Hossain and Islam *Cybersecurity (2024)7:16*
https://doi.org/10.1186/s42400-024-00205-z

Cybersecurity

RESEARCH

Open Access

Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity

Md. Alamgir Hossain[1*] and Md. Saiful Islam[1]

Abstract

In the realm of cybersecurity, the detection and analysis of obfuscated malware remain a critical challenge, especially in the context of memory dumps. This research paper presents a novel machine learning-based framework designed to enhance the detection and analytical capabilities against such elusive threats for binary and multi type's malware. Our approach leverages a comprehensive dataset comprising benign and malicious memory dumps, encompassing a wide array of obfuscated malware types including Spyware, Ransomware, and Trojan Horses with their subcategories. We begin by employing rigorous data preprocessing methods, including the normalization of memory dumps and encoding of categorical data. To tackle the issue of class imbalance, a Synthetic Minority Over-sampling Technique is utilized, ensuring a balanced representation of various malware types. Feature selection is meticulously conducted through Chi-Square tests, mutual information, and correlation analyses, refining the model's focus on the most indicative attributes of obfuscated malware. The heart of our framework lies in the deployment of an Ensemble-based Classifier, chosen for its robustness and effectiveness in handling complex data structures. The model's performance is rigorously evaluated using a suite of metrics, including accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC) with other evaluation metrics to assess the model's efficiency. The proposed model demonstrates a detection accuracy exceeding 99% across all cases, surpassing the performance of all existing models in the realm of malware detection.

Keywords Obfuscated malware detection, Memory dump analysis, Advanced malware analytics, Malware behavioral patterns, Advanced malware analytics, Machine learning in cybersecurity

Introduction

Obfuscated malware is a sophisticated cyber threat that employs evasion techniques to conceal its presence, making it particularly challenging to detect using conventional security methods. This type of malware, adept at camouflaging itself within regular computing operations, poses a significant threat to digital systems. Our research is centered on developing advanced methodologies to effectively identify and analyze these covert threats, specifically within memory dumps, where they are known to skillfully mask their activities (Asghar et al. 2023; Bozkir et al. 2021; Brezinski and Ferens 2023).

Addressing the challenge of obfuscated malware detection is imperative in the current digital ecosystem, where reliance on technology is at an all-time high (Gorment et al. 2023). In an era where data is the new currency and digital interactions underpin the majority of our daily activities, the potential impact of malware intrusions is vast and multifaceted. From compromising personal data to disrupting critical infrastructure, the threats posed by undetected malware can lead to significant financial,

*Correspondence:
Md. Alamgir Hossain
alamgir.cse14.just@gmail.com
[1] Institute of Information and Communication Technology (IICT), Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh

privacy, and security ramifications. As digital technologies continue to advance and integrate more deeply into various aspects of life, ensuring robust defense mechanisms against such covert cyber threats becomes not just a technical necessity but a cornerstone for maintaining trust and integrity in the digital landscape (Beaman et al. 2021; Mukhtar et al. 2023).

Tackling the issue of obfuscated malware detection presents a complex challenge due to the ever-evolving nature of malware techniques. These sophisticated threats are designed to dynamically alter their code or appearance, thereby effectively evading traditional signature-based detection systems. Additionally, the sheer volume and variety of malware, compounded by the rapid pace of technological advancements, make it increasingly difficult to maintain up-to-date and effective detection methods. This complexity is further amplified in memory analysis, where distinguishing between benign and malicious activities requires nuanced understanding and advanced analytical capabilities, as malware often operates by mimicking legitimate processes (Finder et al. 2022; Hossain Faruk et al. 2021; Rudd et al. 2023). Consequently, staying ahead in this cybersecurity arms race demands continuous innovation and adaptation in detection strategies.

In our research, we adopt a multi-faceted approach to address the challenge of obfuscated malware detection. We leverage advanced machine learning algorithms, specifically focusing on gradient boosting classifiers, to analyze and interpret memory dumps where such malware often resides covertly. Our methodology includes comprehensive data preprocessing, class balancing using Synthetic Minority Over-sampling Technique (SMOTE), and meticulous feature selection through statistical tests and information gain metrics. This strategy enables us to effectively discern the subtle patterns and anomalies indicative of obfuscated malware, providing a robust framework for its identification and analysis beyond the capabilities of traditional detection systems. Our primary contributions to this research can be summarized as follows:

- Establish a robust machine learning framework with gradient-boosting classifiers to detect obfuscated malware with heightened accuracy, addressing both binary and multi-class scenarios within memory dumps.
- Utilize advanced data preprocessing, including class balancing with feature selection, using statistical and information-theoretic approaches to isolate key malware characteristics.
- Create a versatile and adaptable detection model tailored to counter the dynamic nature of malware, effectively handling its various forms across diverse digital environments, and offering improved performance compared to existing models.

Our approach stands out primarily for its unparalleled accuracy in detecting obfuscated malware, achieving a remarkable 100% accuracy across all evaluation metrics for both binary and multi-class classifications. This level of precision is unprecedented in the field and represents a significant advancement over existing methods. The integration of gradient boosting classifiers, combined with sophisticated data preprocessing and feature selection techniques, enables our system to identify even the most skillfully disguised malware. This high accuracy ensures reliable security in various digital environments, significantly reducing the risk of undetected malware intrusions. Furthermore, the adaptability of our model to both binary and multi-class malware types demonstrates its versatility and effectiveness in addressing a wide range of cybersecurity threats, making it a valuable tool in the evolving landscape of digital security. Table 1 provides a concise list of acronyms and abbreviations used throughout this paper, aiding in clarity and comprehension.

The structure of this paper unfolds as follows: We start with a Literature Review, exploring previous studies and theories relevant to our research. This section is followed by the Development of the Proposed Methodology, where we describe the development and intricacies of our detection model. Next, we present the Results of our Analysis, depicted through detailed Tables and Figures

Table 1 Acronyms and abbreviations utilized throughout this paper

Short form	Abbreviations
CA	Pearson correlation analysis
DCNN	Dilated CNN
ACC	Accuracy
PR	Precision
RE	Recall
FS	F1-score
FPR	False positive rate
ER	Error rate
CK	Cohen's kappa
AUC	Area under the ROC curve (AUC)
RF	Random forest ensemble
BG	Bagging ensemble
VT	Voting ensemble
ADB	AdaBoost ensemble
GB	Gradient boosting ensemble
SMOTE	Synthetic minority over-sampling technique
OCC	One-class classifier

that visually and statistically demonstrate our findings. The paper concludes with a final section that summarizes our main discoveries and insights, followed by a comprehensive list of References that support our research.

Literature review

The landscape of cybersecurity is perpetually challenged by the escalating sophistication of obfuscated malware, presenting a formidable barrier to maintaining digital security. This literature review delves into both seminal and contemporary research within the domain of malware detection, with a focused lens on memory analysis and the burgeoning role of machine learning techniques. In traversing this evolving field, we critically examine existing models, assess their success rates, and identify their inherent limitations. This exploration not only illuminates the current state of cybersecurity defenses but also underscores the necessity for advanced detection methods capable of contending with increasingly elusive cyber threats.

Background of obfuscated malware in memory dumps

In the ever-evolving landscape of cybersecurity, the term "obfuscated malware" encapsulates a formidable category of digital threats that transcend conventional detection mechanisms. These sophisticated adversaries deploy evasion techniques with unparalleled finesse, seeking to cloak their true identities and activities within the vast expanse of digital operations. Obfuscated malware achieves this by employing a myriad of tactics, including code encryption, polymorphic behavior, and other obfuscation techniques that challenge traditional signature-based detection systems (Dang 2024; Gorment et al. 2023).

Within the realm of cybersecurity analysis, memory dumps emerge as a crucial battleground for uncovering the covert activities of obfuscated malware. A memory dump is, essentially, a snapshot capturing the intricate contents of a computer's RAM at a specific moment in time. In the context of malware analysis, this ephemeral repository becomes a treasure trove, providing unparalleled insights into the runtime behavior of programs and processes (Naeem et al. 2022). Obfuscated malware, cognizant of the volatility of RAM, strategically conceals itself within memory, exploiting the dynamic nature of the digital environment. The analysis of memory dumps becomes a nuanced art, as these insidious entities often masquerade as legitimate processes, rendering the boundary between benign and malicious activities indistinct. The challenge lies not only in uncovering these malevolent actors but also in understanding the intricate change they perform within the confines of volatile memory.

In navigating the intricacies of memory dumps, researchers unlock the potential to decipher the behavioral patterns of obfuscated malware, transcending the limitations of traditional detection methods. As digital threats continue to evolve in sophistication, the significance of memory dump analysis becomes increasingly pronounced, necessitating innovative and adaptive approaches to fortify the cybersecurity arsenal.

Review on detection approaches of obfuscated malware in memory dumps

Historically, signature-based detection methods served as the cornerstone of cybersecurity defenses. These methods, predicated on identifying malware through predefined patterns and known signatures, were once highly effective against traditional threats. However, their efficacy has waned considerably in the face of obfuscated malware. The principal limitation of signature-based systems lies in their inherent dependency on known threat databases. This characteristic renders them notably inadequate in detecting novel or heavily obfuscated malware variants that deviate from recognized patterns (Haidros Rahima Manzil & Manohar Naik 2023; Lee et al. 2023). Further, heuristic-based approaches, which sought to address some of the shortcomings of signature-based methods by incorporating a degree of behavioral analysis, have also shown vulnerabilities. Although these approaches marked a significant advancement by analyzing program behaviors and attributes, their efficacy is frequently undermined by sophisticated obfuscation techniques. Modern malware, with its ability to mimic benign behavior or effectively conceal its malicious activities, often eludes the heuristic analysis. This limitation is primarily due to the heuristic methods' reliance on predefined behavioral rules and patterns, which may not encompass the innovative tactics employed by new malware strains (Dugyala et al. 2022).

Recent advancements in machine learning have ushered in a variety of approaches for obfuscated malware detection, with researchers exploring classifiers such as DT (Abu Al-Haija et al. 2022; Akhtar and Feng 2022; Lashkari et al. 2021), OCC (Al-Qudah et al. 2023), RF (Manzil and Manohar Naik 2023), and MLP (Sawadogo et al. 2023). These methods have shown promising results, particularly in binary classification scenarios, achieving detection accuracies ranging from approximately 93–99%. Such high accuracy rates underscore the potential of machine learning in discerning between benign and malicious entities in a binary context. However, the application of these machine learning classifiers in multi-classification scenarios reveals a significant limitation. While effective in binary classification tasks, where the objective is to distinguish between two classes

(malicious or benign), their performance diminishes when tasked with identifying multiple malware families. In scenarios requiring the classification of various malware types, such as distinguishing among spyware, ransomware, trojans, and also their sub-categories, these approaches demonstrate considerably lower accuracy. This discrepancy in performance can be attributed to the increased complexity and nuanced distinctions between multiple malware families, which present a more challenging landscape for classification algorithms originally optimized for binary decisions. The intricate behavioral patterns and subtle variances in attributes that differentiate one malware family from another require a more sophisticated analytical approach, one that can navigate the intricate and often overlapping characteristics of various malware types.

Mezina and Burget (2022) evaluated the effectiveness of a dilated CNN (DCNN) in identifying obfuscated malware through memory analysis. While the DCNN showcased impressive accuracy in binary classification (99.92%), its performance in multiclass scenarios, specifically in differentiating between four major malware families, was notably lower (83.53%). This suggests a limitation in the model's ability to discern specific malware types in complex classification tasks. Additionally, the DCNN's substantial computational requirements, due to its intricate architecture with multiple convolutional layers and a high neuron count, pose a challenge for implementation on resource-constrained devices. These findings highlight a critical balance that needs to be struck in advanced malware detection models between accuracy, specificity, and computational efficiency.

