Comparative Analysis of Feature Selection Methods with XGBoost for Malware Detection on the Drebin Dataset

Ines Aulia Latifah[1], Fauzi Adi Rafrastara[2]*, Jevan Bintoro[3], Wildanil Ghozi[4], Waleed Mahgoub Osman[5]
Department of Informatics Engineering, Faculty of Computer Science
Universitas Dian Nuswantoro, Indonesia[1][2][3][4]
Mathematics Department, College of Education
Sudan University of Science and Technology, Sudan[5]
111202113408@mhs.dinus.ac.id[1], fauziadi@dsn.dinus.ac.id[2], 111202113433@mhs.dinus.ac.id[3],
wildanil.ghozi@dsn.dinus.ac.id[4], waleedmo@sustech.edu[5]

Abstract*—Malware, or malicious software, continues to evolve alongside increasing cyberattacks targeting individual devices and critical infrastructure. Traditional detection methods, such as signature-based detection, are often ineffective against new or polymorphic malware. Therefore, advanced malware detection methods are increasingly needed to counter these evolving threats. This study aims to compare the performance of various feature selection methods combined with the XGBoost algorithm for malware detection using the Drebin dataset, and to identify the best feature selection method to enhance accuracy and efficiency. The experimental results show that XGBoost with the Information Gain method achieves the highest accuracy of 98.7%, with faster training times than other methods like Chi-Squared and ANOVA, which each achieved an accuracy of 98.3%. Information Gain yielded the best performance in accuracy and training time efficiency, while Chi-Squared and ANOVA offered competitive but slightly lower results. This study highlights that appropriate feature selection within machine learning algorithms can significantly improve malware detection accuracy, potentially aiding in real-world cybersecurity applications to prevent harmful cyberattacks.

Keywords*— android malware detection; drebin; information gain; XGBoost; machine learning.

I. INTRODUCTION

Malware is a type of software specifically designed to damage or exploit computer systems [1]. It includes viruses, worms, ransomware, trojan horses, adware, and spyware. As cyber threats evolve, malware becomes more sophisticated, targeting both individual devices and critical infrastructure. The financial and data losses from malware attacks are significant, with global costs in 2021 estimated to be in the hundreds of billions of dollars. This underscores the urgent necessity for effective detection and prevention strategies [2].

A well-known malware attack is WannaCry, a ransomware that attacked computer systems in May 2017. This attack spread to over 150 countries, encrypting data on infected machines and demanding ransom payments in Bitcoin. This incident demonstrated the massive potential damage a single type of malware can inflict on critical infrastructure [3]. Additionally, the Joker malware targeting Android devices highlights the increasing threat to mobile platforms. Joker infiltrates official applications and secretly registers users for premium services without their consent. This attack financially harms users and violates data privacy, adding a serious threat to the expanding mobile device ecosystem [4].

Traditional signature-based malware detection methods have limitations, particularly against new or polymorphic malware [1]. Machine learning techniques, which focus on behavioral analysis and anomaly detection, have been developed to address this issue but face challenges with high-dimensional data, feature selection, and processing time. To improve efficiency and accuracy, advanced feature selection techniques and classification approaches that combine multiple methods can be applied, as they can tackle the challenges of high-dimensional data by filtering irrelevant features, reducing noise, and simplifying machine learning models. This improves the quality of data input into machine learning models, enhances malware detection, and differentiates between malicious and benign software [5].

Various studies between 2020 and 2024 have explored the combination of machine learning techniques with both static and dynamic analysis to enhance malware detection. The research by Rana and Sung [6] on Android malware detection using ensemble learning techniques like stacking, blending, and boosting achieved an accuracy of 97.96% on the DREBIN dataset. Similarly, Roseline and Geetha [7] applied tree-based ensemble methods, with XGBoost achieving the highest accuracy of 95.59%. Yin, Yuhua, et al. [8] introduced a hybrid feature selection method that improved the accuracy of intrusion detection, raising it from 82.25% to 84.24% on the UNSW-NB15 dataset. Sihwail, Omar, and Ariffin [9] employed memory forensic techniques for feature extraction, with a Support Vector Machine classifier achieving a high accuracy of 98.5%, although time complexity remains a challenge. However, all these studies indicate a gap in the optimization of feature selection and processing speed for better malware

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2294, Copyright ©2024
Submitted : October 9, 2024, Revised : November 6, 2024, Accepted : November 12, 2024, Published : November 22, 2024

detection. The contribution of this research is to conduct a comparative analysis on feature selection methods with the XGBoost algorithm on the Drebin dataset. This research seeks to identify the optimal combination of feature selection and classification methods to enhance the accuracy and efficiency of machine learning-based malware detection compared to previous studies.

