JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY
https://e-journal.uum.edu.my/index.php/jict

How to cite this article:
Rafrastara, F. A., Ghozi, W., Sani, R. R., Handoko, L. B., Abdussalam, Pramudya, E. R., & Abdollah, F. M. (2025). Integrating information gain and chi-square for enhanced malware detection performance. *Journal of Information and Communication Technology, 24*(1), 79-101. https://doi.org/10.32890/jict 2025.24.1.4

Integrating Information Gain and Chi-Square for Enhanced Malware Detection Performance

[1]Fauzi Adi Rafrastara, [*2]Wildanil Ghozi, [3]Ramadhan Rakhmat Sani,
[4]Lekso Budi Handoko, [5]Abdussalam, [6]Elkaf Rahmawan Pramudya &
[7]Faizal M. Abdollah
[1,2,3,4,5&6]Faculty of Computer Science, Universitas Dian Nuswantoro, Indonesia
[7]Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka, Malaysia

[1]fauziadi@dsn.dinus.ac.id
[*2]wildanil.ghozi@dsn.dinus.ac.id
[3]ramadhan_rs@dsn.dinus.ac.id
[4]handoko@dsn.dinus.ac.id
[5]abdussalam@dsn.dinus.ac.id
[6]elkaf.rahmawan@dsn.dinus.ac.id
[7]faizalabdollah@utem.edu.my
[*]Corresponding author

Received: 1/12/2024 Revised: 10/12/2024 Accepted: 15/1/2025 Published: 31/1/2025

ABSTRACT

Malware represents a serious and continuously evolving threat in the modern digital environment. Detecting malware is essential to safeguard devices and systems from risks such as data corruption, data theft, account compromises, and unauthorized access that could result in total system takeover. As malware has progressed from its simpler, monomorphic variants to more sophisticated forms like oligomorphic, polymorphic, and metamorphic, a machine learning-based detection system is now required, surpassing the limitations of traditional signature-based methods. Recent studies have shown that this challenge can be addressed by employing machine learning algorithms for detection. Some studies have also implemented various feature selection methods to optimize detection efficiency. However, they continue to struggle with false positives and false negatives, striving to reach zero tolerance in malware detection. This study introduces the IGCS method, a combined feature selection

approach that integrates Information Gain with Chi-Square (X^2) to enhance both the effectiveness and efficiency of machine learning classifiers. Using IGCS, six classifiers—Random Forest, XGBoost, kNN, Decision Tree, Logistic Regression, and Naïve Bayes—achieved higher performance scores compared to other scenarios, such as when classifiers were combined with Information Gain, Chi-Square, PCA, or even without any feature selection. As a result, Random Forest with 30 features selected by IGCS proved superior to any combination of classifiers and feature selection methods in malware detection, achieving 99.0% accuracy, recall, precision, and F1-Score. This combination also demonstrated efficiency with a 52.5% decrease in training time and a 56.9% decrease in testing time.

Keywords: Malware detection, IGCS, feature selection, Information Gain, Chi-Square.

INTRODUCTION

Malware, derived from "malicious software," refers to software that intentionally performs harmful actions on a victim's computer for personal gain (Kale et al., 2024). These actions can include data manipulation, encryption, and monitoring of users without their knowledge. While genuine file transfers occur with user consent, malicious activities happen without the user's knowledge and are executed automatically by malware. Cybercriminals leverage malware to take advantage of security weaknesses in targeted systems, enabling them to gain access to and control over devices. Hundreds of millions of electronic devices are currently or have been targeted by malware, leading to a significant rise in global financial losses (Wang et al., 2024). As criminal gangs' technology has advanced, the variety and forms of malware have expanded considerably, becoming more diverse and posing an even greater threat to computers. Malware comes in various forms, such as viruses, worms, trojan horses, adware, spyware, rootkits, bots, ransomware, and encrypted malware (e.g. TLS-based malware), each with its own method of attack and impact (Keshkeh et al., 2022; Shahid et al., 2021).

Efforts to combat malware attacks have been underway for decades. The initial documented computer virus, called "Elk Cloner" was developed by Richard J. Skrenta Jr., a teenager aged 15 from the United States (Lu et al., 2023; Zhao et al., 2023). This virus was capable of infecting Apple II computers and spreading itself to other systems. In 1987, Bernd Robert, a computer security specialist from Germany, developed the first antivirus software (Supriyanto et al., 2024). which specifically infected *.com files on DOS operating systems. These early developments marked the beginning of a continuous battle against malware, leading to the sophisticated antivirus and cybersecurity measures we have today. The evolution of both malware and antivirus software highlights the ongoing arms race in the field of computer security, emphasizing the need for constant vigilance and innovation to protect against ever-evolving threats.

The increasing prevalence and evolving nature of malware pose significant threats not only to the public but also to IT teams responsible for safeguarding critical data. These threats affect a diverse array of devices, including personal computers and mobile devices, which operate on different systems and platforms (Aslan & Samet, 2020). Malware has evolved not only in its types but also in its ability to evade detection. Traditional antivirus solutions that rely on signature-based detection techniques are only effective against monomorphic viruses (Bao et al., 2024). However, polymorphic malware, which can generate different signatures for its offspring, poses a significant challenge to conventional antivirus solutions (Rafrastara et al., 2024). This type of malware can randomly create new signatures for each new file, making it difficult for antivirus programs to keep up, as they would need to store an ever-increasing number of signatures in their databases. Consequently, this makes malware handling highly

inefficient. To address this issue, there is a need for intelligent systems capable of analyzing and detecting malware both statically and dynamically (Wang et al., 2024).

Modern virus detection has evolved beyond fingerprinting to include behaviour-based analysis, which has become crucial because of the evolution of malware from monomorphic to oligomorphic, polymorphic, and metamorphic forms (Supriyanto et al., 2024). Behaviour-based detection is more complex than fingerprinting, as it involves examining thousands of features before modelling them with machine learning algorithms. However, evaluating too many features can adversely affect the performance of machine learning algorithms, particularly regarding execution speed. Recent studies have implemented machine learning algorithms and feature selection methods for malware detection, aiming to enhance both effectiveness and efficiency (Abujazoh et al., 2023; Kale et al., 2024; Supriyanto et al., 2024). Despite achieving accuracy rates above 95% and reducing the number of features to 30, these methods still struggle to attain zero-tolerance in malware detection. This research focuses on enhancing the performance of machine learning algorithms by proposing a hybrid feature selection method called IGCS (Information Gain & Chi-Square), which leverages the benefits of Information Gain and Chi-Square to make the detection process more efficient and effective.

RELATED WORK

Researchers have investigated various machine learning methods for detecting malware. Abujazoh et al. (Abujazoh et al., 2023) compared k-Nearest Neighbour (kNN), Support Vector Machine (SVM), and Decision Tree (DT) algorithms. The dataset used in this research is "Malware static and dynamic features VxHeaven and Virus Total Dataset" from the UCI machine learning repository. To deal with imbalance, they divided the dataset into eight parts and applied under-sampling, adding 595 non-malware files to each part. They then selected the top 30 features using Chi-Square and ReliefF to be tested with the three different algorithms. As a result, DT consistently outperformed SVM and kNN, achieving the best results on dataset part 8 with Chi-Square feature selection, reaching an accuracy of 98.53%. Since achieving 100% accuracy in malware detection is crucial to protect user data and systems, the 98.53% accuracy score obtained in this research can still be improved by implementing other algorithms and feature selection methods.