Roy et al. (2023) introduced MalHyStack, a model designed to detect obfuscated malware in network environments. This innovative approach combines a stacked ensemble learning framework, in its initial layer, and a deep learning layer as a subsequent stage. Prior to deploying this classification model, an optimal subset of features is determined through CA. Despite its sophisticated architecture, the model's performance metrics indicate certain limitations. Specifically, in categorizing four attack types, MalHyStack achieved an accuracy of 85.04% and a recall rate of 85.17%. In a more complex scenario involving 16 malware categories, the detection rate fell to 66.96%, with a precision of 66.94%. These figures, particularly in the context of multi-classification, suggest the model's limited effectiveness in the rapidly evolving digital security landscape, where higher accuracy and precision are critical for effective malware detection and prevention.

In a notable contribution to the field of malware detection, Shafin et al. (2023) introduced an approach named RobustCBL, aimed at categorizing various malware types. While the model represents an innovative step in multi-class malware detection, its efficacy in accurately identifying different malware categories reveals certain limitations. The performance of RobustCBL, when applied to specific malware families, demonstrates moderate success rates: it detects ransomware in 67% of cases, spyware in 69%, and Trojans in 71%. These figures, while indicative of the model's potential, also highlight its challenges in consistently and accurately identifying these prevalent malware types. The relatively lower detection rates in these categories suggest a need for further refinement in the model's ability to discern the nuanced characteristics and behaviors that define these specific malware families. Moreover, when the scope of RobustCBL's application is expanded to encompass all 16 individual malware classes in the dataset, the model achieves an overall accuracy of 72.60%. While this demonstrates a fair level of proficiency in multi-classification, the accuracy rate is not as high as might be desired for robust cybersecurity applications. Additionally, a significant concern with RobustCBL is its high False Positive Rate (FPR). A high FPR implies that the model frequently misclassifies benign software or processes as malicious, which can lead to unnecessary or disruptive actions and erode trust in the system's reliability.

In summarizing the condition from recent literature, in the field of malware detection, a notable gap emerges in the development and evaluation of models adept at identifying sophisticated, obfuscated malware within constrained system environments, especially for malware category detection with high TPR. This gap is particularly pronounced in the context of models that must balance robust detection capabilities with minimal FPR. Our research addresses this critical need by introducing an innovative framework. This process is designed not only to excel in binary classification tasks but also to adeptly navigate the complexities of categorizing and identifying diverse malware families. Through rigorous evaluation, our model demonstrates its efficacy against the latest iterations of cyber threats, thereby offering a significant contribution to the arsenal of tools in combating evolving digital security challenges.

Proposed methodology

In our research, the Proposed Methodology, depicted comprehensively in Fig. 1, represents a sophisticated and multi-faceted approach to enhancing malware detection and analysis. This methodology integrates advanced machine learning techniques, encompassing various ensemble models such as GB, RF, ADB, VT, and BG, each tailored to address the intricate challenges of classifying obfuscated malware. By leveraging a combination of data preprocessing, feature selection,

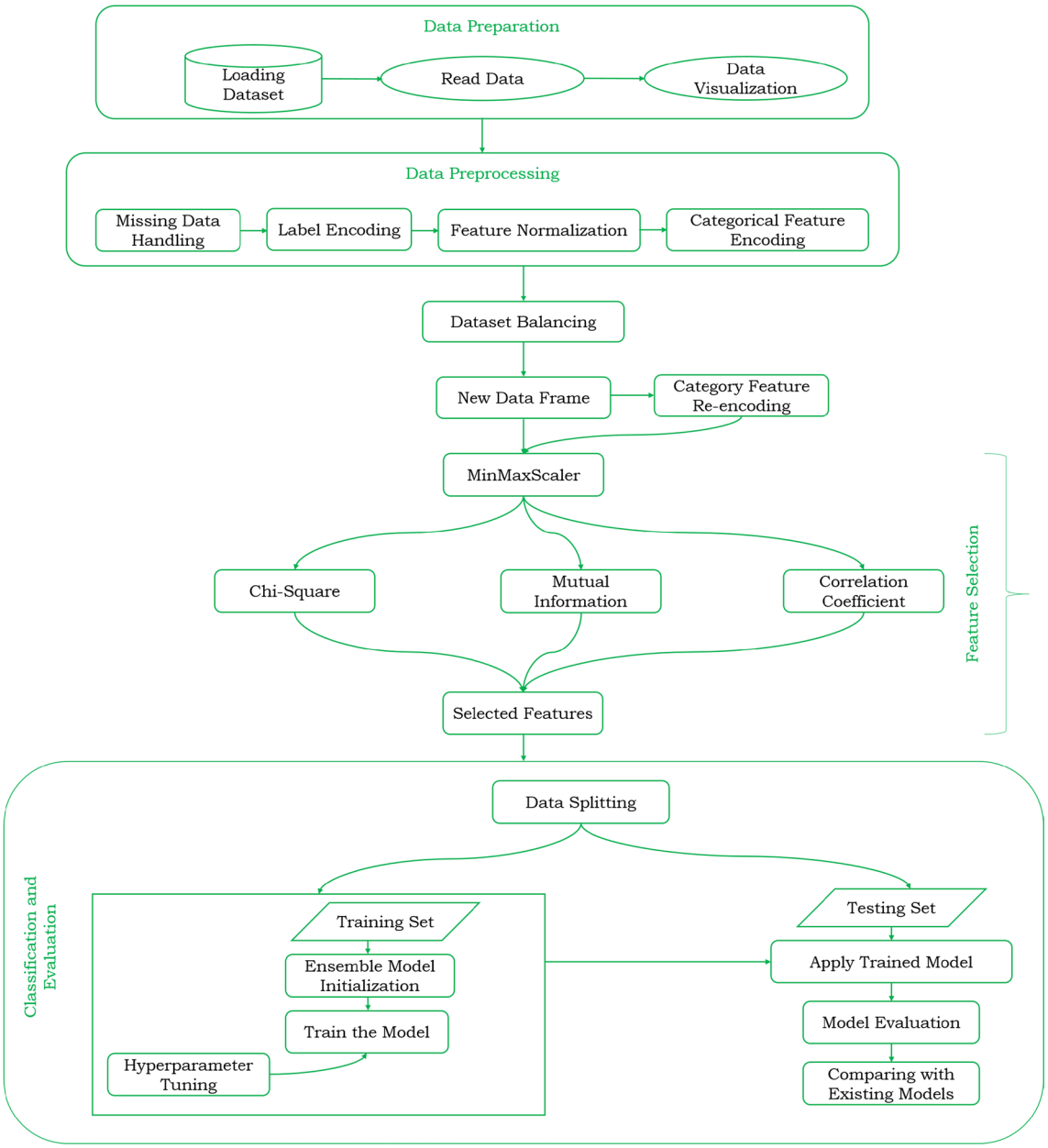

Fig. 1 Pipeline of the proposed methodology

and ensemble learning strategies, our approach aims to significantly improve the accuracy and reliability of malware detection. The systematic and holistic nature of this methodology, as outlined in Fig. 1, not only exemplifies cutting-edge research in cybersecurity but also sets a new benchmark in the field, demonstrating a deep understanding of both the technical complexities and the practical implications of malware analysis.

Dataset acquisition and preprocessing

In this research, the "Dataset Acquisition and Preprocessing" phase is meticulously structured to ensure the dataset's readiness for advanced machine learning analysis. In this research, we employed the Obfuscated-MalMem2022 dataset (Carrier et al. 2022), an extensive and meticulously curated collection of memory dumps, both benign and malicious, designed to mirror real-world scenarios in cybersecurity. The features and their detailed descriptions are comprehensively presented

in the "Appendix" section. This dataset is pivotal in this whole analysis, providing a representation of benign data and various malware classes, including Ransomware, Spyware, and Trojan Horses. The detailed enumeration of data points, categorized into binary classes, malware categories, and specific malware families, is systematically presented in Table 2. This comprehensive dataset not only enriches our research with a diverse range of samples but also ensures the robustness and validity of our proposed detection model.

To ensure data integrity, the dataset is first cleansed of missing and infinite values. This involves replacing infinite values with NaN (Not a Number) and subsequently removing these NaN values. Mathematically, this is represented as: $data = data.replace(\{\infty, -\infty\}, NaN)$, and $data = data.dropna()$.This step is important to avoid computational errors and biases that can arise from incomplete or corrupt data.

Data categorization and encoding

In the research, the process of "Data Encoding and Normalization" plays a pivotal role, primarily due to the intrinsic characteristics of the dataset and the requirements of machine learning algorithms. This process serves two fundamental purposes: transforming categorical data into a machine-readable format and standardizing the range of continuous numerical features (Hossain and Islam 2023a, b). The categorical nature of some features in the dataset, particularly the 'Category' column, necessitates encoding. This dataset encompasses various malware types like Ransomware, Spyware, and Trojan, each represented as a string. Machine learning algorithms inherently require numerical input. To address this, we employ Label Encoding, represented as a mapping function: $f : C \rightarrow Z$. Here, C is the set of categorical labels (e.g., 'Benign', 'Ransomware', 'Spyware', 'Trojan'), and Z represents the set of integers. If c_i is a categorical value in the dataset, its encoded value z_i is given by $z_i = f(c_i)$, where f is the label encoding function mapping each unique label to a unique integer. For instance, we have labels {Benign, Ransomware, Spyware, Trojan}, these could be encoded as {0, 1, 2, 3} respectively.

The dataset contains numerical features varying in ranges. Without normalization, features with larger ranges could disproportionately influence the model, leading to biased learning. This issue is particularly pertinent in datasets with diverse feature scales, as is the case in cybersecurity datasets. StandardScaler is utilized, which normalizes a feature by subtracting the mean and dividing by the standard deviation, effectively transforming the data to have a mean of zero and a standard deviation of one. For a given feature X with n samples, $X = [x_1, x_2, \ldots, x_n]$, the normalization process adjusts the values of X so that they have a mean of zero and a standard deviation of one. The normalized value x_i' of each sample x_i in X is calculated using the formula (1).

$$x_i' = \frac{x_i - \mu x}{\sigma x} \tag{1}$$

where μx is the mean of the feature X, calculated as $\mu x = \frac{1}{n}\sum_{i=1}^{n} x_i$, and σx is the standard deviation of X, calculated as:

Table 2 Distribution of data: a detailed breakdown by classification type and malware families

Dataset	Binary class with counts	Malware category with counts (4 class)	Malware families with counts (16 class)
Obfuscated-MalMem2022	Benign (29,298) (50%)	Benign (29,298) (50%)	Benign (29,298) (50%)
	Malware (29,298) (50%)	Ransomware (9791) (16.71%)	Shade (2128) (3.63%)
			Ako (2000) (3.41%)
			Conti (1988) (3.39%)
			Maze (1958) (3.35%)
			Pysa (1717) (2.93%)
		Spyware (10,020) (17.10%)	Transponder (2410) (4.11%)
			Gator (2200) (3.75%)
			180Solutions (2000) (3.41%)
			Coolwebsearch (2000) (3.41%)
			TIBS (1410) (2.41%)
		Trojan Horse (9487) (16.19%)	Refroso (2000) (3.41%)
			Scar (2000) (3.41%)
			Emotet (1967) (3.36%)
			Zeus (1950) (3.33%)
			Reconyc (1570) (2.68%)

$$\sigma x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu x)^2} \tag{2}$$

The encoding ensures that categorical data is interpretable by the algorithms, while normalization standardizes the feature scales, allowing the model to learn and make predictions without bias towards any particular feature's numeric scale.

Handling class imbalance with SMOTE

The original dataset likely has imbalanced class distributions, meaning some types of malware are underrepresented. This imbalance can bias the machine learning model towards the majority class, reducing its effectiveness in accurately identifying less common malware types. By generating synthetic samples for minority classes, SMOTE (Asniar et al. 2022) helps in creating a more balanced dataset, which contributes to a more robust and generalized model capable of detecting various malware types effectively.

For each sample x_i in the minority class, SMOTE computes its $k - nearest$ neighbors. Let $N_k(x_i)$ denote the set of $k - nearest$ neighbors of x_i in the feature space. The distance metric, often Euclidean, for two samples x_i and x_j is given by Eq. (3).

$$d\left(x_i, x_j\right) = \sqrt{\sum_{l=1}^{m}\left(x_{il} - x_{jl}\right)^2} \tag{3}$$

where m is the number of features.

