FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 9 No. 3, March, 2025, pp 284 - 293
DOI: https://doi.org/10.33003/fjs-2025-0903-3235

COMBATTING BANKING MALWARE THREATS: EVALUATING THE EFFICACY OF HYBRID AND SINGLE-CLASSIFICATION ALGORITHMS

***[1]Dauda, S., [2]Ahmad, M. A., [2]Aliyu, A. A., [2]Ibrahim, M., [1]Abdulkadir, S., [1]Ahmed, A. M., [1]Mukhtar, A. S. and [1]Bello, S.**

[1]Department of Informatics, Kaduna State University, Kaduna – Nigeria.
[2]Department of Secure Computing, Kaduna State University, Kaduna – Nigeria.

*Corresponding authors' email: slmandauda@yahoo.com

ABSTRACT

The increasing sophistication and prevalence of banking malware pose significant challenges to cybersecurity, threatening the confidentiality, integrity, and availability of financial systems and user data. This study evaluates the efficacy of hybrid and single-classification algorithms in detecting banking malware, addressing a critical gap in existing research. A total of eight classification algorithms were analyzed, including three hybrid models—Stacked Ensemble with Gradient Boosting, AdaBoost, and Stacking with Decision Trees and Random Forest. Additionally, five single classifiers—Support Vector Machine (SVM), Decision Tree, k-NN, Random Forest and Logistic Regression were assessed. The research methodology incorporated principal component analysis (PCA) for feature selection and techniques like Adasyn and Tomek Link to address data imbalance. Classification performance was evaluated using key metrics: accuracy, precision, recall, and F1-score. Results demonstrated that hybrid models, particularly an ensemble combining Random Forest and Decision Tree, outperformed other classifiers, achieving superior accuracy (0.98), precision, and recall. While Gradient Boosting and AdaBoost also exhibited robust performance, Logistic Regression showed room for improvement in precision and recall metrics. This research highlights the effectiveness of hybrid classification models in enhancing the detection of banking malware and underscores their potential for strengthening cybersecurity defenses in financial systems. The study contributes to the growing literature on machine learning applications in malware detection and provides insights into the strengths and limitations of diverse classification algorithms.

Keywords: Accuracy, Banking malware, Cybersecurity, Classification algorithms, Hybrid models, Single classifiers

INTRODUCTION

As the term implies, malware refers to software that, when executed maliciously, carries out damaging activities against a victim as intended by its creator. Typically, these programs are designed to disable, disrupt, or destroy computer systems or networks (Zolkipli et al., 2010). The threat posed by malware is steadily increasing, owing to the advancements in communication networks and computer technology (Rieck et al., 2008). In today's digital landscape, malware and other cyber threats are pervasive, leading to a wide range of misconduct, including fraudulent internet activities, invasion of personal information, data theft, and various forms of cybercrime. These incidents expose fundamental vulnerabilities and flaws in software platforms, raising concerns about cybersecurity, with a particular emphasis on malware detection (Pawlicka et al., 2023). Malware has evolved into a profit-oriented endeavor, with creators employing techniques to make it as stealthy and undetectable as possible (Pawlicka et al., 2023). These malware programs are crafted by skilled programmers with a deep understanding of digital forensic techniques, making forensic analysis challenging and complex. The more vulnerable a technology or system is, the greater the risk of exploitation by malware. The sophistication and adaptability of malware contribute to its longevity and pose significant risks to end-users and organizations (Fuhr et al, 2022). However, there exist infection mechanisms and user behaviors that can mitigate these risks.

Internet users encounter malware threats on a daily basis, especially in the field of digital forensics. Malware attacks have become increasingly profit-driven, organized by illicit entities, and executed with a low profile and targeted approach. The proliferation of malware takes on various forms and continues to expand (Pawlicka et al., 2023). Given the widespread use of the internet and networks, the likelihood of encountering malware is amplified, as it exploits the effectiveness of these technologies. Many malware distributions require user permission for access, making user awareness a significant factor in infection risk. Regrettably, many internet users are unaware of security threats and remain oblivious to the presence of malware on their devices (Aboaoja et al., 2022). Moreover, they lack the necessary knowledge to secure their information systems, further elevating the risks associated with malware (Aslan & Samet, 2020).

Targeted cybercrime focuses on compromising the integrity, confidentiality, and availability of networks and systems (Aboaoja et al., 2022). Unauthorized access, often referred to as "hacking," is a prominent example of such cybercrimes. Various methods are employed, including deceiving individuals into providing credentials, using computational power to crack passwords, or exploiting software vulnerabilities. Criminals can employ malicious software or malware to gain remote access to a victim's computer. It is essential to note that creating and distributing such malware is illegal (Aslan & Samet, 2020). Malware can grant unauthorized access to a computer through pre-programmed backdoors or by utilizing remote administrative tools (RATs), enabling remote control over computer systems. Certain features, such as webcams and microphones, can be remotely activated and deactivated (Pawlicka et al., 2023). Additionally, malware controlled by other computers can become part of a botnet, possessing substantial computing power and capable of initiating distributed denial-of-service

(DDoS) attacks. As the malware problem continues to escalate, various approaches have been developed to mitigate the threat and safeguard systems. However, these efforts face significant challenges due to the advanced tactics employed by hacking communities and the lucrative nature of their illicit activities (Aboaoja et al., 2022). This issue remains a prominent topic of debate, necessitating attention in the banking system and contributing to the existing literature on malware intrusion detection systems.

In light of these observations, this research proposes an exploration of eight distinct classification algorithms. This diversified selection comprises three hybrid models, three single classifiers, and one artificial neural network (ANN). The motivation behind this broader selection is to facilitate a comprehensive evaluation of these algorithms, enabling us to discern their relative strengths and weaknesses. The inclusion of hybrid models is particularly significant, as previous research has indicated their potential to outperform certain single algorithms. Additionally, the work seeks to address a recognized concern from the benchmark study: the issue of data imbalance within the malware dataset. Data imbalance can significantly hinder classification accuracy, and thus, this research will employ techniques to mitigate this challenge.

Classification methods

From a machine learning perspective, malware detection can be viewed as either a classification or clustering problem. The objective is to cluster unknown malware types into distinct groups based on specific properties determined by the algorithm. Conversely, when a model is trained on a comprehensive dataset of malicious and benign files, the problem can be simplified to classification. For known malware families, the challenge becomes primarily a matter of classification. With a limited set of predefined classes, each malware sample can be accurately assigned to the appropriate class. This approach tends to yield more accurate results compared to clustering algorithms.