Another study focused on the kNN algorithm, testing different k values and feature selection methods for malware detection (Supriyanto et al., 2024). The dataset utilized in this study is also "Malware static and dynamic features VxHeaven and Virus Total Dataset" from UCI machine learning repository. Firstly, the authors compared the performance of kNN, Naïve Bayes, and Logistic Regression in malware detection. As a result, kNN surpassed the other two classifiers, achieving an accuracy and F1-Score of 95.8%. The researchers then tested the kNN using different k=values (3, 5, and 7) and applied two feature selection techniques: Information Gain (IG) and Principal Component Analysis (PCA). The number of features or components used in this experiment is 32. Finally, the best performance was achieved by the combination of kNN with k=3 and Information gain with 32 features, resulting in 96.9% accuracy and F1-Score. This score is still low compared to the previous state-of-the-art (Abujazoh et al., 2023). Therefore, it is essential to develop a model with improved performance, aiming for zero tolerance toward false positives and false negatives due to the critical consequences of malware attacks.

In their study, Kale et al. (2024) conducted an analysis on a private dataset consisting of 200 Win32 Portable Executable (PE) files evenly split between goodware and malware. The features were extracted using the Cuckoo Sandbox, a widely recognized tool for dynamic malware analysis. The researchers

evaluated the performance of several classification algorithms, including J48, NB, Reduced Error Pruning Tree (REP Tree), Adaptive Boosting Model 1, and Multilayer Perceptron (MLP). Feature selection was performed using Multi-Objective Genetic Algorithms (MOGAs), reducing the initial 335 features to 200. The study achieved a notable accuracy of 93.33%, highlighting the effectiveness of the selected algorithms and feature selection method in distinguishing between goodware and malware. Nevertheless, increasing the number of samples is necessary to enhance the model's robustness and generalizability. A larger dataset would provide a more comprehensive representation of the diverse characteristics of both goodware and malware, potentially leading to improved accuracy and reliability of the detection system. Enhancing the accuracy would ensure more reliable detection of malware, thereby increasing the overall effectiveness and trustworthiness of the system. This could be achieved by exploring additional feature selection methods, incorporating more advanced machine learning algorithms, or further optimizing the existing models.

Lu et al. (2023) employed the Malimg Dataset, which contains 9,339 grayscale images of malware, classified into 25 different categories. The study compared the performance of three algorithms: CPL-Net, Convolutional Neural Networks (CNN), and Long Short-Term Memory Networks (LSTM). CPL-Net, which integrates a CNN-LSTM network with a parallel feature fusion approach, demonstrated superior performance over the other two algorithms. Specifically, CPL-Net achieved an impressive accuracy of 98.7% and an F1 score of 98.6%, underscoring its effectiveness in malware detection. However, employing image datasets for malware detection requires significant computational power for both the training and testing phases, making it less efficient compared to tabular datasets. Additionally, the accuracy score needs further enhancement to approach zero tolerance in malware detection, ensuring minimal false positives and negatives.

Based on the reviewed literature, several challenges need to be addressed. Firstly, detection accuracy must be improved, as false positives and false negatives in malware detection can pose significant risks. Secondly, malware detection should be performed as efficiently and quickly as possible. Therefore, identifying the optimal combination of classification algorithms and feature selection methods, while minimizing the number of features used is crucial for achieving effective and efficient malware detection.

A hybrid feature selection method called Information Gain and Chi-Square (IGCS) is proposed by combining the scores of Information Gain and Chi-Square to identify the best features. These two methods are chosen for this research as they have been effectively utilized by researchers in malware detection, demonstrating excellent performance, especially with the dataset used in this study. Combining two or more feature selection methods is a common practice to leverage the benefits of each method. Recent studies have demonstrated the effectiveness of hybrid methods in feature selection (Dabas et al., 2023; Hossain et al., 2024; Omuya et al., 2021; Yin et al., 2023). One notable example is Dabas et al. (2023), who combined filter-based methods (MRMR, PCA, PCC) and wrapper-based (GLBPSO, RFE) and applied these combinations with various machine learning algorithms (DT, SVM, kNN, LR) to detect ransomware. The best performance was achieved by the combination of PCA and RFE implemented in the Decision Tree algorithm, with 99.91% accuracy and 99.92% F-Score.

METHODOLOGY

This research involves several stages, as shown in Figure 1: Dataset Preparation, Pre-Processing, Feature Selection and Modeling, and Evaluation. The software and hardware used during the experiment in this research will also be discussed in this section.

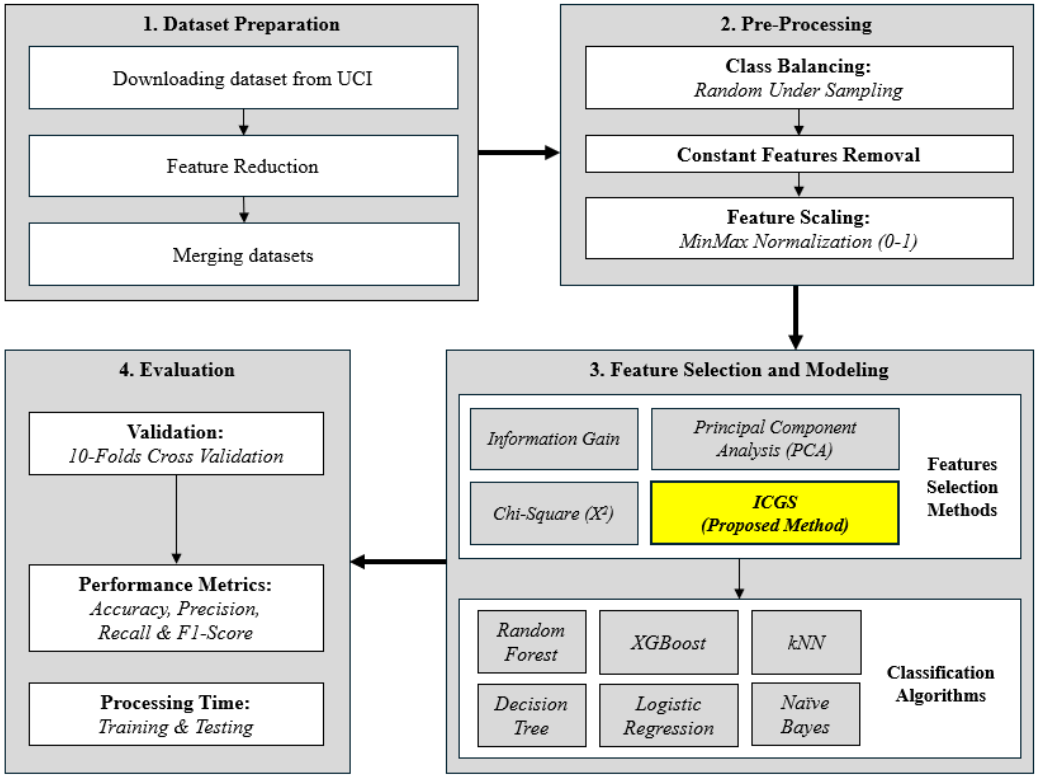
Hardware and Software

The availability of supporting tools is a key factor that greatly influences the success and efficiency of research. In computer science research, support from hardware and software is crucial (Supriyanto et al., 2024). High-quality software cannot perform optimally without adequate hardware, and excellent hardware is ineffective without the right software. A personal computer with the following specifications was utilized for this study:

Processor	: Intel® Xeon® E5620 12M Cache, 2.40 GHz
RAM	: 16 GB DDR3
Graphic Card	: Radeon RX550
Hard Drive	: SSD 1 TB

Figure 1

Research Stages



The main software utilized in this research was Orange (downloaded from https://orangedatamining.com/download/). Orange is a robust data mining tool that can be effectively utilized for data pre-processing tasks such as class balancing, eliminating constant features, scaling features, and selecting

relevant features. In addition, Orange also provides various machine learning algorithms as well as complete evaluation results, such as accuracy, recall, precision, F1-score, train time, and test time.