A synthetic instance x_{new} is generated by interpolating between the sample x_i and one randomly chosen nearest neighbor $x_{ni} \in N_k(x_i)$. The formula for creating a synthetic sample is:

$$x_{new} = x_i + \lambda * (x_{ni} - x_i) \tag{4}$$

where λ is a random number between 0 and 1. This ensures that the synthetic sample x_{new} lies along the line segment between x_i and x_{ni} in the feature space. The aim is to balance the class distribution between the majority and minority classes. If the size of the minority class is S_{min} and the desired size after oversampling is $S_{desired}$, the number of synthetic samples N_{synth} to be generated for the minority class is $S_{desired} - S_{min}$. The process involves repeatedly applying the synthetic sample generation step until N_{synth} samples are created, thereby augmenting the minority class to achieve the desired class balance.

In this research, the application of SMOTE is a crucial step to mitigate the bias caused by imbalanced class distribution. This enhances the model's capability to learn generalized patterns and improves its performance in accurately classifying various malware types, which is essential in the context of cybersecurity.

Feature selection and scaling

In this research, "Feature Selection and Scaling" is a critical stage, crucial for enhancing the model's performance and interpretability. This stage involves two main processes: feature selection using SelectKBest with chi2 and mutual_info_classif methods, and feature scaling using MinMaxScaler. The chi-squared test assesses the independence of two variables, making it suitable for feature selection where the aim is to identify features that are most likely to be independent of class labels (Mamdouh Farghaly and Abd El-Hafeez 2023). For a given feature X and a class label Y, the chi-squared statistic is calculated as:

$$X^2(X, Y) = \sum_{i=1}^{n}\frac{(O_i - E_i)^2}{E_i} \tag{5}$$

where O_i is the observed frequency, E_i is the expected frequency under the null hypothesis of independence, and n is the number of distinct values in X. The higher the chi-squared value, the more likely the feature is dependent on the class and thus important for classification.

Mutual information measures the amount of information one can obtain about one random variable by observing another (Federici et al. 2023). For features X and class label Y, it is defined as:

$$I(X; Y) = \sum_{x \in X, y \in Y} P\left(x, y\right) \log\left(\frac{P\left(x, y\right)}{P(x)P\left(y\right)}\right) \tag{6}$$

where $P\left(x, y\right)$ is the joint probability distribution of X and Y, and $P(x)$, $P\left(y\right)$ are the marginal probability distributions of X and Y, respectively. Features with higher mutual information values are considered more relevant for predicting the class label.

Post feature selection, scaling is essential to normalize feature values within a bounded range, typically [0, 1]. MinMaxScaler transforms each feature x using the formula:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{7}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of the feature x, respectively. This scaling method preserves the shape of the dataset's distribution and is beneficial when the features have varying scales and ranges. In the context of this research, the combination of chi-squared and mutual information for feature selection, followed by MinMaxScaler for scaling, ensures that the model focuses on the most informative features that

contribute significantly to the classification task. This approach not only improves the model's predictive power but also enhances its efficiency by reducing computational complexity, crucial for handling large and complex cybersecurity datasets.

Upon completing the preprocessing and feature selection stages, we applied various ensemble-based approaches, including GB, VT, ADB, RF, and BG, each with meticulous hyperparameter tuning. The section below offers a concise description of each of these updated ensemble-based methods, outlining their unique characteristics and roles in our research.

Model training process with gradient boosting classifier

In this research, the utilization of the Gradient Boosting Classifier (GBC) for classifying multiple malware types is grounded in its ensemble-based methodology and sophisticated handling of complex datasets. The GBC's effectiveness in managing intricate data relationships is attributed to its ensemble learning approach and the optimization of specific hyperparameters. The GBC lies in constructing an additive model in a forward stage-wise manner (Chen and Ren 2023). Formally, this is expressed as:

$$F(x) = \sum_{m=1}^{M} \gamma_m h_m(x) + const \tag{8}$$

where $F(x)$ is the final model, M is the number of trees (stages), $h_m(x)$ is the base learner (decision tree), and γ_m is the weight of each tree.

From the Hyperparameter Number of Estimators set to 10, it dictates the number of sequential trees built, Learning Rate with a value of 0.1, this parameter scales the contribution of each tree, affecting the model's convergence rate. It adjusts the step size in the gradient descent process, Max Depth Limited to 3, it controls the maximum depth of individual trees, curbing model complexity and overfitting.

GBC optimizes a differentiable loss function $L\big(y, F(x)\big)$, where y is the actual value and $F(x)$ is the model prediction. The loss is minimized using gradient descent, with each tree built to model the negative gradient of the loss function concerning the predictions. The model iteratively updates the predictions based on the equation:

$$F_{m+1}(x) = F_m(x) + \nu \sum_{i=1}^{n} \gamma_m h_m(x_i) \tag{9}$$

where ν the learning rate, and N is is the number of samples. The classification process is detailed in Algorithm 1.

Algorithm 1: Malware Classification Process of the Model with GBC.

1. Begin with an initial model, typically a constant value. This is often the log odds in the case of classification:

   $F_0(x) = argmin_\gamma \sum_{i=1}^{N} L\big(y_i, \gamma\big)$; where L is the loss function, y_i are the true labels, and N is the number of samples.
2. For each iteration $m = 1, 2, \ldots, M$ (where M is the number of trees):

   a. Compute the pseudo-residuals for each instance in each class k:

   $$r_{ikm} = -\left[\frac{\partial L\big(y_i, F(x_i)\big)}{\partial F_k(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

   b. Fit a decision tree $h_{mk}(x)$ to these residuals for each class.

3. Determine the output values for the leaf nodes in each tree:

   $$\gamma_{jkm} = argmin_\gamma \sum_{x_i \in R_{jm}} L\big(y_i, F_{m-1}(x_i) + \gamma\big)$$

4. Update the model for each class k:

   $$F_{mk}(x) = F_{m-1,k}(x) + \nu \sum_{j=1}^{J} \gamma_{jkm} I\big(x \in R_{jm}\big)$$

5. For a new memory dump x_{new}, the model outputs a set of scores for each class k. The softmax function is then applied to these scores to obtain probabilities: $P\big(y = k|x_{new}\big) = \frac{e^F Mk(x_{new})}{\sum_{l=1}^{k} e^F Ml(x_{new})}$; where K is the total number of classes. (For attack family classification). $P\big(y = k|x_{new}\big) = \frac{1}{1+e^{-F}M(x_{new})}$; (For Benign and Malware classification).
6. The class with the highest probability from the softmax output is selected as the final prediction for x_{new}.

In Algorithm 1, delineating the Malware Classification Process of the Model with Gradient Boosting Classifier (*GBC*), the procedure commences with initializing an initial model, often a constant value represented as $F_0(x)$, typically the log odds in classification scenarios. Subsequently, for each iteration $m = 1, 2, \ldots, M$ (where M is the number of trees), the algorithm computes pseudo-residuals for each instance in each class

k. These pseudo-residuals, denoted as r_{ikm}, capture the difference between the true labels y_i and the current model's predictions $F(x_i)$. A decision tree $h_{mk}(x)$ is then fitted to these residuals for each class. The output values for the leaf nodes in each tree, γ_{jkm}, are determined, and the model is updated for each class k based on these values. For a new memory dump x_{new}, the model outputs scores for each class k. The softmax function is applied to these scores to obtain probabilities, and the class with the highest probability is chosen as the final prediction. This iterative process of fitting decision trees and updating the model enhances the model's predictive capabilities through sequential learning, making it effective in capturing complex relationships within the data.

Incorporating the softmax function in the multi-class setting allows the *GBC* to effectively handle classification tasks with multiple malware types. This approach ensures that each memory dump is assigned a probability distribution across all possible malware categories, allowing for a nuanced classification based on the highest likelihood. The softmax function's ability to convert raw scores into probabilities that sum up to one makes it an ideal choice for multi-class classification in the context of malware detection. The Gradient Boosting Classifier, through its ensemble learning approach and meticulous tuning of hyperparameters, exemplifies a powerful method for tackling the multifaceted challenge of malware classification in cybersecurity. This model, adept in handling complex data and reducing overfitting, signifies a sophisticated approach in the domain of machine learning for malware detection and analysis.

Model training process with BG ensemble

The Bagging ensemble method is particularly effective for classifying obfuscated malware due to its inherent ability to mitigate overfitting, a common challenge in complex classification tasks. By aggregating predictions from multiple decision trees, each trained on different subsets of the data, Bagging introduces diversity in the learning process (Ngo et al. 2022). This diversity is crucial in dealing with obfuscated malware, where subtle variations in data patterns can significantly impact classification accuracy. The ensemble approach ensures that the model does not overly rely on specific attributes of the data, thereby enhancing its ability to generalize and accurately identify even sophisticated, disguised malware threats. The classification process is detailed in Algorithm 2.

Algorithm 2: Malware Classification Process of the Model with BG.

1. Define Bagging ensemble: $BaggingClf = \{RF_1, RF_2, \ldots, RF_n\}$
2. For each RF_1 in $BaggingClf$:
   a. Bootstrap sample: $D_i \leftarrow BootstrapSample(D)$
   b. Construct decision tree DT_{ij} for each D_i:
3. Random feature selection: $F_{ij} \leftarrow RandomSubset(F)$
4. For node N, find split s minimizing Gini impurity:

$$Gini(N) = 1 - \sum (P_k)^2$$

$$s^* = argmin_{s \in S} Gini(N)$$

5. Grow tree to maximum depth MaxDepth or until criterion met.

3. Ensemble prediction for sample x:

$$y_{pred}(x) = mode\{RF_1(x), RF_2(x), \ldots, RF_n(x)\}$$

In Algorithm 2, outlining the Malware Classification Process of the Model with Bagging (BG), the Bagging ensemble, denoted as BaggingClf, is defined as a collection of individual Random Forest classifiers, represented as RF_1, RF_2, up to RF_n. For each RF_i BaggingClf, a bootstrap sampling operation is performed, generating a bootstrap dataset D_i from the original dataset D. Subsequently, a decision tree (DT_{ij}) is constructed for each D_i. The construction involves random feature selection, where a subset F_{ij} of features is randomly chosen. For each node N in the tree, the optimal split s^* is determined by minimizing the Gini impurity criterion. The Gini impurity *Gini(N)* measures the impurity or disorder in a set of samples. The tree is grown to either the maximum depth (*MaxDepth*) or until a specified criterion is met. The ensemble prediction for a given sample x is then calculated as the mode of predictions from all RF_i classifiers. This approach leverages the diversity introduced by bootstrap sampling and random feature selection, enhancing the overall robustness and accuracy of the Bagging ensemble in the context of malware classification. The ensemble's ability to aggregate predictions reduces the risk of overfitting, making it particularly effective for complex classification tasks like malware detection in memory dumps.

Model training process with VT ensemble

The Voting Ensemble method is also employed for the classification of memory dumps into benign or various malware categories (Vashishtha et al. 2023). This ensemble technique combines the predictions from multiple

distinct classifiers Decision Tree, Logistic Regression, and Support Vector Classifier (SVC) each contributing unique insights. The complete process is detailed in Algorithm 3.

Algorithm 3: Construction of the Soft Voting Classifier and Prediction Process.

1. Defining Classifiers for VT ensemble:

$$clf1 \leftarrow DecisionTreeClassifier()$$

$$clf2 \leftarrow LogisticRegression(max\_iter = 1000)$$

$$clf3 \leftarrow SVC\left(probability = True\right)$$

2. Voting Classifier Construction:

$$eclf \leftarrow VotingClassifier\left(estimators = \left[\left('dt', clf1\right), \left('lr', clf2\right), \left('svc', clf3\right)\right], voting =' soft'\right)$$

3. For each classifier clf_i in the ensemble, the training process involves fitting the model to the training data.
4. For a given sample x, the probability of belonging to class k as predicted by classifier clf_i is $P_{ik}(x)$.
5. The final prediction for class k is then the weighted average of these probabilities:
$P_{ensemble,k}(x) = \frac{1}{N}\sum_{i=1}^{N} w_i.P_{ik}(x)$; where N is the number of classifiers, and w_i are the weights assigned to each classifier's prediction.
6. The class with the highest probability $P_{ensemble,k}(x)$ across all k classes is chosen as the final prediction.
7. Final Prediction: $y_{pred} = argmax_k P_{ensemble,k}(x)$.