XGBoost (XGB)

The XGBoost (eXtreme Gradient Boosting) algorithm, which implements the gradient boosting method introduced by Chen & Guestrin (2016), is a widely employed and versatile tool. It empowers tree-boosting algorithms to attain advanced classification and high efficiency levels (Ma et al., 2020). In XGBoost, a set of regression trees collectively generates the final outcome. The final score is computed using the formula provided below:

$$\hat{y} = \sum_{h=1}^{H} g_h(x) \quad (1)$$

In this equation, there are H trees, and each tree has leaf scores denoted as K. Another advantage of XGBoost is its resilience to multicollinearity. By employing XGBoost, model performance can be maximized. However, XGBoost requires careful parameter tuning to avoid issues like overfitting and excessive misunderstanding, which can be challenging due to the numerous parameters it utilizes. To optimize the hyperparameter values, we applied the grid search approach along with cross-validation.

AdaBoost

The AdaBoost approach involves using weighted duplicates of the training data over multiple rounds to build classifiers, denoted as n = 1,..., N. This process adjusts the weight distribution, D_n, which determines the significance of samples in the dataset for classification, by assigning greater weight to samples that were incorrectly classified. The training set consists of samples (x_1, y_1), (x_2, y_2),..., where x_i falls within a specific domain space X. The label set, Y = 1, +1, encompasses all labels, y_i. D_n represents the weight of this distribution on training sample i in round n.

$$\varepsilon_n = \sum_{i:h_n(x_i)\neq y} D_n(i) \quad (2)$$

Initially, all weights are set to be equal. As each round progresses, the vulnerable learner is compelled to prioritize the challenging samples in the training set, as the weights of incorrectly classified samples increase. The objective of the weak learner is to find a weak hypothesis, denoted as $h_{n()} = X_1, + 1$, that aligns with the distribution D_n. The error rate of a weak hypothesis serves as an indicator of its quality.

Random Forest (RF)

Popular machine learning (ML) techniques like Random Forest (R.F.) are employed for data classification. This method finds applications in various fields, including investment (Jabeur, 2017), physical security (Ozigis et al., 2020; Kaminska, 2018), and marketing research (Salminen et al., 2019). Random Forest relies on multiple trees to enhance robustness, aggregating the mean prediction value after each tree's creation. Each tree is constructed using a randomly chosen subset of input variables. The expression for the estimated model is as follows:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} g_i(x) \quad (3)$$

The set of random trees from the kth learner is denoted as g(x), with x representing the vector of input features. The Random Forest (R.F.) final estimate is derived by averaging the results from all the trees, each contributing to the estimate with specific weights. Yesilkanat (2020) asserts the superiority of the R.F. algorithm over other machine learning techniques. This superiority arises from the R.F.'s ability to generate trees using random methods and autonomously collect training data from subsets. The R.F. approach accomplishes training through bootstrapping on randomly selected independent subsets of datasets, which effectively mitigates overfitting.

Support Vector Machine (SVM)

In the realm of binary classification, Support Vector Machines (SVM) were developed to create or define an ideal hyperplane that effectively separates a dataset into distinct classes (Hagedoorn et al., 2017). SVM, a rapidly evolving field within machine learning, encompasses various concepts, including VC theory, statistical modeling, maximum margin optimal hyperplanes, kernel methods, and other innovative ideas. This sets it apart from more commonly used empirical risk reduction techniques like neural network models. SVMs, which stand for Support Vector Machines, represent supervised learning approaches for both regression and classification tasks. These methods have found successful applications across a wide range of classification and machine learning scenarios, with their foundations laid in 1996 by pioneers such as Vapnik, Harris Drucker, and others.

Decision Trees (DT)

The C4.5 Decision Tree Classification Method, developed by Ross Quinlan, represents an improvement over the ID3 algorithm. In the realm of machine learning, these classifiers construct decision trees based on data samples. They employ an information gain measure to select a variable as the dataset for the tree and utilize an edge-based segmentation approach to build decision tree models. In this research, a trial is conducted with n outcomes that partition the training samples and dataset L into subgroups (L_1, L_2, L_3, ..., L_n). If P is any collection of samples, then |P| represents the total number of samples in P, and (C_i, P) signifies the total number of samples in P belonging to category C_i. The entropy of set P is defined as follows:

$$infor(P) = -\sum_{i=0}^{k} \frac{freq(C_i,P)}{|P|} Log_2 \left(\frac{freq(C_i,P)}{|P|}\right) \quad (4)$$

Once L is partitioned based on the outcomes of a specific feature with respect to z, we can calculate the overall knowledge content of L. By employing computational information, we can estimate the information content of L (L). The total information content of L is equal to the weighted sum of the entropies of each subgroup.

Logistic regression (LR)

Logistic regression models are employed to uncover relationships between categorical and other factors. In many studies, the outcome variable is binary, typically denoted as 1 for the occurrence of a specific event and 0 for its absence. Using a set of predictors, the logistic regression model computes the probability of an event happening. The logistic regression model's predicted outcome can be expressed as follows: While L_i ranges between 0 and 1, Z_i represents a linear combination of the input variables and can span from negative to positive values. Logistic regression encounters various statistical challenges in practice.

$$L_1 = \ln\left(\frac{P_1}{1-P_1}\right) = \mathbb{Z}_i = \beta X_i + \mu_i \quad (5)$$

Two common issues in this context are multicollinearity and reduced performance accuracy. According to Jabeur (2017), logistic regression (L.R.) typically removes most of the input variables that exhibit strong correlations with the analysis's outcome. He emphasized that achieving the best results doesn't always involve using the maximum likelihood estimator.

Malware Classification using Machine learning

The increasing use of internet-enabled devices has heightened vulnerability to cyberattacks, prompting the development of advanced malware detection methods (Enem and Awujoola, 2023). Traditional machine learning techniques, while effective, often suffer from long processing times and are being overshadowed by deep learning, which reduces the need for manual feature engineering. Recent research highlights the effectiveness of machine learning and deep learning in malware detection and classification.

