International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

An Enhanced Social Media Android Malware Detection through Stacking Ensemble Machine Learning Model with Reinforcement Learning-Based Rule Factor

Saqib Malik[1], Narendra Sharma[2]

[1]Research Scholar, Department of Computer Application, Sri Satya Sai University of Technology and Medical Sciences, Sehore, 466001, India.
[2]Associate Professor, Department of Computer Science, Sri Satya Sai University of Technology and Medical Sciences, Sehore, 466001, India.

Abstract

Android malware detection in social media dataset faces challenges such as feature selection, class imbalance, and efficient model construction. Imbalanced datasets reduce detection accuracy, necessitating strategies to balance data and enhance detection precision. This study aims to develop an ensemble learning-based model that tackles class imbalance, and ensures accurate and efficient malware detection. It also introduces a novel classifier for Android malware detection system that leverages reinforcement learning-based rules. Two sampling methods—random undersampling, and random oversampling—are applied to balance imbalanced datasets. The proposed Stacking Model with Reinforcement Learning-Based Rule Factor (SMRLF) combines classifiers such as stochastic gradient tree boosting (SGTB), Decision Tree (DT), Logistic Regression (LR) and Naïve Bias (NB) for android malware detection and employs reinforcement learning-based rules to enhance classification precision and validation. The SMRLF model achieves 99.00% detection accuracy, outperforming alternative classifiers like LR (95.00%), Naïve Bayes (67.50%), DT (93.20%), and meta-learner (98.40%). It also demonstrates reduced computational time (24.872 seconds) compared to others. The study introduces a novel SMRLF classifier that combines reinforcement learning-based rules factor with ensemble learning techniques, offering a robust solution for Android malware detection with high accuracy and efficiency. This work represents a major breakthrough in stacking techniques by presenting a novel combination of classifiers intended to improve accuracy. This study highlight the value of ensemble approaches, especially stacking, in enhancing Android malware detection rates by assessing classifier performances using a new association of two social media datasets. In addition to improving detection accuracy, this method offers a framework that may be used for various classification problems. Beyond malware detection, the knowledge gathered from this effort provides insightful viewpoints for more general categorization applications. Sustained efficacy in detection systems depends on continued research into developing malware approaches in order to preserve and grow these achievements.

Keywords: Android malware detection, social media, ensemble learning, Stacking Model, Reinforcement Learning-Based Rule Factor.

1. Introduction

Internet usage for communication is growing at a very fast rate; thus, social media sites are part of daily lives. Unfortunately, these platforms have also become a target for cyber threats, especially Android malware distribution. Social media malware typically takes advantage of user trust to spread only malicious links or applications that infect computers and sensitive data. Such threats can come from attackers seeking to capture information, launch phishing sprays, or install spyware and adware. Furthermore, user ingenuity, along with insufficient cyber consciousness and unclientele execution, has caused monetary losses and a breach of privacy leading to an increase in such digital assaults resulting in challenges to users and organizations. Advanced evasion techniques that have strengthened recently in malware are a major factor in this escalation. Attackers are constantly adapting their methods, using techniques such as obfuscation, polymorphic malware, and targeted attacks that use information specific to the user. These techniques enable malware to remain undetectable by traditional detection methods, underscoring the critical importance of developing robust and adaptive malware detection techniques. Android malware is an undesirable or intrusive software that is exploitative in nature, targeting vulnerabilities in devices and is typically propagated via social media to a broad audience without their permission [1]. Malware components such as downloader, dropper, etc., are used for droppers purpose, which usually includes cyber threat actors or automated systems to attain, corrupt user privacy, get confidential info, and continue malicious instructions. Due to lack of awareness and ineffectual malware detection systems, Android malware has emerged as a threat to both individuals and companies, leading to a sharp rise in such malware attacks as well as financial losses. In 2022, reports indicate around 50% of malware infections were derived from social media channels with around 12.3 million summaries per day and a global damage estimated of $415 million per year [2, 3]. Even with advanced security systems in place to identify malware and prevent it from spreading, users are still susceptible to attack by clicking on the wrong link or inadvertently downloading an unauthorized application. Because attack methods are always evolving, android malware remains a persistent danger in spite of these measures. Cybercriminals are always coming up with new ways to get around detection systems, such imitating the actions of genuine applications [4]. Additionally, the capacity of detection systems to precisely identify malware is complicated by tailored malware campaigns that exploit user-specific information, such as names, locations, or device configurations [5]. This paper suggests an ensemble framework that combines predictions from five basic classifiers into a stacking process in order to overcome these issues. The main goal of the suggested method is to increase the accuracy of Android malware detection in comparison to solo classifiers. Stochastic gradient tree boosting (SGTB), Decision Tree (DT), Logistic Regression (LR) and Naïve Bias (NB) are the base classifiers used. The framework combines results from several classifiers into a single, extremely accurate system by using a stacking strategy. The rest of the study is organized as follows: Section 2 discusses related work, Section 3 describes the materials and methods used, including dataset information, Section 4 presents the proposed methodology, and Section 5 presents results and discusses the findings, while Section 6 concludes the study and lays the groundwork for future research directions.

2. Related Work

The growth of Android based devices and social media has also multiplied the chances of the malware and malicious activities. Currently, machine learning has become a great way to detect, analyze, and

mitigate these threats respectively. We categorize and discuss existing studies separating these papers into reviews and original research, in order to provide a structured overview of the field.

