Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 6, December 2024, pp. 1663-1672

DOI: https://doi.org/10.52436/1.jutif.2024.5.6.3898
p-ISSN: 2723-3863
e-ISSN: 2723-3871

OPTIMIZING ANDROID MALWARE DETECTION USING NEURAL NETWORKS AND FEATURE SELECTION METHOD

Jevan Bintoro[1], Fauzi Adi Rafrastara[*2], Ines Aulia Latifah[3], Wildani Ghozi[4], Warusia Yassin[5]

[1,2,3,4]Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Indonesia
[5]Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka, Malaysia
Email: [1]111202113433@mhs.dinus.ac.id, [2]fauziadi@dsn.dinus.ac.id, [3]111202113408@mhs.dinus.ac.id, [4]wildanil.ghozi@dsn.dinus.ac.id, [5]s.m.warusia@utem.edu.my

(Article received: October 06, 2024; Revision: November 27, 2024; published: December 29, 2024)

Abstract

Malware poses a serious threat to Android security systems. In recent years, Android malware has rapidly evolved, employing obfuscation techniques such as polymorphic and metamorphic. Unfortunately, signature-based malware detection cannot identify modern variants of Android malware. This study aims to compare various feature selection methods and machine learning algorithms to identify the most effective and efficient combination for classifying Android malware. The dataset used in this research is the Drebin dataset. Four classification algorithms are used in this comparison: Naive Bayes, Logistic Regression, Neural Network, and Random Forest. The best-performing algorithm is then implemented in three different scenarios: without feature selection, with Information Gain, and with Chi-Squared (X^2). In the latter two scenarios, the appropriate number of features was selected using the backward elimination method. Both feature selections achieved the same performance, but Information Gain required fewer features. The evaluation metrics used in this study include AUC, accuracy, F1-score, training time, and testing time. Measuring training and testing time benefits the model by making it more efficient, thus allowing for faster detection in real-world applications. The results show that the combination of the Information Gain feature selection method and the Neural Network algorithm achieves the highest performance, with an accuracy and F1-Score of 98.6%. Additionally, this combination achieves a training time of 81.135 seconds and a testing time of 1.095 seconds. Compared to the Neural Network algorithm without feature selection, this combination results in a 17.7597 % reduction in training time and a 57.9977 % reduction in testing time while maintaining the same performance values. This research contributes to improving the speed and accuracy of malware detection systems, enhancing mobile security.

Keywords: *android malware detection, drebin, information gain, machine learning, neural network.*

1. INTRODUCTION

Over the past five years, the number of Android smartphone users has increased significantly, making it the most popular smartphone operating system (OS) by user count, with 2.5 billion active users in 2019 [1]. The trend indicates that Android's market share has grown from 37% to 75% over seven years [2]. Between April and June 2021, 313 million gadgets (smartphones and tablets) were shipped, with over 80% of these devices running on the Android OS. However, a study found that over 1 billion devices were using an outdated version of the Android operating system, leading to numerous security vulnerabilities [3].

Android security has improved significantly over the years, incorporating features like device and file-based encryption, which are mandatory for all Android devices [4]. The Android security system enforces protections at runtime or during program execution to mitigate malicious code that has already run on the device [5]. Despite these advancements, malware remains a serious threat to the Android OS. Google Play Protect scans 200 billion apps in the Google Play Store daily to safeguard users from malware apps, which are designed to attack smartphone operating systems [6].

In 2010, a mobile malware called FakePlayer became the first trojan targeting the Android OS [7]. FakePlayer generated revenue by sending SMS messages to paid services [8]. These messages could result in charges to the user without their consent.

Another notable mobile malware is Exodus Spyware, which, once installed on a device, grants the attacker full control and access to the device and its data [9]. This malware first verifies the phone number and IMEI to determine if the smartphone is the intended target. If it is, the malware downloads the actual malicious software to hack the phone [10]. Exodus has been used by various public prosecution offices in Italy [11]. In 2016, Google mitigated this malware by closing the vulnerability exploited by DirtyCOW, the program installed by Exodus.

The new generation of malware employs masking techniques such as polymorphic and metamorphic methods, which signature-based

malware detection cannot detect [12]. Polymorphic and metamorphic malware types change their form each time they replicate to infect new files [13]. Signature-based malware detection classifies malware by matching the program's signature with existing malware signatures [14]. The signature of a malware file is created using hash algorithms like SHA1 and MD5. This detection method relies on an existing malware signature database; if the malware file is modified, a new signature must be generated and added to the database, making it less effective [15]. Therefore, an Artificial Intelligence-based detection method is needed to address the weaknesses of signature-based detection.

In this study, Artificial Intelligence (AI) was implemented for malware detection. We aim to identify the best combination of classification algorithms and feature selection methods. The AI malware detection model is trained using classification algorithms, while feature selection methods are employed to enhance the model's process efficiency.

Rana [16] conducted research to evaluate advanced ensemble learning techniques for detecting Android malware. The Drebin dataset was used in this research and underwent substring-based feature selection (SBFS) and data balancing. The model was trained using an advanced ensemble method incorporating Decision Tree, Random Forest, Gradient Boosting Tree, Support Vector Machine, and Logistic Regression. The performance results showed that the highest accuracy achieved was 97.96%. However, this accuracy is not ideal for classifying malware, since it needs zero tolerance for false positive and false negative detection.