This study compares various feature selection techniques with the XGBoost algorithm on the Drebin dataset, a popular Android malware dataset, focusing on machine learning-based malware detection. The goal is to identify the most effective feature selection method for enhancing malware detection accuracy and efficiency. The findings aim to provide insights into optimizing machine learning-based malware detection systems through effective feature selection.

II. RESEARCH METHODS

There are five main steps in this research, namely: data collection, pre-processing, modeling, evaluation, and result. The stages of the research are illustrated in Figure 1.

A. Hardware and Software

In computer science research, the combination of hardware and software is key to success [10]. Powerful hardware needs to be paired with suitable software for optimal performance, ensuring efficient achievement of research goals. In this study, a personal computer with the following hardware specifications was used:

- Processor: 2,3 GHz Dual-Core Intel Core i5
- RAM: 8 GB
- SSD: 128 GB
- Graphics Card: Intel Iris Plus Graphics 640 1536 MB

In this study, Orange (https://orangedatamining.com/) was used for data cleaning and evaluating model performance with the XGBoost algorithm, specifically through the "Test and Score" widget. Additionally, Python was employed for data preprocessing, where the Random Under-Sampling (RUS) method was applied to balance the dataset, enabling a comparison of the model's effectiveness on both balanced and unbalanced datasets.

B. Data Collection

The Drebin dataset, available on Kaggle, consists of a single *.csv file with data on Android applications categorized as either malware or benign. It includes 15,036 entries, with 5,560 labeled as Suspicious (S) and 9,476 as Benign (B). Each entry is described by 215 features, excluding the label column, which classifies each entry as either malware (‘S’) or benign (‘B’). The details of the Drebin dataset are summarized in Table I.