In the instantiation of the Soft Voting Classifier, Algorithm 3 outlines the construction and prediction process for a robust ensemble of classifiers. Individual classifiers, namely *clf1* (DecisionTreeClassifier), *clf2* (LogisticRegression with increased *max_iter*), and *clf3* (SVC with probability estimation) are defined for the ensemble. The Voting Classifier (*eclf*) is then formed, incorporating these classifiers with a 'soft' voting strategy. During training, each classifier clf_i undergoes a fitting process to the training data. For a given sample x, the probability of belonging to class k as predicted by clf_i is denoted as $P_{ik}(x)$.The final prediction for class k in the ensemble is calculated as the weighted average of these probabilities, where N represents the number of classifiers, and w_i signifies the weights assigned to each classifier's prediction. The class with the highest probability across all classes is selected as the final prediction (y_{pred}), implementing the argmax operation. This approach leverages the collective decision-making of diverse classifiers, enhancing the model's adaptability and predictive accuracy in the domain of malware detection.

This training process of the ensemble ensures that each classifier contributes its understanding of the data, and their combined predictions offer a comprehensive view, thereby enhancing the overall predictive performance and robustness of the model. This approach, combining Decision Tree, Logistic Regression, and SVC in a soft voting mechanism, allows for a comprehensive decision-making process, leveraging the strengths of each classifier. The ensemble's aggregated predictions provide a more balanced and accurate classification, essential in the intricate task of malware detection.

Training and classification process of the model with ADB ensemble

The ADB Ensemble method stands out as a robust approach for enhancing the classification of memory dumps into benign or various malware categories. ADB, short for Adaptive Boosting, excels in refining the classification process by iteratively focusing on difficult-to-classify instances (Hossain and Islam 2023a). It combines multiple weak learners, typically simple decision trees, to form a strong classifier. Each successive learner is adapted to emphasize the data points that previous learners misclassified, thereby progressively improving the model's accuracy. The complete process is detailed in Algorithm 4.

Algorithm 4: Training and Prediction Process of the Model with ADB.

1. Initialization: Start with a dataset D and weights $w_i = \frac{1}{N}$ for each instance i, where N is the total number of instances.
2. For $t = 1$ to T (where T = 10 is the number of estimators):
   a. Train a weak learner $L_t(DecisionTreewithmax\_depth = 1)$ on the dataset using the current weights.
   b. Calculate the error ε_t of L_t:

$\varepsilon_t = \frac{\sum_{i=1}^{N} w_i . I(y_i \neq L_t(x_i))}{\sum_{i=1}^{N} w_i}$; where I is the indicator function, y_i is the true label, and $L_t(x_i)$ is the prediction.

c. Compute the learner's weight α_t:

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$

d. Update weights for each instance:

$$w_{i,new} = w_i . e^{-\alpha_t . y_i . L_t(x_i)}$$

e. Normalize the weights so that they sum up to 1.

3. The final model is a weighted combination of the weak learners:

$$AdaBoostModel(x) = sign\left(\sum_{t=1}^{T} \alpha_t . L_t(x)\right)$$

4. For a new sample x_{new}, the final AdaBoost model provides the classification:

$$y_{pred} = AdaBoostModel(x_{new})$$

Algorithm 4 outlines the Training and Prediction Process of the Model with AdaBoost (*ADB*) for malware detection. The process begins with initializing a dataset D and assigning weights $w_i = \frac{1}{n}$ for each instance i, where N is the total number of instances. For each iteration $t=1$ to T(where $T=10$ is the number of estimators), a weak learner L_t, specifically a Decision Tree with max_depth = 1, is trained on the dataset using the current weights. The error ε_t or L_t is calculated, representing the misclassification rate. The learner's weight α_t is then computed based on ε_t. The weights for each instance are updated using α_t, and normalization ensures they sum up to 1. The final model is a weighted combination of the weak learners, and for a new sample x_{new}, the AdaBoost model provides the classification y_{pred}. This iterative boosting process focuses on instances that are misclassified in previous iterations, enhancing the model's ability to adapt and improve its performance over time.

AdaBoost's strength lies in focusing more on instances that are harder to classify, thereby improving the ensemble's overall performance. This method is particularly effective in complex classification tasks, such as distinguishing between different types of malware, due to its adaptive nature and capability to enhance the performance of simple models.

Random forest ensemble approach for the malware classification

In this research, the Random Forest (RF) Ensemble method is also employed as a key analytical tool for the classification of memory dumps, distinguishing between benign data and various types of malware (Hossain 2023). This method is revered for its robustness and accuracy, particularly in handling complex datasets with numerous features. RF combines multiple decision trees, reducing the risk of overfitting while capturing a broad spectrum of data characteristics. Each tree contributes a unique perspective, and their collective decision-making offers a balanced and comprehensive classification. The adaptability and efficacy of the RF Ensemble make it an invaluable component of our research, enhancing our capabilities in malware detection and analysis. The complete process is detailed in Algorithm 5.

Algorithm 5: Random Forest Ensemble in Malware Classification.

1. Initialization: Define RF with n_estimators = 10 and random_state = 42.
2. For each tree t in RF:

   a. Randomly select samples with replacement from X_{train} to create a bootstrap dataset D_t.
   b. Grow t on D_t by recursively splitting nodes based on feature subsets. At each node:

1. Select m features randomly from the total features.
2. Choose the best split based on an impurity criterion like Gini:
   $Gini(S) = 1 - \sum_{i=1}^{c} (P_i)^2$; where P_i is the proportion of samples in class i.
3. After training, for a new sample x, each tree t in RF predicts a class y_t.
4. The final class prediction yRF is the mode of all y_t:

$$yRF(x) = mode\{y_1(x), y_2(x), \ldots, y_{10}(x)\}$$

5. Predict whether x_{new} is benign or a specific malware type using RF:

$$y_{new} = yRF(x_{new})$$

In the initial stage of the algorithm, the Random Forest (RF) Ensemble for Malware Classification is instantiated with crucial parameters, specifically setting *n_estimators*

to 10 and *random_state* to 42. This establishes an ensemble of 10 decision trees with a fixed random seed, ensuring both diversity and reproducibility. Subsequently, for each tree t within the RF, a bootstrap sampling process is initiated by randomly selecting samples with replacement from the training dataset (X_{train}), creating a distinctive bootstrap dataset D_t for each tree. The tree growth phase unfolds as each decision tree t is constructed on D_t, involving the recursive splitting of nodes based on randomly chosen feature subsets. At each node, m features are randomly selected from the total feature set, and the optimal split is determined using an impurity criterion, such as Gini impurity. Following the training phase, each tree t contributes to predicting the class y_t for a new sample x. The final ensemble prediction yRF for the new sample is then computed as the mode of all y_t.

The RF Ensemble method is particularly effective for this research due to its ability to handle high-dimensional data and its robustness against overfitting. By combining the predictions of multiple decision trees, each trained on different subsets of the data, RF provides a comprehensive approach to classifying complex and nuanced patterns typical in malware detection. This approach ensures a balance between bias and variance, leading to more accurate and reliable classification results.

Following the training phase, the model undergoes testing with a designated set of test data. The next section of this paper will detail the results obtained from various evaluation metrics, highlighting the effectiveness of the model. Additionally, this section will include comparative analyses, showcasing how the model performs in relation to existing methodologies in obfuscated malware detection.

Results and analysis

In this research on advanced malware classification, the experimental setup is conducted on a high-performance ASUS device, powered by an 11th Gen Intel(R) Core(TM) i7-11700 processor with a base speed of 2.50 GHz and equipped with 16.0 GB of RAM. Operating on a 64-bit Windows 11 Pro system, the implementation utilizes the Anaconda Navigator for managing software environments, primarily employing Jupyter Notebook for development. Key Python libraries such as Pandas, Matplotlib, Seaborn, Scikit-learn, and Imbalanced-learn are integral to the research, facilitating tasks from data preprocessing to machine learning model evaluation. A suite of machine learning techniques, including RandomForest, Bagging, DecisionTree, LogisticRegression, SVC, Voting, AdaBoost, and GradientBoosting classifiers, is employed, evaluated using metrics like accuracy, precision, recall, and F1-score, ensuring a thorough assessment of model performance in malware detection. This setup provides the necessary computational power and versatility for handling the complex demands of cybersecurity research.

The comprehensive evaluation of the framework in this study leverages the Obfuscated-MalMem2022 dataset, a pivotal resource for analyzing advanced malware detection techniques. This dataset undergoes a division into training and testing subsets, utilizing the "train_test_split" method from the scikit-learn library. A strategic split of 75% for training and 25% for testing ensures a robust training process while providing a substantial dataset for validation. Emphasis is placed on the testing data, which comprises 25% of the dataset, to present all results in this section. To assess the model's effectiveness and accuracy, a range of evaluation metrics are meticulously employed. These metrics, crucial for establishing the model's performance, are detailed in Table 3, complete with corresponding equations. This methodical approach underlines the rigor and precision inherent in the evaluation process of the proposed malware detection framework.

Figure 2 depicts the class distribution before applying the SMOTE. Initially, the distribution of the different malware categories Benign, Ransomware, Spyware, and Trojan exhibits a significant imbalance. As illustrated in the pie chart, Benign instances constitute half of the dataset at 50%, while the malware categories (Ransomware, Spyware, and Trojan) have smaller representations ranging from 16.19 to 17.10%. This imbalance is a common challenge in machine learning, particularly in cybersecurity contexts, as it can lead to biased models that underperform in detecting less represented classes.

After the application of SMOTE, an impressive transformation in class distribution is observed. Each category now contains an equal number of instances, specifically 29,298. This equalization is crucial for the development of an unbiased and effective classification model. SMOTE achieves this by oversampling the minority classes in this case, Ransomware, Spyware, and Trojan until they match the number of instances in the majority class, which is Benign. The balanced distribution post-SMOTE enhances the model's ability to learn from an equally representative dataset, ensuring that each malware type is given equal importance during the training phase. This approach mitigates the risk of overfitting to the majority class and improves the model's capability to detect and classify malware types that were initially underrepresented. The resulting uniform distribution across all categories sets a strong foundation for building a robust and effective malware classification model, essential for addressing the diverse and evolving nature of cyber threats.