Roseline [17] conducted research to detect and classify Android malware using the Leave One Feature Out (LOFO) feature selection method and various tree-based models. The Drebin dataset was utilized for this study. The LOFO method was employed to assess the importance of the dataset's features. The models used in this research included Decision Tree, Random Forest, ExtraTrees, GBM, LightGBM, AdaBoost, and XGBoost. The results indicated that the XGBoost model achieved the best performance with 30 features, attaining an accuracy of 95.59% and an F1-Score of 93.89%. However, this accuracy is not optimal for malware classification, and class balancing was not implemented in this research.

Supriyanto [18] conducted research on detecting malware using the K-Nearest Neighbor (kNN) algorithm with a feature-selected dataset. The datasets used were VxHeaven and VirusTotal. To improve model performance, the dataset underwent Information Gain and Principal Component Analysis (PCA), reducing the number of features to 32. The kNN algorithm was used for classification. The research achieved an accuracy and F1-score of 95.8%. However, the kNN algorithm requires calculating the distance between the query instance and all training instances, making it computationally expensive.

Rafrastara [19] conducted research to detect malware using an ensemble-based stacking method. The dataset used in this research was named "Malware Static and Dynamic Features VxHeaven and VirusTotal Data Set." Due to class imbalance, the dataset underwent class balancing using an undersampling method, reducing the number of instances in each class to 595. Additionally, feature selection reduced the dataset to 1,086 features. The research achieved a score of 95.5% in both accuracy and recall. However, the stacking ensemble method can be computationally intensive, requiring significant resources for training and inference.

Albahar [20] conducted research on the implementation of a modified ResNeXt model for identifying Android malware. The Drebin image dataset was used for this research, and the dataset was transformed into grayscale images for the model to classify. The author used the ResNeXt classification model, a convolutional neural network that employs a strategy called "split-transform-merge," resulting in an accuracy of 98.2%. However, the accuracy achieved in this research is not ideal for classifying malware. Additionally, the effectiveness of the proposed regularization technique is difficult to assess, as the paper does not provide a comparison with other regularization methods.

Based on the literature review above, there are two state-of-the-art studies that utilized the same dataset used in this research, called the Drebin dataset, as conducted by [16], [18]. However, the accuracy of their performance is still not ideal for malware classification. This study aims to identify the optimal combination of machine learning algorithms and feature selection techniques for effective and efficient Android malware detection.

2. RESEARCH METHOD

This study involves five sequential stages, as illustrated in Figure 1. The first stage is dataset collection. The gathered data often has certain issues that need to be addressed, making it crucial to preprocess it in the second stage, Pre-Processing. After preprocessing, the data moves to the Modeling stage, where we implement and compare four algorithms: Naïve Bayes, Neural Network, Logistic Regression, and Random Forest, with and without feature selection. In the Evaluation stage, we use 10-fold cross-validation before evaluating the model's performance based on accuracy, F1-Score, training time and testing time. Lastly, the best combination of feature selection method, number of features, and model becomes the outcome of this research.

A. Hardware and Software

The success of the experiment heavily relies on the use of capable software and hardware. For software, Microsoft Excel is utilized for pre-

processing tasks, such as balancing the data classes in the dataset. For model implementation, the Orange data mining tool (https://orangedatamining.com/) is employed. Orange offers comprehensive features for model development and performance evaluation, enabling quick and precise calculation of metrics like Classification Accuracy, Recall, F1-Score, and Precision, as well as the processing time. Regarding the hardware, this study utilized a computer with the following specifications:

Processor : AMD Ryzen 5 PRO 4650G
RAM : 16 GB
HD : 256 GB SSD + 1 TB SSD
VGA : NVIDIA GeForce RTX 3060

B. Data Collection

The dataset used in this study is Drebin dataset that available in Kaggle dataset repository. The specifications of the dataset are listed below:

Dataset name : Drebin dataset
Number of classes : 2 (B and S)
Number of records : 15036
Number of features : 215 (label excluded)
Missing Value : 5

The dataset contains 5,560 malware records and 9,476 benign records. Due to the differences in the number of records between the classes, the dataset is considered imbalanced. A dataset is said to be imbalanced when one of the target values has significantly fewer instances than the other [21]. Handling imbalanced datasets is recommended, as unequal class distribution can negatively impact the performance of classifiers [22].

C. Pre-Processing

Data preprocessing is defined as a group of methods used to improve the quality of the raw data [23]. Preprocessing reduces data complexity and enhances algorithm performance [24]. In this study, the preprocessing methods used include data cleaning, class balancing, and feature selection. Upon evaluating the dataset, we identified five instances with missing values. If not addressed, these missing values could lead to inaccurate classification [25]. Therefore, a data cleaning method was implemented to remove the instances with missing values.

The Drebin dataset is imbalanced, with the number of records labeled as Benign (B) being approximately 4,000 higher than those labeled as

Malware (S). To address this issue, we balanced the dataset using the Random Under-Sampling (RUS) method, performed with Microsoft Excel. Random Under-Sampling is an algorithm that balances the target distribution by randomly eliminating instances from the majority class [26].

After balancing the data, the dataset underwent feature selection using the Information Gain and Chi-Squared (X^2) methods to make the training process more efficient. Information Gain and X^2 are reliable feature selection algorithms [27], [28]. Both are useful for identifying the features that most significantly impact model performance. Then Backward Elimination method are performed to find the best number of features.