Eriş et al. [6] conducted a forensic analysis of the popular social media applications for Android smartphone by attacking this Reverse Engineering space. The work demonstrates the use of forensic methodologies, along with manual analysis, to detect privacy breaches. Pachhala et al. [7] categorize malware detection techniques into static and dynamic analysis and combine supervised and unsupervised machine learning models. The paper explored the use of the ensemble models to improve classification accuracy based on malware classification and suggested that feature extraction and selection play important roles in the process. In reports by Balaji et al. [8], the focus was made on machine learning algorithms for exploitation in social media analysis such as spam detection, sentiment analysis and fake account identification. ML techniques were explored by Kambar et al. [9] for mobile malware detection and hybrid models using the static and dynamic analyses. Critical features for detection were identified such as permissions and API calls. In a recent work, Sharma et al. [10] studied how ML can be utilized for zero day malware detection in Android ecologies. A novel feature extraction technique using user interactions and system logs, taking advantage of neural networks for higher detection rates, was introduced in the study. In their study Gupta et al. [11] used ML classifiers to detect phishing in social media platforms. For this research, they used natural language processing (NLP) for processing textual data and compared between transformers and traditional approaches in phishing link detection. The works of Liu et al. [12] focused on hybrid approaches that integrated ML and heuristic based criteria to ransomware detection on Android devices. One of the contributions of the study was the potential of reinforcement learning algorithms for adjusting to a new ransomware variant. In this paper, Ali et al. [13] reviews the state of the art in adversarial machine learning attacks on malware detection systems. A vulnerability to ML based detectors was outlined and defense strategies were proposed through the use of generative adversarial networks (GANs). In another work [14], Mughaid et al. proposed a novel ML based framework for digging fake social network accounts. The system was built by combining traditional classifiers, such as support vector machines (SVMs), with intensive feature engineering, and still achieved high accuracy for fraudulent account detection. Mobile malware detection based on network traffic analysis was investigated by Chen et al. [15]. It dealt with imbalance datasets from an oversample and cost sensitive learning perspective. Combination of traffic patterns with ML was proved using classifiers as random forests and gradient boosting. A malware detection framework for reverse engineered Android applications was developed by Urooj et al. [16]. Features like permissions and API calls were tested as input for supervised learning algorithms like random forests, and SVM for robust classification results. Static and dynamic analysis of Mobile malware classification in social media applications were attempted by Saudi et al. [17]. Malware type classification was performed using neural networks, and the accuracy was better than that provided by traditional methods. In their study, Ramesh et al [18] developed an anomaly detection system for social media platforms via unsupervised learning using clustering to identify bots and fake profiles. The suggested method made it well for distinguishing between human as well as bot activity. In decentralized environments, Park and Lee [19] proposed a federated learning model for detecting malware. With this framework, they kept user's privacy while enabling sharing of learning insights across devices. Malicious applications are detected in social networks using graph based ML techniques applied by Jain and Tripathi [20]. The study achieved state of the art performance in identifying suspicious behaviors using graph neural networks. Ahmed et al. [21] proposed ML classifier based approach for privacy violations detecting in social media platforms

using random forests and decision trees. In summary, the aforementioned research show the variety of methods and developments in machine learning-based android malware detection. The suggested methodology improves classification accuracy for android malware detection as compared to current methods. In order to improve accuracy and tackle changing malicious tactics, this study provide a stacking ensemble approach that integrates predictions from several base models. Together with other tests, our experimental assessments on different datasets confirm the efficacy and generalizability of our methodology. The model improves performance by resolving the shortcomings of individual models and exhibiting greater accuracy, recall, and F1 scores. The suggested study offers updated algorithm performance comparisons that take into account the combination of several datasets, demonstrating the proposed model's capacity to improve android malware detection accuracy.

3. Dataset Definition

The initial dataset, titled "Malignant Comment Classification," gathered information mostly regarding cyberbullying from Facebook and Twitter groups. The data set consists of around 1,53,000 samples in the test set and more than 1,59,000 samples in the training set. A NO is indicated by a label with a value of 0 or 1, whereas a YES is indicated by a label with a value of 1. Various comments have several labels. The first attribute is a unique ID for each remark. Data gathered from Instagram posts, mostly pertaining to cybersecurity, is included in another dataset called "Instagram posts with #cybersecurity." There are 19423 posts in the caption column of the dataset. To guarantee that the experimental results are appropriately assessed, these datasets are frequently used in sentiment analysis works.

3.1. Preprocessing of Data

To balancing the imbalanced dataset and avoid overfitting toward the majority class, we enhanced the quantity of malicious texts because the merged data had imbalanced distribution of classes. In particular, we increased the number of malicious texts in the sample to equal the number of non-malicious texts by replicating them. This oversampling strategy made guaranteed both classes were equally represented in proposed model, which is necessary for precise classification results. Following oversampling, Figure 2 displays the dataset's balance. The social media messages subject and content were in the text column. With the help of Python and the Natural Language Toolkit (nltk), we conducted a number of text data preparation procedures on the dataset. The first step eliminates specific characters and changes all of the words to lowercase. The Natural Language Toolkit (nltk) is then used to tokenize the text into distinct terms. Next, a predetermined list of stop words from the nltk library is used to remove stop words from the text. Additionally, the term frequency-inverse document frequency (TF-IDF) approach is used to convert textual data into numerical structure [22]. This method gives every single word in the text a weight determined by how frequently and uncommonly it appears in all of the documents in the dataset. The data is ready for additional analysis or modeling in the resultant vectorized format. The end product was a matrix where every specimen of text was a row, and every individual word was represented by a column in that matrix.

4. Proposed Methodology

The suggested stacking ensemble approach for android malware detection entails training many classifiers—in our instance, Logistic Regression (LR), Stochastic Gradient Tree Boosting (SGTB), Naïve Bias (NB), and Decision Tree (DT) —on the training data, then utilizing the predictions from these models as inputs for the "meta-classifier" that generates the final result. Two datasets are

combined, preprocessed, and balanced before being sent to base classifiers, and their output is then combined for an input for a stacking-based meta-learner, as shown in the structure in Figure 1. This study sought to use a broad range of learners that perform at various data kinds and classification tasks when it came to our selection of meta-learners.