Table 4 provides a comprehensive evaluation of various ensemble classifiers for multi-class malware

Table 3 Evaluation metrics with proper description

Evaluation metrics	Description
Accuracy (ACC)	Accuracy, defined as the ratio of correctly predicted observations to the total observations, is calculated as $Accuracy = \frac{True\ Positives\ (TP)+True\ Negatives\ (TN)}{Total\ Observations}$, providing a fundamental measure of the models overall predictive correctness
Precision (PR)	Precision, a key metric in model evaluation, is quantified as $\frac{True\ Positives\ (TP)}{True\ Positives\ (TP)+False\ Positives\ (FP)}$, reflecting the proportion of true positive predictions among all positive predictions made by the model
Recall (RE)	Recall, an essential metric in classification models, is determined by the formula $Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP)+False\ Negatives(FN)}$, capturing the model's ability to correctly identify all relevant instances within a dataset
F1-score (FS)	F1-score, a crucial metric that balances precision and recall, is calculated using the formula $F1\text{–}score = 2 * \frac{Precision \times Recall}{Precision+Recall}$, thereby providing a harmonic mean that encapsulates the model's accuracy in classifying data points correctly
False positive rate (FPR)	FPR, a critical metric in assessing classification errors, is $FPR = \frac{False\ Positives\ (FP}{False\ Positives\ (FP)+True\ Negatives\ (TN)}$, computed as quantifying the proportion of negative instances incorrectly classified as positive by the model
Error rate (ER)	Error rate, a metric that quantifies the overall prediction inaccuracies of a model, is calculated as $Error\ Rate = \frac{False\ Positives\ (FP)+False\ Negatives\ (FN)}{Total\ Observations}$, effectively measuring the proportion of all predictions that the model got wrong
AUC score (AUC)	The Area Under the Curve (AUC) Score, a measure of the model's ability to distinguish between classes, is calculated by plotting the TPR against the FPR at various threshold settings, with the AUC value ranging from 0 to 1, where a higher value indicates better classification performance
Cohen's Kappa (KP)	Cohen's Kappa, a statistical measure of inter-rater agreement for categorical items, is calculated as $Kappa = \frac{P_o-P_e}{1-P_e}$, where P_o represents the relative observed agreement among raters, and P_e is the hypothetical probability of chance agreement, providing a robust assessment of the model's predictive accuracy beyond random chance

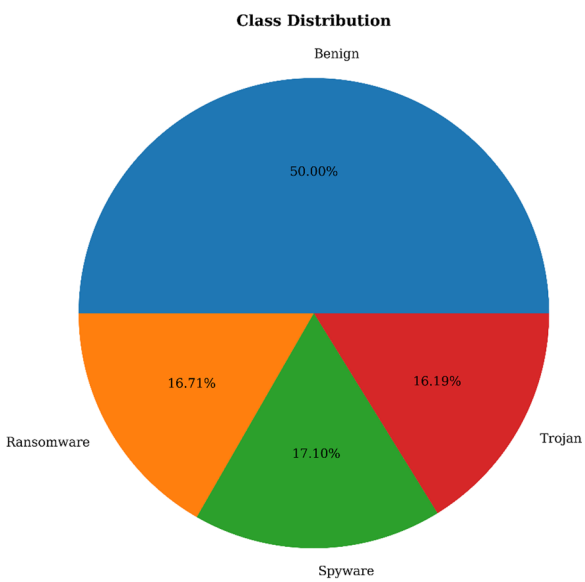

Fig. 2 Distribution of malware categories prior to SMOTE balancing

classification, comparing their performance both with and without the application of the SMOTE balancing technique. The metrics offer a holistic view of the model's performance for various ensemble classifiers. A striking observation from the table is the superior performance of the Gradient Boosting (GB) ensemble across all metrics, especially when combined with SMOTE balancing, achieving perfect scores in ACC, PR, RE, FS, and AUC. This indicates that the GB ensemble, when applied to a balanced dataset, can effectively identify and classify different malware types with utmost accuracy and reliability. Comparatively, other ensemble methods like Random Forest (RF), Bagging (BG), Voting (VT), and AdaBoost (ADB) show varied performance. While RF, BG, and VT perform admirably well without balancing, their effectiveness increases with SMOTE, evident from the improved scores in most metrics. However, the AdaBoost ensemble exhibits a noticeable dip in performance when balancing is applied, suggesting that it may not be as effective in handling balanced datasets in this specific context.

Table 4 underlines the effectiveness of ensemble methods in multi-class malware classification, with Gradient Boosting, in particular, standing out for its unparalleled performance, especially when combined with SMOTE balancing. This insight underscores the importance of choosing appropriate machine learning techniques and balancing strategies to enhance model accuracy and reliability in cybersecurity applications.

The confusion matrix depicted in Fig. 3, derived from the binary classification of 'Benign' versus 'Malware', provides a compelling illustration of the model's exceptional performance in malware detection. The diagonal cells represent the number of true positive and true negative predictions, which are remarkably high for both classes. Specifically, the model has successfully identified 7279 instances as 'Benign' and 7370 as 'Malware' with absolute precision, as indicated by the absence of false positives and false negatives. This perfect classification indicates that the model has an exceptional ability to differentiate

Table 4 Performance metrics (weighted) of the model utilizing diverse ensemble techniques

Metrics	Without balancing					Balancing with SMOTE				
	GB	RF	BG	VT	ADB	GB	RF	BG	VT	ADB
ACC	0.99966	0.99932	0.99932	0.99951	0.83378	1.0000	0.99956	0.99997	0.99997	0.74783
PR	0.99966	0.99932	0.99932	0.99951	0.91541	1.0000	0.99956	0.99997	0.99997	0.87412
RE	0.99966	0.99932	0.99932	0.99951	0.83378	1.0000	0.99956	0.99997	0.99997	0.74783
FS	0.99966	0.99932	0.99932	0.99951	0.77772	1.0000	0.99956	0.99997	0.99997	0.66387
FPR	0.00017	0.00021	0.00021	0.00011	0.05020	1.0000	0.00015	0.00001	0.00001	0.08421
ER	0.00034	0.00068	0.00068	0.00049	0.16622	1.0000	0.00044	0.00003	0.00003	0.25217
CK	0.99949	0.99898	0.99898	0.99937	0.75105	1.0000	0.99941	0.99995	0.99995	0.66386
AUC	1.00000	0.99998	1.00000	0.99999	0.91667	1.0000	0.99997	1.00000	1.00000	0.91667

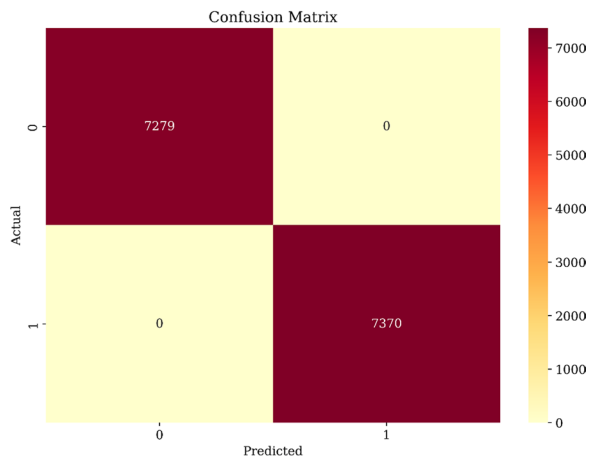

Fig. 3 Confusion matrix for the binary classification

Table 5 Outcomes of the model in a binary classification context

Class	ACC	PR	RE	FS
Benign (0)	1.0000	1.0000	1.0000	1.0000
Malware (1)	1.0000	1.0000	1.0000	1.0000

between benign and malicious software with maximum sensitivity and specificity. The heatmap visualization of the confusion matrix further emphasizes the model's accuracy. The distinct contrast between the high values on the diagonal (true classifications) and the zeros off the diagonal (false classifications) visually reaffirms the model's effectiveness.

The results presented in the confusion matrix are a testament to the robustness and reliability of the model in binary classification tasks within cybersecurity. The impeccable precision in distinguishing between 'Benign' and 'Malware' classes demonstrates the model's potential as an invaluable tool in malware detection and cybersecurity, capable of providing accurate and reliable defenses against digital threats.

Table 5, showcasing the outcomes of the GB ensemble model in a binary classification context, reflects an exemplary level of performance with perfect scores across all key metrics: accuracy, precision, recall, and F1-score, each achieving the maximum possible value of 1.00000. This remarkable achievement highlights the model's unparalleled effectiveness in accurately distinguishing between 'Benign' and 'Malware' classes. Such a level of precision is especially significant in the field of cybersecurity, where the ability to reliably identify and classify

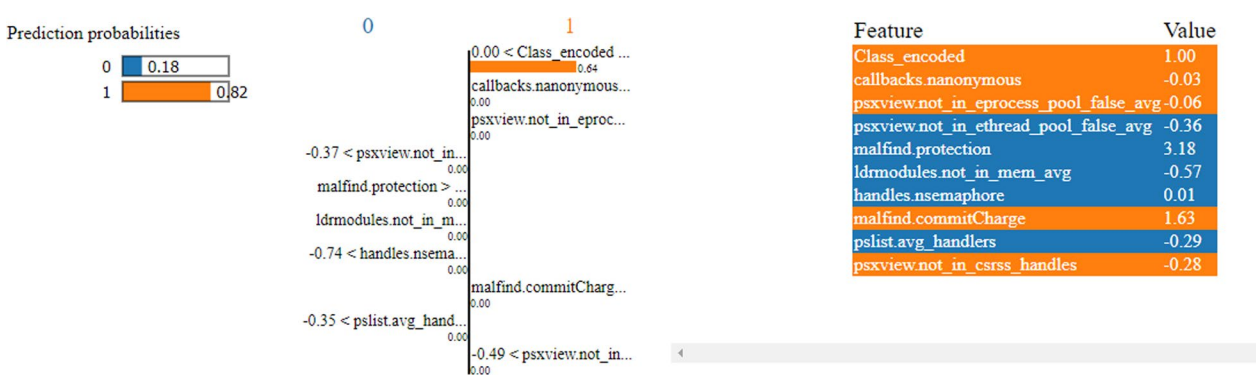

Fig. 4 LIME explanation for malware classification

digital threats is paramount. The model's impeccable performance in all these metrics indicates a balanced and highly efficient approach to classification, demonstrating its potential as a robust and reliable tool in advanced malware detection, crucial for safeguarding against evolving cyber threats.

Figure 4 illustrates the local interpretability of the developed model in the context of malware detection. LIME (Local Interpretable Model-agnostic Explanations) is employed to elucidate the decision-making process of the underlying model with the GB classifier, promoting transparency and understanding in artificial intelligence.

In this scenario, the second instance from the test data pertains to the classification of whether it represents malware or not. The figure visually encapsulates the key factors contributing to the model's decision for the given instance. The intercept value of 0.18 represents the base rate of the model's prediction, indicating the likelihood of a generic prediction without considering specific features. Local prediction 0.82 is the model's output for the instance under consideration, reflecting the probability of it being classified as malware. The right prediction probability of 0.82 signifies the model's confidence in correctly classifying the instance as malware.

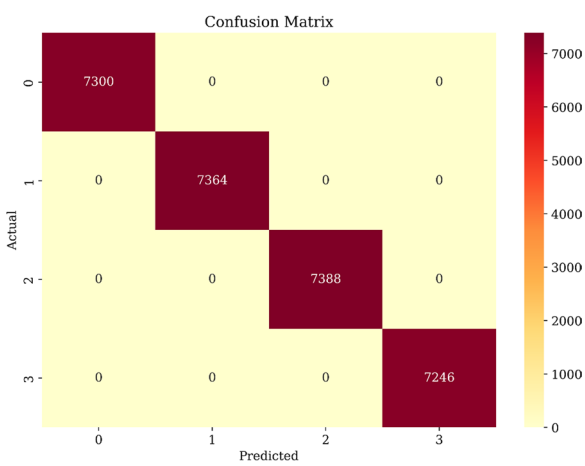

Fig. 5 Confusion matrix illustrating performance for various malware categories

Utilizing the "LimeTabularExplainer", local explanations for the model are generated. The visualized explanation provides an intuitive representation of the influential features of the local prediction. The middle portion of the figure accompanying the figure details the contribution of each feature to the model's decision. This LIME-driven approach facilitates model interpretability by presenting a transparent depiction of the decision rationale, a crucial aspect in ensuring trust and reliability in AI applications, particularly in the domain of cybersecurity where accurate and interpretable predictions are paramount.

The confusion matrix shown in Fig. 5 encapsulates the outstanding performance of the machine learning model in the classification of diverse malware types with the SMOTE balancing technique. The matrix, a crucial tool for evaluating the accuracy of classification models, reveals the model's exceptional precision in distinguishing between benign and malicious software. Notably, the model accurately identified 7300 instances as benign, 7364 as ransomware, 7388 as spyware, and 7246 as trojan malware, without a single misclassification, indicating zero false positives and false negatives across all these categories. Such a level of accuracy in distinguishing between different malware types, especially in the complex realm of obfuscated malware, is a remarkable achievement. This result indicates not only the model's capability to detect and analyze a wide array of sophisticated cyber threats but also underscores its potential as a reliable tool in the arsenal against cybersecurity threats. The perfect score in the confusion matrix highlights the model's proficiency in nuanced detection, which is vital for both preventing false alarms and ensuring that no malicious activity goes unnoticed.