Liu et al. (2017) proposed a machine learning-based system for malware analysis, achieving 98.9% accuracy in classifying unknown malware and detecting 86.7% of new threats. Kedziora et al. (2019) focused on Android malware detection, finding that Support Vector Machines (SVM) and Random Forest (RF) were the most effective algorithms, with accuracy rates up to 80.7%. Vinayakumar et al. (2019) explored deep learning's potential to eliminate manual feature engineering, demonstrating its superiority over traditional methods, especially in zero-day malware detection.

He and Kim (2019) highlighted the limitations of traditional Malware Detection Systems (MDS) and proposed a Convolutional Neural Network (CNN)-based approach, which proved resilient to API injection attacks. Alzaylaee et al. (2020) introduced DL-Droid, a deep learning method for Android malware detection, achieving up to 99.6% accuracy by combining dynamic and static features. Qiu et al. (2020) systematically addressed challenges in applying deep learning to Android malware detection, emphasizing the importance of code semantics and feature extraction.

Singh and Singh (2021) analyzed machine learning-based malware detection systems, highlighting the challenges in creating effective classifiers. Usman et al. (2021) proposed a big data forensics approach to preemptively detect malicious IP addresses, demonstrating its effectiveness in mitigating zero-day attacks. Kazi et al. (2022) focused on detecting Zeus malware using machine learning, with Random Forest achieving 97% accuracy in identifying both old and new variants.

MATERIALS AND METHODS

This study aims to bridge the gap by employing a diverse set of classifiers on the malware dataset. The classifiers include three hybrid models: Ensemble with Gradient Boosting, AdaBoost, and staking Random Forest and Decision Tree.. Additionally, five single classifiers, namely Support Vector Machine (SVM), Decision Trees (DT), K-Nearest Neighbors (k-NN), Random Forest (RF) and Logistic Regression (LR) will be used.

Before proceeding with classification, a feature selection technique such as principal component analysis will be applied to the dataset. Furthermore, to address class imbalance issues, the dataset will be pre-processed using techniques like Adasyn and Tomek link. Following data preprocessing, the dataset will be split into training and testing subsets for evaluation.

The step towards achieving the research objectives commenced with a comprehensive grasp of the research context and its potential advantages. The procedure for implementing the machine learning algorithm for an intrusion detection system focused on identifying banking malware within network traffic aimed at customers' accounts. The workflow of the research methodology, depicted in Figure 1, illustrates the path towards realizing the research objectives and goals. This path comprises four essential steps to facilitate the process:

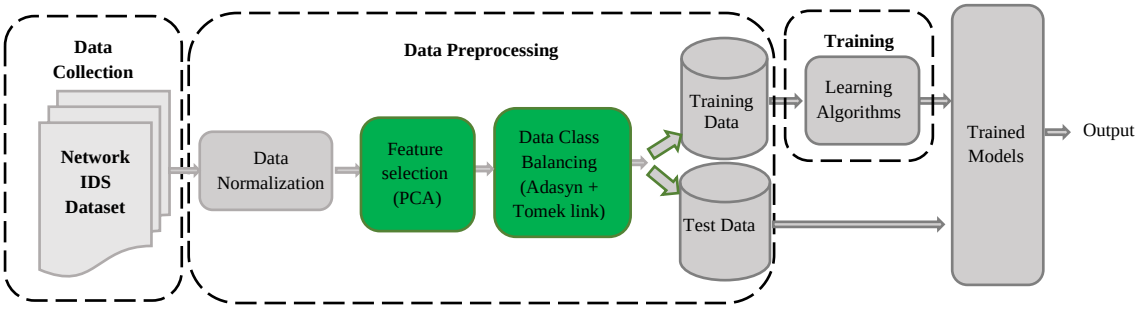

Figure 1: Proposed methodology Flow

System Description

Figure 1 illustrates the flow of the proposed methodology. While various approaches have been employed by researchers, this work seeks to adopt the benchmark method outlined by Kazi et al. (2019) as a foundation while identifying gaps that require further exploration. Furthermore, this study sources an open-source dataset from the University of Irvine, California repository website, which will serve as the input for the algorithms under evaluation during performance assessment.

Dataset Preprocessing

The first crucial step involves preprocessing the malware dataset. This preprocessing phase aims to prepare the data for subsequent analysis. The following steps are part of the dataset preprocessing:

Normalization: The dataset will undergo normalization to ensure that all features are on the same scale. This is essential for machine learning algorithms to perform optimally, as features with different scales can lead to biased model outcomes.

Feature Selection

Feature selection techniques will be employed to identify the most relevant and informative features. Two potential methods under consideration are Principal Component Analysis (PCA) and Correlation-Based Feature Selection. PCA reduces dimensionality while preserving important information, while Correlation-Based Feature Selection identifies features that are highly correlated with the target variable. However, only one will be selected for this study.

Class Balancing

Addressing class imbalance is critical in malware classification. Two class balancing techniques, namely Adasyn and Tomek link, will be applied to ensure a balanced distribution of samples from different classes. Adasyn generates synthetic samples for the minority class to balance the dataset, while Tomek link identifies and removes overlapping samples from different classes.

Dataset Splitting

Once preprocessing is complete, the dataset will be divided into two subsets: a training set and a testing set. The training set will be used to train machine learning models, while the testing set will be employed to evaluate their performance.

With these preprocessing steps in place, the dataset will be ready for classification using various machine learning algorithms, and their performance will be rigorously assessed.

Evaluation Method

The evaluation method is a pivotal component of any research or study, as it serves as the compass guiding the assessment of hypotheses, methodologies, and the ultimate success of a project. In the context of this research, the evaluation method takes center stage as we aim to measure the performance, accuracy, and efficacy of the machine learning models employed in the classification of malware (Awujoola et al., 2021). An effective evaluation method is essential in ensuring the credibility and reliability of the research findings. It provides a structured framework for systematically analyzing the outcomes and drawing meaningful conclusions. In this section, we delve into the core principles of the evaluation method that will be applied in our study.

The primary objective of the evaluation method is to answer critical questions related to the performance of the machine learning models in classifying malware. We seek to determine how well these models can discern malicious software from benign files, and whether they can effectively address the challenges posed by evolving threats in the cybersecurity landscape.