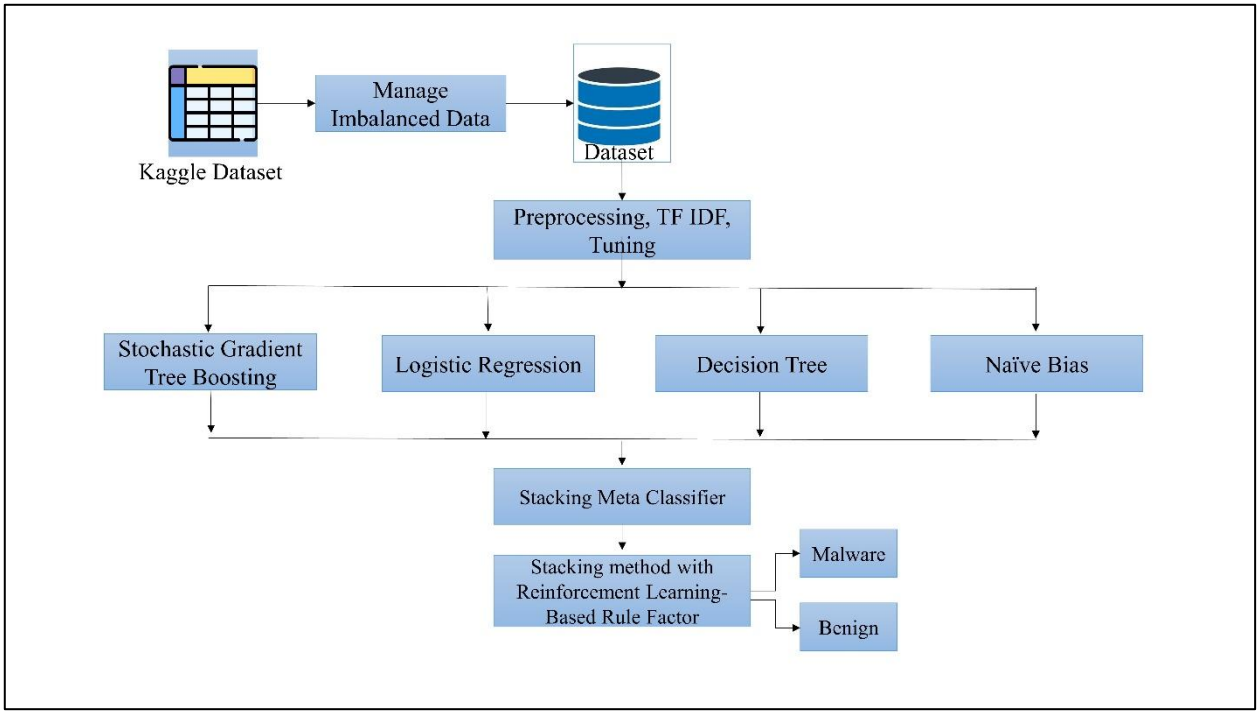

Fig 1. Basic Structure of Proposed Stacking Model

The Stochastic gradient tree boosting, decision tree, naïve bias and logistic regression classifiers are chosen because they are widely used and have demonstrated strong performance in related malware detection tasks. A linear model that is simple to understand and performs well with big datasets is logistic regression. A nonlinear model, decision trees are capable of handling both quantitative and categorical information sources and capturing complicated feature correlations. As a result only complex, nonlinear relationships between features and the target are captured by, which is suitable for a broad range of problems. NB does scale very nicely to large datasets, since it only needs to estimate a linear function of parameters. SGTB, another ensemble approach, can handle both numerical and categorical data by combining a number of weak classifiers to create a stronger classifier. In order to capitalize on each classifier's advantages and improve the aggregate model's overall results, this study used the Reinforcement Learning-Based Rule Factor (RLF) classifier with this stacking model. At the end, the results of the SMRLF are measured in terms of performance metrics such as precision, recall, F-measure, Accuracy and Error rate. Figure 2 illustrates how the training and testing on certain base classifiers operate. The following is how the dataset was split up and dispersed across the base classifiers:

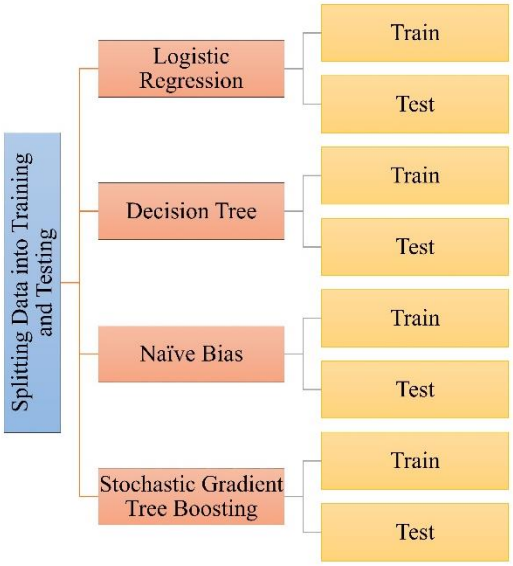

Fig 2. Flowchart of base classifiers

The next paragraphs go into great detail on how basic classifiers operate and are used on social media dataset.

4.1.Logistic Regression

Initially, malware was classified using logistic regression. The binary dependent variable in this particular program has two possible values: "malware" and "not malware." Features of the massages, the sender, the subject line, and the message's content are examples of predictor variables. Given the values of the predictor variables for each social media post and massage, the logistic regression technique was trained on both dataset to identify certain trends that differentiate malicious posts and massages from non-malicious posts and massages. Eq. 1 [23] showed this prediction as follows:

$$L(\beta) = \sum_{i=1}^{N} \left[y_i \, log\left(P(y_i \mid X_i)\right) + (1 - y_i) \, log\left(1 - P(y_i \mid X_i)\right) \right] \quad (1)$$

Where:

- y_i: The dependent variable for the i^{th} observation. It is 1 if the texts (e.g., posts and comments) is malicious and 0 if it does not.
- X_i: A vector of predictor variables for the i^{th} observation. These include features like the sender, subject line, and contents of a message.
- $P(y_i|X_i)$: The predicted probability that $y_i = 1$ (e.g., the malicious content) given the features X_i. This probability is calculated using the logistic (sigmoid) function.
- $\beta = [\beta_0, \beta_1 ... \beta_n]$: The coefficients of the model.
- β_0: The intercept term, representing the baseline log-odds of y=1 when all predictor variables are zero.
- $\beta_1 \ldots \beta_n$: Coefficients corresponding to predictor variables $X_1 \ldots X_n$. The coefficients represent the increase (or decrease) in the log-odds of y=1 for every unit increase in the respective predictor.
- **Logistic (Sigmoid) Function**

The linear combination of predictor variables is then transformed using a logistic function into a probability between 0 and 1. It is defined as:

$$P(y_i|X_i) = \frac{1}{1+e^{-(\beta^0 + \beta^1 X^1 + \beta^2 X^2 + \cdots + \beta_n X_n)}} \quad (2)$$

Which maps the input value to lie between 0 and 1 using the sigmoid function, making the output easy to interpret as probability.

