Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Hybrid Optimization Technique for Malicious Malware Detection for Feature Selection with Classification

Paramjeet kaur[1], Dr. Vijay Laxmi[2]

[1]Research Scholar, Department Of Computer Applications, Guru Kashi University Talwandi Sabo(Bathinda)
[2]Professor Faculty of Computer Applications, Guru Kashi University Talwandi Sabo(Bathinda)

This study presents a novel technique for identifying malware links through the integration of genetic and PSO algorithms to select feature, subsequent to classification in which Random Forest is utilized. The presented methodology offers an accuracy rate of 99.63% as compared to traditional methods such as XGBoost, SVM, Random Forest, Decision Tree, and the Stacking Classification algorithm. This technique is implemented to accurately flag malicious links when the false alarms and missed detections are diminished. The latest computer methods are employed in this research to illustrate an important progress in online security. An effective approach is obtained to design more advanced tools for combating phishing attempts. In real-world scenarios, future work will aim to evaluate the scalability and efficiency. In conclusion, this innovative methodology suggests a great insight to improve internet security and effectively thwart the phishing threats.

Keywords: Energy efficiency, Smart Room, Co2, Humidity, Hybrid approaches, SMOTE-ENN, KNN imputation.

1. Introduction

The swift advancement of mobile internet technology in recent years has significantly contributed to the growth of the software industry. At the same time, the number of malware threats has been rising sharply. The open nature of the Android application market has made the Android system a prime target for many mobile-based malware attacks. As security threats from Android malware continue to grow, developing efficient and innovative mobile malware detection methods becomes crucial [1][2]. Traditional malware detection techniques often require manually setting detection rules, which limits their effectiveness in identifying new malware variants in a world with an ever-increasing volume of malware. Artificial Intelligence (AI) has gained significant attention over the past few decades as a field of study focused on creating intelligent machines capable of solving problems autonomously without human intervention.

Recent advances in artificial intelligence (AI) have significantly improved malware detection

techniques. These AI-based methods are more accurate, robust, and generalizable compared to traditional detection techniques [3][4], reducing the risk of false detection for newly generated malware. This makes AI-based malware detection an area of considerable scientific interest. Malware detection using AI algorithms typically involves two main phases: data preprocessing, which focuses on extracting software features, and model classification, where the extracted feature data are used to train the model for classification tasks. In the data preprocessing phase, common extraction methods include static extraction and dynamic extraction of feature data. Static extraction involves gathering features without running the software, focusing on bytecode, file header information, API call information, application interface details, and application permission information, among others [5][6]. The key principle behind static analysis is decompiling software to obtain source code or bytecode, then analyzing the semantic features and information contained within. This approach generally incurs lower overhead and is more stable. For example, MaMadroid uses Android application permission information, API call data, and other static data for malware detection. In contrast, dynamic extraction analyses the behavioural activities of software during runtime. This approach tends to yield more accurate feature data since it captures the software's behaviour in real-time. For instance, DL-Droid uses software log files from actual devices to extract feature data. However, many malware programs hide their malicious behaviour when running in a virtual environment [7] [8], making the setup for dynamic analysis more complex and resource-intensive. This can lead to decreased stability in accuracy and higher overhead, complicating the implementation of such malware classification models. In the model training and classification phase of malware detection, the primary approaches involve methods based on machine learning (ML) algorithms and deep learning (DL) models. ML is a branch of artificial intelligence (AI) that enables machines to automatically learn from experience and make decisions accordingly [9][10]. Deep learning, a subset of ML, utilizes neural networks with architectures inspired by the human brain to analyse complex sets of variables. Cybersecurity researchers have explored a variety of AI-based techniques for detecting malware attacks. Machine learning-based methods commonly use algorithms like Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Naive Bayes (NB), and Random Forest (RF), among others.

