DOI: https://doi.org/10.52436/1.jutif.2024.5.6.3903
p-ISSN: 2723-3863
e-ISSN: 2723-3871

IMPROVING MALWARE DETECTION USING INFORMATION GAIN AND ENSEMBLE MACHINE LEARNING

Arsabilla Ramadhani[1], Fauzi Adi Rafrastara[*2], Salma Rosyada[3], Wildanil Ghozi[4], Waleed Mahgoub Osman[5]

[1,2,3,4]Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Indonesia
[5]Mathematics Department, College of Education, Sudan University of Science and Technology, Sudan
Email: [1]111202113384@mhs.dinus.ac.id, [2]fauziadi@dsn.dinus.ac.id, [3]111202113382@mhs.dinus.ac.id, [4]wildanil.ghozi@dsn.dinus.ac.id, [5]waleedmo@sustech.edu

(Article received: October 14, 2024; Revision: November 22, 2024; published: December 29, 2024)

Abstract

Malware attacks pose a serious threat to digital systems, potentially causing data and financial losses. The increasing complexity and diversity of malware attack techniques have made traditional detection methods ineffective, thus AI-based approaches are needed to improve the accuracy and efficiency of malware detection, especially for detecting modern malware that uses obfuscation techniques. This study addresses this issue by applying ensemble-based machine learning algorithms to enhance malware detection accuracy. The methodology used involves Random Forest, Gradient Boosting, XGBoost, and AdaBoost, with feature selection using Information Gain. Datasets from VirusTotal and VxHeaven, including both goodware and malware samples. The results show that Gradient Boosting, strengthened with Information Gain, achieved the highest accuracy of 99.1%, indicating a significant improvement in malware detection effectiveness. This study demonstrates that applying Information Gain to Gradient Boosting can improve malware detection accuracy while reducing computational requirements, contributing significantly to the optimization of digital security systems.

Keywords: *Ensemble-based Algorithms, Gradient Boosting, Information Gain, Machine Learning, Malware Detection.*

1. INTRODUCTION

Malware, short for "malicious software," refers to software specifically designed to harm, disrupt, or gain unauthorized access to computer systems [1]. The diversity of malware types, including viruses, worms, trojans, ransomware, and spyware, presents significant challenges for detection systems. Each operates in distinct ways to compromise data security [2], often causing severe consequences, such as network-wide infections, data theft, system damage, and the loss of critical information [3].

A notorious example of malware is WannaCry, a ransomware that exploits vulnerabilities in the Windows operating system. WannaCry encrypts victims' data and demands a ransom in Bitcoin for its decryption. The attack had a massive impact on hospitals, government organizations, and companies globally, forcing them to halt operations due to the inability to access critical data. This attack resulted in billions of dollars in losses and exposed significant weaknesses in global cybersecurity systems [4].

More recently, other soophisticated malware such as Emotet have emerged, evolving into complex threats. Initially recognized as a banking trojans, these malware variant now serve as delivery vehicles for other harmful software, spreading through malicious email attachments disguised as legitimate documents or links. Once a device is infected, these trojans can steal sensitive information, including passwords, financial data, and personal details, amplifying the potential damge. Emotet, in particular, has been widely used in high-profile cyberattacks, significanly amplifying the potential damage. These attacks have targeted sectors such as healthcare and government, resulting in massive data breaches and substantial financial losses, especially for companies and organizations dependent on robust information security [5].

While malware detection has advanced significantly, numerous challenges persist, particularly with the rise of more sophisticated threats like polymorphic malware. Traditional detection methods, such as signature-based detection [1], often fail to recognize new or unseen malware, as even minor code changes can evade detection. Moreover, advanced evasion techniques, including obfuscation and encryption, further complicate the detection process [5].

Machine learning has emerged as a promising solution for improving malware detection, offering greater accuracy and the ability to detect previously unseen threats. However, challenges remain, including high false positive rates and difficulty identifying zero-day attacks. Ensemble-based machine learning algorithms, such as Random Forest, Gradient Boosting, XGBoost, and AdaBoost, have gained attention for their potential to enhance detection systems by leveraging multiple models to improve performance and reduce errors.

This research aims to evaluate the performance of these ensemble-based machine learning algorithms in malware detection, with a focus on optimizing both accuracy and computational efficiency. Specifically, it compares the performance of Random Forest, Gradient Boosting, XGBoost, and AdaBoost algorithms using features selected by Information Gain. The research also aims to address limitations of existing methods by investigating how feature selection using Information Gain can improve classification results, particulary in terms of accuracy, F1-score, and computational efficiency. The findings aim to provide insights into optimizing malware detection systems by balancing detection accuracy and resource utilization.

Recent studies have shown promising results in machine learning-based malware detection. For example, the k-Nearest Neighbor (kNN) algorithm achieved a high accuracy of 97.0% accuracy with Information Gain feature selection [1], while research incorporating Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRU) attained 93% accuracy [6]. Other studies have demonstrated even higher accuracies, such as 98.58% using Recurrent Neural Networks (RNN) [7] and 98.65% using a combination of OC-SVM, LOF, and Mi-iForest [8]. However, these methods still face challenges related to handling large datasets and capturing temporal dependencies, highlighting the need for more efficient and scalable approaches.

2. RESEARCH METHODOLOGY

This research follows a sequential approach, with each stage building on the previous one. Figure 1 provides an overview of the process. Success in this research depends on the seamless integration of hardware and software. Optimal software performance requires robust hardware, while high-performance hardware benefits from efficient software.

For this study, the computer specifications included an 11th Gen Intel® Core™ i5-1135G7 processor (2.40 GHz), 8 GB of DDR4-3200 SDRAM (1 x 64-bit), and a 256 GB NVMe SSD for fast data access. Intel® Iris® Xe Graphics provided enhanced visual performance, ensuring smooth graphical operations during data processing and visualization. These hardware specifications enabled efficient handling of large datasets and computationally intensive machine learning algorithms.

The primary software used was Orange Data Mining (https://orangedatamining.com/), which supported various stages of the research, including data pre-processing, feature selection, and model building. Orange's tools for class balancing, normalization, and machine learning algorithms, such as Random Forest, Gradient Boosting, XGBoost, and AdaBoost, were crucial in model evaluation and optimization.