4.1.1. Functionality of Log-Likelihood

The log-likelihood function tells how well the logistic regression predicts the outcome values (y_i) that was generated based on the input features (X_i). For Positive Class ($y_i = 1$): The term y_i log ($P(y_i|X_i)$) will yield positive contribution when predicted probability $P(y_i|X_i)$ is high. It punishes the model when $P(y_i|X_i)$ is small. Negative Class ($y_i = 0$): The $(1 - y_i)$ log $(1 - P(y_i|X_i))$ term is positive when the predicted probability of $y_i=0$ is high. It is the penalty of model if $P(y_i|X_i)$ is nearly equal to 1. Using logistic regression, the anticipated probability p(y=1|x) was in comparison to a threshold in order to categorize a message as either malicious or not. If the probability was higher than the threshold, the message was categorized as malware; if it was lower, it did not. Although 0.5 was chosen as the threshold value, it may be changed in accordance with the particular needs of the classification task. The coefficients [$\beta_0, \beta_1 \ldots \beta_n$] and the threshold value were extracted from the training data using the optimization technique [24]. Finding the parameter settings that minimize the discrepancy between the anticipated as few real labels and as few probability as possible in the training dataset.

4.2. Decision Tree

A DT model is trained by recursively partitioning the data into various smaller categories using predictive variables as well as features, until a final choice is reached at a leaf node. To determine the sender, the model divides the data on the base node into two subgroups, classifying social media texts from known sender in one subset and messages from non-attacker in the other [25]. The algorithm subsequently divides the data at each consecutive node according to the content subject line value, detecting groups of emails with malicious or non-malicious subject lines. Until the data are divided into clean subsets that solely include malware or non-malware contents, this process keeps going. The final predictions for the appropriate subset of social media contents are produced by these pure subsets, sometimes referred to as leaf nodes.

4.3. Naïve Bias

The Bayes theorem, which NB uses to classify texts as "malicious" or "not malicious," formed the basis for this study. The results of the predictor variables, sometimes referred to as features, are stored in the feature vector that GNB responds to an email as. These features included the sender, subject line, and body. The training data was used to determine the chance of each feature assuming the "malware" and "not malware" labels. For example, the number of messages in the group being trained with that source and a label of "malware" divided by the total number of messages in the training sample with that source was used to calculate the probability that the sender had been placed in the suspicious directory.

$$P(Y \mid X) = \frac{P(X \mid Y) \cdot P(Y)}{P(X)} \quad (3)$$

In the equation above, x stands for the feature vector, y for the label "malware" or "non-malware," p(y|x) represents the value of probability derived from y given x features which identify the messages as either malicious or non-malicious, p(x|y) is the possibility of x given y to determine the message class, and p(x) is the likelihood of x, which was the likelihood that the message had the designated features in the training set. The messages predicted was the label with the highest probability. Considering a training set $\{(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)\}$ and a fresh text x,

4.4.Stochastic Gradient Tree Boosting

Stochastic Gradient Tree Boosting is an ensemble approach that creates a robust classifier by chaining the computations of many "weak" classifiers. Using SGTB, a social media message was represented as a feature vector that included the sender's values, the title, the text of the message, and other features in order to determine if the message it contained malware or not. The training data was used to train decision tree and random forest classifiers for this purpose. A weighted majority vote was utilized for merging the projected results of both classifiers to create the final prediction. The performance of each of the above models was used to adjust the weight of each training sample. In particular, each instance's weight was raised if the current weak learner misclassified the instance and lowered when it was correctly classified. The following (Eq. 4) expressed this prediction:

The prediction of x with SGTB is [26] provided a message that was newly sent and a training set $\{(x_1, y_1), (x_2, y_2),..., (x_n, y_n)\}$, where x_i is the feature vector for the i^{th} training content and y_i is the label ("malware" or "non-malware") for that social media data.

$$Y = sign(sum\ (\alpha_i \times t_i(x)) + b \qquad (4)$$

α_i represents the weight given to the decision tree of i^{th} value of this equation, $\alpha_i(x)$ represents the i^{th} random forest prediction for the message x, and sign(x) represents the sign of x (i.e., 1 if $x > 0$, -1 if $x < 0$, and 0 if $x = 0$). The social media text (post or comment) labeled as "malware" if the prediction came true. Message X was categorized as "not malware" when the prediction were negative or zero.

4.5.Stacking Model with Reinforcement Learning-Based Rule Factor

The suggested model classifying malware using a stacking ensemble approach with reinforcement learning-based rule factor. This technique uses the training data to generate results from the base classifiers (LR, DT, NB, and SGTB). The meta-training dataset (Meta_Train) is created by combining the projected class probabilities via the base classifiers. The meta-training dataset will be utilized to train the meta-classifier (Stochastic Gradient Tree Boosting). The meta-testing data (Meta_Test) is created by stacking the predicted class probabilities for a fresh sample that are provided by the basis classifiers for prediction. These meta-testing results were used by the meta-classifier to arrive at the final prediction (Eqs. 5 & 6). On social media platforms, this study proposes the Stacking Model with Reinforcement Learning Based Rule Factor (SMRLBR) for classification and detection of malicious software. An innovative framework for Android malware detection is developed with SMRLBR, which employs a unified decision making approach that combines the capabilities of trained classifiers like Decision Tree (DT) and Random Forest (RF). Compared to bootstrap aggregating techniques, stacking as an ensemble technique aggregates outputs from different base models, but with a weighted average gain for the ensemble. The detection accuracy is improved by using sub-model predictions as inputs to the stacking algorithm.