Dataset Collection and Preparation

This research utilized a publicly available malware dataset called "Malware Static and Dynamic Features VxHeaven and VirusTotal Data Set," which can be downloaded (*Malware Static and Dynamic Features VxHeaven and Virus Total*, 2019). The downloaded file contains three CSV dataset files: staDynBenignLab, staDynVxHeaven2698Lab, and staDynVt2955Lab. The staDynBenignLab or goodware dataset comprises 595 instances sourced from program files directories on Microsoft Windows 7 and 8. Meanwhile, the staDynVxHeaven2698Lab or VxHeaven dataset contains 2,698 instances collected from various malware samples in the VxHeaven lab. Lastly, the staDynVt2955Lab or Virus Total dataset includes 2,955 instances obtained from diverse malware types in the VirusTotal lab. VirusTotal focuses on analyzing suspicious files and URLs, while VxHeaven provides information and samples related to the creation of viruses and malware. Both websites contain a wide range of malware commonly found in public, such as Trojan Horses, Ransomware, Spyware, Rootkits, Worms, and Keyloggers.

Regarding the number of features, the three files differ. The malware datasets from both VxHeaven and VirusTotal have 1,087 features, whereas the goodware dataset has 1,085 features. This dataset has no missing values, so no imputation is necessary. The dataset details are shown in Table 1. This dataset is not immediately suitable for modeling purposes due to several drawbacks. These include the presence of three different files with a varying number of features, class imbalance, numerous constant features, and non-normalized data. Therefore, several pre-processing steps are required to address these issues. Due to the differing number of features in the datasets, the first step is to equalize them by removing features that are not present in all datasets. This process, known as feature reduction, involves removing irrelevant features that could lead to overfitting. Two particular features, 'vbaVarIndexLoad' and 'SafeArrayPtrOfIndex', are only available in the VxHeaven and VirusTotal malware datasets. Therefore, they need to be removed. Additionally, features like 'filename' and 'compiled_date', which exist in all three datasets, should also be removed. Afterwards, merging the three datasets into a single dataset can be done, preparing it for the pre-processing phase in the next step. In this phase, instances from the malware datasets will be labelled as 1, while instances from the goodware dataset will be labelled as 0.

Table 1

The Details of the Dataset

Dataset File	staDynBenignLab.csv	staDynVxHeaven2698Lab.csv	staDynVt2955Lab.csv
Source	Goodware from Program Files Folder in MS Windows 7 & 8	Various malware from VxHeaven lab	Various malware from Virus Total lab
Number of Instances	595	2698	2955
Number of Features	1085	1087	1087
Missing Values	-	-	-

Pre-Processing

The second step is called Pre-Processing. Pre-processing plays a vital role in guaranteeing that the data is clean, consistent, and ready for analysis. This step helps in removing noise, normalizing data, and transforming it into a format that machine learning algorithms can effectively process. Proper pre-processing can significantly improve the performance and accuracy of the models (Yan & Razmjooy, 2023). This research involves several tasks, including class balancing, removing constant features, and feature scaling.

After analyzing the dataset, it turns out to have an imbalanced class, with an Imbalance Ratio (IR) of 1:9.5, which is considered a medium imbalance (Fan et al., 2017). Therefore, this issue needs to be addressed through class balancing. Class balancing is a method used to tackle the problem of imbalanced datasets, where one class significantly outnumbers the other(s). If class imbalances are not addressed, predictions for the minority class may become inaccurate. Accurate prediction of the minority class is vital, as misclassifications can lead to significant consequences or high costs (Rafrastara et al., 2024). The class balancing technique employed in this study is Random Under Sampling (RUS), which involves decreasing the number of instances in the majority class to align with the number of instances in the minority class (Rafrastara et al., 2023). This is achieved by randomly choosing a subset of instances from the majority class, which aids in balancing the distribution of classes. With RUS, the number of samples in the malware class will be reduced to match the number of samples in the goodware class, which is 595.

The second step in this stage is called Constant Features Removal. This step is performed to eliminate features that have the same value across all instances in the dataset. Such features do not provide any useful information for distinguishing between classes and can be safely removed to decrease the dataset's dimensionality. Removing constant features helps enhance the efficiency of machine learning algorithms and improves the model's overall performance (Supriyanto et al., 2024). In this dataset, 933 out of 1083 features have been identified as constant features. Therefore, these features should be removed. Next, upon examining the remaining features, it was found that they were on different scales. When features have varying scales, those with larger ranges can overshadow those with smaller ranges, potentially distorting the results and causing the model to perform poorly. To address this, feature scaling is essential (Pandey & Jain, 2017). The feature scaling method used in this research is MinMax Normalization (0,1), which transforms the features to a fixed range, typically (0, 1). This ensures that all features contribute equally to the model, enhancing the convergence speed of algorithms and improving overall performance. By scaling the features to a common range, Min-Max normalization helps achieve more accurate and reliable predictions as well as minimizing bias (Singh & Singh, 2020). Equation 1 is the formula of MinMax Normalization.

$$v_i = \left(\frac{x_i - x_{min}}{x_{max} - x_{min}} (v_{max} - v_{min}) \right) + v_{min} \tag{1}$$

In (1), v_i represents the newly calculated value for row i, where x_i is the original value to be normalized. Here, x_{min} and x_{max} represent the minimum and maximum values of the feature, respectively. Additionally, v_{max} is set to 1 as the new upper limit, and v_{min} is set to 0 as the new lower limit. The partial results of implementing feature scaling are shown in Table 2.

Table 2

Illustration of the Data Features Before (Left) and After (Right) Scaling

No.	Before Scaling			After Scaling		
	byte	dll	dd	byte	dll	dd
1	0	11	17703	0	0.0328358	0.0964084
2	0	0	1273	0	0	0.00693261
3	0	0	230	0	0	0.00125255
4	0	0	1527	0	0	0.00831586
5	0	13	107609	0	0.038806	0.586026
6	0	0	52	0	0	0.000283186
7	1	1	12926	0.0188679	0.00298507	0.0703935
8	0	0	2025	0	0	0.0110279
9	0	101	54254	0	0.301493	0.295461
10	0	0	687	0	0	0.00374132

Feature Selection and Modeling

This step comprises two activities: feature selection and modelling. Feature selection is crucial for identifying and preserving the most important features that significantly enhance the predictive capability of the machine learning model. This process helps to decrease the dataset's dimensionality, boost model performance, and reduce the risk of overfitting (Warwicker & Rebennack, 2024). On the other hand, modelling involves training the machine learning algorithm using the selected features to create a predictive model. This model is then evaluated using various metrics to ensure its effectiveness and efficiency. By integrating these two processes, we aim to develop a robust and efficient machine learning system capable of accurately detecting malware.

There are five experimental steps in this research. First, we compared the performance of six classifiers: Decision Tree, Random Forest, XGBoost, kNN, Logistic Regression, and Naïve Bayes. These classifiers are well-known for their strong performance across various datasets. We measured their performance to determine the best classifier in terms of accuracy, recall, precision, and F1-Score. The initial comparison was conducted without applying any feature selection techniques.

The hyperparameters used for the six algorithms in this research are as follows:

1. Random Forest- The number of trees is set to 50, and the maximum depth of trees is 3.
2. kNN- The number of neighbours is set to 5, using the Euclidean distance metric, and the weight function is by distance.
3. Decision Tree- The maximum depth is set to 10, with a minimum samples split of 5 and a minimum of 1 sample per leaf.
4. XGBoost- The learning rate is 0.300, the number of trees is 100, lambda is 1, and the maximum depth is 6.
5. Logistic Regression- The regularization strength is 0.900, using Lasso (L1) regularization.
6. Naïve Bayes - The type of distribution used is multinomial, with a smoothing parameter (alpha).