While malware detection using individual machine-learning classifiers has been extensively studied, the performance of each model can vary due to differences in training datasets and feature selection strategies. Additionally, each classifier has its own limitations and uncertainties. By combining multiple classifiers, you can reduce the variance in expected errors and improve classification accuracy compared to using a single classifier [11][12]. Ensemble learning is a machine learning strategy that relies on multiple methods to create a more accurate classifier, rather than depending on a single approach. This strategy has proven effective in many fields. Theoretically and practically, ensemble learning techniques have been shown to outperform weaker single-method approaches, particularly when addressing complex and high-dimensional prediction problems. The most common ensemble learning techniques are bagging, boosting, and stacking [13][14]. Stacking is a technique that merges multiple machine learning models (base models) into one stage and then applies a different machine learning model (meta model) to create a more accurate classification system. Boosting is a sequential ensemble technique designed to enhance the performance of a model by focusing on data that was previously misclassified. At the start, each data point in the

dataset is given an equal weight. When a model (N) makes an error, the weight of the misclassified data increases, so that in the next iteration (N+1), the model focuses more on these challenging cases [15][16]. This approach aims to stabilize the entire learning process by reducing errors from earlier classifiers. The two commonly used boosting methods include AdaBoost and XGBoost. Bagging is an approach that builds an ensemble of classifiers by altering the training sets given to a base classifier. It starts by selecting a single base classifier and then using it multiple times with different training subsets, which are created by randomly selecting samples from the original training set. Each subset is used to train a base classifier. The predictions from each base classifier are then combined based on a predefined rule. An ensemble learning approach addresses the limitations of single-model methods by combining multiple separate classifiers, typically achieving better generalization performance than a single model [17][18]. The core concept behind ensemble methods is to rearrange training datasets in various ways (through resampling or reweighting), train a base classifier on each rearranged dataset, and create an ensemble of base classifiers. A new ensemble classifier is then developed using the stacked ensemble method, where a new model learns to integrate predictions from multiple base models to improve accuracy.

2. Literature Review

Z. Chen, et.al (2023) suggested a bicubic interpolation (BI)-based method to identify and classify malware on enhancing the security of plant protection information terminal (PPIT) system [17]. The BI algorithm was implemented for reconstructing the generated malware images so that the issue image size imbalance was tackled. The Cycle-GAN model was adopted to augment data for balancing the number of samples among malware families and an effectual malware classification framework was developed on the basis of CNNs for enhancing the efficacy to identify and classify malware. The experimental outcomes depicted that the suggested method was effective to classify malware. Moreover, this method offered an accuracy of 99.76% for RGB images and 99.62% for gray images of MMCC (BIG2015) dataset.

D. A. Kumar, et.al (2023) introduced a mechanism for identifying and classifying several files and API calls into benign and harmful on the basis of two-level classification algorithms: Macro (to detect malware) and Micro (to classify malware files into a Trojan, Spyware, Adware, etc.) [18]. A classifier was adopted for data mining (DM) to discover malware. The features and behaviors of every virus were considered to project diverse classifiers which assisted in identifying malware. A dynamic analysis method was implemented for recognizing the malware traits. The Cuckoo Sandbox was employed for executing sample files in a virtual setting so that the static and dynamic analysis reports were created. The Weka tool and training datasets were exploited for developing the Machine learning (ML) methods. The experiments exhibited that the introduced mechanism was effective to detect and classify malware with the help of diverse ML models.

R. Alguliyev, et.al (2024) presented a method to classify CPS malware relied on pre-trained deep neural network (DNN) algorithms [19]. Various visual representations of malware and detection models were integrated on the basis of transfer learning (TL). Two algorithms called AlexNet and MobileNet were adopted to differentiate diverse malware families in grayscale

images. Radon transform (RT) was executed on resulting grayscale malware images for enhancing the accuracy to classify novel malware binaries. Three datasets: Microsoft Malware Classification, IoT_Malware and MalNet-Image, were executed for simulating the presented method. The experimental results demonstrated that the presented method was performed better against the traditional schemes concerning efficacy to classify malware families which affected the cyber-physical systems.

S. H. Khan, et.al (2023) designed a novel model called Deep Squeezed-Boosted and Ensemble Learning (DSBEL) to detect malware in which a new Squeezed-Boosted Boundary-Region Split-Transform-Merge (SB-BR-STM)-CNN and ensemble learning were utilized [20]. The STM block was focused on deploying multi-path dilated convolutional (MDC), Boundary, and regional operations for capturing the homogenous and heterogeneous global malevolent patterns. Furthermore, the transfer learning (TL) and multi-path-based squeezing and boosting (MSB) were deployed at initial and final levels for attaining various feature maps which learned the miniature pattern variations. At last, this model aimed to extract discriminative features from the initial utilized method. The ensemble algorithm of (SVM, MLP, and AdabooSTM1) was fed with these features for enhancing the generalized ability of hybrid learning. The experiments revealed that the designed model offered an accuracy of 98.50%, F1-Score of 97.12%, MCC of 91.91%, recall of 95.97% and precision of 98.42%.