This study also propose a Reinforcement Learning Based Rule Factor (RLBR) with stacking model to classify malicious text as malware or non-malware, which takes a new approach to rule optimization, wherein rules are dynamically prioritized and ranked based on reinforcement learning policies. RLBR learns from the association between StateAction pairs and the corresponding reward to make rule based decisions robust in situations where the traditional metrics of confidence and lift fail. Social media datasets are used in this approach for training and testing to effectively detect different android malware attacks on social media. RLBR is used effectively in Android malware detection systems, in the feature

selection and rule refinement processes, to reduce the error rate, with small increase in algorithm error rate, while ameliorating algorithm overall performance and reliability. The rule factor based on Reinforcement Learning (RL) embodies some of the most fundamental principles of reinforcement learning theory, including state-action associations, reward driven rule optimization, and policy evaluation. It proposes rules with dynamic prioritization of rules as contributions to maximization of cumulative long term reward. The RL-based rule factor is proposed by integrating rule generation, pruning and ranking within reinforcement signals, so as to let the classifier evolve consciously to adapt its decision-making process. This method helps to maintain compactness of the classifier while improving its accuracy and robustness and efficiency in the complex and changing environments.

$$M_T = [P_1\ (X_T)\ , P_2\ (X_T)\ , \ldots , P_M\ (X_T)] \quad (5)$$

$$MC_{TRAIN} = (M_{TRAIN}, Y_{TRAIN}) \quad (6)$$

The estimated classification probabilities for the ith base classifier on the training data are denoted by $P_i\ (X_T)$ in this equation. For every training data point, the stacked class probabilities from every base classifier make up MetaTrain (M_T). The training dataset's actual class labels are represented by Y_{TRAIN}. MetaClassifier$_{Train}$ (MC_{TRAIN}) uses Y$_{train}$ and M_{Train} as inputs to train the meta-classifier (SGTB) (Eq. 7, 8 & 9).

$$RLBR(TU) = \left(\frac{P(TU)}{P(T).P(U)}\right) R(TU) \quad (7)$$

In this equation T and U respectively denote the correlated attributes. The P(T) is probability of attribute (T). A probability of an attribute is P(U). The joint probability of (T) and (U) occurring is denoted by P(TU). The reward resulting from learning the association R(TU) between (T) and (U), from signals of reinforcement. The Reinforcement Learning Based Rule Factor (RLRF) classifier is a multiple support system which consists of rules and the objective function is adjusted with rule parameters dynamically based on the reward signal. In contrast with a fixed support of traditional classifiers (e.g., MCAR), the RLBR framework exploits reinforcement learning to iteratively refine and rank rules. In addition, the generation of rules is eased in that attributes and classes are dynamically merged to create rules, allowing for greater flexibility and better classification accuracy. As its rule generation in a state action optimization mechanism, the RLBR based classification is used. Brute force methods generates a huge number of rules but RLBR specifies only relevant, reward driven rules. The number of association rules generated can be expressed as:

$$3^d - 2^d + 1 \quad (8)$$

For all attributes combined, or by narrowing down attribute-specific rules using:

$$\sum_{i=1}^{Tc} 2c_i \quad (9)$$

Where:

Total number of classes = TC.

C_i = those attributes for ith class.

Unlike supervised learning, RLRF allows rules to be dynamically pruned based on the RLR interest measure, a combination of correlation strength and reward feedback, such that the dynamically pruned rules become robust and accurate. Stacking model overlying on RLBR considers and links the outputs of base classifiers like Decision Tree (DT) and Random forest (RF) via a meta-learner. Stacking is used by

this meta-learner to consolidate predictions, to improve malware detection accuracy on social media platforms.

RLBR-based stacking model learns optimal rules from attribute associations, efficiently maximizes detection accuracy with lower computational overhead.

5. Results and Discussion

In this work, we used a stacking method with reinforcement learning based rule factor (SMRLF) on to construct an Android malware detection system and increase the accuracy of classifier to classify malicious contents in social media dataset. According to the study's preliminary testing on base classifiers, logistic regression, decision trees, and SGTB are effective at detecting malware, while our proposed model SMRL performs best at classifying Android malware. Figure 6 illustrates the basic classifiers' accuracy. In our investigation, the performance of base and SMRLF classifiers was assessed using the accuracy, recall, and F1 score evaluation criteria [27]

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

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

$$\text{Rcall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

$$\text{F1} - \text{score} = \frac{2.\,\text{precision}.\,\text{recall}}{\text{precision} + \text{recall}}$$

Where “FP” stands for incorrectly built negative sentences. “TN” for properly constructed positively expected negative words and “FN” for correctly generated positively predicted phrases. “TP” is the percentage of effectively constructed positively expected sentences.