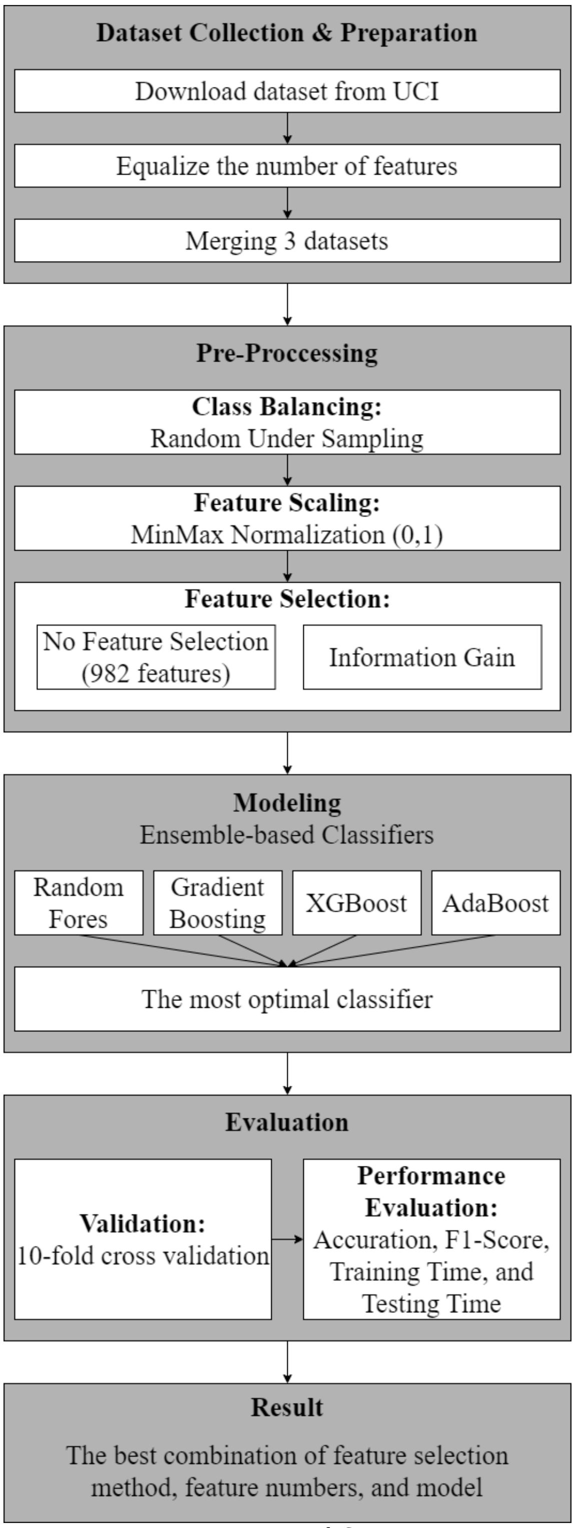

Dataset Collection & Preparation
Download dataset from UCI
Equalize the number of features
Merging 3 datasets
Pre-Proccessing
Class Balancing: Random Under Sampling
Feature Scaling: MinMax Normalization (0,1)
Feature Selection:
No Feature Selection (982 features)
Information Gain
Modeling
Ensemble-based Classifiers
Random Fores
Gradient Boosting
XGBoost
AdaBoost
The most optimal classifier
Evaluation
Validation: 10-fold cross validation
Performance Evaluation: Accuration, F1-Score, Training Time, and Testing Time
Result
The best combination of feature selection method, feature numbers, and model

Figure 1. Research Stages

2.1. Dataset Collection and Preparation

At this stage, the malware dataset is downloaded from the UCI Machine Learning Repository, as described in Table 1, details of the dataset used in research. The dataset consists of three files: a goodware dataset and two malware datasets, one from VirusTotal and another from VxHeaven. Each dataset captures the activity of files executed in a virtual environment or sandbox, presenting the results in a tabular format with over 1,085 features.

Table 1. Details of The Dataset Used in Research

Dataset Name	Malware static and dynamic features VxHeaven and VirusTotal
Number of Files	3 (goodware, malware from VirusTotal and malware from VxHeaven file)
Number of Rows	Goodware: 595; VirusTotal: 2955; VxHeaven: 2698
Number of Features	Goodware: 1086; VirusTotal: 1087; VxHeaven: 1087 (without label)
Class	Goodware: 0; VirusTotal: 1; VxHeaven: 1
Missing Values	None

The goodware dataset contains 595 benign file activities, each with 1,086 features. The VirusTotal malware dataset comprises 2,955 samples, each with 1,087 features. The VxHeaven malware dataset contains 2,698 samples, also with 1,087 features. Due to the differing number of features across the three datasets, they could not be directly merged. To standardize the feature count, certain attributes were removed. Specifically, features such as ‘vbaVarIndexLoad’ and ‘SafeArrayPtrOfIndex,’ which are absent in the goodware dataset, were eliminated from the malware datasets. Conversely, the feature ‘Feature 1,’ which is not present in the malware datasets, was removed from the goodware dataset. After these adjustments, all datasets contained 1,085 features.

The 'filename' feature was deemed irrelevant and removed from both the goodware and VirusTotal malware datasets. Labels were then assigned, with goodware labeled as ‘0’ and malware labeled as ‘1’. The three datasets were subsequently merged into a single dataset containing 1,190 rows, 983 columns, 982 features, and two target classes.

2.2. Pre-Processing

Pre-processing is a critical initial step in machine learning, involving the transformation or encoding of data to make it suitable for efficient analysis by machine learning algorithms [9]. It can also be seen as a process that prepares the data for modeling by refining features. The impact of pre-processing on machine learning performance is significant, offering benefits such as reduced training time and improved analysis speed [7].

This study performed three primary pre-processing tasks: class balancing, feature scaling, and feature selection. Class balancing is essential to prevent the model from becoming biased toward the majority class. Techniques such as oversampling, undersampling, or SMOTE can be used to address this imbalance [1]. In this study, undersampling was applied to reduce the number of instances from the majority class to match the number of instances in the minority class. Initially, the dataset exhibited an imbalance ratio of 1:9.5, but after undersampling, this was adjusted to 1:1. Without class balancing, the model may underperform in predicting the minority class, leading to inaccurate results. Undersampling helps balance the data distribution, allowing the machine learning model to perform more effectively and improving prediction accuracy for the minority class. Thus, class balancing plays a vital role in developing robust and accurate models [1].

The second stage of the pre-processing phase is feature scaling, which standardizes the range of values across features [10]. This study employs MinMax normalization for feature scaling, setting a lower bound of 0 and an upper bound of 1. MinMax normalization is a commonly used technique in data pre-processing that scales numerical data to a specific range, typically between 0 and 1 [11], [12]. This method involves a linear transformation of the original data to ensure that all values fall within the desired range [12]. As a result, the feature values are scaled between 0 and 1 without altering the relationships or distribution among them.

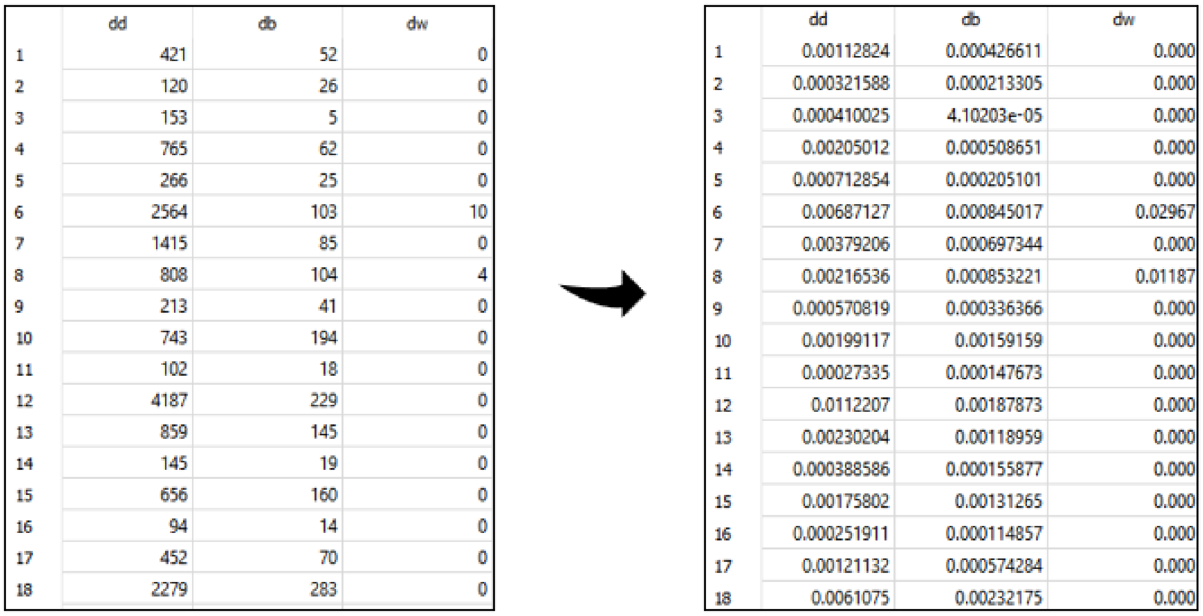
MinMax normalization is particularly beneficial for ensemble-based algorithms, such as Random Forest, Gradient Boosting, XGBoost, and AdaBoost. Although these algorithms are not highly sensitive to feature scale, normalization can improve stability and computational efficiency. By applying normalization, each feature contributes more evenly, preventing features with larger scales from dominating the calculations, particularly in models that utilize decision trees. The formula for MinMax normalization is provided in Equation 1.