In the second experiment, we evaluated the six classifiers using the Information Gain feature selection technique. Recent studies indicate that incorporating Information Gain in feature selection can enhance the performance of machine learning classifiers (Omuya et al., 2021; Supriyanto et al., 2024). After obtaining the information gain score for each feature, feature selection is conducted using forward selection until we find the best evaluation metric scores from the top 60 features, which are better than those from the first experiment (without feature selection). Information Gain is a widely used and popular feature selection method (Kurniabudi et al., 2020). The most significant features are determined according to their entropy values (as shown in Equation 2).

$$Entropy\ (S) = -\sum_{i}^{c} P_i\ Log_2\ P_i \tag{2}$$

The entropy formula measures the amount of uncertainty or disorder within a dataset (S). In this formula, entropy (S) represents the entropy of the set (S). The summation symbol indicates that we sum over all possible classes (i) from 1 to (c), where (c) is the total number of classes. (P_i) is the probability of an instance being classified into class (i), and ($\log_2$ (P_i)) is the logarithm base 2 of (P_i). This calculation helps quantify the unpredictability in the data, with higher entropy indicating greater disorder and lower predictability.

Once the entropy values are obtained, Information Gain can be calculated using Equation 3. The Information Gain (IG) formula is used to measure the reduction in entropy when a dataset (S) is split based on an attribute (A). In this formula, (Gain(S, A)) represents the Information Gain of attribute (A) with respect to the dataset (S). (Entropy(S)) is the entropy of the entire dataset (S), which measures the amount of uncertainty or impurity in the dataset. The summation formula runs over all possible values (v) of the attribute (A). For each value (v), ($|S_v|$) is the number of instances in the subset (S_v) where the attribute (A) has the value (v), and (|S|) is the total number of instances in the dataset (S). (Entropy(S_v)) is the entropy of the subset (S_v). The term ($|S_v|/|S|$) represents the proportion of the dataset that falls into subset (S_v). Information Gain measures the effectiveness of an attribute (A) in minimizing uncertainty by subtracting the weighted sum of the entropies of the subsets from the overall entropy of the dataset. A higher IG score signifies that the attribute offers the most valuable information for classification, highlighting its importance in the decision-making processes of machine learning models.

$$Gain\ (S, A) = Entropy(s) - \sum |Sv|\ |S|\ Entropy(Sv)\ Values(A) \tag{3}$$

In the third experiment, we used the Chi-Square (X^2) feature selection method in the same manner as the second experiment. Chi-Square is also a well-known feature selection method that has been used in several studies to improve the performance of classifiers (Abujazoh et al., 2023; Yang et al., 2024). The Chi-Square test is frequently employed as a feature selection technique to identify the most significant features within a dataset. This process involves computing the Chi-Square statistic for each feature and then selecting the top (k) features that exhibit the highest Chi-Square values. By focusing on these top features, the model can be more efficient and accurate, as it leverages the most relevant data points that have a strong association with the target variable. This method not only enhances the model's performance but also reduces computational time by eliminating less important features (Rasheed et al., 2023). The formula for Chi-Square is outlined in Equation 4.

$$x^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{4}$$

The Chi-Square formula (i.e., Equation 4) is used to measure the difference between observed and expected frequencies in a dataset. Here, (x^2) represents the Chi-Square statistic, (O_{ij}) is the observed frequency in cell (i, j), and (E_{ij}) is the expected frequency in the same cell. The summation runs over all attribute values (i) and class labels (j), with (m) being the number of attribute values and (k) the number of class labels. This formula helps identify features that have a significant association with the target variable, where higher Chi-Square values suggest stronger relationships. This, in turn, facilitates effective feature selection for machine learning models.

In the fourth experiment, we implemented PCA to six algorithms to reduce the dataset's dimensionality while preserving as much information as possible. Recent studies have also incorporated PCA to improve the performance of classifiers (Omuya et al., 2021; Supriyanto et al., 2024). PCA converts the original features into a new set of uncorrelated features arranged in order of the variance they explain. This method helps in simplifying the data, making it easier and faster for machine learning algorithms to process by focusing on the most significant components (Dissanayake et al., 2022). Each algorithm in this experiment is combined with PCA to achieve the optimum evaluation score. PCA effectively balances the trade-off between dimensionality reduction and maintaining the integrity of the data, making data analysis more efficient and manageable.

Finally, in the fifth experiment, we proposed a combination of the Information Gain and Chi-Square methods for feature selection, leveraging their complementary strengths to enhance the performance of malware detection systems. This new approach is called the IGCS, and since it produces scores in different ranges, it is necessary to normalize both scores to the same range. The normalization process can use MinMax Normalization (0,1), as discussed in the previous sub-section. The formula of IGCS is presented in Equation 5.

$$IGCS = Normalized(IG) + Normalized(X^2) \quad (5)$$

Normalized (IG) represents the Information Gain score that has been normalized using the MinMax Normalization (0,1) method. Similarly, Normalized (X^2) denotes the Chi-Square score that has been normalized using the same MinMax Normalization (0,1) method. These normalized scores are then summed to produce an IGCS score for each feature. Subsequently, the IGCS scores are sorted in descending order. Using forward selection, the top 10 to 60 features are selected to measure the best performance of each classifier. In a technical context, Algorithm 1 provides the pseudocode for the IGCS feature selection method. The function 'Calculate_IGCS' sums the scores of Information Gain (IG) and Chi-Square (CS or X^2) after they have been normalized using MinMax Normalization (range 0 to 1). On the other hand, the function 'Normalize_MinMax' is used to perform normalization on the IG and X^2 scores using MinMax Normalization method, with a minimum value of 0 and a maximum value of 1.

Algorithm 1: IGCS

```
Function Calculate_IGCS(IG, CS):
  | IGCS = Normalize_MinMax(IG) + Normalize_Minmax(CS)
  | Return IGCS

Function Normalize_MinMax(Value):
  | #Perform MinMax Normalization(0 to 1)
  | Value_Min = Minimum_Value(Data)
  | Value_Max = Maximum_Value(Data)
  | Normalized_Value = (Value – Value_Min) / (Value_Max –
  | Value_Min)
  | Return Normalized_Value

# Example usage:
IG = Calculate_InformationGain(Data)
CS = Calculate_ChiSquared(Data)
IGCS_Score = Calculate_IGCS(IG, CS)
```

To demonstrate that our proposed feature selection method is superior to other methods, we apply it to six different machine learning algorithms in the fifth experiment and compare the results with those from the first experiment (without feature selection), the second experiment (with Information Gain), the third experiment (with Chi-Square), and the fourth experiment (with PCA). The discussion of the performance evaluation metrics used in these experiments is available in the Subsection Evaluation.

Evaluation

Assessment is a vital component in data mining, as it allows for the evaluation of the effectiveness and accuracy of the developed models. This process involves analyzing various performance metrics, including accuracy, recall, precision, and F1-score, to confirm that the model meets the required standards. This step is essential to identify any potential issues, validate the results, and make necessary adjustments to improve the model's performance. Without proper evaluation, the reliability and applicability of the data mining results could be compromised, leading to incorrect conclusions and decisions.