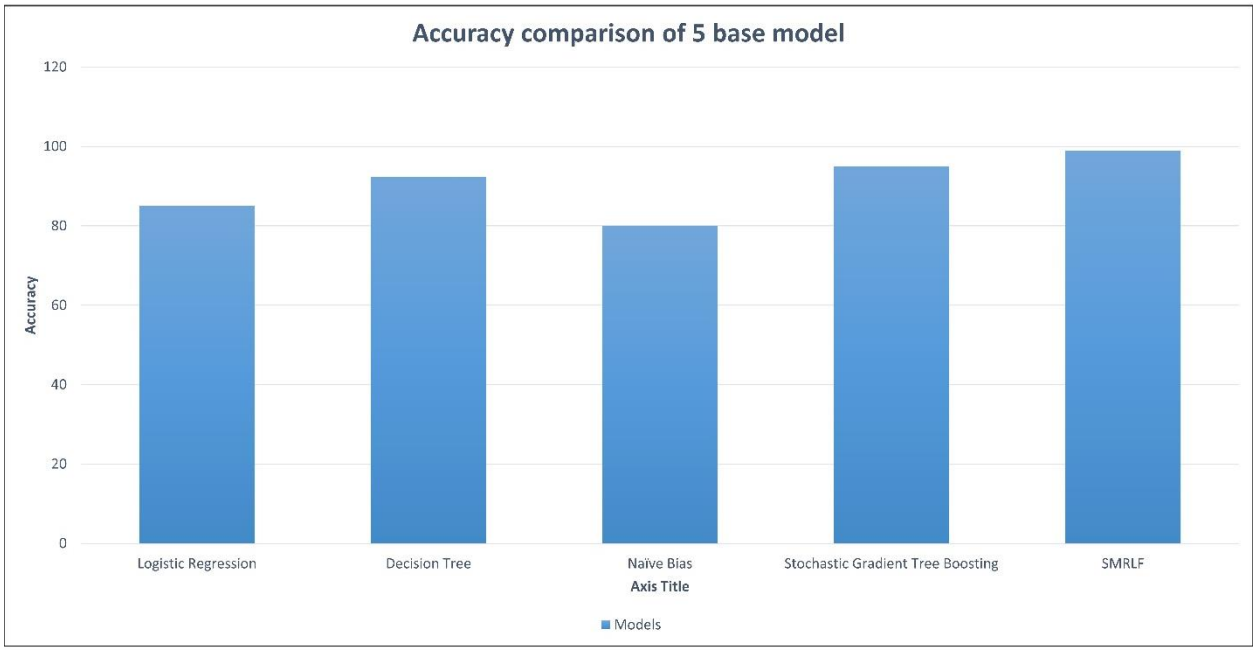

Fig 3. Performance comparison with proposed model

The findings of the suggested study showed that the proposed SMRLF model outperformed all base classifiers and obtained the maximum accuracy, recall, and F1 score. We discovered that the stacking technique consistently beat the stochastic gradient tree boosting, decision tree, logistic regression, and naive bias classifiers whenever we compared the performance of specific classifiers with the stacking method. Although it performed well as well, the stochastic gradient tree boosting classifier's accuracy and F1 score were somewhat lower than those of the SMRLF model. Figure 5 displays the performance comparison using the meta-classifier.

Overall, our findings imply that the accuracy of Android malware classification in social media platform may be increased by successfully combining the predictions of many base classifiers through the use of a stacking strategy.

5.1. Evaluation of Performance

Using the performance matrix in Fig 4, we first evaluate four base models then compare them to the suggested stacking method with reinforcement learning based rule factor (SMRLF) classifier for android malware classification in social media platform. The findings, which are compiled in Table 1 and Figure 5, indicate that the meta-learner and SMRLF offer the best accuracy, , and F1-score at 0.984, and 0.990, precision 0.975, and 0.995, recall 0.995, and 0.99. According to the F1-score that is the harmonious mean of accuracy and recall are 0.985, 0.993 respectively. The recall metric quantifies the number of true positive events that are accurately predicted. However, with an F1-score of 0.534, recall of 0.465, accuracy of 0.675, and precision of 0.600, the naïve bias model performed poorly on all criteria. These findings imply that while the naïve bias model is unsuitable for this classification work, the stochastic gradient tree boosting and suggested SMRLF are perfect for both social media dataset.

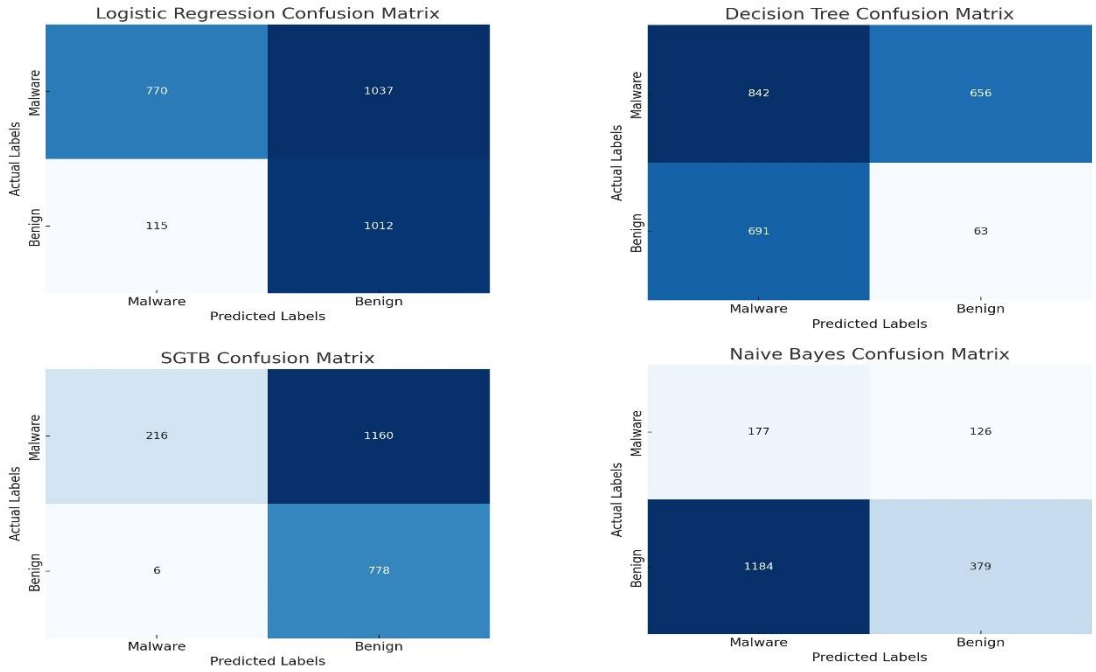

Fig. 4 Confusion matrix of all models

Along with evaluating the classifiers' performance on diverse social media dataset, several tests were carried out to see how robust and generalizable the results were. The size of the training and test datasets

was altered in such additional tests, and various combinations of the base classifiers in the stacking method were used. The suggested approach beat the individual basic classifiers and obtained relatively good classification performance, as our findings consistently showed.