$$v' = \frac{v - min(A)}{max(A) - min(A)} \tag{1}$$

In Equation 1, v' denotes the normalized value, where v represents the original value, and min(A) and max(A) indicate the minimum and maximum values of the attribute (A) respectively. This formula scales all data points within the range of 0 to 1 while maintaining their relative relationships. Figure 2 shows the dataset before and after MinMax normalization, highlighting how the values are adjusted to fall within the specified range. Prior to normalization, the feature values can vary widely, which may lead to disproportionate contributions from certain features during model training

After normalization, all feature values are scaled to a consistent range, preventing any single feature from dominating because of its scale. This method also reduces the standard deviation, which helps diminish the effect of outliers in the dataset. The left plot displays the original range of feature values, while the right plot illustrates the transformed values, now scaled between 0 and 1

DOI: https://doi.org/10.52436/1.jutif.2024.5.6.3903
p-ISSN: 2723-3863
e-ISSN: 2723-3871



	dd	db	dw
1	421	52	0
2	120	26	0
3	153	5	0
4	765	62	0
5	266	25	0
6	2564	103	10
7	1415	85	0
8	808	104	4
9	213	41	0
10	743	194	0
11	102	18	0
12	4187	229	0
13	859	145	0
14	145	19	0
15	656	160	0
16	94	14	0
17	452	70	0
18	2279	283	0

	dd	db	dw
1	0.00112824	0.000426611	0.000
2	0.000321588	0.000213305	0.000
3	0.000410025	4.10203e-05	0.000
4	0.00205012	0.000508651	0.000
5	0.000712854	0.000205101	0.000
6	0.00687127	0.000845017	0.02967
7	0.00379206	0.000697344	0.000
8	0.00216536	0.000853221	0.01187
9	0.000570819	0.000336366	0.000
10	0.00199117	0.00159159	0.000
11	0.00027335	0.000147673	0.000
12	0.0112207	0.00187873	0.000
13	0.00230204	0.00118959	0.000
14	0.000388586	0.000155877	0.000
15	0.00175802	0.00131265	0.000
16	0.000251911	0.000114857	0.000
17	0.00121132	0.000574284	0.000
18	0.0061075	0.00232175	0.000

Figure 2. Before (left) and After (right) Applying This MinMax Normalization

In the final pre-processing stage, feature selection is performed to identify the most relevant features from the dataset, enhancing the performance of the machine learning model. This study employs the forward selection method, a technique used in machine learning and statistics that operates in stages. The process begins with no features and adds them sequentially, selecting the feature that provides the greatest improvement in model performance at each step [13].

For feature selection, this study uses the Information Gain method. Information Gain measures the reduction in entropy or uncertainty about the class label given the presence of a particular feature. Many studies have effectively used Information Gain to reduce dimensionality by selecting the most relevant features based on weight calculations [14], [1]. This approach ensures that the selected features contribute meaningfully to the model, improving both accuracy and efficiency.

The choice of Information Gain for feature selection stems from its ability to provide a straightforward yet effective method for identifying features that offer the most valuable information for classification. By measuring how much a feature contributes to reducing uncertainty about the target class, Information Gain helps focus on the most relevant features while ignoring redundant or irrelevant ones. This results in a more compact and efficient model, reducing overfitting and improving generalization.

Information Gain is commonly chosen for feature selection because it effectively quantifies the information contained in each feature. It ranks features based on the reduction in entropy (uncertainty) achieved when a specific feature is known, making it a powerful filter-based technique. This method has been successfully applied in data classification, consistently helping identify features most relevant to the task at hand. By retaining only the most informative features, it enhances model accuracy and computational efficiency, leading to better performance in real-world applications.

The concept of Information Gain can be likened to a scoring system, indicating how much a feature contributes to improving predictions. A higher Information Gain score suggests greater usefulness. Its simplicity and efficiency in filtering out irrelevant or redundant features have allowed it to outperform other techniques in some cases [15], [16], [17], [1]. Further discussions on this topic can be found in Chapter 4.

Before selecting the features, it is essential to understand how Information Gain is calculated. The process begins by determining the dataset's entropy, as represented in Equation 2:

$$Entropy(S) = -\sum P_i \log_2 P_i \qquad (2)$$

Table 2. A List of The Top 47 Features with The Highest Information Gain Scores

No.	Features	IG Score	No.	Features	IG Score
1.	minor_image_version	0.728	25.	sizeOfHeaders.1	0.182
2.	minor_operatimg_system_version	0.701	26.	number_of_imports.1	0.181
3.	size_of_stack_reserve	0.613	27.	int	0.137
4.	major_operating_system_version	0.600	28.	ret	0.120
5.	minor_linker_version	0.558	29.	nop	0.107
6.	compile_date	0.540	30.	bt	0.099
7.	major_image_version	0.538	31.	ent_max	0.096
8.	dll_characteristics	0.506	32.	AddressOfEntryPoint	0.086

9.	major_subsystem_version	0.480	33.	rol	0.082
10.	minor_subsystem_version	0.470	34.	jge	0.081
11.	CheckSum	0.410	35.	BaseOfCode	0.081
12.	major_linker_version	0.396	36.	free	0.080
13.	characteristics	0.371	37.	subsystem	0.079
14.	number_of_IAT_entires	0.288	38.	rcl	0.079
15.	number_of_IAT_entires.1	0.288	39.	ent_whole_file	0.078
16.	pushf	0.253	40.	xor	0.078
17.	files_operations	0.246	41.	test	0.076
18.	count_dll_loaded	0.204	42.	fistp	0.074
19.	count_file_opened	0.197	43.	sbb	0.072
20.	not	0.197	44.	filesize	0.072
21.	size_of_stack_commit	0.191	45.	dll	0.072
22.	number_of_sections.1	0.186	46.	in	0.072
23.	sizeOfHeaders	0.182	47.	file_alignment	0.071
24.	size_of_headers	0.182			

Here, Pi represents the probability of each class in the dataset. Once the entropy values are determined, the Information Gain is calculated using Equation 3:

$$Gain(S,A) = Entropy(S) - \sum_{values}(A)\frac{|S_v|}{S}Entropy(S_v) \quad (3)$$

After calculating the Information Gain for each feature, the highest Information Gain score was found for the feature minor_image_version, which scored 0.728. This score indicates that minor_image_version plays a critical role in distinguishing between malware and goodware. Its high relevance is attributed to the significant differences it captures between these two categories. With the highest Information Gain score, this feature serves as a key indicator for classification, demonstrating minimal overlap with other features and making a unique contribution to malware detection models. From an initial set of 982 features, the feature selection process using Information Gain reduced the number of features to 47. For more details on this selected feature, can be found in Table 2, List of Top 47 Features with the Highest Information Acquisition Score.

2.3. Modeling

Modeling aims to identify factors that predict various data dimensions and evaluate the impact of changes within those dimensions. The benefits of modeling include enhanced prediction accuracy and consistent performance across different scenarios. Additionally, modeling enables the incorporation of larger datasets, which facilitates more informed decision-making and ultimately leads to increased efficiency and effectiveness in the analysis [18], [19].

The focus of the modeling in this research is on implementing ensemble-based algorithms. Ensemble methods combine the outputs of independently trained weak models and aggregate or weight their predictions to enhance overall accuracy. Techniques such as Bagging, Stacking, and Boosting are employed to achieve improvements in predictive performance [2].