Metrics

To achieve these objectives, this work employed a range of evaluation metrics and techniques, each tailored to specific aspects of model performance. These metrics will encompass measures of accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (AUC-ROC). Additionally, we will explore the use of confusion matrices and visualizations to gain deeper insights into model behaviour as shown in Table 1

Table 1: Confusion Matrix for binary classification

Parameter	**Prediction**	
	Malware	**Benign**
Malware	True Positive	False Positive
Benign	False Negative	True Negative

The evaluation metrics are calculated using the confusion matrix, from which the values are generated. These variables serve as performance indicators for the model. IDS is assessed based on the following common standards:

True Positive Rate (TPR): This quantity is determined by dividing all attacks by the share of attacks that have been properly anticipated. If every incursion is identified, the TPR is 1, which is especially fantastic for an IDS. The Detection Rate (DR) or Sensitivity are other names for TPR. The TPR is mathematically represented in equation 6 (Awujoola et al., 2023):

$$TPR = \frac{TP}{FN+TP} \quad (6)$$

False Positive Rate (FPR): This statistic is calculated by dividing the total dollar amount of its occurrences by the percentage of incident types that are incorrectly classified as assaults.

$$\frac{Sensitivity}{FPR} = \frac{FP}{FP+FN} \quad (7)$$

False Negative Rate (FNR): False negatives arise while a detector incorrectly classifies an aberration as regular rather than detecting it. The FNR can be mathematically stated as shown in equation 8.:

$$\frac{Recall}{FNR} = \frac{FN}{FN+TP} \quad (8)$$

Classification Rate (CR): The CR measures how well the IDS can distinguish among standard and surprising traffic conduct. It is described as the proportion of activities to all times wherein the forecasts had been correct:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (9)$$

Finally the F1 score can be identified as follows:

$$F\,1Score = 2 * \frac{Precision * Sensitivity}{Precision + Sensitivity} \quad (10)$$

The accuracy rate (ACC) is derived using the total sample size (FN + FP + TP + TN) and the sum of all classified observations (TP + TN). This allows us to assess the classification model's estimate result as 1 when the actual value of a class is 1 and as 0 when the true value of a class is 0. ACC can be determined using the following formula:

$$ACC = \frac{(TP + TN)}{(FN + FP + TP + TN)} \quad (11)$$

Area Under Curve: Additionally, using a confusion matrix, we can calculate sensitivity and specificity rates. AUC values between 0 and 1 imply a more accurate model when they are closer to 1. When the area under the ROC curve is substantial and the distributions of T.N. and T.P. do not overlap, it indicates that the classes have been sufficiently separated (Mai and Liao, 2019). Finally, we compare the areas under the curves using a suggested method.

RESULTS AND DISCUSSION

Results Analysis

In this section, the evaluation results of eight classification algorithms, including an ensemble, are presented. Tables 2 through 9 depict the classification reports obtained from the evaluations of these algorithms, while Figures 2 through 10 visualize their confusion matrix and ROC.

Table 2 shows the classification report of the eight evaluated machine learning models without class balancing technique with Api calls malware Dataset

Table 2: Precision, Recall and FI-Score Values of the Evaluated Machine Learning Models without Balancing Technique

S/N	Classifier	Benign Instances			Malicious Instances			Overall			Accuracy
		Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	
1	GB	0.95	0.97	0.96	0.96	0.95	0.96	0.96	0.96	0.96	0.96
2	AB	0.92	0.93	0.93	0.93	0.92	0.93	0.93	0.93	0.93	0.93
3	Stacking(RF + DT)	0.98	0.99	0.98	0.99	0.98	0.98	0.98	0.98	0.98	0.98
4	SVM	0.99	1.00	0.99	0.99	0.98	0.98	0.99	0.99	0.99	0.99
5	DT	0.98	0.99	0.98	0.99	0.98	0.98	0.98	0.98	0.98	0.98
6	KNN	0.95	1.00	0.97	1.00	0.95	0.97	0.97	0.97	0.97	0.97
7	RF	0.99	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00
8	LR	0.90	0.60	0.54	0.90	0.60	0.54	0.90	0.60	0.54	0.51

Tables 2 illustrate the classification reports resulting from the evaluation of eight algorithms. The results obtained from the analysis of the eight evaluated machine learning models on the API calls malware dataset, without the application of class balancing techniques, provide insights into the performance of these models in handling imbalanced data. The metrics used to assess the models include precision, recall, F1-score, and overall accuracy for both benign and malicious instances. These results highlight the strengths and weaknesses of each model under unbalanced conditions.

Among the models, the Random Forest (RF) classifier demonstrated the highest performance across all metrics, achieving an overall accuracy of 1.00. This result indicates that RF effectively handles the dataset's imbalance without requiring class balancing techniques. Similarly, the Support Vector Machine (SVM) and Decision Tree (DT) classifiers performed exceptionally well, both achieving an overall accuracy of 0.99 and 0.98, respectively. These models exhibited strong precision and recall scores for both benign and malicious instances, showcasing their reliability in distinguishing between the two classes.

The Stacking ensemble classifier also performed remarkably, with an overall accuracy of 0.98. This model displayed balanced precision, recall, and F1-scores for both classes, suggesting its robustness in combining the strengths of multiple classifiers to improve classification outcomes. The Gradient Boosting (GB) and Adaptive Boosting (AB) classifiers achieved slightly lower accuracies of 0.96 and 0.93, respectively. These models maintained consistent performance across both benign and malicious instances, although their results indicate a marginal decline in effectiveness compared to the top-performing models.

The K-Nearest Neighbors (KNN) classifier achieved an overall accuracy of 0.97. Despite its relatively high accuracy, the performance was slightly affected by the imbalanced data, as seen in its precision and recall scores for benign and malicious instances. The Logistic Regression (LR) classifier, however, displayed the weakest performance, with an overall accuracy of 0.51. This result reflects significant challenges in distinguishing between the two classes, as evidenced by its lower precision, recall, and F1-scores.