Y. Wang, et.al (2024) developed a Deep Learning (DL) Based Malware Attack DetectoR in Android Smartphones using LinkNET (MADRAS-NET) for identifying and mitigating the kinds of malwares in Android devices [21]. The Max Abs Scaler was fed with a set of data to pre-process it. Afterward, LinkNET deployed its output to classify the malware. The pre-processed data was assisted in identifying malware, and splitting the output into 3 classes, namely real users, Penetho malware, and FakeAV malware. In the end, the developed method was computed on AndMal2020 dataset that assisted in detecting and classifying malware and its families. The developed method offered an accuracy of 99.81% in comparison with traditional methods.

B. T. Hammad, et.al (2022) established a malware classification (MC) technique to identify malware in five phases [22]. In these phases, the dataset was prepared to generate 2D malware images from the malware binary files; the visualized malware was pre-processed to scale the visual malware images to be suitable in the input size of CNN model. Moreover, the hand-engineering (Tamura) and deep learning (GoogLeNet) methods were adopted for extracting the features. At last, the K-Nearest Neighbor (KNN), Support Vector Machines (SVM), and Extreme Learning Machine (ELM) algorithms were implemented for classifying malware. The Malimg unbalanced dataset was executed for simulating the established technique. The accuracy of this technique was found higher. The results revealed that the established technique was performed better and offered an accuracy of 95.42% in contrast to manual attributes and 96.84% than deep feature methods.

F. Nawshin, et.al (2024) suggested a new technique called DP-RFECV-FNN for detecting android malware in which Differential Privacy (DP) was exploited in a Feedforward Neural Network (FNN), for IoT networks under the zero trust framework [23]. The DP was combined for ensuring that the data was kept confidential in the detection procedure for which a novel standard was established for privacy in cybersecurity methods. Moreover, the potentials of DP

were and zero trust security integrated with robust learning potential of the FNN for recognizing known and new malware kinds and offering superior accuracy when the privacy controls were maintained. The results indicated that the suggested technique offered an accuracy of 97.78%-99.21% for static features and 93.49%-94.36% for dynamic features of Android applications while detecting them as malware and benign.

S. S. Alshamrani, et.al (2022) devised a novel machine learning (ML)-based system for classifying PDF malware [24]. The given PDF file was examined in a statistical and dynamic way to maximize the accuracy to discover the precise nature of the document. This system was capable of distinguishing obscure and zero-day malware. Five diverse methods were implemented in experimentation for computing the finest approach. Diverse metrics: true positive rate (TPR), precision, false positive rate (FPR), false negative rate (FNR), and F1-score were considered for analyzing the devised system. A malicious attack was launched on this system for obfuscating the malicious code within the PDF file. The PDF parser was utilized to conceal this file. The devised system provided a F1-measure up to 98.6% with random forest (RF) in contrast to other techniques.

S. J. I. Ismail, et.al (2024) introduced a self-supervised learning (SSL)-based technique called MalSSL in which image was represented for classifying malware [25]. The contrastive learning (CL) and data augmentation (DA) technique were utilized for classifying unlabeled malware images. First of all, an unlabeled Imagenette dataset was employed for training this technique and an unlabeled malware dataset was exploited for retaining this technique in downstream tasks such as to classify malware family and malware benign. The introduced technique yielded an accuracy up to 98.4% in initial task on Malimg dataset and 96.2 % on Maldeb dataset. The results depicted that the introduced technique was effective for classifying malware at superior accuracy in contrast to other technique for which no labeled data was required.