This study utilizes four ensemble algorithms: Random Forest, Gradient Boosting, XGBoost, and AdaBoost. The Random Forest algorithm is a supervised learning method used for classification and regression tasks. It improves prediction accuracy by aggregating the outputs of multiple decision trees, each trained on a random subset of the data using ensemble learning techniques [20]. This algorithm is a homogeneous ensemble learning method, where each decision tree or base learner is trained using a random subset of feature vectors [21]. The feature vector is represented as shown in Equation 4:

$$x = \{x_1, x_2, \dots, x_P\} \quad (4)$$

Here, p refers to the dimensionality of the feature vector available to each base learner. The primary objective is to identify the prediction function *f(x)*, which estimates the target variable *Y*. This prediction function can be expressed as:

$$L(Y, f(x)) \quad (5)$$

Where L is the loss function, and the goal is to minimize the expected value of this loss. In regression and classification applications, squared error loss and zero-one loss are commonly used, as defined in Equations 6 and 7:

$$L(Y, f(x)) = (Y - f(x)^2) \quad (6)$$

$$L(Y, f(x)) = I(Y \neq f(x)) = \begin{cases} 0, if\ Y = f(x), \\ 1, otherwise. \end{cases} \quad (7)$$

To create an ensemble, a collection of base learners is combined. Let the base learners be denoted as:

$$h_1(x), h_2(x), \dots, h_J(x) \quad (8)$$

The averaging process for regression tasks is given by Equation 9, while the voting mechanism for classification tasks is defined in Equation 10:

$$f(x) = \frac{1}{J}\sum_{j=1}^{J} h_j(x) \quad (9)$$

$$f(x) = arg\,max \sum_{j=1}^{J} I(y = h_j(x)) \quad (10)$$

Gradient Boosting is a machine learning algorithm employed for both classification and regression tasks. It constructs a prediction model by combining a collection of weak models, aiming to minimize the loss function and reduce the discrepancy between predicted and actual values [22]. Introduced by Friedman in 2001, Gradient Boosting operates as a supervised learning technique that integrates multiple weak learners into an additive ensemble. The learning process is sequential, with each new base learner trained to correct the residual errors from the predictions of the current ensemble. A learning rate is applied to scale the output of each new learner before it is added to the ensemble [23].

XGBoost, or Extreme Gradient Boosting, is a robust classification and regression algorithm that builds on the gradient boosting framework by incorporating decision trees as weak learners. This algorithm is designed to create a more powerful and accurate model while mitigating overfitting [24]. XGBoost is a supervised learning method that has proven effective in a wide range of applications, from healthcare and finance to government and education. Its computational efficiency and strong predictive performance have made it a choice for many machine learning tasks, especially in machine learning applications and real-world data analysis [25].

AdaBoost, or Adaptive Boosting, is a widely used boosting algorithm based on decision trees, introduced by Freund and Schapire in 1996 [26]. It starts with a single weak learner and assigns equal weights to each observation. The algorithm then iteratively trains weak learners, increasing the weights of incorrectly predicted observations while decreasing those of correctly predicted ones. The stopping criterion is usually defined by a specified number of learners (M) or a threshold change in prediction error. After reaching the stopping criterion, the weak learners are combined, with their weights reflecting their accuracy, to form a strong final classifier that is essentially a weighted average of the weak classifiers. The algorithm operates as follows:

1. Initialize weights: Set observation weights $w_i = \frac{1}{n}$ for all i.
2. Fit a classifier: For each weak learner *m*, fit a classifier $f_m(x)$.
3. Compute weighted error: Calculate the weighted error err_m of the classifier.

$$err_m = \frac{\sum_{i=1}^{N} w_i \cdot I(y_i \neq f_m(x_i))}{\sum_{i=1}^{N} w_i} \quad (11)$$

4. In Equation 11, *m* denotes the *m*-th weak learner, y_i represents the class of the *i*-th observation, and $f_m(x_i)$ indicates the prediction made by the *m*-th classifier for the *i*-th observation. Furthermore, $I(y_i \neq f_m(x_i))$ is an indicator function that takes a value of 0 when the prediction is correct and 1 when it is incorrect.

$$\alpha_m = log\left(\frac{1-err_m}{err_m}\right) \quad (12)$$

5. Compute classifier weight: Determine the weight α_m for the classifier based on this error.

$$w_i^{new} = w_i^{old} \cdot \left(\alpha_m \cdot I\big(y_i \neq f_m(x_i)\big)\right),\ i = 1, 2, \dots, n \quad (13)$$

6. Update weights: Adjust the weights of observations for the next iteration.
7. Repeat: Continue until the stopping criterion is reached.

Finally, the strong classifier *f(x)* is given by the sign of the sum of weighted weak classifiers:

$$f(x) = sign(\sum_{m=1}^{M} \alpha_m \cdot f_m(x)) \quad (14)$$

In this study, the primary objective is to compare the performance of four ensemble-based algorithms: Random Forest, Gradient Boosting, XGBoost, and AdaBoost. These algorithms were chosen because ensemble methods, which combine the outputs of multiple base models, tend to improve predictive accuracy and generalization compared to single models. Random Forest and AdaBoost are known for their ability to handle complex, high-dimensional datasets like those in malware detection, as they reduce overfitting and improve robustness. XGBoost and Gradient Boosting are particularly effective due to their boosting mechanism, which focuses on correcting the errors of previous models, making them highly efficient in handling unbalanced and noisy data like malware classification tasks.

The algorithms will be evaluated based on several performance metrics: accuracy, F1-score, training time, and testing time. These metrics will help identify which algorithm performs most effectively and efficiently in terms of both prediction accuracy and computational resource usage. The results of these comparisons will be discussed in Section 2.5 to determine which of the four algorithms offers the best overall performance for this specific malware dataset.

2.4. Evaluation

After completing the machine learning algorithm modeling, the next step is to evaluate the model's performance. In this study, data validation is performed first, followed by the assessment of the performance of the four algorithms to determine the most effective one. The validation method used in this process is 10-fold cross-validation. Cross-validation is a robust technique for evaluating model performance, optimizing hyperparameters, and ensuring the classification algorithm's reliability, particularly when the dataset is limited [1]. The k-fold cross-validation method divides the dataset into 'k' approximately equal parts. The model is trained using 'k-1' parts, while the remaining part is used for validation. This procedure is repeated ‘k’ times, each time using a different subset as the validation set. The average performance across these ‘k’ iterations provides a more accurate assessment of the model's generalizability. When ‘k’ is set to 10, it is referred to as 10-fold cross-validation. This approach helps mitigate overfitting and produces a more consistent estimate of model performance, offering a more generalizable evaluation of the model’s effectiveness [27], [28], [29].

After data validation, the performance of the ensemble-based algorithms, such as Random Forest, Gradient Boosting, XGBoost, and AdaBoost, is evaluated based on four key metrics: accuracy, F1-Score, training time, and testing time. Accuracy, as a measure of the model's efficiency, reflects its ability to make correct predictions. It is calculated as the proportion of true positives and true negatives among all examined cases. In contrast, the F1-Score is a crucial metric in machine learning, particularly when balancing different types of errors is important. The F1-Score combines precision and recall into a single value, providing a balanced evaluation of models in situation where it is essential to account for both the relevance of predictions and the model's ability to capture all relevant cases [1].

$$Accuracy = \frac{TP+TN}{(TP+FP+TN+FN)} \quad (15)$$

Accuracy is calculated using Equation 15, where the TP (True Positive) value indicates the correct identification of positive instances, while TN (True Negative) refers to the correct identification of negative instances. In contrast, FP (False Positive) represents the misclassification of negative data as positive, and FN (False Negative) refers to the misclassification of positive data as negative [30]. The F1-Score, on the other hand, provides a more comprehensive evaluation of model performance, especially in cases where accuracy alone does not adequately reflect the trade-off between precision and recall. To compute the F1-Score, both precision and recall must be calculated, using Equation 17 and Equation 18, respectively.

$$F1 - Score = 2 \times \frac{precision \times recall}{precision + recall} \quad (16)$$

$$Precision = \frac{TP}{FP+FN} \quad (17)$$

$$Recall = \frac{TP}{TP+FN} \quad (18)$$

In Equation 16, precision represents the proportion of true positive predictions out of all instances predicted as positive. At the same time, recall is the proportion of true positives identified out of all actual positive instances. The F1-Score harmonizes precision and recall, making it especially useful for imbalanced datasets, where accuracy alone may not fully reflect model performance. A high F1-Score indicates a well-balanced trade-off between false positives (FP) and false negatives (FN), ensuring the model's reliability across different datasets.