Overall, the results suggest that ensemble-based models, such as RF and Stacking, and algorithms like SVM, are better suited for handling imbalanced datasets without requiring class balancing techniques. Conversely, simpler models like LR struggle under these conditions, emphasizing the importance of model selection when addressing class imbalance challenges in machine learning

Confusion Matrix

Table 3 shows the results of the confusion matrix for eight machine learning models evaluated with with Api calls malware Dataset without class balancing techniques

Table 3: Confusion Matrices for the Evaluated Machine Learning Models without Balancing Technique

S/N	Classifier	True Positives	False Positives	True Negatives	False Negatives	Total Instances
1	Gradient Boosting	10349	373	10150	527	21399
2	AdaBoost	10022	700	9791	886	21399
3	Stacking (RF+ DT)	10586	136	10462	215	21399
4	Support Vector Machine	10707	15	10579	98	21399
5	Decision Tree	10592	130	10461	216	21399
6	K Nearest Neighbor	10707	15	10095	582	21399
7	Random Forest	10698	24	10607	70	21399
8	Logistic Regression	9617	1105	1193	9484	21399

The confusion matrices obtained for the eight evaluated machine learning classifiers on the API calls malware dataset without class balancing techniques reveal key insights into the performance of these models in distinguishing between benign and malicious instances as shown in Table 2. Each classifier's true positives, false positives, true negatives, and false negatives highlight their strengths and weaknesses in

correctly identifying the respective classes under imbalanced data conditions.

The Random Forest classifier achieved the best performance, with 10,698 true positives and 10,607 true negatives, and only 24 false positives and 70 false negatives. This indicates a remarkable ability to correctly classify both benign and malicious instances, with minimal misclassifications. Similarly, the Support Vector Machine displayed exceptional results, with 10,707 true positives, 10,579 true negatives, 15 false positives, and 98 false negatives. These metrics reflect its robustness and precision in handling the dataset's imbalance.

The Stacking ensemble classifier, which combines the strengths of Random Forest and Decision Tree models, also performed well, recording 10,586 true positives, 10,462 true negatives, 136 false positives, and 215 false negatives. The Decision Tree classifier, analyzed individually, achieved comparable results with 10,592 true positives, 10,461 true negatives, 130 false positives, and 216 false negatives. These results suggest that ensemble-based methods and tree-based algorithms are effective in this context.

The Gradient Boosting classifier displayed solid performance, with 10,349 true positives and 10,150 true negatives, but it recorded a higher number of false positives (373) and false negatives (527) compared to the top-performing models. Similarly, the K-Nearest Neighbor classifier achieved 10,707 true positives and 10,095 true negatives, with 15 false positives and a higher count of false negatives (582), suggesting some difficulty in managing the dataset's imbalance.

AdaBoost showed relatively moderate performance, with 10,022 true positives, 9,791 true negatives, 700 false positives, and 886 false negatives. This reflects its limitations in distinguishing between the classes compared to other ensemble models. Logistic Regression, however, demonstrated the weakest performance, with 9,617 true positives and only 1,193 true negatives. It also recorded the highest number of false positives (1,105) and false negatives (9,484), indicating significant challenges in handling the imbalanced dataset.

The confusion matrices underscore the superiority of ensemble-based and tree-based models in managing imbalanced data without the application of class balancing techniques. Conversely, simpler models like Logistic Regression struggle to achieve accurate classification under these conditions. The results emphasize the importance of selecting robust classification algorithms for applications involving imbalanced datasets.

Table 4 provides a summary of the total number of correct classifications and misclassification for each of the eight machine learning models. For the ensemble models, the stacked Random Forest and Decision Tree model has the highest number of correct classifications followed by the Gradient Boosting and AdaBoost models. For the single classifiers, the Random Forest model has the highest number of correct classification and lowest number of misclassified instances, followed by the Support Vector Machine model then the Decision Tree, K Nearest Neighbor and Logistic Regression models.

Table 4: Summary of Correct and Misclassified Instances of Banking Malware

S/N	Classifier	Correct Classifications	Misclassifications
1	Gradient Boosting	20499	900
2	AdaBoost	19813	1586
3	Stacking (RF+ DT)	21048	351
4	Support Vector Machine	21286	113
5	Decision Tree	21053	346
6	K Nearest Neighbor	20802	597
7	Random Forest	21305	94
8	Logistic Regression	10810	10589

Generally, the Random Forest and Support Vector Machine models have the highest number of correctly classified instances of 21,305 and 21,286 instances respectively for the analysis conducted using API calls malware dataset with Adasyn and Tomek Link data balancing technique.

These insights derived from the confusion matrices offer valuable guidance for refining and optimizing classification algorithms for detecting banking malware. By understanding the patterns of correct and incorrect classifications, cybersecurity professionals can tailor strategies to enhance the accuracy and reliability of malware detection systems.

Classification Results

The results in Table 5 show the performance of the eight machine learning models. These algorithms were evaluated based on precision, recall, and F1-score for detecting banking malware using the "API Calls Malware Dataset.

Table 5: Precision, Recall and FI-Score Values of the Evaluated Machine Learning Models with Balancing Technique

S/N	Classifier	Benign Instances			Malicious Instances			Overall			Accuracy
		Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	
1	GB	0.96	0.95	0.96	0.95	0.97	0.96	0.96	0.96	0.96	0.96
2	AB	0.93	0.92	0.93	0.92	0.93	0.93	0.93	0.93	0.93	0.93
3	Stacking (RF+DT)	0.99	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.98	0.98
4	SVM	1.00	0.99	0.99	0.99	1.00	0.99	0.99	0.99	0.99	0.99
5	DT	0.99	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.98	0.98
6	KNN	1.00	0.95	0.97	0.95	1.00	0.97	0.97	0.97	0.97	0.97
7	RF	1.00	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00
8	LR	0.90	0.89	0.89	0.89	0.90	0.89	0.89	0.89	0.89	0.89

The classification report presented in Table 5 highlights the performance metrics of eight machine learning classifiers evaluated using precision, recall, and F1-score for detecting benign and malicious instances with class balancing techniques applied to the dataset. Gradient Boosting exhibited consistent performance, achieving a precision, recall, and F1-score of 0.96 across benign and malicious instances, resulting in an overall accuracy of 96%. Similarly, AdaBoost demonstrated reliable outcomes with an overall accuracy of 93%, supported by balanced precision, recall, and F1-score values of 0.93 for both benign and malicious instances.

The stacking ensemble model, combining Random Forest and Decision Tree, displayed superior performance with an overall accuracy of 98%. It achieved precision, recall, and F1-scores of 0.98 and 0.99 for benign and malicious instances, respectively, reflecting the robustness of ensemble techniques in enhancing classification outcomes. Support Vector Machine achieved near-perfect results with an overall accuracy of 99%. It demonstrated precision and recall values of 1.00 and 0.99, respectively, for benign instances, while attaining 0.99 and 1.00 for malicious instances.