Ö. Aslan, et.al (2021) recommended a new deep learning (DL)-based model for classifying malware variants on the basis of hybrid algorithm [26]. A novel hybrid framework was developed in which 2 extensive pre-trained network algorithms were implemented. Four phases were executed in which data acquisition was performed, a deep neural network (DNN) model was developed, trained, and evaluated at last. The Malimg, Microsoft BIG 2015, and Malevis datasets were applied for computing the recommended model. The experimental outcomes proved the supremacy of recommended model for classifying malware at higher as compared to other methods and offered an accuracy of 97.78% on initial dataset.

3. Research Methodology

In the previous year's many techniques has been proposed for the detection of malware which are based on the machine learning models. The techniques which are already been proposed are unable to achieve designed accuracy to overcome that drawback novel model is proposed in this research work for the malicious malware detection. The proposed model is discussed in the detail below: -

The focus of this research is to detect malicious malware. The process of detecting malicious malware involves several stages, such as pre-processing, feature extraction, and classification.

Various techniques have been developed in recent years on the basis of machine learning models to detect phishing URL. But many have not been capable of achieving the desired level of accuracy. For addressing this limitation, a novel model is introduced in this study for detecting phishing URLs. This developed model is defined as:

1. Dataset Input and pre-processing: - In the first stage the dataset will be collected from the authentic source. The collected dataset has 48 feature sets and approx. 1000 instances. The dataset has two class which are malicious and non-malicious. The dataset is well balanced and in the pre-processing phase redundant, missing values will be replaced with mean of the dataset.

2. Feature Extraction: - The second phase is of feature extraction in which attribute set establish relationship with target set. The hybrid optimization algorithm is the combination of genetic and PSO algorithm. The proposed flowchart is the hybrid version of Genetic and PSO algorithm. This algorithm is useful to select the optimization attributes and encoding an effective solution for an issue into an individual. In fact, every individual is considered as an entity supporting features of chromosomes. A number of individuals collectively creates a population. The major task is to generate a population of chromosomes randomly and surround it with variables of problem prior to deploy Genetic Algorithm (GA). The next phase emphasizes on assessing the created data chromosomes. The chromosomes, which are capable of clearly demonstrating an optimal method for tackling the issue, are useful for building other chromosomes. The population is defined as the primary set of random solutions available in this algorithm. A chromosome is utilized for illustrating every member of the population in order to perform coding for a solution for dealing with the issue. The decoding formula is expressed as:

$$X = X_{min} + \frac{X_{max} - X_{min}}{2^{Nx} - 1} \sum_{n=0}^{N^x - 1} b_n^X 2^n \tag{1}$$

In which, $b_0^X, \ldots\ldots, b_{N^x-1}^X$ denote the binary representations of X's. Various iterations called generations are exploited for creating the chromosomes. In every generation, a number of fitness indicators are executed for evaluating the fitness value of the chromosomes. Every particle i has a relation with 2 vectors, such as the position vector denoted with $X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$ and the velocity vector $V_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,n}]$. The positions $x_{i,d}$of novel solutions are adjusted at constant rate for performing their searching process. For every particle, this algorithm focuses on reminding the historical location of an individual as $pbest_i$, and the current global optimal position which the entire particle swarm has discovered is defined with gbest. The discovery of these locations lead to update the velocity and position of every particle in according with the given equations as:

$$v_{i,d}(t+1) = \omega \cdot v_{i,d}(t) + c_1 \cdot rand_1 \cdot \left(pbest_{i,d} - x_{i,d}(t)\right) + c_2 \cdot rand_2 \cdot \left(gbest_d - x_{i,d}(t)\right) \tag{2}$$

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \tag{3}$$

In this, the t-th iteration is illustrated with t, d is used to represent the d-th dimension of the

particle, ω denotes the inertia weight, c_1 and c_2 are used to demonstrate the acceleration constants, the random numbers are specified with $rand_1$ and $rand_2$ whose distribution is done at random within the interval [0, 1]. The mitigation of the inertia weight ω leads to enhance the efficiency of this algorithm. this weight is defined as:

$$\omega = \omega_{max} - (\omega_{max} - \omega_{min}) \cdot \frac{t}{t_{max}} \quad (4)$$

In this, ω_{max} is used to denote the maximal weight and ω_{min} shows the minimum weight, t defines the number of the current iteration, and the number of the maximum iteration is specified with t_{max}. The crossover operator or a mutation operator (MO) are implemented to integrate 2 chromosomes taken from the current generation for generating the offspring. A steady population size is maintained through an innovative generation. Some parents and children are selected on the basis of fitness values and others are rejected for producing this generation. Several possibilities are available for fitter chromosomes.