To evaluate the models, we first employed 10-cross fold validation, followed by calculating the scores of various classification performance metrics. 10-fold cross-validation is a reliable technique for evaluating a model's generalizability. This method entails dividing the dataset into ten equal segments, known as "folds." The model is trained on nine of these folds while the remaining fold is used for testing. This procedure is repeated ten times, ensuring that each fold acts as the test set exactly once. The results are then averaged to provide a comprehensive evaluation of the model's performance (Supriyanto et al., 2024). This method helps to minimize bias and variance, providing a more accurate estimate of the model's effectiveness. By using 10-fold cross-validation, we ensure that our evaluation is thorough and reliable, leading to more trustworthy conclusions about the model's capabilities (Battineni et al., 2019; Orrù et al., 2020).

The performance metrics used in this research are Accuracy, Recall, Precision, and F1-Score. Accuracy represents the proportion of correctly predicted instances relative to the total number of instances in the

dataset. It serves as a clear indicator of the model's overall performance. When using a balanced dataset like we use in this research, accuracy becomes a more reliable metric because it reflects the model's ability to correctly classify instances across all classes without bias. To measure accuracy, we calculate the ratio of correct predictions to the total number of predictions made by the model. The formula for accuracy is presented in Equation 6. True Positives (TP) refer to the instances where the model accurately identifies the positive class, while True Negatives (TN) denote the instances correctly classified as negative. False Positives (FP) arise when the model mistakenly predicts the positive class, and False Negatives (FN) occur when the model inaccurately identifies the negative class.

$$Accuracy = \frac{TP + TN}{(TP + FP + TN + FN)} \tag{6}$$

Recall is a performance metric in machine learning that assesses a model's capability to detect all positive instances within a dataset. It is calculated as the ratio of true positives to the total of true positives and false negatives. Recall is crucial in scenarios where identifying all positive cases is more important than avoiding false positives, such as in disease detection or malware identification. It complements precision and is often used together with it to provide a comprehensive evaluation of a model's performance, particularly through the F1-score. The recall formula is presented in Equation 7.

$$Recall = \frac{TP}{TP + FN)} \tag{7}$$

Precision is a metric in machine learning that evaluates the accuracy of positive predictions. It is calculated as the ratio of true positives to the total of true positives and false positives. This metric is particularly important in scenarios where the consequences of false positives are significant, such as in spam or fraud detection. In the context of malware detection, high precision ensures that most of the files identified as malware are indeed malicious, minimizing the risk of falsely flagging legitimate files. Precision, along with recall, provides a balanced view of a model's performance, often summarized through the F1-score. The formula of precision is presented in Equation 8.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Considering the importance of false positives and false negatives in malware detection, the F1-Score plays a vital role as it takes both precision and recall into account, offering a more comprehensive assessment of the model's performance. The F1-Score calculates the harmonic mean of precision and recall, thus providing a balanced evaluation of the model's effectiveness. The formula of the F1-score is presented in Equation 9.

$$F1\ Score = \frac{2\ x\ (Precision\ x\ Recall)}{Precision + Recall} \tag{9}$$

By employing accuracy, recall, precision, and F1-score on a balanced dataset, we can attain a thorough and fair assessment of the model's performance, ensuring strong performance across all classes. These metrics offer valuable insights into the efficacy of the proposed method in identifying malicious software. Additionally, processing time was evaluated to gauge the efficiency of the proposed method in malware detection. Training time indicates the length of time necessary to train the model on the

dataset, encompassing feature selection and model fitting. On the other hand, testing time represents the duration required to assess the trained model on the test dataset, which includes the time spent on feature selection and making predictions.

RESULT AND ANALYSIS

This section discusses the result of each step explained in the previous section, starting with the dataset preparation. There were three dataset files that needed to be merged into a single dataset file. Due to differing features among these datasets, we first equalized them by removing features that were not available in all datasets. This step is called feature reduction, led to the removal of 'vbaVarIndexLoad' and 'SafeArrayPtrOfIndex' features from the VxHeaven and VirusTotal datasets, as well as the 'filename' and 'compiled_date' features from all three datasets.

After ensuring that the three datasets shared the same number of features, we merged them into a single dataset and added a label: 0 for instances from the goodware dataset and 1 for instances from the malware datasets, sourced from both VxHeaven and VirusTotal. This process resulted in a dataset containing 6,248 instances, 1,083 features, and a target variable with two classes (0 and 1). Due to the dataset's imbalance with a ratio of 1:9.5, which is classified as medium imbalance, random under-sampling was necessary to equalize the number of instances. As a result, the number of instances in both the goodware and malware classes is equal, with 595 instances each. Next, we applied constant feature removal, eliminating 933 features that had constant values, which left us with 150 features. Removing constant features is essential because they do not contribute to the model's predictive power and can lead to overfitting, thereby degrading model performance. Subsequently, we applied feature scaling using MinMax Normalization (1,0). This step ensures that all remaining features lie within the same range, enhancing model stability and improving convergence rates during training. Feature scaling also prevents features with larger scales from disproportionately influencing the model.

In the experiment, we implemented six different classifiers with four different feature selection methods. The classifiers used were kNN, LR, NB, DT, XGBoost, and RF, with hyperparameters as mentioned in chapter three. The feature selection methods used were IG, X^2, PCA, and our proposed method, IGCS. The optimal number of features was determined based on experimental results using forward selection technique from the top 60 features. Firstly, we compared the performance of these six classifiers without performing any feature selection. The outcomes of this comparison are shown in Table 3. In this experiment, the most effective classifier for malware detection was Random Forest, achieving an Accuracy, F1-Score, Precision, Recall, and Specificity of 0.987. In contrast, Naïve Bayes had the lowest scores compared to the other five algorithms. On the other hand, Decision Tree was the fastest algorithm in terms of training and testing time, whereas XGBoost was the slowest. By applying feature selection, the performance metrics scores can be increased, while the time taken for training and testing can be decreased.

Table 3

Performance of Various Classifiers Without Feature Selection

Classifiers	Performance Metrics				Processing Time	
	Accuracy	Precision	Recall	F1 Score	Training	Testing
Random Forest	0.987	0.987	0.987	0.987	3.979	0.843
XGBoost	0.982	0.982	0.982	0.982	5.611	0.257
kNN	0.965	0.965	0.965	0.965	1.541	1.058
Decision Tree	0.962	0.962	0.962	0.962	1.379	0.002
Logistic Regression	0.962	0.962	0.962	0.962	2.898	0.518
Naïve Bayes	0.918	0.918	0.918	0.918	2.499	0.746

In the next step, we implement various feature selection methods in each classifier using forward selection technique. Feature selection is widely known and can be used to speed up the process as well as increasing accuracy, precision and other score on performance metrics. The first feature selection method that was implemented is Information Gain. The Information Gain of each feature was calculated first and then sorted in descending order. A higher Information Gain score indicates that the feature has a greater ability to distinguish between different classes. After identifying the top features based on Information Gain, we integrated these features into our machine learning models. Table 4 shows the highest performance metrics score of each algorithm with the optimum number of features. The performance of most classifiers was successfully increased, especially in terms of performance metrics. The overall performance of Random Forest was increased from 98.7% to 98.9% with only 35 out of 150 features. kNN performed the same score in evaluation metrics, but the number of features significantly decreased to 40. A significant improvement was achieved by Naïve Bayes when combined with Information Gain. By using 45 features, the evaluation score of Naïve Bayes was increased from 91.8% to 97.1% of accuracy. Other evaluation metrics were also increased as well to 97.1%.