Decision Tree yielded an accuracy of 98%, with consistent precision, recall, and F1-scores of 0.98 and 0.99 across benign and malicious instances. The K-Nearest Neighbor classifier showed satisfactory performance, achieving an overall accuracy of 97%. It attained a precision of 1.00 and a recall of 0.95 for benign instances, while achieving a precision of 0.95 and a recall of 1.00 for malicious instances, indicating slight variations in class predictions. Random Forest emerged as the top-performing classifier, achieving a perfect accuracy of 100%. It consistently scored 1.00 across precision, recall, and F1-score for both benign and malicious instances, demonstrating its capability to handle the balanced dataset effectively.

Logistic Regression, while comparatively less effective, achieved an overall accuracy of 89%. It demonstrated consistent precision, recall, and F1-scores of 0.89 and 0.90 for benign and malicious instances, respectively. These results suggest that while Logistic Regression is functional, its performance is relatively limited compared to other classifiers. Overall, the application of class balancing techniques significantly enhanced the performance of the machine learning models, with Random Forest and Support Vector Machine standing out as the most effective classifiers in this evaluation.

Confusion matrix

The confusion matrix results obtained from the experiment with the class balancing technique.

Table 6: Confusion Matrices for the Evaluated Machine Learning Models with the class balancing Technique

S/N	Classifier	True Positives	False Positives	True Negatives	False Negatives	Total Instances
1	Gradient Boosting	164	119	13	10673	10969
2	AdaBoost	125	158	33	10653	10969
3	Staking (RF + DT)	192	122	70	10585	10969
4	Support Vector Machine	128	155	2	10684	10969
5	Decision Tree	181	102	83	10603	10969
6	K Nearest Neighbor	129	154	11	10675	10969
7	Random Forest	170	113	8	10678	10969
8	Logistic Regression	135	111	6	10717	10969

The results of the confusion matrices in Table 6 provide insight into the classification performance of the evaluated machine learning models after applying class balancing techniques. The metrics, including true positives, false positives, true negatives, and false negatives, reveal the models' ability to correctly identify benign and malicious instances while minimizing errors.

Gradient Boosting displayed a relatively balanced performance, with 164 true positives and 119 false positives, alongside 13 true negatives and 10,673 false negatives. This indicates the model effectively identified some malicious instances but struggled with benign detection, as reflected by the higher number of false negatives.

AdaBoost exhibited a slightly lower true positive count of 125 and a higher false positive count of 158. It recorded 33 true negatives and 10,653 false negatives, suggesting moderate performance with a tendency to misclassify a higher proportion of benign instances as malicious.

Staking (RF + DT) performance with 192 true positive. Balanced performance with relatively high true negative of 70 and fewer false negative of 10,585 compared to others.

Support Vector Machine achieved 128 true positives and 155 false positives while maintaining one of the lowest false negative counts, with 10,684 instances. Its true negative count stood at 2, indicating its ability to correctly classify benign instances was limited but effective in identifying malicious ones.

The Decision Tree model had the highest true positive count at 181, coupled with 102 false positives and 83 true negatives. However, it recorded 10,603 false negatives, showcasing strong benign detection at the cost of some misclassifications in malicious instances.

The K-Nearest Neighbor classifier demonstrated a true positive count of 129 and a relatively low false positive count of 154. Its true negatives were recorded at 11, with false negatives totaling 10,675. This performance highlights its capability to balance benign and malicious instance classification, though it still exhibited a notable number of misclassifications.

Random Forest emerged as one of the most consistent models, with a true positive count of 170 and only 113 false positives. It achieved a very low false negative count of 10,678 and recorded 8 true negatives, showcasing its effectiveness in both benign and malicious classification tasks.

Logistic Regression emerged with relatively low false positive of 111 with the weakness of high false negatives of 10,717 and moderate of true positives of 135.

The confusion matrices reveal that Random Forest performed most effectively, with a well-balanced capability to minimize false positives and false negatives. Other models, such as Support Vector Machine and Gradient Boosting, showed specific strengths in certain aspects but faced challenges in achieving a balanced classification. The class balancing techniques contributed to improving the models' performance by addressing prior imbalances, though the effectiveness varied across classifiers. Summary of Correct and Misclassified Instances of Banking Malware is presented in table 7

Table 7: Summary of Correct and Misclassified Instances of Banking Malware

S/N	Classifier	Correct Classifications	Misclassifications
1	Gradient Boosting	177	10792
2	AdaBoost	158	10811
3	Staking (RF + DT}	262	10707
4	Support Vector Machine	130	10839
5	Decision Tree	264	10705
6	K Nearest Neighbor	140	10829
7	Random Forest	178	10791
8	Logistic Regression	241	10828

The summary of correct and misclassified instances for the evaluated machine learning models highlights their ability to distinguish between benign and malicious cases of banking malware after applying data class balancing techniques. The results reflect varying degrees of accuracy and misclassification across the classifiers.

Gradient Boosting achieved 177 correct classifications, but it misclassified 10,792 instances, indicating that while the model performed relatively well in identifying some cases correctly, it struggled with a significant proportion of errors. Similarly, AdaBoost demonstrated 158 correct classifications, with a higher count of 10,811 misclassifications. This outcome suggests that AdaBoost, although functional, faced challenges in accurately separating the two classes.

Stacking (Random Forest + Decision Tree) Closely followed with 262 correct classifications and 10,707 misclassifications. Support Vector Machine correctly classified 130 instances while misclassifying 10,839. This performance reflects a strong focus on identifying one class over the other, potentially at the expense of overall balance in the classification task. In contrast, the Decision Tree classifier demonstrated a higher count of correct classifications, reaching 264, with 10,705 misclassifications. This suggests that Decision Tree achieved a better balance in detecting both classes compared to several other models.

The K-Nearest Neighbor classifier achieved 140 correct classifications but exhibited 10,829 misclassifications. This performance highlights its limitations in accurately identifying a substantial portion of the instances. Random Forest achieved one of the higher correct classification counts, with 178 correctly identified instances and 10,791 misclassified cases. Its performance indicates a more consistent ability to differentiate between classes, although misclassifications remained notable.

While the summary underscores variations in the classifiers' abilities, Decision Tree and Random Forest stood out with comparatively higher correct classification counts, showcasing better adaptability to the balanced data. Gradient Boosting, Support Vector Machine, and K-Nearest Neighbor struggled with significant misclassification rates, reflecting areas where further optimization may enhance their performance. Logistic Regression moderately performed with 241 correct classifications, performing better than many models but with slightly higher misclassifications of 10,828. These findings emphasize the importance of selecting suitable algorithms and fine-tuning them to improve classification accuracy for banking malware detection tasks.