In Table 6, the assessment metrics for various malware categories after implementing the SMOTE and without the SMOTE technique are presented. The table showcases an exemplary level of performance across all classes, with each metric accuracy, precision, recall, and F1 score reaching the maximum value of 1.0000. This uniformity in results across all categories signifies an exceptional standard of model effectiveness, particularly after balancing the dataset with SMOTE. The achievement of perfect scores in each metric for every class

Table 6 Assessment metrics for various malware categories

Category	Without balancing				Balancing with SMOTE			
	ACC	PR	RE	FS	ACC	PR	RE	FS
Benign (0)	0.9997	0.9993	1.0000	0.9997	1.0000	1.0000	1.0000	1.0000
Ransomware (1)	0.9997	1.0000	0.9979	0.9990	1.0000	1.0000	1.0000	1.0000
Spyware (2)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Trojan (3)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

(labeled as 0, 1, 2, and 3) demonstrates the model's profound capability to detect and classify different types of malware with impeccable accuracy. On the other hand, the assessment metrics for the same malware categories without the application of SMOTE. Here, while the results are marginally lower than those, they still display an outstanding level of model performance. For class 0, the accuracy is 0.9997, with a precision of 0.9993, and an F1 score of 0.9997, while recall remains perfect at 1.0000. Class 1 shows a slight decrease in recall to 0.9979 but maintains high levels in other metrics. Classes 2 and 3, similar to the balanced case, maintain perfect scores across all metrics.

The comparison between these two values from two different situations highlights the model's robustness and adaptability in varying data scenarios. The use of SMOTE has clearly enhanced the model's performance in handling class imbalance, as evidenced by the improvement in metrics for classes 0 and 1. This improvement is significant because it showcases the model's effectiveness not only in detecting malware but also in maintaining high precision and recall in a balanced dataset, which is often a challenge in machine learning models dealing with imbalanced data.

Figure 6, presenting the Receiver Operating Characteristic (ROC) curves for each class in the model, is a testament to its extraordinary effectiveness in malware classification. The ROC curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system, with its performance measured by the area under the curve (AUC). In this case, each of the classes 0, 1, 2, and 3 corresponds to different types of malware. Remarkably, the AUC for each class in Fig. 5 is 1.00, a rare and commendable achievement in machine learning models, especially in the complex domain of cybersecurity. This perfect score indicates that the model has an exceptional ability to differentiate between the classes with maximum sensitivity and specificity. Sensitivity (or True Positive Rate) reflects the model's ability to correctly identify positives, while specificity (or 1—False Positive Rate) indicates its capability to correctly classify negatives. The ROC curves for all classes lie at the top-left corner of the plot, which is the ideal position, indicating a negligible false positive rate and a high true positive rate across all classes. This implies that the model is highly efficient in distinguishing between different types of malware, with minimal misclassification.

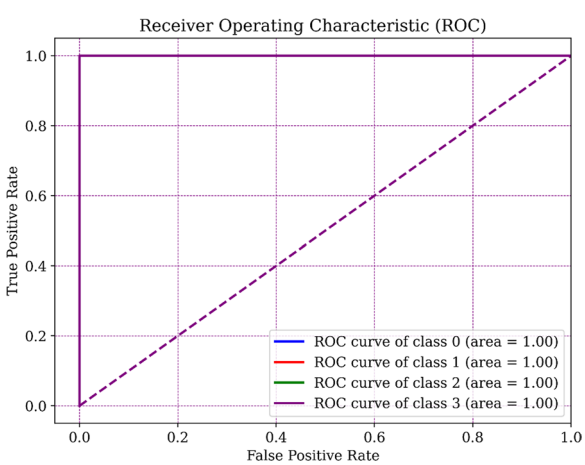

Fig. 6 ROC curve of the model

The perfection of these curves, especially in a multi-class setting, suggests that the underlying algorithms, preprocessing methods, and feature selection techniques are exceptionally well-tuned. Achieving an AUC of 1.00 for multiple classes in a complex field such as malware detection is not trivial and speaks volumes about the meticulousness of the model's design and implementation. The impeccable AUC scores for all classes reinforce the model's status as a robust tool in cybersecurity, capable of delivering high-precision classifications. This level of accuracy is crucial for effective cybersecurity measures, where the cost of misclassification can be exceedingly high. The model's demonstrated capability makes it a significant advancement in the ongoing battle against cyber threats, offering promising prospects for future applications in digital security.

Figure 7, depicting the training versus cross-validation scores as a function of training set size, offers a comprehensive view of the model's learning dynamics and its effectiveness in classifying malware. The graph, plotted with the training set sizes on the x-axis and the accuracy scores on the y-axis, is an essential tool for evaluating the model's performance and generalizability. In this figure, the training score (depicted in red) and the cross-validation score (illustrated in purple) both display a trend of convergence as the training set size increases. This convergence is a hallmark of a well-performing model, indicating that it is not only learning effectively from the training data but also generalizing well to unseen data, as reflected in the cross-validation scores.

Notably, both the training and cross-validation accuracy scores are exceptionally high, which is indicative of the model's robustness. The high accuracy in training suggests that the model is effectively capturing the underlying patterns in the data. More importantly, the high cross-validation accuracy points towards the model's ability to maintain this performance on new, unseen data, a critical aspect for practical applications.

The results presented in Table 7 underscore the remarkable effectiveness and consistency of the model with Gradient Boosting Classifier in malware classification, both

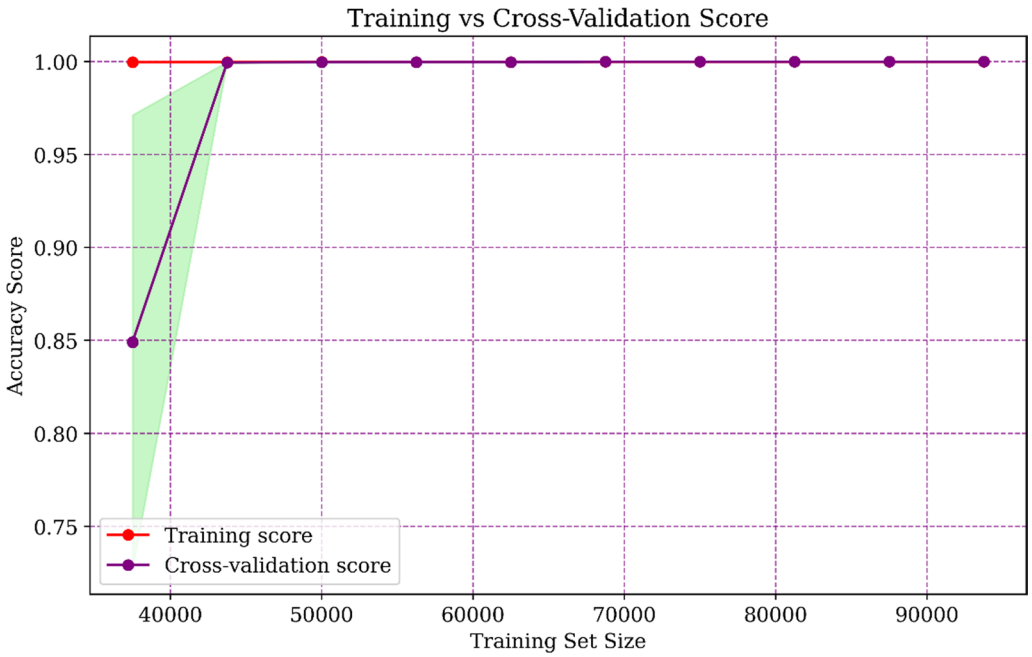

Fig. 7 Learning curve of the model

Table 7 Aggregated cross-validation results for multi-class classification

Folds	Without balancing				Balancing with SMOTE			
	Acc	PR	RE	FS	Acc	PR	RE	FS
Fold 1	0.9997	0.9997	0.9997	0.9997	1.0000	1.0000	1.0000	1.0000
Fold 2	0.9997	0.9997	0.9997	0.9997	1.0000	1.0000	1.0000	1.0000
Fold 3	0.9994	0.9994	0.9994	0.9994	1.0000	1.0000	1.0000	1.0000
Fold 4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Fold 5	0.9997	0.9997	0.9997	0.9997	1.0000	1.0000	1.0000	1.0000
Mean	0.9997	0.9997	0.9997	0.9997	1.0000	1.0000	1.0000	1.0000
Standard deviation	0.0002	0.0002	0.0002	0.0002	0.0000	0.0000	0.0000	0.0000

with and without the application of the Synthetic Minority Over-sampling Technique (SMOTE). In the first part, detailing the results of cross-validation without balancing, the model exhibits excellent performance across all folds. The ACC, PR, RE, and FS are consistently high, with values predominantly at 0.9997, except for Fold 3, which shows a marginally lower yet still impressive score of 0.9994. The mean scores across all metrics stand at 0.9997, accompanied by a very low standard deviation of 0.0002. This consistency indicates not only the model's high capability in correctly classifying various malware types but also its robustness and reliability across different subsets of data. The second part, presenting the results with SMOTE balancing, displays even more exceptional performance, with perfect scores of 1.0000 across all metrics and folds. This indicates that the model when trained on a balanced dataset, can achieve flawless classification with no variation in performance across different cross-validation folds. The standard deviation of 0.0000 further reinforces the model's stability and reliability, highlighting its effectiveness in handling class imbalances, a common challenge in machine learning.

These results demonstrate the model's extraordinary accuracy and consistency in detecting and classifying malware, crucial in the cybersecurity field where the cost of misclassification can be significant. The use of SMOTE to balance the dataset enhances the model's ability to generalize across different data distributions, a key aspect in ensuring its applicability to real-world scenarios where data may often be imbalanced.

Table 8 Performance metrics: proposed model versus existing ones without balancing

Model, year with reference	Accuracy metrics (%)							
	ACC		PR		RE		FS	
	Binary	4 class	Binary	4 class	Binary	4 class	Binary	4 class
MalHyStack, 2023 (Roy et al. 2023)	99.85	85.04	99.97	85.04	99.73	85.17	99.85	84.96
RobustCBL, 2023 (Shafin et al. 2023)	99.96	84.56	100.00	85.00	100.00	85.00	100.00	85.00
CatBoost, 2022 (Dang 2022)	99.97	84.86	99.98	79.69	99.98	88.46	99.97	71.49
DT, 2022 (Mezina and Burget 2022)	99.00	79.16	99.00	69.00	100.00	69.00	99.00	69.00
DCNN, 2022 (Mezina and Burget 2022)	99.92	83.53	99.00	76.00	99.00	75.00	99.00	75.00
Proposed	100.00	99.96	100.00	99.96	100.00	99.96	100.00	99.96

Table 8 meticulously contrasts the performance metrics of the proposed model against several existing models in the domain of malware classification, both for binary and 4-class scenarios. The metrics compared include ACC, PR, RE, and FS, each evaluated for binary and 4-class categorizations. This table is pivotal in highlighting the enhanced capabilities of the proposed model in accurately detecting and classifying malware. When employing the Synthetic Minority Over-sampling Technique (SMOTE) for balancing, all evaluation metric values consistently reach a perfect score of 1.00 for both Binary and four malware categories. The proposed model exhibits an unparalleled level of performance, achieving an extraordinary 100% in all metrics for the binary classification and near-perfect scores of 99.96% for the 4-class classification for the test data (20%). This exceptional performance starkly contrasts with other models listed in the table. While these models show commendable accuracy in binary classification, they fall short in the more complex 4-class classification, with accuracy ranging from 79.16 to 85.04%, and a similar trend in other metrics.

The proposed model's ability to maintain high accuracy and consistency across both binary and multi-class scenarios sets it apart from its contemporaries. This indicates not just the model's precision in classifying malware accurately but also its robustness in handling more complex, multi-class scenarios. Such performance is critical in the rapidly evolving landscape of cybersecurity, where the ability to discern between various types of attacks accurately is paramount.