Training time refers to the time taken by the classifier to learn patterns from the training data [31]. This metric is crucial because it reflects the model's efficiency in processing and learning from large datasets. Models with faster training times are particularly beneficial when working with high-dimensional data or when frequent retraining is required.

Meanwhile, testing time measures the duration it takes for a classifier to make predictions on unseen data after training [32]. This metric is essential for evaluating model performance in real-time applications, where fast predictions are often critical. Shorter testing times are advantageous for classifiers that require real-time decision-making, ensuring that the model is both efficient and responsive.

Both training and testing times contribute to assessing the overall computational efficiency of the model. Faster times for both metrics enhance the model's practicality, especially in time-sensitive environments or resource-constrained systems, while maintaining accurate predictions and balanced F1-Scores.

2.5. Result

The final stage of this research involves identifying the optimal combination of results from the ensemble-based algorithms. This process is crucial for optimizing model performance and ensuring reliable predictions in real-world applications. By comparing the accuracy, training time, and testing time of each algorithm, this stage allows for the selection of the most effective model for malware detection. This approach focuses on selecting the best-performing algorithm, balancing predictive accuracy with computational efficiency. The combination of results forms the foundation for developing a model that can be effectively applied in practice.

3. RESULT AND DISCUSSION

This study evaluates the performance of ensemble-based machine learning algorithms for malware detection, focusing on Random Forest, Gradient Boosting, XGBoost, and AdaBoost to identify the most efficient and effective option. By conducting a series of experiments, we compare key performance indicators, including accuracy, F1-score, training time, and testing time, to determine the optimal algorithm for this dataset.

The data collection yielded a combined dataset of 5,653 malware entries and 595 goodware entries. To ensure feature uniformity and address the significant class imbalance, irrelevant features were removed, resulting in a consistent set of 1,085 features across all datasets.

This step ensured that the final dataset used for analysis had a consistent and reliable structure, making it more suitable for machine learning applications. The uniformity of the dataset enhances the potential accuracy of malware detection during modeling. Additionally, the well-structured dataset streamlines the preprocessing phase, establishing a solid foundation for the modeling and evaluation stages that follow.

The research is divided into two experimental phases. In the first phase, all 982 features are used without feature selection, allowing a comprehensive comparison of the four ensemble-based algorithms. The algorithm that demonstrates the highest accuracy, as shown in Table 3, "Performance of Ensemble-Based Algorithms without Feature Selection," is XGBoost. Given its superior accuracy, XGBoost is selected for further analysis in the second phase.

Table 3. Performance of Ensemble-based Algorithms without Feature Selection

Number of Features	No Feature Selection							
	Random Forest		Gradient Boosting		XGBoost		AdaBoost	
	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
All 982 features	98.4%	98.4%	98.9%	98.9%	**98.7%**	**98.7%**	97.8%	97.8%

In the second phase, forward selection with Information Gain is employed as the feature selection technique. This method incrementally adds features based on their Information Gain scores, optimizing the feature set to achieve the highest possible performance. This approach enables a comparison of algorithm performance as the dataset is progressively reduced. The results of this phase are presented in Table 4, "Performance of Ensemble-Based Algorithms with Information Gain Feature Selection." Notably, the Gradient Boosting algorithm achieves an accuracy of 99.1% with a reduced set of 47 selected features, maintaining strong performance.

Table 4. Performance of Ensemble-based Algorithms with Information Gain Feature Selection Method

Number of Features	Information Gain							
	Random Forest		Gradient Boosting		XGBoost		AdaBoost	
	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
43	98.70%	98.70%	98.80%	98.80%	98.80%	98.80%	97.90%	97.90%
44	98.80%	98.80%	98.70%	98.70%	98.80%	98.80%	97.70%	97.70%
45	98.60%	98.60%	98.80%	98.80%	98.80%	98.80%	98.00%	98.00%
46	98.80%	98.80%	98.70%	98.70%	98.80%	98.80%	98.00%	98.00%
47	**98.50%**	**98.50%**	**99.10%**	**99.10%**	**98.80%**	**98.80%**	**98.10%**	**98.10%**
48	98.70%	98.70%	99.00%	99.00%	99.00%	99.00%	98.10%	98.10%
49	98.70%	98.70%	99.00%	99.00%	99.00%	99.00%	98.20%	98.20%
50	98.50%	98.50%	99.10%	99.10%	99.00%	99.00%	98.00%	98.00%
51	98.40%	98.40%	99.10%	99.10%	98.90%	98.90%	98.00%	98.00%
52	99.10%	99.10%	99.00%	99.00%	99.00%	99.00%	98.20%	98.20%

A comparison of Table 3 and Table 4 reveals that the Gradient Boosting algorithm demonstrates the highest accuracy both without feature selection and with 47 features selected using Information Gain. The identification of this optimal feature set enhances the Gradient Boosting algorithm's performance, underscoring that reducing the number of features can improve computational efficiency while preserving or even boosting accuracy. These findings highlight the effectiveness of feature selection in optimizing model performance, particularly for Gradient Boosting.

The next step involves analyzing the computational efficiency of two feature sets: all 982 features without feature selection and the 47 features selected using Information Gain. Computational efficiency is assessed by comparing training time and testing time, with the goal of accelerating the computational process while maintaining accuracy. This metric is vital as it demonstrates the algorithm's ability to process the dataset effectively in terms of both time and computational resources.

As shown in Table 5, "Computational Efficiency of Gradient Boosting: All Feature Set vs. Information Gain Feature Selection," reducing the number of features not only improves accuracy but also significantly enhances computational efficiency in both training and testing times. With Information Gain, the Gradient Boosting algorithm achieves an accuracy of 99.1%, compared to 98.9% without feature selection, thus alleviating the computational burden.

Table 5. Computational Efficiency of Gradient Boosting: All Feature Set vs. Information Gain Feature Selection Method

Number of Features	Feature Selection	Gradient Boosting			
		Accuracy	F1-Score	Training Time	Testing Time
All 982 features	No Feature Selection	98.9%	98.9%	35.583	0.202
47 features	Information Gain	99.1%	99.1%	10.819	0.041

DOI: https://doi.org/10.52436/1.jutif.2024.5.6.3903
p-ISSN: 2723-3863
e-ISSN: 2723-3871

As shown in Table 6, the comparison of training time between feature selection without (NFS) and with Information Gain (IG), highlights the effect of feature reduction on training time. Training time is significantly reduced from 35.583 seconds to 10.819 seconds when using the 47 features selected through Information Gain. This reduction is further illustrated in Figure 3, the training time bar chart, which clearly shows the correlation between fewer features and a substantial decrease in training time. This reduction offers significant benefits, particularly when the training process needs to be repeated on large datasets.