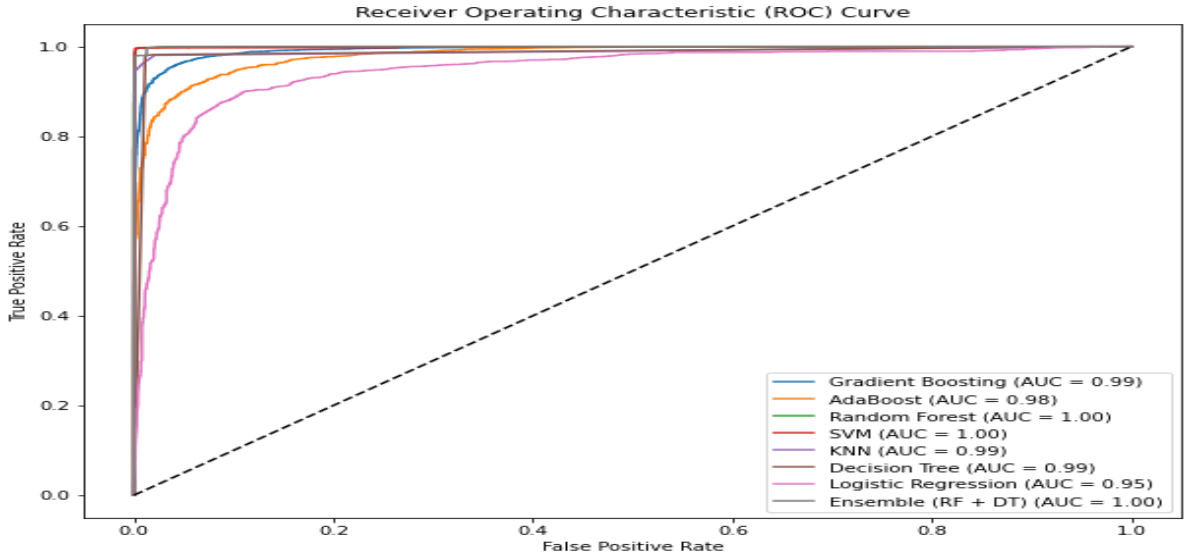

Figure 2: Receiver Operating Characteristic (ROC) curves

The analysis of the performance of multiple machine learning models, as illustrated by their confusion matrices and Receiver Operating Characteristic (ROC) curves, provides valuable insights into their effectiveness in classification tasks. The models under consideration include K-Nearest Neighbors (KNN), Decision Tree, Logistic Regression, Ensemble (Random Forest + Decision Tree), Gradient Boosting, Random Forest, and Support Vector Machine (SVM). Each model demonstrates varying levels of accuracy, precision, and robustness, as reflected in their respective confusion matrices and AUC scores.

The KNN model exhibits a strong performance with a high number of true positives (10,094) and true negatives (10,707). It maintains a minimal false positive rate, with only 15 instances, but has a moderate false negative count of 583. This suggests that while the model effectively identifies most

positive cases, there is room for improvement in minimizing missed classifications. Similarly, the Decision Tree model performs well, achieving 10,461 true positives and 10,592 true negatives. Although it records slightly more false positives (130) compared to KNN, it reduces the number of false negatives to 216, indicating a better balance between precision and recall.

Logistic Regression, in contrast, struggles to match the performance of the other models. It has a significantly higher number of false positives (1,105) and false negatives (1,193), resulting in lower overall accuracy. This model's predictive capability appears weaker, which is reflected in its performance metrics. On the other hand, the ensemble model combining Random Forest and Decision Tree offers robust results with 10,462 true positives and 10,586 true negatives. Its false positive and false negative rates, at 136 and 215 respectively, are comparable to those of the standalone Decision Tree model, suggesting that the ensemble approach can enhance reliability while maintaining precision.

Gradient Boosting emerges as a competitive model, delivering a commendable performance with 10,349 true positives and 10,373 true negatives. However, it records a relatively higher number of false positives (373) and false negatives (527) compared to some other models, which may limit its suitability in scenarios requiring strict error minimization. Among all the models analyzed, Random Forest stands out as the top performer. With 10,607 true positives and 10,698 true negatives, it achieves minimal errors, recording only 70 false positives and 98 false negatives. This exemplary performance highlights its robustness and reliability in classification tasks.

SVM also ranks as one of the best-performing models, with 10,579 true positives and 10,602 true negatives. Its false positive and false negative rates, at 15 and 98 respectively, align closely with those of the Random Forest model. Furthermore, the ROC curve analysis reinforces these observations, with Random Forest and SVM achieving perfect AUC scores of 1.0, indicating their superior classification capabilities. Gradient Boosting, KNN, and Decision Tree also perform well, with AUC values of approximately 0.99, while Logistic Regression lags slightly with a comparatively lower AUC.

CONCLUSION

This study demonstrates the efficacy of hybrid and single-classification algorithms in detecting banking malware, with a particular emphasis on hybrid models. The ensemble approach, combining Random Forest and Decision Tree, emerged as the most robust classifier, achieving superior accuracy, precision, recall, and F1-score. The findings underscore the importance of employing advanced classification techniques and addressing data imbalance to enhance malware detection in cybersecurity applications. Future research could explore the integration of deep learning architectures, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), to capture complex malware behavior patterns. Additionally, extending the dataset with real-time malware samples and investigating adversarial resilience in detection models could further strengthen banking malware detection systems.

REFERENCES

Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-Rimy, B. A. S., Eisa, T. A. E., & Elnour, A. A. H. (2022). Malware detection issues, challenges, and future directions: A survey. *Applied Sciences, 12*(17), 8482.

Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep learning-based android malware detection using real devices. *Computers & Security, 89*, 101663.

Angelo Oliveira. (2019). Malware analysis datasets: API call sequences. *IEEE Dataport.* https://dx.doi.org/10.21227/tqqm-aq14

Aslan, Ö. A., & Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access, 8*, 6249–6271.

Awujoola, O. J., Ogwueleka, F. N., Irhebhude, M. E., & Misra, S. (2021). Wrapper based approach for network intrusion detection model with combination of dual filtering technique of resample and SMOTE. In *Artificial Intelligence for Cyber Security: Methods, Issues and Possible Horizons or Opportunities* (pp. 139-167). Cham: Springer International Publishing.