The third experiment involved implementing the Chi-Square (X^2) feature selection method on six different classifiers. After combining the classifiers with Chi-Square, all of them performed better than without feature selection. Since Chi-Square and Information Gain are ranking-based feature selection methods, the process is similar. First, the Chi-Square score for each feature is computed and then arranged in descending order. Using forward selection up to 60 features, we chose the most optimal number of features that produced the highest performance metrics. The accuracy of Random Forest increased from 0.987 to 0.989 with 60 Chi-Square features. XGBoost also showed improvement in performance metrics, increasing from 0.982 to 0.985. Logistic Regression had a significant improvement, increasing from 0.962 to 0.966 with 54 features, which is even better than the combination of Logistic Regression with Information Gain, which had 0.965 in accuracy, precision, recall, and F1-Score. The Decision Tree also performed better with Chi-Square, achieving a score of 0.969 across all metrics. This performance was superior to the Decision Tree without feature selection and even better than with Information Gain. Table 4 presents the top performance metrics scores for each algorithm, achieved with the optimal number of features.

Table 4

Performance of Various Classifiers With Information Gain, Chi-Square, and PCA

Classifiers	Information Gain					Chi-Square (X^2)					PCA				
	Feat No.	Acc	Prec	Rec	F1	Feat No.	Acc	Prec	Rec	F1	Comps. No.	Acc	Prec	Rec	F1
RF	35	**0.989**	**0.989**	**0.989**	**0.989**	60	**0.989**	**0.989**	**0.989**	**0.989**	10	**0.988**	**0.988**	**0.988**	**0.988**
XGBoost	30	0.985	0.985	0.985	0.985	55	0.985	0.985	0.985	0.985	10	0.982	0.982	0.982	0.982
kNN	40	0.965	0.965	0.965	0.965	55	0.968	0.968	0.968	0.968	10	0.965	0.965	0.965	0.965
DT	45	0.963	0.963	0.963	0.963	27	0.969	0.969	0.969	0.969	10	0.962	0.962	0.962	0.962
LR	35	0.965	0.965	0.965	0.965	54	0.966	0.966	0.966	0.966	10	0.962	0.962	0.962	0.962
NB	45	0.971	0.971	0.971	0.971	40	0.966	0.939	0.939	0.939	10	0.918	0.918	0.918	0.918

Note: RF: Random Forest; XGBoost: Extreme Gradient Boosting; kNN: k-Nearest Neighbour; DT: Decision Tree; LR: Logistic Regression; NB: Naïve Bayes; Feat No.: Number of Features used in feature selection method; Acc: Classification Accuracy; Prec: Precision; Rec: Recall; F1: F1-Score; Comps. No.: Number of Components used in PCA.

The fourth experiment involves applying PCA across six different algorithms. PCA is a well-known feature extraction technique that enhances the performance of classifiers by decreasing the dataset's dimensionality while preserving the majority of its variance. This reduction helps in mitigating the curse of dimensionality, improving computational efficiency, and potentially enhancing the model's performance. In this experiment, we applied PCA to transform the original features into a set of principal components and then used these components as inputs for the classifiers. We used 10 principal components, which explained 80% of the variance. As a result, the accuracy of Random Forest was slightly improved from 0.987 to 0.988, while the performance of the remaining algorithms was successfully maintained. The results of the PCA experiment with various machine learning algorithms are presented in Table 4.

The final experiment employed our proposed method, called the IGCS (Information Gain – Chi-Square) hybrid method, on six classifiers. This method combines the Information Gain and Chi-Square feature selection methods. Both feature selection methods performed well in most situations, inspiring us to combine their strengths to create a more robust feature selection method. This claim is supported by our second and third experiments, in which Information Gain and Chi-Square successfully improved the performance of all algorithms, unlike PCA, which only slightly enhanced the performance of Random Forest.

The IGCS score was obtained using Equation 5. First, we calculated the Information Gain Score, followed by the Chi-Square Score. These scores were then normalized using MinMax Normalization (0,1) before being summed up to yield the final IGCS Score. Descending sorting was conducted to rank the IGCS Score from high to low. A high IGCS Score indicates a feature with a significant impact, while a low IGCS Score indicates a feature with minimal impact. The illustration and result of the IGCS process are shown in Table 5.

Table 5

Top Six Features and Their Scores Using Information Gain, Chi-Square, and IGCS

Feature Ranking	Information Gain			Chi-Square			IGCS (Proposed)	
	Features Name	**IG Score**	**Normalized (IG)**	**Features Name**	**X^2 Score**	**Normalized X^2**	**Features Name**	**IGCS Score**
1	Minor_image_ version	0.761	1.000	Minor_image_ version	781.947	1.000	Minor_image_ version	2.000
2	minor_operating_ system_version	0.716	0.941	minor_operating_ system_version	705.179	0.902	minor_operating _ system_version	1.843
3	major_operating_ system_version	0.639	0.839	characteristics	560.419	0.717	major_image_ version	1.371
4	size_of_stack_ reserve	0.626	0.822	major_image_ version	501.791	0.642	major_operating _ system_version	1.346
5	minor_linker_ version	0.566	0.743	file_operations	453.0.5	0.579	characteristics	1.227
6	major_image_ version	0.555	0.729	major_linker_ version	437.405	0.559	dll_ characteristics	1.169

The results of the fifth experiment are shown in Table 6. Random Forest achieved the highest score when combined with IGCS, with 99% accuracy, precision, recall, and F1-Score. This score is better than Random Forest without feature selection (98.7%), with 35 features of Information Gain (98.9%), with 60 features of Chi-Square (98.9%), and with 10 components of PCA (98.8%). This combination is also superior to other algorithms when combined with various feature selection methods, including IGCS (See Figure 2).

Table 6

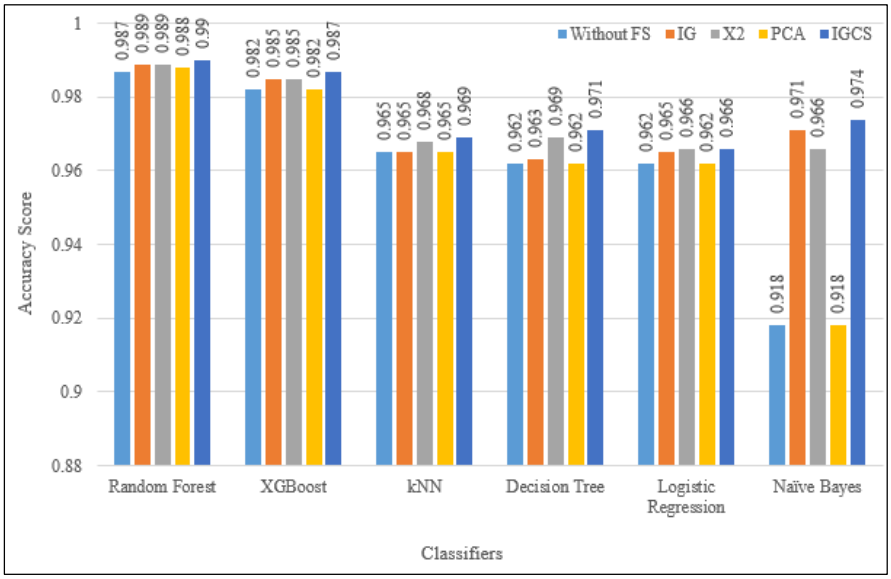
Performance of Various Algorithms with IGCS

Classifiers	Number of Features	Performance Metrics			
		Accuracy	Precision	Recall	F1-Score
Random Forest	30	**0.990**	**0.990**	**0.990**	**0.990**
XGBoost	30	0.987	0.987	0.987	0.987
kNN	41	0.969	0.969	0.969	0.969
Decision Tree	14	0.971	0.971	0.971	0.971
Logistic Regression	30	0.966	0.966	0.966	0.966
Naïve Bayes	40	0.974	0.974	0.974	0.974

The other five classifiers also improved when combined with IGCS. XGBoost achieved 98.7% when combined with 30 features of IGCS. This is better than XGBoost combined with 30 features of Information Gain (98.5%), 55 features of Chi-Square (98.5%), 10 components of PCA (98%), or even without feature selection (98.2%). Having fewer features can lead to optimal performance in terms of processing time. kNN classifiers achieved 96.9% accuracy with 41 features of IGCS. This was an improvement compared to kNN with 40 features of Information Gain (96.5%), 55 features of Chi-Square (96.8%), 10 components of PCA (96.5%), or without feature selection (96.5%).

