolamprologus brevis,” Int J Evol Biol, vol. 2012, 2012.

[24] M. Babazadeh, O. Rezayfar, and E. Jahani, “Interval reliability sensitivity analysis using Monte Carlo simulation and mouth brooding fish algorithm (MBF),” Appl Soft Comput, vol. 142, p. 110316, 2023.

[25] M. Sokolova, N. Japkowicz, and S. Szpakowicz, “Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation,” in Australasian joint conference on artificial intelligence, Springer, 2006, pp. 1015–1021.

[26] A. Kulkarni, D. Chong, and F. A. Batarseh, “Foundations of data imbalance and solutions for a data democracy,” in Data democracy, Elsevier, 2020, pp. 83–106.

[27] K. Rieck, P. Trinius, C. Willems, and T. Holz, “Automatic analysis of malware behavior using machine learning,” J Comput Secur, vol. 19, no. 4, pp. 639–668, 2011.

[28] D. Farhat and M. S. Awan, “A brief survey on ransomware with the perspective of internet security threat reports,” in 2021 9th International Symposium on Digital Forensics and Security (ISDFS), IEEE, 2021, pp. 1–6.

[29] A. V Test, “The independent it-security institute.” 2019.

[30] Hamed Ghorban Tanhaei, Payam Boozary & Sogand Sheykhan, " Analyzing the Impact of Social Media Marketing, Word of Mouth and Price Perception on Customer Behavioral Intentions through Perceived Interaction", in 2024 International Journal of Business and Social Science Vol. 15, No. 1, pp. 69-77, URL: https://doi.org/10.15640/jehd.v15n1a8