Table 6. Comparison of Training Time: Feature Selection Without (NFS) vs. Information Gain (IG) Feature Selection

Algorithm	Number of Feature	Training Time
Gradient Boosting – NFS	All 982 features	35.583
Gradient Boosting - IG	47 features	10.819

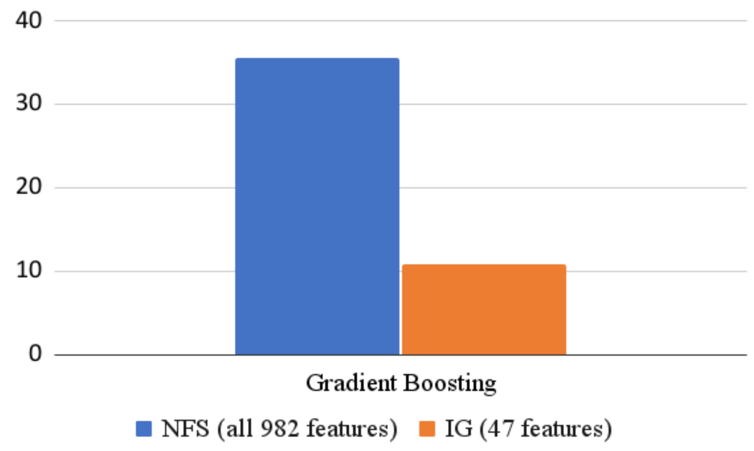


Figure 3. Training Time Bar Chart

Meanwhile, Table 7 compares the testing time between feature selection without (NFS) and with Information Gain (IG), showing a significant reduction in testing time. The testing time decreases from 0.202 seconds with 982 features to just 0.041 seconds when using the 47 features selected through Information Gain. This improvement is further depicted in Figure 4, the testing time bar chart, which visually demonstrates how feature selection effectively speeds up the testing process.

Table 7. Comparison of Training Time: Feature Selection Without (NFS) vs. Information Gain (IG) Feature Selection

Algorithm	Number of Feature	Training Time
Gradient Boosting – NFS	All 982 features	35.583
Gradient Boosting - IG	47 features	10.819

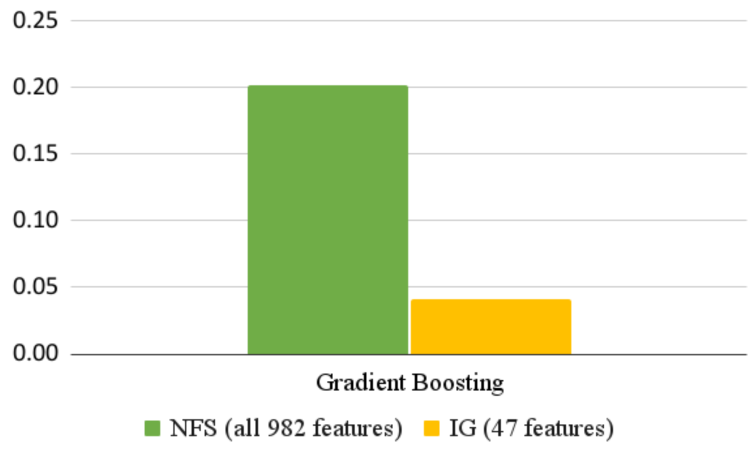
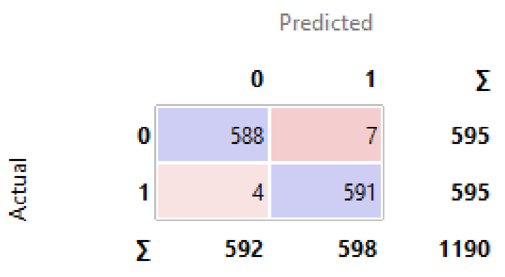


Figure 4. Testing Time Bar Chart

In addition to improving training and testing efficiency, feature selection using the Information Gain method simplifies the model overall. By removing irrelevant features, the algorithm can focus on the most influential ones, enhancing model interpretability and reducing the risk of overfitting.

This leads to a more efficient model that maintains an optimal level of detection. Such advantages are particularly crucial in real-world applications, especially in malware detection systems that require quick response times. In large datasets, selecting the right features is pivotal to creating efficient and responsive models. The experimental results demonstrate that applying Information Gain not only boosts accuracy but also significantly enhances computational efficiency during both the training and testing phases.



Actual \ Predicted	0	1	Σ
0	588	7	595
1	4	591	595
Σ	592	598	1190

Figure 5. Confusion Matrix in Gradient Boosting

In addition to accuracy and efficiency metrics, the confusion matrix shown in Figure 5 provides valuable insights into the Gradient Boosting model's classification performance with Information Gain feature selection. The matrix highlights the breakdown of true positives, true negatives, false positives, and false negatives, offering a detailed view of the model's ability to correctly classify malware and goodware. This granular information helps assess the model's precision in distinguishing between the two classes, providing a deeper understanding of its strengths and areas for improvement in the classification process.

In this confusion matrix, reveals that the Gradient Boosting model, with Information Gain feature selection, correctly classifies 591 entries as malware (True Positives) and 588 entries as goodware (True Negatives). The misclassification rates are low, with only 7 entries incorrectly classified as malware (False Positives) and 4 entries incorrectly classified as goodware (False Negatives). From these values, we can calculate key performance metrics such as accuracy and F1-score.

- Accuracy is calculated as:

$$\text{Accuracy} = \frac{True\ Positives + True\ Negatives}{Total\ Entries}$$
$$= \frac{591+588}{591+588+7+4}$$
$$= \frac{1179}{1190}$$
$$= 99.1\%$$

- Precision is calculated as:

$$\text{Precision} = \frac{True\ Positives}{True\ Positives+False\ Positives}$$
$$= \frac{591}{591+7}$$
$$= \frac{591}{598}$$
$$= 98.8\%$$

- Recall is calculated as:

$$\text{Recall} = \frac{True\ Positives}{True\ Positives+False\ Negatives}$$
$$= \frac{591}{591+4}$$
$$= \frac{591}{595}$$
$$= 99.3\%$$

- F1-Score is calculated as the mean of precision and recall:

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision+ Recall}$$
$$= 2 \times \frac{0.988 \times}{0.988+0.993}$$
$$= 0.990$$

These results highlight the excellent performance of the model in distinguishing between malware and goodware, as well as the precision, recall, and balanced F1 score of 99.0%. The low misclassification rate is indicated by a small number of false positives (7) and false negatives (4). This indicates that the Gradient Boosting model with Information Gain feature selection maintains strong classification performance, which is very important for practical malware detection tasks. This balance minimizes misclassification, making it a reliable choice for accurately detecting both malware and goodware. The model's strong performance in the confusion matrix further supports its high accuracy (99.1%) and reinforces its suitability for malware detection tasks.

The confusion matrix thus validates the computational efficiency and accuracy benefits observed with Information Gain feature selection, as outlined in Table 8. By reducing the feature set, the model not only achieves optimal accuracy but also improves detection precision, particularly in distinguishing malware from goodware. This is crucial for real-world applications that require rapid and reliable decision-making, highlighting the model's effectiveness in practical scenarios.

Table 8. Performance Analysis Result

No.	Methods	Accuracy	F1-Score
1.	**Gradient Boosting (47) [proposed]**	**99.1%**	**99.1%**
2.	kNN (32) [1]	97.0%	97.0%
3.	CNN-GRU [6]	93%	93%

When compared to other studies, such as the one conducted by [1], which utilized k-Nearest Neighbor (kNN) with 32 features, the results from the Information Gain feature selection demonstrate that the Gradient Boosting algorithm achieves superior accuracy and F1-Score, with an accuracy of 99.1%. While the kNN algorithm's results are commendable, achieving an accuracy and F1-Score of 97.0%, kNN tends to struggle with large datasets, especially when feature selection is not properly applied. This limitation can hinder the classification process, underscoring the advantages of using Information Gain to reduce the feature set and enhance the performance of algorithms like Gradient Boosting.