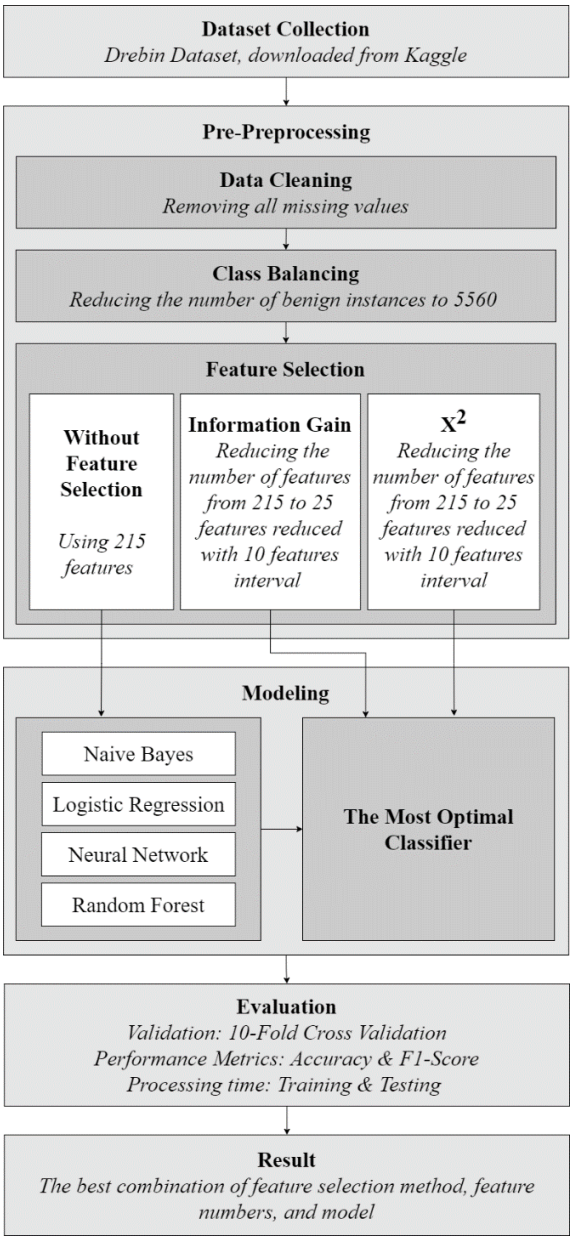

Figure 1. Research stages

Backward elimination is a variable selection method that removes features from the dataset until the optimal number of features is found. The least significant features will be removed first [29]. In this experiment, the least significant feature is removed in each iteration.

Information Gain quantifies a feature's importance for classification. If the Information Gain value increased, the feature would have greater significance [27]. The significant feature is chosen by computing the entropy value. Entropy is a measure of uncertainty that effectively summarizes feature distributions in a concise manner [30]. The equation of entropy can be seen in Equation 1.

$$Entropy(S) = \sum_i^c -P_i log_2 P_i \quad (1)$$

In this equation, C is the total number of instances that exist in the class. P_i is the total count of samples of class i. When entropy value is obtained, the calculation of the Information Gain (IG) can be performed using Equation 2.

$$IG(S,A) = Entropy(s) - Entropy(S_v) \quad (2)$$

In this equation, if A is an attribute, all the possible value of A are represented by v. The collection of all potential values for A is shown by values A. Whereas S is the total number of samples in the collection, Sv is the count of samples that have the value v. The entropy of the samples with value v is denoted by $Entropy(S_v)$.

On the other hand, the Chi-Squared feature selection measures the significance of each original feature by doing Chi-Squared statistic test [28]. By thoroughly analyzing the Chi-Squared statistics, we can identify valuable features for the specific class [31]. This method's equation can be seen in equation 3. O depicts as number of observed class while E depicts as number of expected class. A high Chi-squared test score suggests that the feature and the target class are likely dependent. Therefore, the final dataset with selected features will be used in the modeling and evaluation stage.

$$X^2 = \sum \left[\left(\frac{(O-E)^2}{(E)}\right]\right) \quad (3)$$

D. Modeling

In the modeling stage, this experiment involves four models: Random Forest, Neural Network, Naïve Bayes, and Logistic Regression. These algorithms are selected due to their popularity and effectiveness in data classification. Alternative algorithms like Support Vector Machines and AdaBoost will be explored in future research. The experiment is initially conducted without any feature selection. It is then repeated using the two feature selection methods mentioned in the previous sub-chapter. The four algorithms are implemented simultaneously during the first experiment. In the second and third experiments, only the best algorithm from the first experiment is used.

Random Forest is an algorithm method that is based on ensemble tree. The method predict by averaging many individual trees [32]. The detailed process of Random Forest is illustrated in Figure 2.

A Neural Network is a type of artificial intelligence that mimics the functioning of the human brain. It processes input by creating connections between processing elements, also known as neurons. The output is determined by the arrangement and weights of these connections [33]. Figure 3 provides detailed information about how a neural network processes data. The Neural Network used in this experiment consists of 2 layers with 100 neurons in the first layer and 40 neurons in the second layer. The algorithm uses the ReLU activation function and is configured with a maximum of 200 iterations.

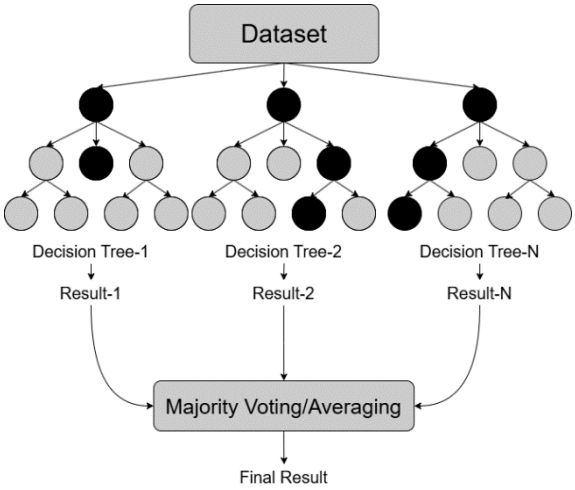

Figure 2. Block Diagram of Random Forest Algorithm

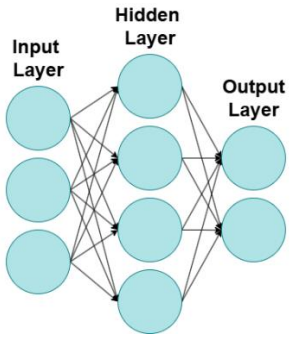

Figure 3. Block Diagram of Neural Network Algorithm

Naive Bayes is a classification method based on Bayes' theorem. This method predict using conditional probability [34]. The following Bayes formula (Equation 4) serves as the foundation for the Naïve Bayes theorem.