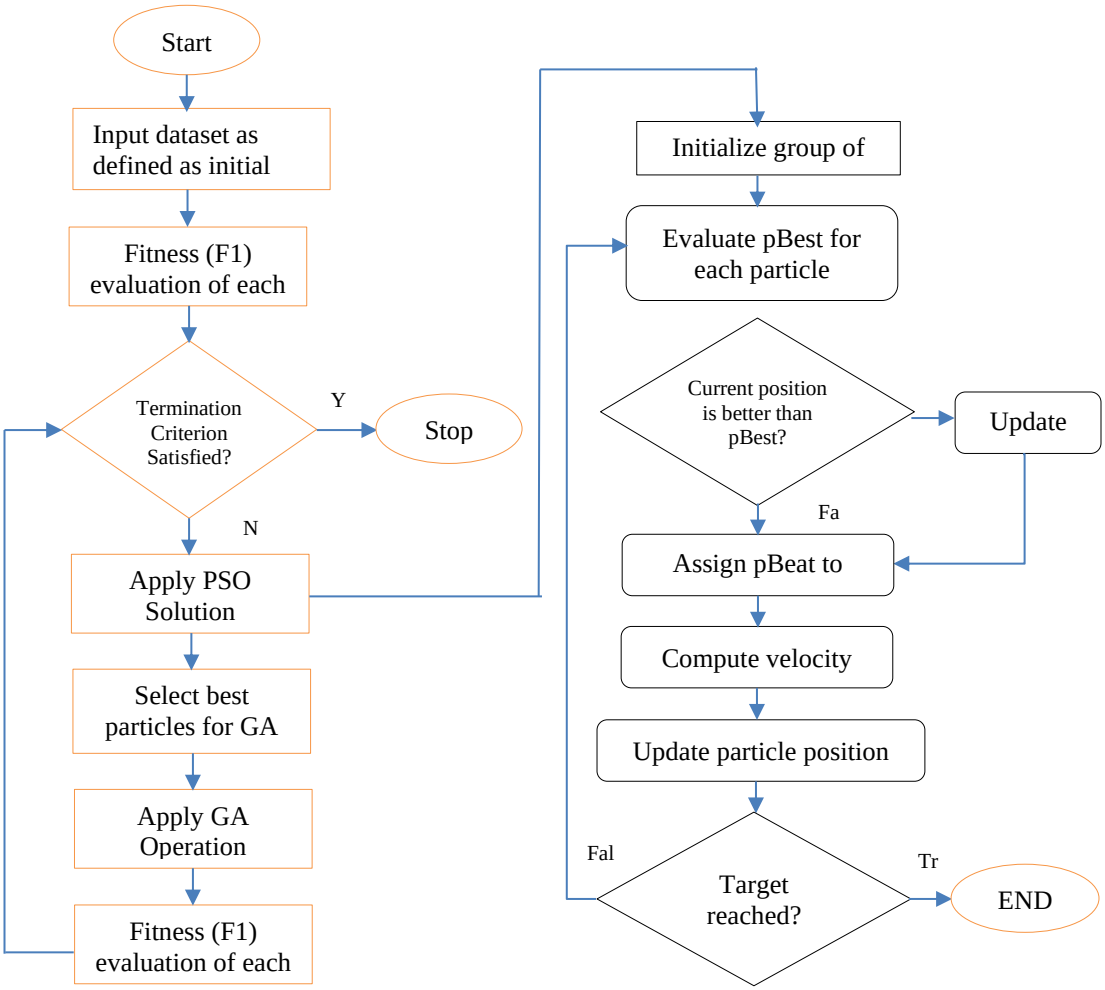

Figure 2: Proposed Model

3. Classification: - The random forest classifier is applied for the classification. The random forest model takes out of optimization algorithm as input for the classification. In Random Forest the trees are combined to create a single, strong learner after averaging or getting the majority vote when numerous tiny, weak DTs are formed in tandem. The RFs are frequently studied as the most precise learning algorithms for training to date. Formally, an RF be a predictor built of a set of randomly generated base regression trees, where $\{r_n(x,\ominus_m, D_n), m \geq 1\}$, where $\ominus_1,\ominus_2$ .... are the independently distributed outputs of a randomly generated variable $\ominus$. For the purpose of creating the aggregated regression estimate, these RT integrations are performed.