Figure 2

Accuracy Performance of Various Classifiers using Different Feature Selection Methods, Including IGCS



Decision Tree also reached peak performance when combined with our proposed method. With IGCS, Decision Tree needed only 14 features to obtain 97.1% accuracy. This score is higher than the performance of Decision Tree when combined with 10 components of PCA (96.2%), 27 features of Chi-Square (96.9%), 45 features of Information Gain (96.3%), or without feature selection (96.2%). Logistic Regression, by employing 30 features of IGCS, achieved 96.6% accuracy. This score is similar to Logistic Regression combined with 54 features of Chi-Square (96.6%). However, the number of features used by Logistic Regression and IGCS to reach that score is significantly reduced (30 features) compared to Chi-Square (54). This performance is also better than other combinations, such as 10 components of PCA (96.2%), 35 features of Information Gain (96.5%), and plain Logistic Regression without feature selection (96.2%). Lastly, Naïve Bayes also performed well when combined with IGCS, achieving 97.4% accuracy with 40 features. This performance is significantly improved compared to Naïve Bayes without feature selection (91.8%), with 10 components of PCA (91.8%), with 40 features of Chi-Square (96.6%), and with 45 features of Information Gain (97.1%).

According to these experiments, it is proven that IGCS can enhance the performance metrics of machine learning classifiers. The highest performance was achieved by the Random Forest classifier using 30 features selected through IGCS. This performance surpasses the two state-of-the-art methods proposed by Abujazoh et al. (2023) and Supriyanto et al. (2024) using similar datasets, as shown in Table 7. The combination of Random Forest and IGCS achieved the highest performance with 99.0% for both accuracy and F1-Score. The number of features used in our proposed work is 30, the same as the number of features used by Abujazoh et al. (2023). Additionally, our method is also superior to the work proposed by Supriyanto et al. (2024), which implemented PCA on the kNN algorithm for malware detection, using 32 components and achieving the best performance of 96.9% for both Accuracy and F1-Score. This means that our proposed method is more effective and efficient, as it uses fewer features. The comparison results are shown in Table 7.

Table 7

Performance Comparison of the Proposed Work with the State-of-the-arts

Classifiers	Feature Selection/ Extraction Method	Number of Features/ Components Used	Accuracy	F1-Score
Random Forest	IGCS (proposed)	30	99.0%	99.0%
Decision Tree (Abujazoh et al., 2023)	Chi-Square	30	98.53%	-
k-NN (Supriyanto et al., 2024)	PCA	32	96.9%	96.9%

The next experiment measures the efficiency of our method compared to without implementing any feature selection. By using the optimal number of features of each algorithm, the processing time of classifiers can also be accelerated, as shown in Table 8. This experiment reveals a substantial reduction in processing time for both training and testing phases when implementing the IGCS feature selection method compared to without feature selection. Specifically, the training time decreased from 3.884 seconds to 1.845 seconds, a reduction of approximately 52.5%. Similarly, the testing time reduced from 0.843 seconds to 0.363 seconds, marking a 56.9% decrease. This significant decrease indicates that IGCS effectively reduces the processing time through minimizing the number of features the model needs to process, thereby enhancing overall efficiency. Consequently, IGCS not only boosts model performance but also ensures faster execution, making it highly advantageous for real-time applications.

The same improvement was observed across several other algorithms tested in this study, such as XGBoost, kNN, Decision Tree, Logistic Regression, and Naïve Bayes. There was a significant increase in processing speed when applying IGCS feature selection compared to without it.

Table 8

Comparison of Processing Time Among Classifiers with and without IGCS (in Seconds)

Classifiers	Without Feature Selection		With IGCS	
	Training Time	Testing Time	Training Time	Testing Time
Random Forest	3.884	0.843	1.845	0.363
XGBoost	4.405	0.257	1.042	0.052
kNN	1.356	1.058	0.544	0.457
Decision Tree	0.843	0.002	0.219	0.001
Logistic Regression	2.760	0.518	0.775	0.107
Naïve Bayes	2.116	2.499	0.506	0.165

CONCLUSION

Malware poses a significant and evolving threat in today's digital landscape. Detecting malware is crucial as it protects devices and systems from dangers such as data loss or damage, data theft, account breaches, and unauthorized access that can lead to full system control. Given that malware has evolved from traditional forms (monomorphic) to more advanced forms (oligomorphic, polymorphic, and metamorphic), a machine learning-based detection system is needed instead of the traditional signature-based approach. This research proposes the IGCS method, a hybrid feature selection technique that combines Information Gain and Chi-Square (X^2) to enhance the performance of machine learning classifiers in terms of accuracy, recall, precision, F1-Score, as well as training and testing time. Using IGCS, six classifiers—Random Forest, XGBoost, kNN, Decision Tree, Logistic Regression, and Naïve Bayes—achieved higher performance scores compared to other scenarios, such as when classifiers were combined with Information Gain, Chi-Square, PCA, or even without any feature selection. As a result, Random Forest with 30 features of IGCS proved superior to any combination of classifiers and feature selection methods in malware detection including state-of-the-art methods, achieving 99.0% accuracy. This combination was also efficient in both training and testing.

Future work is needed to enhance the performance of classifiers to reach zero tolerance for false positives and false negatives, given the severe impact of malware attacks. Additionally, this method also needs to be tested alongside other hybrid methods and on different datasets to ensure its robustness and generalizability across various types of malware. Exploring the integration of other advanced feature selection techniques and machine learning algorithms could further improve detection accuracy and efficiency. Future research should also evaluate both the computational complexity and overhead introduced by the IGCS method using Big O Notation to understand its impact on system performance and scalability when applied to larger datasets. Continuous updates and adaptations to the evolving nature of malware are essential to maintain the effectiveness of the detection system.

ACKNOWLEDGMENT

This work was supported by the Directorate General of Higher Education, Research, and Technology (DGHERT) of the Ministry of Education, Culture, Research, and Technology (MOECRT) of the Republic of Indonesia, which funded this research through the Fundamental Research - Regular scheme.

REFERENCES

Abujazoh, M., Al-Darras, D., A. Hamad, N., & Al-Sharaeh, S. (2023). Feature selection for high-dimensional imbalanced malware data using filter and wrapper selection methods. *2023 International Conference on Information Technology (ICIT)*, 196–201. https://doi.org/10.1109/ICIT58056.2023.10226049

Aslan, O., & Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access*, *8*, 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

Bao, H., Li, W., Chen, H., Miao, H., Wang, Q., Tang, Z., Liu, F., & Wang, W. (2024). Stories behind decisions: Towards interpretable malware family classification with hierarchical attention. *Computers & Security*, *144*, 103943. https://doi.org/10.1016/j.cose.2024.103943

Battineni, G., Sagaro, G. G., Nalini, C., Amenta, F., & Tayebati, S. K. (2019). Comparative machine-learning approach: A follow-up study on type 2 diabetes predictions by cross-validation methods. *Machines*, *7*(4), 74. https://doi.org/10.3390/machines7040074

Dabas, N., Ahlawat, P., & Sharma, P. (2023). An effective malware detection method using hybrid feature selection and machine learning algorithms. *Arabian Journal for Science and Engineering*, *48*(8), 9749–9767. https://doi.org/10.1007/s13369-022-07309-z