$$P(A|B) = \frac{p(A)p(B|A)}{p(B)} \quad (4)$$

Where,

- B : attributes
- A : class
- P(A|B) : probability of even A given B has occurred
- P(B|A) : probability of even B given A has occurred
- P(A) : probability of event A
- P(B) : probability of event B

Logistic Regression is a statistical method commonly used to predict the dichotomous dependent variable using one or more than one variables. This method can be used to derive how the independent variables can influence dependent variable [35]. The logistic equation can be found in Equation 5.

$$LR = \frac{e(\beta_0 x_0 + \beta_1 x_1 + \cdots \beta_i x_i)}{1 + e(\beta_0 x_0 + \beta_1 x_1 + \cdots \beta_i x_i)} \quad (5)$$

In this equation, the coefficients of regression in the preceding equation are denoted as β_0, β_1, β_i while the independent variables are represented by x_0, x_1,

x_i. At the conclusion of the modeling stage, the accuracy, F1-Score, and training and testing times of each experiment are compared to identify the algorithm that performs best and most efficiently in classifying the data.

E. Evaluation

The evaluation stage is conducted at the end of this experiment. Prior to evaluating the model, 10-fold cross-validation is implemented. The model's performance is assessed by calculating the accuracy, F1-Score, training time and testing time.

10-fold cross-validation is used to prevent overfitting in classification tasks and to evaluate the performance of models when applied to new data [36]. This method divides the dataset into 10 approximately equal subgroups. One subgroup is utilized for testing and the other nine subgroups are used for training for each of the ten iterations [37]. After implementing 10-fold cross-validation, a confusion matrix is used to assess the model to calculate its AUC, accuracy and F1-Score.

A confusion matrix is a visual review tool that shows the actual class results in rows, while columns indicate the predicted class results [38]. The four components of a confusion matrix are: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

True Positive (TP) means that the predicted positive result matches the actual positive result. True Negative (TN) means that the predicted negative result matches the actual negative result. False Positive (FP) occurs when the predicted positive result differs from the actual negative result. False Negative (FN) occurs when the predicted negative result differs from the actual positive result [39]. These four components are used to calculate accuracy, recall, precision, and F1-Score. Given the potentially fatal consequences of false positives and false negatives in malware detection, this research measures the AUC (area under the curve), accuracy and F1-score for each model variation.

Area Under the Curve (AUC) is a metric derived from the Receiver Operating Characteristic (ROC) curve, which summarizes the performance of a predictor. The ROC curve method illustrates how well a predictor distinguishes between different outcomes by plotting True Positive Rate (TPR) against False Positive Rate (FPR). ROC analysis employs a series of thresholds to evaluate a model's performance [40]. The equations for AUC, FPR, and TPR are shown below.

$$AUC = \sum_{i=1}^{n-1}(FPR_{i+1} - FPR_i) \times \frac{TPR_{i+1}+TPR_i}{2} xf_{osc} \quad (6)$$

Where,

- FPR_i : False Positive Rate at threshold i
- TPR_i : True Positive Rate at threshold i

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

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

The accuracy value provides a measure of how often a model correctly classifies instances [41]. The equation for calculating accuracy is shown in Equation 6. In this equation, accuracy is obtained by dividing the total number of correct predictions (TP and TN) by the total number of predictions (TP, FP, TN, FN).

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

Meanwhile, the precision and recall harmonic mean yields the F1-Score value [42]. A harmonic mean is a kind of average that is computed by multiplying the total number of values in a dataset by the reciprocal of each value in the dataset. An averaging metric that is helpful for rates and ratios is the harmonic mean. The following formula is used to compute it:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (10)$$

In this context, Recall is the ratio of true positive predictions to all actual positives, whereas precision is the ratio of true positive predicted results to all positive predictions. The formula of precision can be seen in Equation 8, whereas formula of recall is in Equation 9. The F1-Score is a matrix that represents precision and recall, making it a valuable metric for evaluating the performance of a classification model, especially when dealing with imbalanced datasets.

$$Precision = \frac{TP}{TP+FP} \quad (11)$$

$$Recall = \frac{TP}{TP+FN} \quad (12)$$

Additionally, the model's performance in term of efficiency is measured based on its training and testing time. Training time refers to the period during which a model is trained using the complete dataset [43]. Testing time refers to the duration the model takes to make predictions on the test data [44]. Therefore, we evaluate the training time of a model both with and without feature selection. The increase in processing speed is calculated to assess the efficiency of the model.

3. RESULT

In the data collection stage, the dataset was obtained from Kaggle dataset repository. The dataset contains 15036 instances and had two classes in its label, which consists of 5560 instances that labeled as 'S' (suspicious) and 9476 instances that labeled as 'B' (benign). The dataset also had 215 features that excluding the label and 5 instances with missing values. Those 5 instances with missing values were removed from the dataset.