$$\bar{r}_n(X, D_n) = \mathbb{E}_{\ominus}[r_n(X,\ominus, D_n)], \tag{5}$$

In where, subject to X and the data set D_n, $\mathbb{E}_{\ominus}$ denotes what is expected as a function of the random parameter. The dependence of the estimations would be eliminated from the sample in the following notation to simplify it a little and given in the form $\bar{r}_n(X)$ rather than $\bar{r}_n(X, D_n)$. When the M RTs are generated and the average of the individual outcomes is obtained, Monte Carlo was used to calculate the aforementioned expectation. When creating individual trees, where the choice of the split coordinate and split position are constructed, the randomising variable $\ominus$ is used to assess how well subsequent cuts work. As the independent of X and the training sample D_n, the variable $\ominus$ is inferred.

4. Result and Discussion

In this section results of the proposed model are presented and also compared with existing machine learning model. The results of proposed model are tested on authentic data source and compared in terms of accuracy, precision and recall.

4.1. Performance Analysis Metrics

In this section performance analysis metric are presented. The details of the metrics are presented below: -

- Accuracy: - Accuracy is used to measure the performance in the evidence domain recovery and processing of the data. The fraction of the results that are successfully classified can be represented by equation as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

- Precision: - Precision is a performance assessment that measures the ratio of correctly identified positives and the total number of identified positives. This can be seen as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall: - The recall is also referred to as the sensitivity, which is the ratio of connected instances retrieved over the total number of retrieved instances and can be seen as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

4.2 Results

In this section results of proposed model are presented and also compared with existing machine learning models like Decision tree, Random forest, SVM, XGBoost.

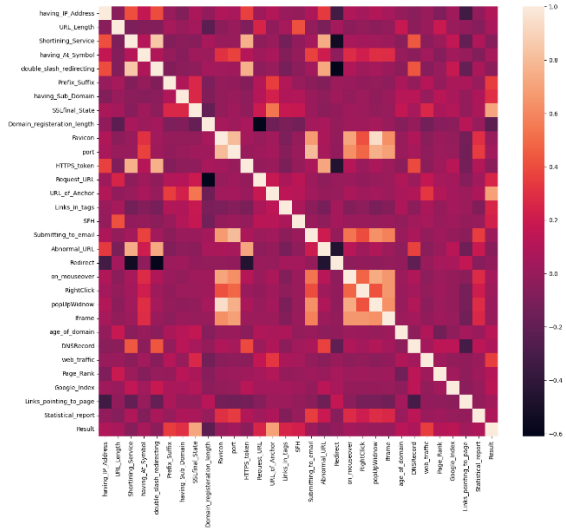

Figure 3: Correlation Matrix

As shown in figure 3, the dataset has various attributes and each attribute has correlation with another attribute. The correction matrix is drawn which illustrates relation with each attribute.