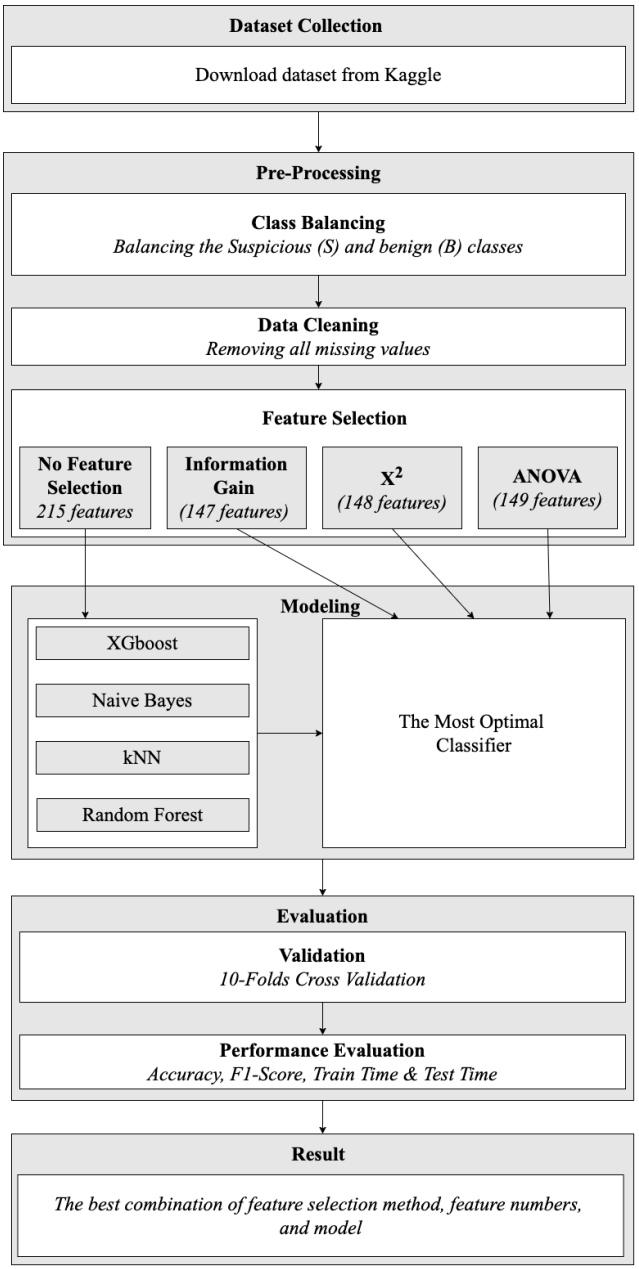

Fig. 1. Research Stages

TABLE I. DETAILS ON DREBIN DATASET

Dataset Name	Drebin Android Malware
Number of Files	1
Number of Rows	15,036
Number of Features	215
Missing Values	5

C. Pre-processing

Pre-processing is the initial stage in data processing, involving cleaning invalid or missing values, data transformation, normalization, and encoding categorical

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2294, Copyright ©2024
Submitted : October 9, 2024, Revised : November 6, 2024, Accepted : November 12, 2024, Published : November 22, 2024

variables [11]. This step improves data quality and model accuracy by providing representative, noise-free features for the machine learning model. Class balancing tackles the issue of class imbalance, where one class contains far more samples than others, which can lead to poor performance in predicting the minority class [12]. Therefore, class balancing is very important because this imbalance can make the model more inclined to predict software as safe, which risks missing malware detection. By balancing the classes, the model can better recognize and accurately predict all classes.

Before applying Random Under-Sampling (RUS), the Drebin dataset shows a significant class imbalance, with the majority class, goodware (B), dominating over the malware (S) class. RUS reduces the number of goodware samples to match the malware class, balancing the dataset and improving the model's focus on both classes. In the original dataset, there are 5,560 malware samples and 9,476 goodware samples, leading to bias. After applying RUS, both classes are balanced at 5,560 data points each, reducing the total dataset size from 15,036 to 11,120. By addressing the imbalance, RUS enhances the model's performance, particularly in identifying minority class instances, such as malware.

Data cleaning ensures data quality in analysis, with a key step being the removal of missing values [13]. Missing data can result from collection errors or incomplete inputs, and removing them helps prevent distortion in models. However, excessive removal may reduce data representation and analysis accuracy. In this study, before the removal of missing values, there were 11,120 data points. After the removal of missing values, the number of data points decreased to 11,115.

Meanwhile, Feature selection is the technique of selecting the most important variables from a dataset to be used in building a machine learning model. The main goal of feature selection is to improve model performance by reducing data dimensionality, eliminating irrelevant or redundant features, and speeding up processing time [14]. In the context of malware detection, feature selection becomes crucial because malware datasets often contain a large number of varied features, which may include irrelevant or redundant information. These less important features not only slow down the model training process but can also negatively impact the model's accuracy, thereby reducing its performance in detecting malicious threats. By reducing the number of features, making the model simpler, enhances prediction accuracy [15].

In this study, three feature selection techniques were used: Information Gain, Chi-Squared, and ANOVA. The selection was based on their effectiveness in significantly reducing data dimensions without compromising accuracy and processing time. Based on previous research, these three methods have demonstrated reliable capabilities in improving the accuracy and efficiency of models on the Drebin dataset.

Information Gain (IG) measures how much information about a target class is gained by knowing a feature's value. It assesses feature relevance by observing the reduction in entropy (uncertainty) after the feature is known [16]. The formula for Information Gain can be seen in Equation 1.

$$\text{Gain}(X,Y)=\text{Entropy}(x)-\sum_{v \in Values}\left(\frac{|X_v|}{|X|}Entropy(Xv)\right) \quad (1)$$

In this context, T represents the dataset, and X represents the feature being evaluated. The probability of feature X having the value x is denoted by $\frac{|X_v|}{|X|}$. Additionally, Entropy(x) refers to the entropy of the dataset before considering the feature X.