Chen, T., & Guestrin, C. (2016, August). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794).

Etaher, N., & Weir, G. (2014, June). Understanding the threat of banking malware. In *Cyberforensics 2014-International Conference on Cybercrime, Security & Digital Forensics.*

Enem, T. A., & Awujoola, O. J. (2023). Malware detection and classification using embedded convolutional neural network and long short-term memory technique. *Science World Journal, 18*(2), 204-211.

Fuhr, J., Wang, F., & Tang, Y. (2022). MOCA: A network intrusion monitoring and classification system. *Journal of Cybersecurity and Privacy, 2*(3), 629–639.

Gauthama Raman, M. R., Somu, N., & Mathur, A. P. (2019). Anomaly detection in critical infrastructure using probabilistic neural network. In *Applications and Techniques in Information Security: 10th International Conference, ATIS 2019, Thanjavur, India, November 22–24, 2019, Proceedings 10* (pp. 129–141). Springer Singapore.

Guo, Q. Y., Zhang, S. J., Liu, H., Wang, C. L., Wei, F. L., Lv, T., ... & Liu, D. X. (2011). Three-dimensional evaluation of upper anterior alveolar bone dehiscence after incisor retraction and intrusion in adult patients with bimaxillary protrusion malocclusion. *Journal of Zhejiang University SCIENCE B, 12*, 990–997.

Guyon, I., & Elisseeff, A. (2006). An introduction to feature extraction. In *Feature extraction: Foundations and applications* (pp. 1–25). Springer Berlin Heidelberg.

Hagedoorn, T. R., & Spanakis, G. (2017, November). Massive open online courses temporal profiling for dropout prediction. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 231–238). IEEE.

He, K., & Kim, D. S. (2019, August). Malware detection with malware images using deep learning techniques. In *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE*

International Conference on Big Data Science and Engineering (TrustCom/BigDataSE) (pp. 95–102). IEEE.

Jabeur, S. B. (2017). Bankruptcy prediction using partial least squares logistic regression. *Journal of Retailing and Consumer Services, 36*, 197–202.

Kamińska, J. A. (2018). The use of random forests in modelling short-term air pollution effects based on traffic and meteorological conditions: A case study in Wrocław. *Journal of Environmental Management, 217*, 164–174.

Kazi, M. A., Woodhead, S., & Gan, D. (2019, November). Comparing and analysing binary classification algorithms when used to detect the Zeus malware. In *2019 Sixth HCT Information Technology Trends (ITT)* (pp. 6–11). IEEE.

Kazi, M. A., Woodhead, S., & Gan, D. (2022). An investigation to detect banking malware network communication traffic using machine learning techniques. *Journal of Cybersecurity and Privacy, 3*(1), 1–23.

Kedziora, M., Gawin, P., Szczepanik, M., & Jozwiak, I. (2019). Malware detection using machine learning algorithms and reverse engineering of Android Java code. *International Journal of Network Security & Its Applications (IJNSA), 11*.

Kim, T., Kang, B., Rho, M., Sezer, S., & Im, E. G. (2018). A multimodal deep learning method for Android malware detection using various features. *IEEE Transactions on Information Forensics and Security, 14*(3), 773–788.

Liu, L., Wang, B. S., Yu, B., & Zhong, Q. X. (2017). Automatic malware classification and new malware detection using machine learning. *Frontiers of Information Technology & Electronic Engineering, 18*(9), 1336–1347.

Mathur, A., Podila, L. M., Kulkarni, K., Niyaz, Q., & Javaid, A. Y. (2021). NATICUSdroid: A malware detection framework for Android using native and custom permissions. *Journal of Information Security and Applications, 58*, 102696.

Mathur, M. K., Verma, A. K., Makwana, G. E., & Sinha, M. (2013). Study of opportunistic intestinal parasitic infections in human immunodeficiency virus/acquired immunodeficiency syndrome patients. *Journal of Global Infectious Diseases, 5*(4), 164.

Mitchell, T. M. (1997). Does machine learning really work? *AI Magazine, 18*(3), 11.

Ozigis, M. S., Kaduk, J. D., Jarvis, C. H., da Conceição Bispo, P., & Balzter, H. (2020). Detection of oil pollution impacts on vegetation using multifrequency SAR, multispectral images with fuzzy forest and random forest methods. *Environmental Pollution, 256*, 113360.

Pawlicka, A., Pawlicki, M., Kozik, R., & Choraś, M. (2023). What will the future of cybersecurity bring us, and will it be ethical? The hunt for the black swans of cybersecurity ethics. *IEEE Access.*

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., & Xiang, Y. (2020). A survey of Android malware detection with deep neural models. *ACM Computing Surveys (CSUR), 53*(6), 1–36.

Rieck, K., Holz, T., Willems, C., Düssel, P., & Laskov, P. (2008, July). Learning and classification of malware behavior. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 108–125). Springer Berlin Heidelberg.

Salminen, J., Yoganathan, V., Corporan, J., Jansen, B. J., & Jung, S. G. (2019). Machine learning approach to auto-tagging online content for content marketing efficiency: A comparative analysis between methods and content type. *Journal of Business Research, 101*, 203–217.

Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture, 112*, 101861.

Tahtaci, B., & Canbay, B. (2020, October). Android malware detection using machine learning. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1–6). IEEE.

Tobiyama, S., Yamaguchi, Y., Shimada, H., Ikuse, T., & Yagi, T. (2016, June). Malware detection with deep neural network using process behavior. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 2, pp. 577–582). IEEE.

Usman, N., Usman, S., Khan, F., Jan, M. A., Sajid, A., Alazab, M., & Watters, P. (2021). Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics. *Future Generation Computer Systems, 118*, 124–141.

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., & Venkatraman, S. (2019). Robust intelligent malware detection using deep learning. *IEEE Access, 7*, 46717–46738.

Yeşilkanat, C. M. (2020). Spatio-temporal estimation of the daily cases of COVID-19 in worldwide using random forest machine learning algorithm. *Chaos, Solitons & Fractals, 140*, 110210.

Zolkipli, M. F., & Jantan, A. (2010, May). A framework for malware detection using combination technique and signature generation. In *2010 Second International Conference on Computer Research and Development* (pp. 196–199). IEEE.