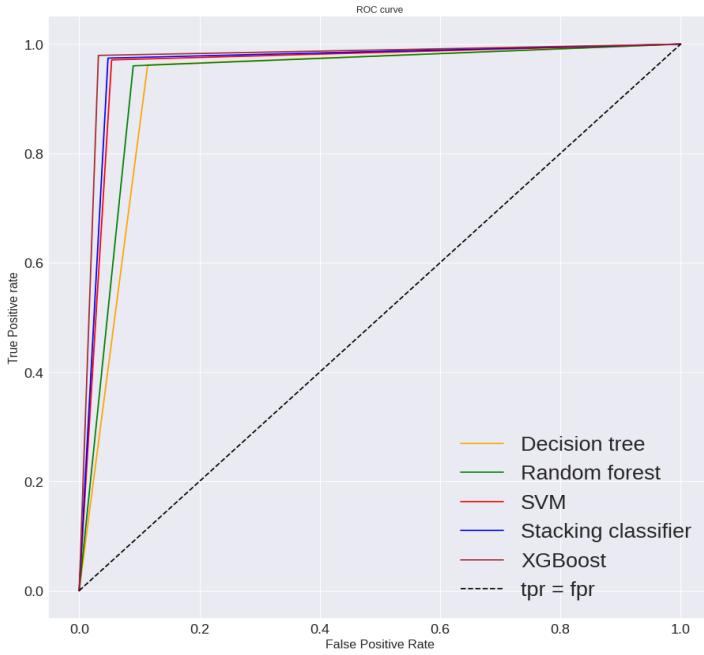

Figure 4: ROC Curve

As shown in figure 4, the ROC curve is drawn and compared from the prediction of the various machine learning algorithm. The ROC curve of decision tree, random forest, SVM, XBboost and stacking is compared for the malicious malware detection.

Table 1: Performance Analysis

Model	Accuracy	Precision	Recall
XGBoost	97.42	97.37	97.93
SVM	96.43	96.9	97.44
Random Forest	93.76	92.81	96.03
Decision tree	92.76	91.08	96.20
Stacking Classifier	97.42	97.37	97.93
Proposed model	99.63	99	99

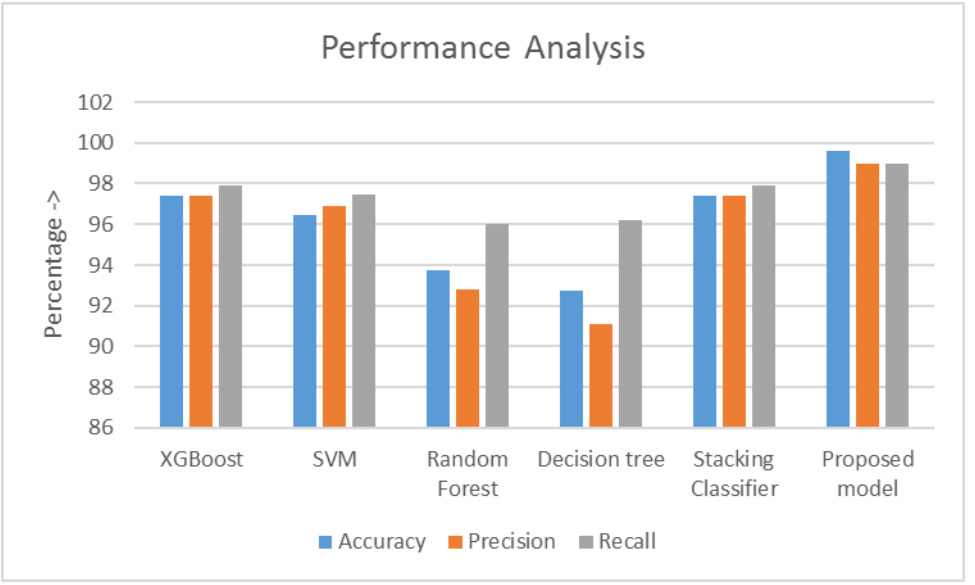

Figure 5: Performance Analysis

As shown in figure 5, the performance of proposed model is compared with other machine learning models for the malicious malware detection. The proposed model is compared with XGBoost, SVM, Random forest, decision tree and stacking in terms of accuracy, precision and recall. It is analysed that proposed model achieves maximum accuracy of 99.63 percent as compared to other machine learning models.

5. Conclusion

In conclusion, the hybrid model proposed in this study, which combines genetic and PSO algorithms for feature extraction and Random Forest for classification, stands out with an outstanding accuracy of 99.63%, surpassing that of traditional models. Compared to existing methods, such as XGBoost with an accuracy of 97.42%, SVM with 96.43%, Random Forest

with 93.76%, Decision Tree with 92.76%, and the Stacking Classifier with 97.42%, the hybrid model demonstrates superior performance in malicious malware detection. Its high precision and recall rates further underscore its efficacy in minimizing false positives and false negatives. This research underscores the potential of integrating advanced optimization algorithms with machine learning techniques to address evolving cybersecurity challenges effectively. Further exploration of its scalability and real-world applicability will be crucial for its practical deployment and continued advancement in safeguarding against malicious malware detection.