Chi-Squared (X^2), this method measures the independence between features and the target label. To ascertain whether categorical variables and the target variable have a significant relationship, the Chi-squared test is employed. Features with a higher X^2 value are considered more relevant. The formula for Chi-Squared can be seen in Equation 2.

$$X^2 = \sum \frac{(f_O - f_E)2}{f_E} \quad (2)$$

When f_O represents the observed frequency and f_E the expected frequency, a lower Chi-squared score occurs when the observed and expected frequencies are close to each other, indicating that the two features are likely independent [17].

The statistical technique known as Analysis of Variance (ANOVA) compares the ratio of variances between groups, such as the variances of two distinct samples. In classification analysis, ANOVA is particularly helpful when the data contains a categorical target variable and numerical input factors. This approach can be employed to evaluate the impact of specific factors on the observed outcomes and to identify if there are significant differences in the mean values across the groups being analyzed [18], [19].

D. Modeling

The next stage after performing feature selection is Modeling. Modeling in data science is used to make predictions or decisions based on historical data. Some commonly used algorithms include XGBoost, Naive Bayes, k-Nearest Neighbors (kNN), and Random Forest.

Extreme Gradient Boosting (XGBoost) is a boosting-based machine learning algorithm designed to enhance prediction accuracy by combining several simple models, such as decision trees. XGBoost is recognized for its ability to efficiently handle large and complex datasets due to its support for parallelization, which accelerates the training process. This technique makes XGBoost one of the most reliable algorithms for classification and regression tasks on large-scale data. The basic formula for XGboost can be seen in Equation 3.

$$\hat{y}i = \sum_{k=1}^{K} f_k(x_i), f_k \in \Gamma \quad (3)$$

In this case, xi refers to the input feature vector, f_k denotes the prediction score of the *k*-th tree, f_k represents the

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2294, Copyright ©2024
Submitted : October 9, 2024, Revised : November 6, 2024, Accepted : November 12, 2024, Published : November 22, 2024

number of regression trees, and Γ represents the space of all possible trees. XGBoost minimizes a regularized objective function [20].

Naive Bayes is a classification technique that assumes all features are independent of one another and is based on the Bayes theorem. Although this assumption is rarely realistic, the algorithm remains effective in many classification tasks. The bayes formula is shown in Equation 4.

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

P(A|B)is the likelihood that event A will occur, given that event B has already taken place. While P(B|A) shows the chance of event B occurring provided that event A has occurred, P(A) shows the independent probability of event A. The total probability of event B is denoted by P(B). These ideas are commonly applied in statistical analysis and are crucial for computing conditional probabilities [21].

The k-Nearest Neighbors (kNN) algorithm is a simple, non-parametric approach used for classification and regression tasks. Data points are categorized by taking into account the majority class of their k nearest neighbors, which is frequently determined by distance measurements like Euclidean distance. The kNN formula is presented in Equation 5.

$$d_i = \sqrt{\sum_{i=1}^{p}(x_{2i} - x_{1i})^2} \quad (5)$$

This formula calculates distances in multi-dimensional space to identify the nearest neighbors for kNN classification or regression [22].

One ensemble learning approach that belongs to the class of homogeneous ensemble methods is random forest. This method uses a randomly chosen subset of the available characteristics to train each decision tree. By using these random subsets, random forest aims to reduce the model's variance and improve prediction performance. This approach can be explored to test the effect of data variation on prediction performance [23].

E. Evaluation

Validation is performed using 10-Fold Cross Validation, where the data is divided into 10 parts. The model is trained on 9 parts and validated on the remaining part. This process repeats 10 times, and the average performance across all folds gives a more accurate estimate of the model's performance [24]. This method aims to reduce bias and ensure the model can generalize well on unseen data.

Following the validation process, Performance Evaluation is carried out using several metrics, namely Accuracy, F1-Score, Training Time, and Testing Time. Accuracy measures the percentage of correct predictions compared to the total predictions made, and it can be formulated as seen in Formula 6.