Dissanayake, S., Gunathunga, S., Jayanetti, D., Perera, K., Liyanapathirana, C., & Rupasinghe, L. (2022). An analysis on different distance measures in KNN with PCA for Android malware detection. *2022 22nd International Conference on Advances in ICT for Emerging Regions (ICTer)*, 178–182. https://doi.org/10.1109/ICTer58063.2022.10024079

Fan, Q., Wang, Z., Li, D., Gao, D., & Zha, H. (2017). Entropy-based fuzzy support vector machine for imbalanced datasets. *Knowledge-Based Systems*, *115*, 87–99. https://doi.org/10.1016/j.knosys.2016.09.032

Hossain, Md. A., Haque, M. A., Ahmad, S., Abdeljaber, H. A. M., Eljialy, A. E. M., Alanazi, A., Sonal, D., Chaudhary, K., & Nazeer, J. (2024). AI-enabled approach for enhancing obfuscated malware detection: A hybrid ensemble learning with combined feature selection techniques. *International Journal of System Assurance Engineering and Management*. https://doi.org/10.1007/s13198-024-02294-y

Kale, G., Bostancı, G. E., & Çelebi, F. V. (2024). Evolutionary feature selection for machine learning based malware classification. *Engineering Science and Technology, an International Journal*, *56*, 101762. https://doi.org/10.1016/j.jestch.2024.101762

Keshkeh, K., Jantan, A., & Alieyan, K. (2022). A machine learning classification approach to detect tls-based malware using entropy-based flow set features. *Journal of Information and Communication Technology*, *21*. https://doi.org/10.32890/jict2022.21.3.1

Kurniabudi, Stiawan, D., Darmawijoyo, Bin Idris, M. Y., Bamhdi, A. M., & Budiarto, R. (2020). CICIDS-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access*, *8*, 132911–132921. https://doi.org/10.1109/ACCESS.2020.3009843

Liu, J., Xiao, Q., Xin, L., Wang, Q., Yao, Y., & Jiang, Z. (2023). M3F: A novel multi-session and multi-protocol based malware traffic fingerprinting. *Computer Networks*, *227*, 109723. https://doi.org/10.1016/j.comnet.2023.109723

Lu, J., Ren, X., Zhang, J., & Wang, T. (2023). CPL-Net: A malware detection network based on parallel CNN and LSTM feature fusion. *Electronics*, *12*(19), 4025. https://doi.org/10.3390/electronics12194025

Malware static and dynamic features VxHeaven and Virus Total. (2019). [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C58K6H

Omuya, E. O., Okeyo, G. O., & Kimwele, M. W. (2021). Feature selection for classification using principal component analysis and information gain. *Expert Systems with Applications*, *174*, 114765. https://doi.org/10.1016/j.eswa.2021.114765

Orrù, G., Monaro, M., Conversano, C., Gemignani, A., & Sartori, G. (2020). Machine learning in psychometrics and psychological research. *Frontiers in Psychology*, *10*, 2970. https://doi.org/10.3389/fpsyg.2019.02970

Pandey, A., & Jain, A. (2017). Comparative analysis of KNN algorithm using various normalization techniques. *International Journal of Computer Network and Information Security*, *9*(11), 36–42. https://doi.org/10.5815/ijcnis.2017.11.04

Rafrastara, F. A., Supriyanto, C., Amiral, A., Amalia, S. R., Fahreza, M. D. A., & Ahmed, F. (2024). Performance comparison of k-Nearest Neighbour algorithm with various k values and distance metrics for malware detection. *Jurnal Media Informatika Budidarma, 8,* 450-458.

Rafrastara, F. A., Supriyanto, C., Paramita, C., Astuti, Y. P., & Ahmed, F. (2023). Performance improvement of random forest algorithm for malware detection on imbalanced dataset using random under-sampling method. *Jurnal Informatika: Jurnal Pengembangan IT*, *8*(2), 113–118. https://doi.org/10.30591/jpit.v8i2.5207

Rasheed, A. F., Zarkoosh, M., & Al-Azzawi, S. S. (2023). The impact of feature selection on malware classification using Chi-square and machine learning. *2023 9th International Conference on Computer and Communication Engineering (ICCCE)*, 211–216. https://doi.org/10.1109/ICCCE58854.2023.10246084

Root, S. J., Throckmorton, P., Tacke, J., Benjamin, J., Haney, M., & Borrelli, R. A. (2023). Cyber hardening of nuclear power plants with real-time nuclear reactor operation, 1. Preliminary operational testing. *Progress in Nuclear Energy*, *162*, 104742. https://doi.org/10.1016/j.pnucene.2023.104742

Sachdeva, S., Bhatia, S., Al Harrasi, A., Shah, Y. A., Anwer, Md. K., Philip, A. K., Shah, S. F. A., Khan, A., & Ahsan Halim, S. (2024). Unraveling the role of cloud computing in health care system and biomedical sciences. *Heliyon*, *10*(7), e29044. https://doi.org/10.1016/j.heliyon.2024.e29044

Shahid, N., Aziz-ur Rehman, M., Khalid, A., Fatima, U., Sumbal Shaikh, T., Ahmed, N., Alotaibi, H., Rafiq, M., Khan, I., & Sooppy Nisar, K. (2021). Mathematical analysis and numerical investigation of advection-reaction-diffusion computer virus model. *Results in Physics*, *26*, 104294. https://doi.org/10.1016/j.rinp.2021.104294

Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, *97*, 105524. https://doi.org/10.1016/j.asoc.2019.105524

Supriyanto, C., Rafrastara, F. A., Amiral, A., Amalia, S. R., Daffa, M., & Fahreza, A. (2024). Malware detection using K-nearest neighbour algorithm and feature selection. *Jurnal Media Informatika Budidarma*, *8*(1), 412-420

Teodorescu, C. A. (2022). Perspectives and reviews in the development and evolution of the zero-day attacks. *Informatica Economica*, *26*(2/2022), 46–56. https://doi.org/10.24818/issn14531305/26.2.2022.05

Vostoupal, J. (2024). Stuxnet vs WannaCry and Albania: Cyber-attribution on trial. *Computer Law & Security Review*, *54*, 106008. https://doi.org/10.1016/j.clsr.2024.106008

Wang, H., Cui, B., Yuan, Q., Shi, R., & Huang, M. (2024). A review of deep learning based malware detection techniques. *Neurocomputing*, *598*, 128010. https://doi.org/10.1016/j.neucom.2024.128010

Warwicker, J. A., & Rebennack, S. (2024). Support vector machines within a bivariate mixed-integer linear programming framework. *Expert Systems with Applications*, *245*, 122998. https://doi.org/10.1016/j.eswa.2023.122998

Yan, C., & Razmjooy, N. (2023). Optimal lung cancer detection based on CNN optimized and improved snake optimization algorithm. *Biomedical Signal Processing and Control*, *86*, 105319. https://doi.org/10.1016/j.bspc.2023.105319

Yang, F., Xu, Z., Wang, H., Sun, L., Zhai, M., & Zhang, J. (2024). A hybrid feature selection algorithm combining information gain and grouping particle swarm optimization for cancer diagnosis. *PLOS ONE*, *19*(3), e0290332. https://doi.org/10.1371/journal.pone.0290332

Yin, Y., Jang-Jaccard, J., Xu, W., Singh, A., Zhu, J., Sabrina, F., & Kwak, J. (2023). IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset. *Journal of Big Data*, *10*(1), 15. https://doi.org/10.1186/s40537-023-00694-8

Zhao, Z., Yang, S., & Zhao, D. (2023). A new framework for visual classification of multi-channel malware based on transfer learning. *Applied Sciences*, *13*(4), 2484. https://doi.org/10.3390/app13042484