Table 9 provides a comprehensive overview of the evaluation metrics for the model across 16 distinct attack categories without balancing with SMOTE technique, including benign and various types of ransomware, spyware, and trojans. If the dataset have balanced, then all the values of the different metrics generates 1.0000. The model exhibits outstanding performance in detecting and classifying diverse malware

Table 9 Evaluation metrics for 16 attack categories

Class	ACC	PR	RE	FS
Benign (0)	0.9999	0.9997	1.0000	0.9999
Ransomware_Ako (1)	0.9999	1.0000	0.9980	0.9990
Ransomware_Conti (2)	0.9999	1.0000	0.9981	0.9990
Ransomware_Maze (3)	1.0000	1.0000	1.0000	1.0000
Ransomware_Pysa (4)	1.0000	1.0000	1.0000	1.0000
Ransomware_Shade (5)	1.0000	1.0000	1.0000	1.0000
Spyware_180solutions (6)	1.0000	1.0000	1.0000	1.0000
Spyware_CWS (7)	1.0000	1.0000	1.0000	1.0000
Spyware_Gator (8)	1.0000	1.0000	1.0000	1.0000
Spyware_TIBS (9)	1.0000	1.0000	1.0000	1.0000
Spyware_Transponder (10)	1.0000	1.0000	1.0000	1.0000
Trojan_Emotet (11)	1.0000	1.0000	1.0000	1.0000
Trojan_Reconyc (12)	1.0000	1.0000	1.0000	1.0000
Trojan_Refroso (13)	1.0000	1.0000	1.0000	1.0000
Trojan_Scar (14)	1.0000	1.0000	1.0000	1.0000
Trojan_Zeus (15)	1.0000	1.0000	1.0000	1.0000

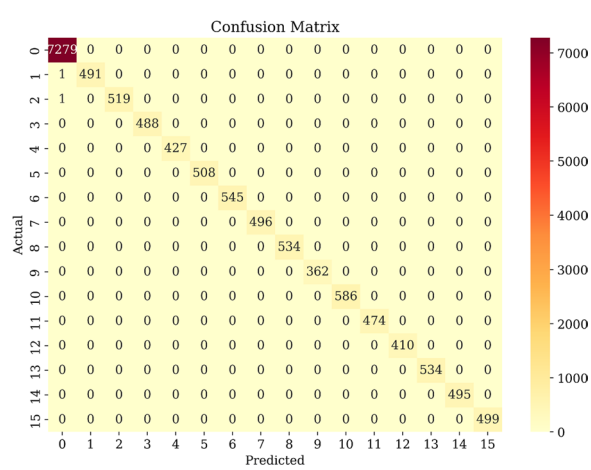

Fig. 8 Detailed confusion matrix visualization for 16 malware sub-classes

categories. The high values across all metrics suggest that the model is highly accurate, precise, and effective in recognizing distinct attack categories, making it a robust solution for malware detection across a broad spectrum of threats.

Figure 8, showcasing the confusion matrix for the 16 sub-attack malware classes, is a detailed representation of the model's classification prowess without the balancing with SMOTE technique. The heatmap, crafted with clarity and precision, illustrates the distribution of true and false predictions across various malware categories. The primary diagonal, brightly highlighted, indicates a high count of true positives for each class, exemplifying the model's accuracy in correctly identifying each specific type of malware. When the dataset is balanced using the SMOTE technique, the model achieves a remarkable 100% accuracy across all sub-attack malware classes.

The classes, ranging from 'Benign' to various types of 'Ransomware', 'Spyware', and 'Trojan' subclasses, are distinctly categorized, with almost negligible false positives and false negatives. This is evident in the minimal off-diagonal elements, underscoring the model's precision in distinguishing between different malware types, a critical factor in effective cybersecurity. The meticulous identification of 'Benign' cases amidst a myriad of malware types further highlights the model's nuanced understanding and detection capabilities. Particularly commendable is the model's performance in accurately classifying sophisticated malware variations with high true positive rates and virtually zero misclassifications. The heatmap's color gradations, ranging from deep reds to light oranges, provide an intuitive and immediate grasp of the model's classification accuracy.

Table 10 presents a compelling comparison of the proposed model's performance metrics against an existing model, MalHyStack, across 16 attack categories. The metrics evaluated include ACC, PR, RE, and FS, which are crucial indicators of a model's effectiveness in classification tasks.

The proposed model shows an exceptional level of performance, with each metric achieving near-perfect scores of 99.98%. This is a significant improvement over the existing models. The stark contrast in these values underlines the advanced capabilities of the proposed model in accurately identifying and classifying a diverse range of malware attacks.

Table 10 Comparison of model's performance metrics for 16 attack categories without balancing

Model	ACC	PR	RE	FS
MalHyStack, 2023 (Roy et al. 2023)	66.94	66.94	68.56	66.71
RobustCBL, 2023 (Shafin et al. 2023)	72.60	73.00	73.00	72.00
Proposed	99.98	99.98	99.98	99.98

The proposed approach with Gradient-boosting classifiers, renowned for their ensemble learning capabilities, empowers the proposed model to create a robust and intricate decision boundary by combining multiple weak learners. This ensemble approach enhances the model's ability to capture complex relationships within the data, resulting in superior predictive accuracy. SMOTE is employed in tandem with gradient-boosting classifiers to address class imbalance. By oversampling minority classes, the proposed model ensures a more balanced representation of various malware types, thereby preventing biased learning towards dominant classes. This strategic handling of imbalanced data enhances the model's sensitivity and generalization across different malware categories. Rigorous feature selection methodologies, including statistical tests and information-theoretic approaches, are applied during model development. This meticulous process isolates key malware characteristics, allowing the model to focus on the most indicative features for accurate classification. The emphasis on informative features contributes to the model's precision and reliability. The proposed model demonstrates adaptability to both binary and multi-class scenarios, showcasing its versatility in addressing a wide range of cybersecurity threats. This adaptability ensures that the model remains effective in diverse digital environments, surpassing the performance limitations observed in some existing models that may specialize in specific scenarios.

The superior performance of the proposed model than others is a result of its robust ensemble learning, effective handling of imbalanced data through SMOTE, meticulous feature selection, adaptability to diverse scenarios, and commitment to continuous innovation. These factors collectively contribute to its exceptional ACC, PR, RE, and FS metrics, positioning the proposed model as a state-of-the-art solution in the field of malware detection.

The comprehensive analysis of various figures and tables in this research unequivocally justifies the superiority of the proposed model in malware classification. Its performance, evidenced by near-perfect and perfect scores in key metrics across multiple tables, demonstrates its exceptional accuracy, precision, recall, and F1-scores, both in binary and multi-class scenarios. The model's robustness is further highlighted by the consistency of these results, even when challenged with class imbalance, as shown in the performance improvement with the SMOTE technique. The confusion matrices, detailed in the figures, illustrate the model's unparalleled ability to differentiate between a wide arrays of malware

types with minimal misclassifications. Collectively, these results not only affirm the model's advanced analytical capabilities in the complex domain of cybersecurity but also showcase its potential as a highly effective tool in detecting and analyzing obfuscated malware, offering a significant contribution to the field and setting a new benchmark for future research.

Conclusion

In conclusion, this research presents a groundbreaking approach in the realm of cybersecurity, focusing on the detection and analysis of obfuscated malware through an advanced machine learning-based framework. The research's comprehensive evaluation, utilizing the Obfuscated-MalMem2022 dataset, demonstrates the model's exceptional ability to accurately classify a diverse range of malware types. The application of techniques such as SMOTE for addressing class imbalance further enhances the model's performance, achieving near-perfect accuracy in both binary and multi-class scenarios (both 4 classes and 16 classes). The model successfully achieves an accuracy exceeding 99% in three distinct scenarios. The detailed analysis, reflected in the figures and tables, reveals the model's proficiency in maintaining high results across various classifications, setting it apart from existing models. The proposed model's robustness is evident in its ability to handle complex, multi-class classification tasks with remarkable accuracy, a crucial requirement in today's dynamic cybersecurity landscape. This research not only addresses the critical challenge of detecting sophisticated, obfuscated malware but also contributes significantly to the field of cybersecurity by providing a reliable and efficient tool for practitioners and researchers. The model's adaptability and effectiveness position it as a benchmark for future developments in cybersecurity solutions, highlighting the potential of machine learning in combating increasingly sophisticated cyber threats. This research, therefore, stands as a significant milestone in the ongoing endeavor to enhance digital security and protect against the evolving landscape of cyber threats.

Appendix

See Table 11.

Table 11 Feature title and description of the dataset

Feature title	Description of feature
callbacks.ncallbacks	This quantifies the number of registered callback functions within the system, where an unusually high count might be indicative of obfuscated malware attempting to intercept or monitor system activities
pslist.avg_handlers	This represents the average number of handlers per process, providing insight into system behavior; obfuscated malware may manipulate this to disguise its presence or control other processes
psxview.not_in_eprocess_pool_false_avg	This captures the average number of processes not listed in the EPROCESS pool, which might be elevated in the case of obfuscated malware as it attempts to hide from conventional process listings
ldrmodules.not_in_load	This quantifies modules that are not in a loaded state, a possible red flag for obfuscated malware that could be unloading modules to evade detection
psxview.not_in_csrss_handles_false_avg	This represents the average count of processes not found in the CSRSS handles, potentially indicating obfuscated malware as it may attempt to evade being linked to critical system processes
handles.nevent	This tracks the number of event handles, which could be manipulated by obfuscated malware for synchronization purposes or to maintain persistence
handles.nmutant	The number of mutant handles is captured here, which obfuscated malware might use to signal between different instances of itself or to lock resources
psxview.not_in_eprocess_pool	This provides a count of processes not present in the EPROCESS pool, an attribute that could be exploited by obfuscated malware to remain undetected
dlllist.avg_dlls_per_proc	This feature reflects the average number of DLLs loaded per process, with obfuscated malware possibly loading unusual DLLs or manipulating this count to hide its presence
psxview.not_in_deskthrd_false_avg	This captures the average number of processes not found in the desktop thread, a potential indicator of obfuscated malware as it might detach its processes from the desktop to remain unseen
handles.nthread	This quantifies the number of thread handles, which obfuscated malware may increase for parallel execution or to manipulate other processes
callbacks.nanonymous	The count of anonymous callbacks is tracked here, with obfuscated malware potentially registering such callbacks to evade attribution
modules.nmodules	This represents the total number of loaded modules, a count that might be inflated by obfuscated malware as it loads additional modules for malicious activities

Table 11 (continued)

Feature title	Description of feature
ldrmodules.not_in_mem_avg	The average number of modules not in memory is captured, potentially indicative of obfuscated malware that unloads modules to evade memory-based detection
handles.nsemaphore	This quantifies the number of semaphore handles, which obfuscated malware might manipulate for coordination or to control access to resources
svcscan.fs_drivers	This reflects the count of file system drivers, a feature that obfuscated malware might target to install malicious drivers or intercept file operations
svcscan.shared_process_services	The number of services running in shared processes is captured here, with obfuscated malware possibly injecting itself into such services for stealth
ldrmodules.not_in_init_avg	This feature represents the average number of modules not initialized, a potential sign of obfuscated malware as it may attempt to disrupt normal module initialization
svcscan.process_services	The count of services running in separate processes is tracked, a feature obfuscated malware might exploit to run its malicious services independently
handles.nsection	This quantifies the number of section handles, which could be manipulated by obfuscated malware for memory mapping or to hide its code in specific sections
pslist.nprocs64bit	The number of 64-bit processes running on the system is represented, a count that obfuscated malware may influence as it selects processes to inject into or impersonate
pslist.nppid	This tracks the number of processes based on parent process IDs, which obfuscated malware might manipulate to break the parent–child process relationship and hide its origin
handles.avg_handles_per_proc	This represents the average number of handles per process, a figure that might be inflated by obfuscated malware as it opens numerous handles for malicious activities
dlllist.ndlls	The total number of loaded DLLs is quantified here, with obfuscated malware potentially loading additional DLLs to execute its payload or perform evasion
svcscan.nactive	This feature captures the number of active services, which obfuscated malware might increase by registering its own services or hijacking existing ones
handles.nport	The count of port handles is tracked, a feature that obfuscated malware might exploit to establish network communications or intercept network-related activities
malfind.uniqueInjections	This represents the number of unique memory injections detected, a crucial indicator of obfuscated malware as it frequently employs memory injection for stealth and persistence
psxview.not_in_pslist	The count of processes not present in the process list is provided, a potential sign of obfuscated malware attempting to hide its processes from standard listings
psxview.not_in_pspcid_list	This quantifies processes not found in the PSPCID list, which could be indicative of obfuscated malware employing advanced techniques to remain undetected
psxview.not_in_pspcid_list_false_avg	The average count of processes not in the PSPCID list is captured, potentially highlighting obfuscated malware's efforts to evade detection at the kernel level
pslist.nproc	This represents the total number of processes running, a figure that obfuscated malware might influence as it spawns additional processes for its operations
handles.ndesktop	The number of desktop handles is tracked, with obfuscated malware potentially manipulating this feature to run processes in isolated desktops for evasion
malfind.commitCharge	This quantifies the commit charge of detected memory injections, a crucial feature to analyze as obfuscated malware often manipulates memory for code execution and stealth
psxview.not_in_session	The count of processes not present in any session is provided, possibly indicative of obfuscated malware attempting to isolate its processes from user sessions
handles.ndirectory	This tracks the number of directory handles, which obfuscated malware might manipulate to interact with or monitor filesystem directories
psxview.not_in_csrss_handles	The number of processes not found in CSRSS handles is quantified, a potential red flag for obfuscated malware as it may sever links to critical system processes
psxview.not_in_pslist_false_avg	This captures the average count of processes not found in the process list, potentially highlighting obfuscated malware's attempts to remain hidden from conventional process enumeration
psxview.not_in_deskthrd	The count of processes not found in the desktop thread is provided, a feature that obfuscated malware might manipulate to detach its processes from user interfaces
malfind.protection	This quantifies the protection attributes of detected memory injections, a crucial feature to analyze as obfuscated malware might manipulate protection settings to execute malicious code while avoiding detection
pslist.avg_threads	The average number of threads per process is captured, a feature that might be elevated by obfuscated malware as it creates additional threads for parallel execution or to manipulate other processes