In contrast, Gradient Boosting with Information Gain feature selection exhibits superior performance in both accuracy and computational efficiency. This highlights the advantage of employing feature selection techniques, as they not only enhance model performance but also optimize the overall efficiency of the algorithm. As a result, the model becomes more suitable for practical applications, particularly in contexts that require rapid and reliable decision-making, such as malware detection systems.

Additionally, research by [6] which utilized Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRU), reported an accuracy of 93% in cross-platform malware classification. However, this performance falls short when compared to Gradient Boosting. The primary limitation of the CNN-GRU approach is its difficulty in capturing temporal dependencies and handling precision-recall imbalances, making it less optimal for malware classification. Addressing these challenges would require a more in-depth analysis of the relationships within sequential data, which is crucial for improving its performance in this domain.

Gradient Boosting outperforms both CNN-GRU and kNN in terms of accuracy and demonstrates greater efficiency when handling features selected through Information Gain. This comparison highlights the advantages of using Gradient Boosting for malware detection tasks, where both performance and computational efficiency are crucial. The model's ability to achieve high accuracy while maintaining low computational overhead makes it a more suitable choice for real-world applications that require fast and reliable decision-making.

Gradient Boosting demonstrates not only superior accuracy but also greater efficiency when handling features selected through Information Gain, particularly in comparison to other methods like k-Nearest Neighbor (kNN) and Convolutional Neural Networks with Gated Recurrent Units (CNN-GRU). This enhanced performance is vital in applications such as malware detection, where rapid and accurate predictions are essential for timely and reliable decision-making. The combination of high accuracy and computational efficiency makes Gradient Boosting a more suitable choice for such critical tasks.

Table 8 presents the results of the performance analysis, comparing the accuracy and efficiency of each method. This table highlights the significant advantages of our proposed approach, which combines Gradient Boosting with Information Gain for feature selection, showcasing its effectiveness in achieving optimal results. The combination of high accuracy and enhanced computational efficiency underscores the strength of this approach, making it a powerful solution for malware detection tasks.

4. DISCUSSION

This study demonstrates the effectiveness of ensemble-based algorithms for malware detection, emphasizing the enhancement of performance through feature selection using the Information Gain method. When coupled with Information Gain, the Gradient Boosting algorithm achieves the highest accuracy at 99.1%, outperforming Random Forest, XGBoost, and AdaBoost. This exceptional accuracy highlights the model's proficiency in correctly classifying both malware and goodware, showcasing its potential for reliable malware detection.

The model's performance is further validated through the confusion matrix, which shows that the Gradient Boosting model correctly identifies 591 malware entries (True Positives) and 588 goodware entries (True Negatives). However, it misclassifies only 7 goodware entries as malware (False Positives) and 4 malware entries as goodware (False Negatives). With these values, the model achieves an accuracy of 99.1%, precision of 98.8%, recall of 99.3%, and an F1-score of 0.990. These metrics demonstrate a strong balance between accuracy, precision, and recall, reinforcing the model's reliability for malware detection tasks.

In comparison, the k-Nearest Neighbor (kNN) algorithm, as reported in [1], only achieved 97.0% accuracy using 32 features. This disparity highlights the limitations of kNN in handling large-scale data without effective feature selection, while Gradient Boosting with Information Gain excels in both computational efficiency and accuracy, making it a more suitable choice for malware detection in real-world scenarios.

Compared to [6], which used a CNN and GRU model for cross-platform malware classification and achieved 93% accuracy, our approach shows superior precision and recall balance. While the CNN-GRU model excels at capturing temporal dependencies, its complexity can lead to increased computational demands, making it less suitable for high-speed, real-time malware detection. In contrast, the Gradient Boosting structure enables efficient handling of large-scale data, and when paired with Information Gain, the algorithm effectively reduces computational overhead without sacrificing accuracy. This makes Gradient Boosting with Information Gain ideal for real-time malware detection systems that require both accuracy and speed.

An essential finding in our study is the reduction of features from 982 to 47, achieved through Information Gain. This reduction not only improves computational efficiency but also maintains high accuracy, highlighting the crucial role of feature selection. Information Gain is highly effective in identifying relevant features, helping to reduce the training and testing time for Gradient Boosting, Random Forest, XGBoost, and AdaBoost. This efficiency boost is critical for implementing these models in real-world applications where resource constraints, such as on mobile devices, are a concern.

Although Gradient Boosting achieved the best results in this study, each algorithm has its unique strengths and limitations. Random Forest offers stability and interpretability but can become computationally intensive with high-dimensional data. XGBoost, while typically faster, may not achieve the same level of accuracy without extensive tuning. AdaBoost tends to perform well with smaller datasets but may struggle with complex malware patterns. These limitations highlight potential areas for future improvement, such as exploring hybrid feature selection techniques or optimizing algorithms to handle larger datasets more efficiently.

In terms of practical implications, our findings suggest that the Gradient Boosting model with Information Gain is an effective choice for real-time malware detection systems. However, applying this model in real-world scenarios may present challenges, including the need for continuous feature adjustments and potential computational limitations in real-time environments. Future research can address these limitations by experimenting with additional feature selection methods, such as Principal Component Analysis (PCA) or Recursive Feature Elimination (RFE), to further refine the model's performance and efficiency.

Thus, this research demonstrates the importance of ensemble methods combined with Information Gain for enhancing malware detection accuracy and computational efficiency. The results not only contribute to the field's understanding of the impact of feature selection on ensemble algorithms but also provide insights into practical applications, laying the foundation for future research aimed at optimizing malware detection in diverse and resource-constrained environments.

5. CONCLUSION

As malware threats evolves, traditional signature-based detection methods face increasingly challenges. This study evaluates the effectivesness of four ensemble-based machine learning algorithms, both with and without the application of Information Gain feature selection. The results show that Gradient Boosting with Information Gain achieved the highest accuracy of 99.1%, while reducing the features set from 982 to 47 features. This reduction enhances computational efficiency and simplifies thes model without compromising accuracy.

By prioritizing the most relevant features, Information Gain optimizes the training and testing processes, ensuring the model remains efficient while maintaining optimal performance. Integrating ensemble-based algorithms and feature selection offers a balanced approach to achieving high accuracy and computational efficiency in malware detection systems. This approach is particularly beneficial in real-world applications where fast and accurate threat identification is critical for organizations in the cybersecurity field.

Future research could explore the use of additional ensemble techniques or hybrid approaches to further enhance malware detection efficiency and accuracy across different systems. Additionally, testing the model on larger and more diverse datasets could provide insights into its scalability and generalizability in various environments. The findings of this study also suggest that combining ensemble algorithms with alternative feature selection methods could lead to better performance in future malware detection systems. Ultimately, this study provides a foundation for developing faster, more accurate, and more efficient malware detection tools, benefiting organizations seeking to strengthen their cybersecurity defenses.

REFERENCES

[1] F. A. Rafrastara, C. Supriyanto, A. Amiral, S. R. Amalia, M. D. Al Fahreza, and F. Ahmed, "Performance Comparison of k-Nearest Neighbor Algorithm with Various k Values and Distance Metrics for Malware Detection," *mib*, vol. 8, no. 1, p. 450, Jan. 2024, doi: 10.30865/mib.v8i1.6971.

[2] D. Sigh and J. S. Samagh, "A COMPREHENSIVE REVIEW OF HEART DISEASE PREDICTION USING MACHINE LEARNING," *jcr*, vol. 7, no. 12, Jun. 2020, doi: 10.31838/jcr.07.12.54.

[3] S. Aurangzeb, H. Anwar, M. A. Naeem, and M. Aleem, "BigRC-EML: big-data based ransomware classification using ensemble machine learning," *Cluster Comput*, vol. 25, no. 5, pp. 3405–3422, Oct. 2022, doi: 10.1007/s10586-022-03569-4.