The number of instances between malware and benign class were different, it could be said that the dataset is imbalance. To solve the issue, class balancing is carried out to balance the classes between benign and malware using Random Undersampling Method. 9476 instances with the benign label were reduced to match the number of instances with malware label after data cleaning which is 5555 instances.

In the feature selection method and modeling stage, three experiments were conducted. Backward Elimination method was used to search the optimal number of features. The first experiment aimed to compare four machine learning models to determine the best one for classifying Android malware. The results showed that the Neural Network model achieved the accuracy and F1-score at 98.6%. The training and testing times were recorded as 98.656 seconds and 2.607 seconds, respectively. Detailed performance results of the experiment are presented in Table 1. Since the Neural Network performed the best, it was used in the subsequent experiments.

The second experiment employed the Information Gain feature selection method with the Neural Network algorithm. Our findings indicate that the combination of the Neural Network and Information Gain, using 131 features, achieved an accuracy and F1-score of 98.6%. Additionally, the combination attained an AUC score of 99.7%. The confusion matrix for this combination shows 5488 true positives, 5461 true negatives, 94 false positives, and 67 false negatives. The detailed results are presented in Table 2.

Table 1. Performance of Classification Algorithms without Feature Selection

Machine Learning Methods	Accuracy (%)	F1-Score (%)	Train Time (S)	Test Time (S)
Random Forest	98.2	98.2	4.259	1.003
Neural Network	**98.6**	**98.6**	**98.656**	**2.607**
Naïve Bayes	84.9	84.8	0.523	0.065
Logistic Regression	97.6	97.6	24.853	0.993

Table 2. The experiment test result with and without Feature Selection (FS).

Number of Features	Neural Network			
	Information Gain		X^2	
	Acc (%)	F1-Score (%)	Acc (%)	F1-Score (%)
126	98.5	98.5	98.5	98.5
127	98.5	98.5	98.2	98.2
128	98.5	98.5	98.4	98.4
129	98.5	98.5	98.4	98.4
130	98.4	98.4	98.5	98.5
131	**98.6**	**98.6**	98.4	98.4
132	98.5	98.5	98.3	98.3
133	98.5	98.5	98.3	98.3
134	98.5	98.5	98.5	98.5
135	98.5	98.5	98.4	98.4
136	98.5	98.5	98.5	98.5
137	98.6	98.6	98.5	98.5
138	98.6	98.6	**98.6**	**98.6**

In the third experiment, we applied the Chi-Squared method for feature selection. The results demonstrated that the combination of the Neural Network and Chi-Squared achieved 98.6% for both accuracy and F1-score using 138 features. Additionally, this combination attained an AUC score of 99.7%. The confusion matrix for this combination indicates 5480 true positives, 5476 true negatives, 79 false positives, and 75 false negatives. The detailed results of this experiment are presented in Table 2.

Table 3. Neural Network comparison results before and after using feature selection

Number of Features	Feature Selection	Acc (%)	F1-Score (%)	Train Time (S)	Test Time (S)
All (215)	No Feature Selection	98.6	98.6	98.656	2.607
131 Features	Information Gain	98.6	98.6	81.135	1.095
Time Reduction				17.7597 %	57.9977 %

Table 4. Performance analysis results

No.	Methods	AUC (%)	Accuracy (%)	F1-Score (%)	Train Time (S)	Test Time (S)
1.	**NN + IG (131)**	**99.7**	**98.6**	**98.6**	**81.135**	**1.095**
2.	NN + X^2(138)	99.7	98.6	98.6	84.755	1.299
3.	Stacking + SBFS (8 extracted features) [16]	-	97.96	-	-	-
4.	XGBoost + LOFO (30) [17]	-	95.59	93.89	-	-