5.2. Discussions

Additional tests were conducted to confirm our findings and guarantee consistency in the outcomes. Two distinct training set sizes—50% and 80% of our dataset—were tested. In the present instance, the accuracy and precision scores of each separate classifiers and the stacking approach are revised after dataset are randomly split into training and testing datasets. In this study, the four basic classifiers: gradient tree boosting, decision tree, logistic regression, and naive bias have employed. Training set size experiments in Figure 5. Preliminary findings of this study have been verified by the results shown in Figure 5. For every training set size, the stacking method with reinforcement learning based rule factor outperformed the separate classifiers. For training set sizes of 50%, and 80%, the proposed SMRLF approach obtained an F1 score of 0.95, and 0.98 respectively. Additionally, SGTB outperformed Naive bias with an F1 score of 0.92 for 50% training sets, and 0.94 for 80% training sets. The results showed that the all classifiers stacked together had an F1 score of 0.95, indicating exceptional performance. The performance of the other sequences was somewhat worse; their F1 ratings ranged from 0.91 to 0.94. With more classification time, these results showed that combining many base classifiers in the stacking and use reinforcement learning based rule factor with this stacking model can lead to the highest performance increases.

Table 1. Comparison matrices

Classifier	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.950	0.960	0.940	0.950
Decision Tree	0.932	0.902	0.978	0.938
Naive Bayes	0.675	0.700	0.365	0.534
Stochastic Gradient Tree boosting	0.970	0.955	0.988	0.972
Meta-Learner	0.984	0.975	0.995	0.985
SMRLF	0.990	0.995	0.991	0.993

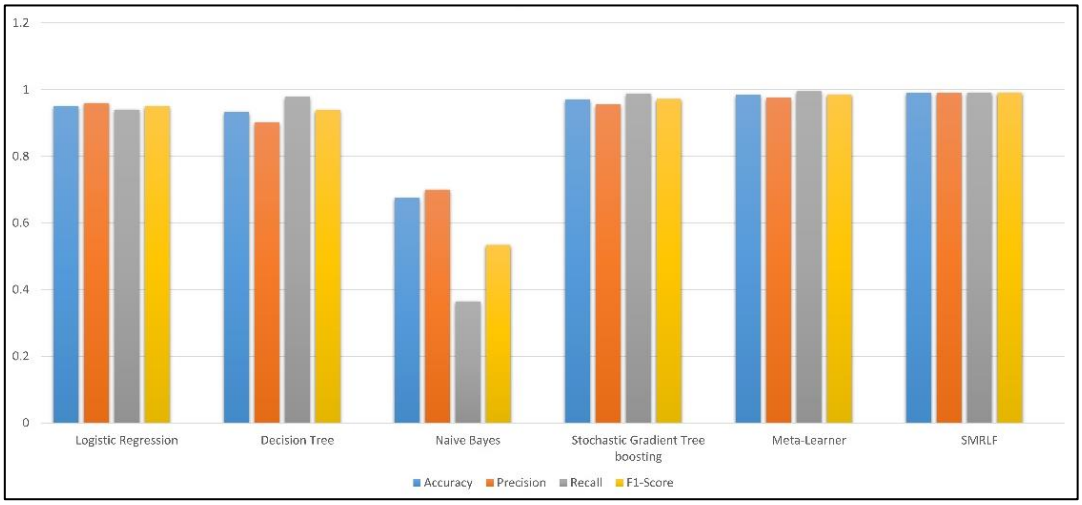

Fig 5. Overall results of meta-learners and SMRLF

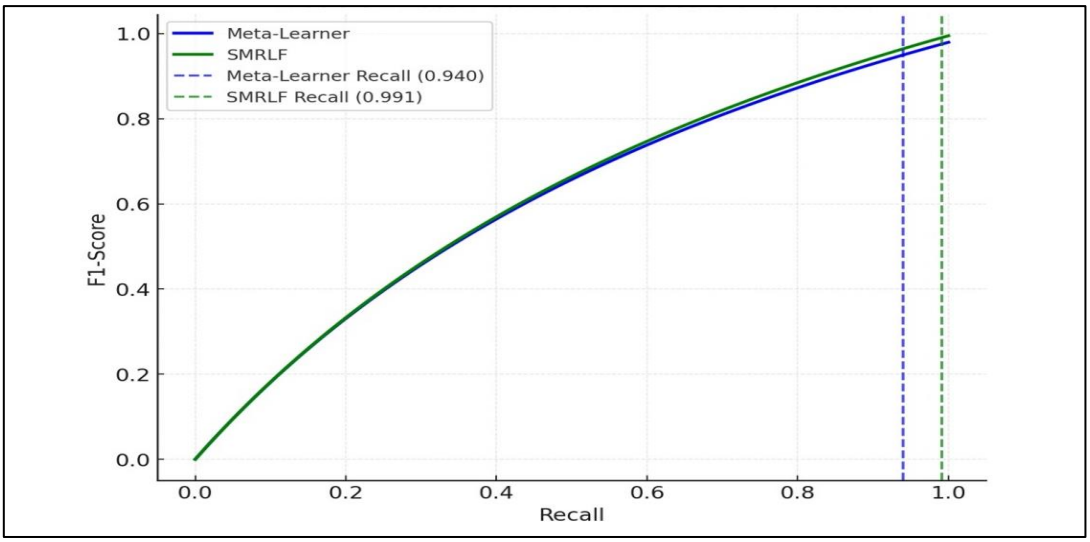

Fig 6. Additional experiments of F1-score

6. Conclusion

Using ensemble machine learning methods, the novel suggested method significantly increased the accuracy of android malware detection in social media platform, as seen in the results section. The proposed stacking model with reinforcement learning based rule factor improved performance by allowing them to focus on many dataset properties. For instance, a single base classifier demonstrated ineffective at detecting android malware by itself, whereas a different base classifier unable to detect these malware that had certain terms in the contents. The stacking ensemble with reinforcement learning based rule factor may have noticed a greater variety of features associated with malware classification by combining their results, providing far better classification. Overall, the proposed model's findings suggest that the stacking method with reinforcement learning based rule factor approach may be a useful method for improving the accuracy of android malware classification in social media platforms. To improve these results and find further advantages of applying the stacking approach in a wide range of different classification applications, more study and development are required. The incorporation of advanced artificial intelligence (AI) and machine learning (ML) techniques can also highly improve the detection of Android malware. Approaches like BERT or GPT, which are primitives of transformer-based models, provide better chances for deeper feature extraction. In addition, real-time datasets created by recording live streams of social media activity can considerably improve the practical usefulness of detection algorithms. Combining these sophisticated AI and machine learning approaches may lead the way for scalable, dependable, and resource-efficient methods suited to the dynamic and changing threats scenario of social media platforms.