[4] S. Ghafur, S. Kristensen, K. Honeyford, G. Martin, A. Darzi, and P. Aylin, "A retrospective impact analysis of the WannaCry cyberattack on the NHS," *npj Digit. Med.*, vol. 2, no. 1, p. 98, Oct. 2019, doi: 10.1038/s41746-019-0161-6.

[5] M. S. Akhtar and T. Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," *Symmetry*, vol. 14, no. 11, p. 2304, Nov. 2022, doi: 10.3390/sym14112304.

[6] N. Pachhala, S. Jothilakshmi, and B. P. Battula, "Cross-Platform Malware Classification: Fusion of CNN and GRU Models," *IJSSE*, vol. 14, no. 2, pp. 477–486, Apr. 2024, doi: 10.18280/ijsse.140215.

[7] M. Almahmoud, D. Alzu'bi, and Q. Yaseen, "ReDroidDet: Android Malware Detection Based on Recurrent Neural Network," *Procedia Computer Science*, vol. 184, pp. 841–846, 2021, doi: 10.1016/j.procs.2021.03.105.

[8] O. AbuAlghanam, H. Alazzam, M. Qatawneh, O. Aladwan, M. A. Alsharaiah, and M. A. Almaiah, "Android Malware Detection System Based on Ensemble

Learning," Jan. 31, 2023. doi: 10.21203/rs.3.rs-2521341/v1.

[9] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 91–99, Jun. 2022, doi: 10.1016/j.gltp.2022.04.020.

[10] A. Mihoub, S. Zidi, and L. Laouamer, "Investigating Best Approaches for Activity Classification in a Fully Instrumented Smarthome Environment," *IJMLC*, vol. 10, no. 2, pp. 299–308, Feb. 2020, doi: 10.18178/ijmlc.2020.10.2.935.

[11] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, Dec. 2020, doi: 10.1016/j.asoc.2019.105524.

[12] A. M. Priyatno, L. Ningsih, and M. Noor, "Harnessing Machine Learning for Stock Price Prediction with Random Forest and Simple Moving Average Techniques," *jesa*, vol. 1, no. 1, pp. 1–8, Mar. 2024, doi: 10.69693/jesa.v1i1.1.

[13] Y.-W. Chong, T. Emad Ali, S. Manickam, M. N. Yusoff, K.-L. Alvin Yau, and S.-L. Keoh, "A Ddos Attack Detection Framework: Leveraging Feature Selection Integration and Random Forest Optimization for Improved Security," 2023. doi: 10.2139/ssrn.4651305.

[14] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020, doi: 10.1109/ACCESS.2020.3009843.

[15] T. A. Alhaj, M. M. Siraj, A. Zainal, H. T. Elshoush, and F. Elhaj, "Feature Selection Using Information Gain for Improved Structural-Based Alert Correlation," *PLoS ONE*, vol. 11, no. 11, p. e0166017, Nov. 2016, doi: 10.1371/journal.pone.0166017.

[16] M. I. Prasetiyowati, N. U. Maulidevi, and K. Surendro, "Determining threshold value on information gain feature selection to increase speed and prediction accuracy of random forest," *J Big Data*, vol. 8, no. 1, p. 84, Dec. 2021, doi: 10.1186/s40537-021-00472-4.

[17] D. Theng and K. K. Bhoyar, "Feature selection techniques for machine learning: a survey of more than two decades of research," *Knowl Inf Syst*, vol. 66, no. 3, pp. 1575–1637, Mar. 2024, doi: 10.1007/s10115-023-02010-5.

[18] M. Pompiliu Cristescu, "Tools Used in Modeling of the Economic Processes," *KSS*, Jan. 2020, doi: 10.18502/kss.v4i1.5985.

[19] D. Goretzko and M. Bühner, "One model to rule them all? Using machine learning algorithms to determine the number of factors in exploratory factor analysis.," *Psychological Methods*, vol. 25, no. 6, pp. 776–786, Dec. 2020, doi: 10.1037/met0000262.

[20] S. Tamilselvi and S. Ragul, "Classification of IoT Network Traffic using Random Forest Classifier," vol. 3, no. 02, pp. 16–25, Feb. 2024.

[21] M. Savargiv, B. Masoumi, and M. R. Keyvanpour, "A New Random Forest Algorithm Based on Learning Automata," *Computational Intelligence and Neuroscience*, vol. 2021, no. 1, p. 5572781, Jan. 2021, doi: 10.1155/2021/5572781.

[22] A. Farki, R. Baradaran Kazemzadeh, and E. Akhondzadeh Noughabi, "A Novel Clustering-Based Algorithm for Continuous and Noninvasive Cuff-Less Blood Pressure Estimation," *Journal of Healthcare Engineering*, vol. 2022, pp. 1–13, Jan. 2022, doi: 10.1155/2022/3549238.

[23] T. Duan *et al.*, "NGBoost: Natural Gradient Boosting for Probabilistic Prediction," vol. 119, pp. 2690–2700, 2020.

[24] N. Fragkis, "Assessment and comparison of existing methods and datasets for sentiment analysis of Greek texts," Πανεπιστήμιο Δυτικής Αττικής, 2022. Accessed: Sep. 20, 2024. [Online]. Available: https://polynoe.lib.uniwa.gr/xmlui/handle/11400/2411

[25] Kartina Diah Kusuma Wardani and Memen Akbar, "Diabetes Risk Prediction using Feature Importance Extreme Gradient Boosting (XGBoost)," *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 7, no. 4, pp. 824–831, Aug. 2023, doi: 10.29207/resti.v7i4.4651.

[26] B. Perry, "AdaBoost And Its Variants: Boosting Methods For Classification With Small Sample Size And Brain Activity In Schizophrenia," 2023. [Online]. Available: http://hdl.handle.net/10464/17817

[27] G. Battineni, G. G. Sagaro, C. Nalini, F. Amenta, and S. K. Tayebati, "Comparative Machine-Learning Approach: A Follow-Up Study on Type 2 Diabetes Predictions by Cross-Validation Methods," *Machines*, vol. 7, no. 4, p. 74, Dec. 2019, doi: 10.3390/machines7040074.

[28] G. Orrù, M. Monaro, C. Conversano, A. Gemignani, and G. Sartori, "Machine Learning in Psychometrics and Psychological Research," *Front. Psychol.*, vol. 10, p. 2970, Jan. 2020, doi: 10.3389/fpsyg.2019.02970.

[29] C. Supriyanto, F. A. Rafrastara, A. Amiral, S. R. Amalia, M. D. Al Fahreza, and Mohd. F. Abdollah, “Malware Detection Using K-Nearest Neighbor Algorithm and Feature Selection,” *mib*, vol. 8, no. 1, p. 412, Jan. 2024, doi: 10.30865/mib.v8i1.6970.

[30] C. Ciciana, R. Rahmawati, and L. Qadrini, “The Utilization of Resampling Techniques and the Random Forest Method in Data Classification,” *tin*, vol. 4, no. 4, pp. 252–259, Sep. 2023, doi: 10.47065/tin.v4i4.4342.

[31] M. D. Ramasamy, K. Periasamy, L. Krishnasamy, R. K. Dhanaraj, S. Kadry, and Y. Nam, “Multi-Disease Classification Model Using Strassen’s Half of Threshold (SHoT) Training Algorithm in Healthcare Sector,” *IEEE Access*, vol. 9, pp. 112624–112636, 2021, doi: 10.1109/ACCESS.2021.3103746.

[32] S. Gupta and B. Singh, “An intelligent multi-layer framework with SHAP integration for botnet detection and classification,” *Computers & Security*, vol. 140, p. 103783, May 2024, doi: 10.1016/j.cose.2024.103783.