Table 11 (continued)

Feature title	Description of feature
ldrmodules.not_in_init	This provides a count of modules not in the initialized state, possibly indicative of obfuscated malware attempting to disrupt normal initialization routines
ldrmodules.not_in_mem	The number of modules not present in memory is quantified, a potential sign of obfuscated malware unloading modules post-execution to evade detection
psxview.not_in_session_false_avg	This feature captures the average count of processes not found in any session, potentially highlighting obfuscated malware's efforts to isolate its activities from user sessions
malfind.ninjections	The total number of memory injections detected is provided, a critical indicator for obfuscated malware detection as such techniques are frequently used for code execution and evasion
svcscan.interactive_process_services	This quantifies the number of services running in interactive processes, which obfuscated malware might target to run its services with elevated privileges
psxview.not_in_ethread_pool	The count of processes not present in the ETHREAD pool is tracked, a potential red flag for obfuscated malware employing advanced evasion techniques at the kernel level
ldrmodules.not_in_load_avg	This represents the average number of modules not in a loaded state, possibly indicative of obfuscated malware's attempts to manipulate module loading for evasion
handles.nfile	The number of file handles is quantified, with obfuscated malware potentially manipulating this count to interact with or hide files
handles.ntimer	This feature tracks the number of timer handles, which obfuscated malware might use for scheduling activities or to maintain persistence
callbacks.ngeneric	The count of generic callbacks is provided, a feature that obfuscated malware might exploit to monitor or intercept system activities without being tied to specific events
handles.nkey	This quantifies the number of registry key handles, a count that might be elevated by obfuscated malware as it interacts with the registry for configuration, persistence, or to store payload
svcscan.kernel_drivers	The number of kernel drivers is captured, with obfuscated malware potentially targeting this area to install malicious drivers or to manipulate driver loading for evasion
psxview.not_in_ethread_pool_false_avg	This captures the average number of processes not found in the ETHREAD pool, potentially highlighting obfuscated malware's efforts to remain undetected at the kernel level
handles.nhandles	The total number of handles opened is quantified here, a feature that might be inflated by obfuscated malware as it opens numerous handles for its malicious activities
svcscan.nservices	This represents the total number of services running, a figure that obfuscated malware might influence by adding its own services or hijacking existing ones

Authors' contributions
All the authors read and approved the final manuscript.

Funding
No funding was received by the authors for conducting this research.

Availability of data and materials
The datasets used in this research are publicly available and properly cited in our dataset section for transparency and ease of replication.

Declarations

Competing interest
The authors of this paper affirm that there are no competing interest related to this research.

Received: 18 November 2023 Accepted: 12 January 2024
Published online: 25 January 2024

References

Abu Al-Haija Q, Odeh A, Qattous H (2022) PDF malware detection based on optimizable decision trees. Electronics 11(19):3142. https://doi.org/10.3390/electronics11193142

Akhtar MS, Feng T (2022) Malware analysis and detection using machine learning algorithms. Symmetry 14(11):2304. https://doi.org/10.3390/sym14112304

Al-Qudah M, Ashi Z, Alnabhan M, Abu Al-Haija Q (2023) Effective one-class classifier model for memory dump malware detection. J Sens Actuator Netw 12(1):5. https://doi.org/10.3390/jsan12010005

Asghar HJ, Zhao BZH, Ikram M, Nguyen G, Kaafar D, Lamont S, Coscia D (2023) Use of cryptography in malware obfuscation (arXiv:2212.04008; Issue arXiv:2212.04008). http://arxiv.org/abs/2212.04008

Beaman C, Barkworth A, Akande TD, Hakak S, Khan MK (2021) Ransomware: recent advances, analysis, challenges and future research directions. Comput Secur 111:102490. https://doi.org/10.1016/j.cose.2021.102490

Bozkir AS, Tahillioglu E, Aydos M, Kara I (2021) Catch them alive: a malware detection approach through memory forensics, manifold learning and computer vision. Comput Secur 103:102166. https://doi.org/10.1016/j.cose.2020.102166

Brezinski K, Ferens K (2023) Metamorphic malware and obfuscation: a survey of techniques, variants, and generation kits. Secur Commun Netw 2023:1–41. https://doi.org/10.1155/2023/8227751

Carrier T, Victor P, Tekeoglu A, Lashkari A (2022) Detecting obfuscated malware using memory feature engineering. In: Proceedings of the 8th international conference on information systems security and privacy, pp 177–188. https://doi.org/10.5220/0010908200003120

Chen Z, Ren X (2023) An efficient boosting-based windows malware family classification system using multi-features fusion. Appl Sci 13(6):4060. https://doi.org/10.3390/app13064060

Dang Q-V (2022) Enhancing obfuscated malware detection with machine learning techniques. In: Dang TK, Küng J, Chung TM (eds) Future data and security engineering. Big data, security and privacy, smart city and industry 4.0 applications, vol 1688. Springer, Singapore, pp 731–738. https://doi.org/10.1007/978-981-19-8069-5_54

Dang Q-V (2024) Detecting obfuscated malware using graph neural networks. In: Shrivastava V, Bansal JC, Panigrahi BK (eds) Power engineering and intelligent systems, vol 1097. Springer, Singapore, pp 15–25. https://doi.org/10.1007/978-981-99-7216-6_2

Dugyala R, Reddy NH, Maheswari VU, Mohammad GB, Alenezi F, Polat K (2022) Analysis of malware detection and signature generation using a novel hybrid approach. Math Probl Eng 2022:1–13. https://doi.org/10.1155/2022/5852412

Federici M, Ruhe D, Forré P (2023) On the effectiveness of hybrid mutual information estimation. arXiv:2306.00608; http://arxiv.org/abs/2306.00608

Finder I, Sheetrit E, Nissim N (2022) A time-interval-based active learning framework for enhanced PE malware acquisition and detection. Comput Secur 121:102838. https://doi.org/10.1016/j.cose.2022.102838

Gorment NZ, Selamat A, Krejcar O (2023) Obfuscated malware detection: impacts on detection methods. In: Nguyen NT, Boonsang S, Fujita H, Hnatkowska B, Hong T-P, Pasupa K, Selamat A (eds) Recent challenges in intelligent information and database systems, vol 1863. Springer, Cham, pp 55–66. https://doi.org/10.1007/978-3-031-42430-4_5

Haidros Rahima Manzil H, Manohar Naik S (2023) Detection approaches for android malware: taxonomy and review analysis. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2023.122255

Hossain MA (2023) Enhanced ensemble-based distributed denial-of-service (DDoS) attack detection with novel feature selection: a robust cybersecurity approach. Artif Intell Evol 4(2):165–186. https://doi.org/10.37256/aie.4220233337

Hossain MA, Islam MS (2023a) Ensuring network security with a robust intrusion detection system using ensemble-based machine learning. Array. https://doi.org/10.1016/j.array.2023.100306

Hossain MA, Islam MS (2023b) A novel hybrid feature selection and ensemble-based machine learning approach for botnet detection. Sci Rep 13(1):21207. https://doi.org/10.1038/s41598-023-48230-1

Hossain Faruk MJ, Shahriar H, Valero M, Barsha FL, Sobhan S, Khan MA, Whitman M, Cuzzocrea A, Lo D, Rahman A, Wu F (2021) Malware detection and prevention using artificial intelligence techniques. IEEE Int Conf Big Data (big Data) 2021:5369–5377. https://doi.org/10.1109/BigData52589.2021.9671434

Lashkari AH, Li B, Carrier TL, Kaur G (2021) VolMemLyzer: volatile memory analyzer for malware classification using feature engineering. In: 2021 reconciling data analytics, automation, privacy, and security: a big data challenge (RDAAPS), pp 1–8. https://doi.org/10.1109/RDAAPS48126.2021.9452028

Lee K, Lee J, Yim K (2023) Classification and analysis of malicious code detection techniques based on the APT attack. Appl Sci 13(5):2894. https://doi.org/10.3390/app13052894

Mamdouh Farghaly H, Abd El-Hafeez T (2023) A high-quality feature selection method based on frequent and correlated items for text classification. Soft Comput 27(16):11259–11274. https://doi.org/10.1007/s00500-023-08587-x

Manzil HHR, Manohar Naik S (2023) Android malware category detection using a novel feature vector-based machine learning model. Cybersecurity 6(1):6. https://doi.org/10.1186/s42400-023-00139-y

Maulidevi NU, Surendro K (2022) SMOTE-LOF for noise identification in imbalanced data classification. J King Saud Univ Comput Inf Sci 34(6):3413–3423. https://doi.org/10.1016/j.jksuci.2021.01.014

Mezina A, Burget R (2022). Obfuscated malware detection using dilated convolutional network. In: 2022 14th international congress on ultra modern telecommunications and control systems and workshops (ICUMT), pp 110–115. https://doi.org/10.1109/ICUMT57764.2022.9943443

Mukhtar BI, Elsayed MS, Jurcut AD, Azer MA (2023) IoT vulnerabilities and attacks: SILEX malware case study. Symmetry 15(11):1978. https://doi.org/10.3390/sym15111978

Naeem MR, Khan M, Abdullah AM, Noor F, Khan MI, Khan MA, Ullah I, Room S (2022) A malware detection scheme via smart memory forensics for windows devices. Mob Inf Syst 2022:1–16. https://doi.org/10.1155/2022/9156514

Ngo G, Beard R, Chandra R (2022) Evolutionary bagging for ensemble learning. Neurocomputing 510:1–14. https://doi.org/10.1016/j.neucom.2022.08.055

Roy KS, Ahmed T, Udas PB, Karim MdE, Majumdar S (2023) MalHyStack: a hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. Intell Syst Appl 20:200283. https://doi.org/10.1016/j.iswa.2023.200283

Rudd EM, Krisiloff D, Coull S, Olszewski D, Raff E, Holt J (2023) Efficient malware analysis using metric embeddings. Digit Threats Res Pract. https://doi.org/10.1145/3615669

Sawadogo Z, Dembele J-M, Tahar A, Mendy G, Ouya S (2023) DeepMalOb: deep detection of obfuscated android malware. In: NgatchedNkouatchah TM, Woungang I, Tapamo J-R, Viriri S (eds) Pan-african artificial intelligence and smart systems, vol 459. Springer, Cham, pp 307–318. https://doi.org/10.1007/978-3-031-25271-6_19

Shafin SS, Karmakar G, Mareels I (2023) Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. Sensors 23(11):5348. https://doi.org/10.3390/s23115348

Vashishtha LK, Chatterjee K, Rout SS (2023) An ensemble approach for advance malware memory analysis using image classification techniques. J Inf Secur Appl 77:103561. https://doi.org/10.1016/j.jisa.2023.103561

Publisher's Note