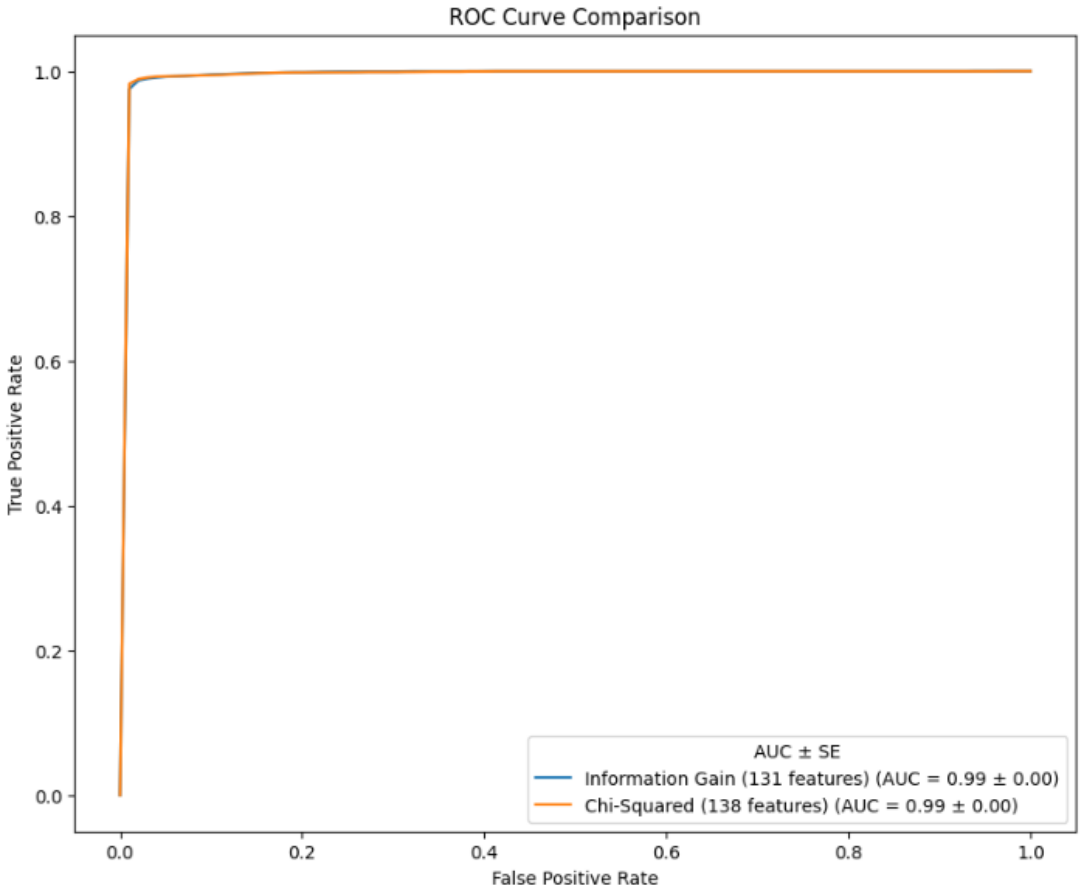

Figure 4. ROC curve comparison of Neural Networks with different feature selection methods

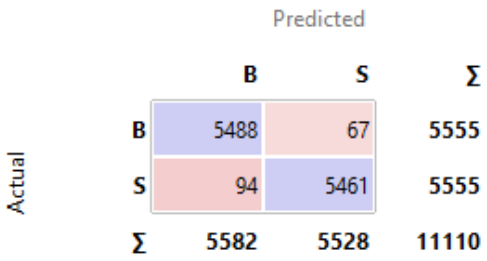

Figure 5. NN + IG confusion matrix

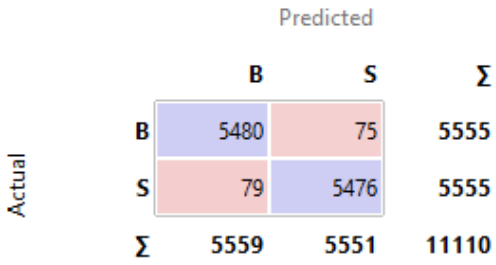

Figure 6. NN + X^2 confusion matrix

4. DISCUSSION

In the first experiment, the Neural Network algorithm achieved the best performance but had a longer training time compared to other algorithms. In the second experiment, based on Table 3, there was a notable improvement in both training and testing times while retaining the same performance after implementing Information Gain, which removed 84 of the least significant features. The training time decreased by 17.7597 %, resulting in a reduction of 17.521 seconds, and the testing time decreased by 57.9977 %, reducing the time by 1.512 seconds. This feature selection significantly enhances the speed of malware identification, which is crucial for preventing the spread of malicious software. In the third experiment, the combination of the Neural Network and Chi-Squared (X^2) achieved the same performance score as the second experiment's combination, but required 3 more features to attain the same performance.

Based on Figures 5 and 6, both confusion matrices indicate that both combinations achieved high performance, despite a slight difference in the number of false positives and false negatives. The ROC curve, shown in Figure 4, demonstrates that both combinations exhibit nearly identical results with almost perfect predictive accuracy. Additionally, the standard error for both methods is consistent, further validating the reliability of the models' performance.

As shown in Table 4 and Figure 7, the combination of the Neural Network and Information Gain provided the best performance with an accuracy score of 98.6%, outperforming three other methods. The second-best method used the same algorithm but employed a different feature selection method, the

chi-squared method (X2). In terms of performance, the second-best method achieved the same AUC ,accuracy, and F1-Score as the best method, but the training and testing times were longer.

The third and fourth methods were researched by [16] and [17], respectively. The third method consisted of a Stacking algorithm and a substring-based feature selection method. The accuracy of this method was 0.64% lower than that of the best method. The fourth method consisted of the XGBoost algorithm and the Leave-One-Feature-Out (LOFO) technique for feature selection. This method achieved an accuracy of 95.59% and an F1-Score of 93.89%. Compared to the best method, this method's performance dropped considerably by 3.05274% in accuracy and 4.77688% in F1-Score.

The combination of the Neural Network algorithm and the Information Gain feature selection method outperforms other algorithms for two reasons. First, with only the most significant features, the Neural Network can focus on the most relevant data, leading to more accurate predictions. Second, the reduction in the number of features makes the Neural Network process faster and more efficient.

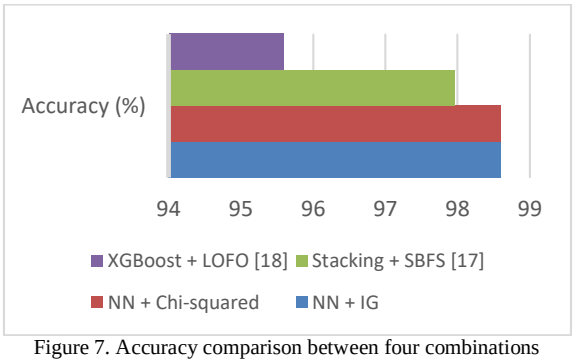

Figure 7. Accuracy comparison between four combinations

5. CONCLUSION

In recent years, polymorphic and metamorphic malware have become prevalent, rendering traditional malware classification methods based on signature matching ineffective. To effectively classify these types of malwares, Artificial Intelligence (AI) is essential to enhance malware detection capabilities. This study explores the implementation of the Neural Network algorithm and the Chi-Squared (X^2) feature selection method to detect malware using the Drebin dataset.

In addition to feature selection, two preprocessing methods, data cleaning and class balancing, were applied to the dataset to ensure accurate classification. For data cleaning, five instances with missing values were removed. Class balancing was achieved using the Random Under Sampling method.