References

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

[2] M. Sun, X. Li, J. C. S. Lui, R. T. B. Ma and Z. Liang, "Monet: A User-Oriented Behavior-Based Malware Variants Detection System for Android," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 5, pp. 1103-1112, May 2017

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

[4] A. Guerra-Manzanares, M. Luckner and H. Bahsi, "Android malware concept drift using system calls: Detection, characterization and challenges", Expert Systems with Applications, vol. 12, no. 5, pp. 859-867, 21 April 2022

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

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

[7] S. H. Moghaddam and M. Abbaspour, "Sensitivity analysis of static features for Android malware detection," 2014 22nd Iranian Conference on Electrical Engineering (ICEE), 2014, pp. 920-924

[8] G. Canbek, S. Sagiroglu and T. Taskaya Temizel, "New Techniques in Profiling Big Datasets for Machine Learning with a Concise Review of Android Mobile Malware Datasets," 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), 2018, pp. 117-121

[9] J. H. Abawajy and A. Kelarev, "Iterative Classifier Fusion System for the Detection of Android Malware," in IEEE Transactions on Big Data, vol. 5, no. 3, pp. 282-292, 1 Sept. 2019

[10] M. Samara and E. -S. M. El-Alfy, "Benchmarking Open-Source Android Malware Detection Tools," 2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), 2019, pp. 1-6,

[11] Y. Xue et al., "Auditing Anti-Malware Tools by Evolving Android Malware and Dynamic Loading Technique," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 7, pp. 1529-1544, July 2017

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

[13] R. B. Hadiprakoso, I. K. S. Buana and Y. R. Pramadi, "Android Malware Detection Using Hybrid-Based Analysis & Deep Neural Network," 2020 3rd International Conference on Information and Communications Technology (ICOIACT), 2020, pp. 252-256

[14] S. Liang and X. Du, "Permission-combination-based scheme for Android mobile malware detection," 2014 IEEE International Conference on Communications (ICC), 2014, pp. 2301-2306

[15] E. C. Bayazit, O. Koray Sahingoz and B. Dogan, "Malware Detection in Android Systems with Traditional Machine Learning Models: A Survey," 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2020, pp. 1-8

[16] J. Wu and A. Kanai, "Utilizing obfuscation information in deep learning-based Android malware detection," 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), 2021, pp. 1321-1326

[17] Z. Chen, S. Xing and X. Ren, "Efficient Windows malware identification and classification scheme for plant protection information systems", Frontiers in Plant Science, vol. 14, pp. 125-136, 2023

[18] D. A. Kumar and S. K. Das, “Machine Learning Approach for Malware Detection and Classification Using Malware Analysis Framework”, International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 1, pp. 330–338, 2023
[19] R. Alguliyev, R. Aliguliyev and L. Sukhostat, “Radon transform based malware classification in cyber-physical system using deep learning”, Results in Control and Optimization, vol. 14, pp. 103-110, 4 February 2024
[20] S. H. Khan, T. J. Alahmadi and A. O. Almagrabi, “A new deep boosted CNN and ensemble learning based IoT malware detection”, Computers & Security, vol. 133, pp. 385-396, 7 July 2023
[21] Y. Wang and S. Jia, “MADRAS-NET: A deep learning approach for detecting and classifying android malware using Linknet”, Measurement: Sensors, vol. 33, pp. 41-49, 19 March 2024
[22] B. T. Hammad, N. Jamil, I. T. Ahmed, Z. M. Zain and S. Basheer, “Robust Malware Family Classification Using Effective Features and Classifiers”, Applied Sciences, vol. 12, no. 15, pp. 887-895, 2022
[23] F. Nawshin, D. Unal and P. N. Suganthan, “AI-powered malware detection with Differential Privacy for zero trust security in Internet of Things networks”, Ad Hoc Networks, vol. 6, no. 4, pp. 169-175, April 2024
[24] S. S. Alshamrani, “Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document Format Files”, Security and Communication Networks, vol. 1, pp. 17-27, 2022
[25] S. J. I. Ismail, Hendrawan, B. Rahardjo, T. Juhana and Y. Musashi, "MalSSL – Self-Supervised Learning for Accurate and Label-Efficient Malware Classification," IEEE Access, vol. 2, no. 1, pp. 368-376, 2024
[26] Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021