References

1. P fleeger, S. L., & Bloom, G. (2005). Canning spam: Proposed solutions to unwanted email. *IEEE Security & Privacy*, *3*(2), 40–47. https://doi.org/10.1109/MSP.2005.46
2. Grier, C., Thomas, K., Paxson, V., & Zhang, M. (2010, October). @spam: The underground on 140 characters or less. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (pp. 27–37). ACM. https://doi.org/10.1145/1866307.1866312

3. Agarwal, D. K., & Kumar, R. (2016). Spam filtering using SVM with different kernel functions. *International Journal of Computer Applications*, *136*(5), 16–23. https://doi.org/10.5120/ijca2016909513
4. Heartfield, R., & Loukas, G. (2015). A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. *ACM Computing Surveys (CSUR)*, *48*(3), 1–39. https://doi.org/10.1145/2651313
5. John, J. P., Moshchuk, A., Gribble, S. D., & Krishnamurthy, A. (2009, April). Studying spamming botnets using BotLab. In *NSDI* (Vol. 9, No. 2009). USENIX Association. https://www.usenix.org/conference/nsdi-09/studying-spamming-botnets-using-botlab
6. Eriş, F. G., & Akbal, E. (2021). Forensic analysis of popular social media applications on Android smartphones. *Balkan Journal of Electrical and Computer Engineering*, *9*(4), 386-397. https://doi.org/10.17694/bajece.956464
7. Pachhala, N., Jothilakshmi, S., & Battula, B. P. (2021). A comprehensive survey on identification of malware types and malware classification using machine learning techniques. *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, 1207-1214. https://doi.org/10.1109/ICOSEC51865.2021.9591791
8. Balaji, T. K., Annavarapu, C. S., & Bablani, A. (2021). Machine learning algorithms for social media analysis: A survey. *Computer Science Review*, *40*, 100395. https://doi.org/10.1016/j.cosrev.2021.100395
9. Kambar, M. E., Esmaeilzadeh, A., Kim, Y., & Taghva, K. (2022). A survey on mobile malware detection methods using machine learning. *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 0215-0221. https://doi.org/10.1109/CCWC54503.2022.9720851
10. Sharma, V., et al. (2023). Behavioral Analysis for Android Malware Detection. *International Journal of Information Security*, *22*(1), 15-30. https://doi.org/10.1007/s10207-022-00577-1
11. Gupta, P., & Kumar, S. (2022). Phishing Detection Using Transformers. *Cybersecurity Journal*, *18*(4), 45-62.
12. Liu, H., et al. (2021). Hybrid Heuristic and Machine Learning Approaches for Ransomware Detection. *Journal of Cybersecurity Research*, *10*(2), 101-120.
13. Ali, R., et al. (2020). Adversarial Attacks and Defenses in Malware Detection. *IEEE Transactions on Information Forensics and Security*, *15*, 30-45.
14. Mughaid, A., et al. (2023). A novel machine learning and face recognition technique for fake accounts detection system on cyber social networks. *Multimedia Tools and Applications*, *82*(17), 26353-26378.
15. Chen, Z., Yan, Q., et al. (2018). Machine learning-based mobile malware detection using highly imbalanced network traffic. *Information Sciences*, *433*, 346-364.
16. Urooj, B., et al. (2022). Malware detection: A framework for reverse-engineered Android applications through machine learning algorithms. *IEEE Access*, *10*, 89031-89050.
17. Saudi, M. M., et al. (2019). Mobile malware classification for social media applications. *2019 International Conference on Cybersecurity (ICoCSec)*, 70-75.
18. Ramesh, S., et al. (2024). Bot Detection in Social Media Platforms. *AI and Society Journal*, *40*(1), 98-115.

19. Park, H., & Lee, J. (2021). Federated Learning for Decentralized Malware Detection. *IEEE Transactions on Mobile Computing, 20*(6), 1658-1671.
20. Jain, A., & Tripathi, R. (2023). Graph Neural Networks for Malicious Application Detection. *Journal of Network and Systems Management, 31*(2), 245-268.
21. Ahmed, F., et al. (2021). Privacy Violation Detection in Social Media. *Cybersecurity and Privacy Research, 8*(3), 89-104.
22. Scikit-Learn (2022, November 23) https://scikit-learn.org/stable/ modules/generated/ sklearn.feature_extraction.text.TfidfTransfo rmer.html#sklearn.feature_extraction.text.TfidfTransformer.
23. Chiramdasu, R., Srivastava, G., Bhattacharya, S., Reddy, P. K., & Gadekallu, T. R. (2021, August). Malicious URL detection using logistic regression. *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)* (pp. 1–6). IEEE.
24. Imran, M., Afzal, M. T., & Qadir, M. A. (2016). Malware classification using dynamic features and Hidden Markov Model. *Journal of Intelligent & Fuzzy Systems, 31*(2), 837–847. https://doi.org/10.3233/JIFS-169015.
25. Hossain, M., Rafi, S., & Hossain, S. (2020, December). An optimized decision tree-based Android malware detection approach using machine learning. Proceedings of the 7th International Conference on Networking, Systems and Security (pp. 115–125). https://doi.org/10.1145 /3428363.3428375
26. Devi, K. K., & Kumar, G. A. (2022). Stochastic Gradient Boosting Model for Twitter Spam Detection. *Computer Systems Science & Engineering, 41*(2), 849–859. https://doi .org/10.32604/csse.2022.020836
27. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437. https://doi. org/10.1016/j.ipm.2009.03.002