The experiment aimed to identify the best combination of feature selection methods and classification algorithms, including Random Forest, Neural Network, Naïve Bayes, and Logistic Regression. The results indicated that the combination of Neural Network and Information Gain achieved the highest accuracy and F1-score, with a score of 98.6% on 131 features. This study demonstrates that combining Neural Networks with Information Gain significantly enhances both the accuracy and efficiency of Android malware detection. In the future, this combination of algorithms can be applied in AI-based antivirus systems to detect viruses more effectively and efficiently.

However, the combination's effectiveness relies on high-quality labeled malware data, and its accuracy is not yet at the ideal 100% for malware classification. Future work could optimize classification using other methods, such as Deep Learning.

REFERENCES

[1] S. Garg and N. Baliyan, "Comparative analysis of Android and iOS from security viewpoint," *Comput Sci Rev*, vol. 40, p. 100372, May 2021, doi: 10.1016/J.COSREV.2021.100372.

[2] T. Sharma and D. Rattan, "Malicious application detection in android — A systematic literature review," *Comput Sci Rev*, vol. 40, p. 100373, May 2021, doi: 10.1016/J.COSREV.2021.100373.

[3] J. D. Ndibwile, E. T. Luhanga, D. Fall, and Y. Kadobayashi, "A demographic perspective of smartphone security and its redesigned notifications," *Journal of Information Processing*, vol. 27, pp. 773–786, 2019, doi: 10.2197/ipsjjip.27.773.

[4] S. Garg and N. Baliyan, "Android security assessment: A review, taxonomy and research gap study," *Comput Secur*, vol. 100, p. 102087, Jan. 2021, doi: 10.1016/J.COSE.2020.102087.

[5] R. Mayrhofer, J. Vander Stoep, C. Brubaker, and N. Kralevich, "The Android Platform Security Model," *ACM Transactions on Privacy and Security*, vol. 24, no. 3, Apr. 2021, doi: 10.1145/3448609.

[6] R. Sikder, M. S. Khan, M. S. Hossain, and W. Z. Khan, "A survey on android security: Development and deployment hindrance and best practices," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 18, no. 1, pp. 485–499, Feb. 2020, doi: 10.12928/TELKOMNIKA.V18I1.13288.

[7] A. Qamar, A. Karim, and V. Chang, "Mobile malware attacks: Review, taxonomy & future directions," *Future Generation Computer Systems*, vol. 97, pp. 887–909, Aug. 2019, doi: 10.1016/J.FUTURE.2019.03.007.

[8] T. Yerlikaya and S. Sen, "Hacking Android Mobile Phone with Phishing," *Journal" Fundamental Sciences and Applications"*, vol. 27, pp. 1–7, Dec. 2021.

[9] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms," *IEEE Access*, vol. 11, pp. 121118–121141, 2023, doi: 10.1109/ACCESS.2023.3328351.

[10] L. Franceschi-Bicchierai and R. Coluccini, "Researchers find google play store apps were actually government malware," 2019.

[11] M. Caianiello, "Criminal Process faced with the Challenges of Scientific and Technological Development," *European Journal of Crime, Criminal Law and Criminal Justice*, vol. 27, no. 4, pp. 267–291, Dec. 2019, doi: 10.1163/15718174-02704001.

[12] O. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," 2020, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2019.2963724.

[13] D. Gibert, C. Mateu, J. Planes, and J. Marques-Silva, "Auditing static machine learning anti-Malware tools against metamorphic attacks," *Comput Secur*, vol. 102, Mar. 2021, doi: 10.1016/j.cose.2020.102159.

[14] D. H. Gillani, "A perspective study on Malware detection and protection, A review," 2022, doi: 10.22541/au.166308976.63086986/v1.

[15] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, vol. 112, p. 101861, Jan. 2021, doi: 10.1016/J.SYSARC.2020.101861.

[16] M. S. Rana and A. H. Sung, "Evaluation of Advanced Ensemble Learning Techniques for Android Malware Detection," *Vietnam Journal of Computer Science*, vol. 7, no. 2, pp. 145–159, May 2020, doi: 10.1142/S2196888820500086.

[17] S. A. Roseline and S. Geetha, "Android Malware Detection and Classification using LOFO Feature Selection and Tree-based Models," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jun. 2021. doi: 10.1088/1742-6596/1911/1/012031.

[18] 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," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 8, no. 1, p. 412, Jan. 2024, doi: 10.30865/mib.v8i1.6970.

[19] F. A. Rafrastara, C. Supriyanto, C. Paramita, and Y. P. Astuti, "Deteksi Malware menggunakan Metode Stacking berbasis Ensemble," *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 8, no. 1, pp. 11–6, 2023.

[20] M. A. Albahar, M. S. Elsayed, and A. Jurcut, "A Modified ResNeXt for Android Malware Identification and Classification," *Comput Intell Neurosci*, vol. 2022, 2022, doi: 10.1155/2022/8634784.

[21] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, "Data imbalance in classification: Experimental evaluation," *Inf Sci (N Y)*, vol. 513, pp. 429–441, Mar. 2020, doi: 10.1016/J.INS.2019.11.004.

[22] K. Md. Hasib *et al.*, "A Survey of Methods for Managing the Classification and Solution of Data Imbalance Problem," *Journal of Computer Science*, vol. 16, no. 11, pp. 1546–1557, Dec. 2020, doi: 10.3844/jcssp.2020.1546.1557.

[23] C. Fan, M. Chen, X. Wang, J. Wang, and B. Huang, "A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data," Mar. 29, 2021, *Frontiers Media S.A.* doi: 10.3389/fenrg.2021.652801.

[24] S. Rao, P. Poojary, J. Somaiya, and P. Mahajan, "A COMPARATIVE STUDY BETWEEN VARIOUS PREPROCESSING TECHNIQUES FOR MACHINE LEARNING," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 3, pp. 431–438, Jul. 2020.

[25] V. Çetin and O. Yıldız, "A comprehensive review on data preprocessing techniques in data analysis," *Pamukkale University Journal of Engineering Sciences*, vol. 28, no. 2, pp. 299–312, 2022, doi: 10.5505/pajes.2021.62687.

[26] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," *2020 11th International Conference on Information and Communication Systems, ICICS 2020*, pp. 243–248, Apr. 2020, doi: 10.1109/ICICS49469.2020.239556.

[27] S. Tangirala, "Evaluating the Impact of GINI Index and Information Gain on Classification using Decision Tree Classifier Algorithm*," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 2, pp.

612–619, 2020.

[28] R. Spencer, F. Thabtah, N. Abdelhamid, and M. Thompson, "Exploring feature selection and classification methods for predicting heart disease," *Digit Health*, vol. 6, 2020, doi: 10.1177/2055207620914777.

[29] M. Z. I. Chowdhury and T. C. Turin, "Variable selection strategies and its importance in clinical prediction modelling," *Fam Med Community Health*, vol. 8, no. 1, Feb. 2020, doi: 10.1136/fmch-2019-000262.

[30] K. Kurniabudi, A. Harris, and A. E. Mintaria, "Komparasi Information Gain, Gain Ratio, CFs-Bestfirst dan CFs-PSO Search Terhadap Performa Deteksi Anomali," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 5, no. 1, p. 332, Jan. 2021, doi: 10.30865/mib.v5i1.2258.

[31] S. K. Trivedi, "A study on credit scoring modeling with different feature selection and machine learning approaches," *Technol Soc*, vol. 63, Nov. 2020, doi: 10.1016/j.techsoc.2020.101413.

[32] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," *Stata Journal*, vol. 20, no. 1, pp. 3–29, Mar. 2020, doi: 10.1177/1536867X20909688.

[33] M. Islam, G. Chen, and S. Jin, "An Overview of Neural Network," *American Journal of Neural Networks and Applications*, vol. 5, no. 1, p. 7, 2019, doi: 10.11648/j.ajnna.20190501.12.

[34] S. Bhatia and J. Malhotra, "Naïve bayes classifier for predicting the novel coronavirus," in *Proceedings of the 3rd International Conference on Intelligent Communication Technologies and Virtual Mobile Networks, ICICV 2021*, Institute of Electrical and Electronics Engineers Inc., Feb. 2021, pp. 880–883. doi: 10.1109/ICICV50876.2021.9388410.

[35] N. R. Panda, J. K. Pati, J. N. Mohanty, and R. Bhuyan, "A Review on Logistic Regression in Medical Research," *National Journal of Community Medicine*, vol. 13, no. 04, pp. 265–270, Apr. 2022, doi: 10.55489/NJCM.134202222.

[36] B. G. Marcot and A. M. Hanea, "What is an optimal value of k in k-fold cross-validation in discrete Bayesian network analysis?," *Comput Stat*, vol. 36, no. 3, pp. 2009–2031, Sep. 2021, doi: 10.1007/S00180-020-00999-9/METRICS.

[37] S. M. Malakouti, M. B. Menhaj, and A. A. Suratgar, "The usage of 10-fold cross-validation and grid search to enhance ML methods performance in solar farm power generation prediction," *Clean Eng Technol*, vol. 15, Aug. 2023, doi: 10.1016/j.clet.2023.100664.

[38] J. Xu, Y. Zhang, and D. Miao, "Three-way confusion matrix for classification: A measure driven view," *Inf Sci (N Y)*, vol. 507, pp. 772–794, Jan. 2020, doi: 10.1016/J.INS.2019.06.064.

[39] M. Te Wu, "Confusion matrix and minimum cross-entropy metrics based motion recognition system in the classroom," *Sci Rep*, vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-07137-z.

[40] J. Muschelli, "ROC and AUC with a Binary Predictor: a Potentially Misleading Metric," *J Classif*, vol. 37, no. 3, pp. 696–708, Oct. 2020, doi: 10.1007/s00357-019-09345-1.

[41] F. A. Rafrastara, C. Supriyanto, C. Paramita, Y. P. Astuti, and F. Ahmed, "Performance Improvement of Random Forest Algorithm for Malware Detection on Imbalanced Dataset using Random Under-Sampling Method," *urnal Informatika: Jurnal Pengembangan IT*, vol. 8, no. 2, pp. 113–118, May 2023.

[42] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, Jan. 2020, doi: 10.1186/s12864-019-6413-7.

[43] I. D. Apostolopoulos, I. Athanasoula, M. Tzani, and P. P. Groumpos, "An Explainable Deep Learning Framework for Detecting and Localising Smoke and Fire Incidents: Evaluation of Grad-CAM++ and LIME," *Mach Learn Knowl Extr*, vol. 4, no. 4, pp. 1124–1135, Dec. 2022, doi: 10.3390/make4040057.

[44] J. Woo, S. H. Jo, G. S. Byun, B. S. Kwon, and J. H. Jeong, "Wearable airbag system for real-time bicycle rider accident recognition by orthogonal convolutional neural network (O-cnn) model," *Electronics (Switzerland)*, vol. 10, no. 12, Jun. 2021, doi: 10.3390/electronics10121423..