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

In addition to accuracy, the F1-Score is a metric that combines precision and recall, making it especially useful in cases of class imbalance. The formula for calculating the F1-Score is shown in Formula 7.

$$\text{F1-Score} = 2\ \frac{(Precision\ x\ Recall)}{Precision+Recall} \quad (7)$$

Precision measures how accurately the model identifies positive instances, while recall assesses the model's ability to capture all true positive cases. Precision aims to minimize false positives, emphasizing prediction quality, whereas Recall focuses on maximizing true positives, highlighting prediction quantity. Equation 8 and 9 illustrate these concepts.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (8)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (9)$$

Finally, Training Time and Testing Time are evaluated to assess the model's efficiency. Training Time refers to how long it takes to train the model, whereas Testing Time indicates how quickly the model can make predictions after training is complete.

III. RESULT AND DISCUSSION

The experiment used a dataset with 5,560 malware and 9,476 benign applications, initially imbalanced at a 1:1.7 ratio. Random Under-Sampling (RUS) was applied to balance the dataset to a 1:1 ratio, with 5,560 samples in each class, ensuring unbiased training. The next step involved using the backward elimination technique for feature selection, where all features are initially included, and the least significant ones are gradually removed. This method helps identify the most important features to enhance the model's performance in detecting both malware and benign applications effectively [25].

TABLE II. THE EXPERIMENT RESULTS WITHOUT FEATURE SELECTION

Algorithms	Accuracy	F1-Score	Training Test	Testing Time
XGBoost	**98.7%**	**98.7%**	7.989	0.818
Naive Bayes	85.2%	85.1%	0.778	0.099
kNN	97.8%	97.8%	0.978	0.978
Random Forest	98.2%	98.2%	0.981	0.981

In the first experiment, four classification algorithms were compared without feature selection. As shown in Table II, Naïve Bayes, despite its faster training time, shows lower accuracy at 85.2%, indicating that the feature independence assumption in Naïve Bayes is less suitable for malware detection on this Drebin dataset. The kNN algorithm achieves an accuracy of 97.8% but has a longer testing time due to the complexity of calculating distances between neighbors on large

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2294, Copyright ©2024
Submitted : October 9, 2024, Revised : November 6, 2024, Accepted : November 12, 2024, Published : November 22, 2024

datasets. Random Forest, with an accuracy of 98.2%, performs closely to XGBoost, although with a higher training time. Among the algorithms, XGBoost achieves the highest accuracy at 98.7% and demonstrates an optimal balance between accuracy and time efficiency.

Information Gain, Chi-Squared, and ANOVA were applied to XGBoost and evaluated using 10-fold cross-validation (as shown in Tables III, IV, and V). XGBoost with Information Gain using 147 features achieved the highest performance with an accuracy and F1-score of 98.7%. Chi-Squared and ANOVA, using 148 and 149 features respectively, achieved slightly lower scores of 98.3%. The highest accuracy and F1-score were obtained by XGBoost without feature selection and with Information Gain. Reducing the number of features with Information Gain proved to be the most effective and efficient approach for XGBoost in this comparison.

TABLE III. PERFORMANCE OF THE XGBOOST WITH INFORMATION GAIN (IG)

Number of Features	XGBoost + Information Gain (IG)			
	Accuracy	F1-Score	Training Time	Testing Time
146	98.5%	98.5%	5.572	0.557
147	**98.7%**	**98.7%**	**5.793**	**0.748**
148	98.7%	98.7%	5.293	0.556
149	98.6%	98.6%	5.346	0.659
150	98.6%	98.6%	5.302	0.66

TABLE IV. PERFORMANCE OF THE XGBOOST WITH CHI-SQUARED (X2)

Number of Features	XGBoost + X2			
	Accuracy	F1-Score	Training Time	Testing Time
146	98.1%	98.1%	5.199	0.539
147	98.1%	98.1%	5.077	0.545
148	**98.3%**	**98.3%**	**5.068**	**0.605**
149	98.3%	98.3%	5.246	0.566
150	98.3%	98.3%	5.559	0.549

TABLE V. PERFORMANCE OF THE XGBOOST WITH ANOVA

Number of Features	XGBoost + ANOVA			
	Accuracy	F1-Score	Training Time	Testing Time
146	98.1%	98.1%	5.239	0.557
147	98.1%	98.1%	5.102	0.641
148	98.2%	98.2%	5.715	0.702
149	**98.3%**	**98.3%**	**5.939**	**0.665**
150	98.3%	98.3%	5.577	0.609

A small difference in accuracy and F1-score between the Information Gain method (98.7%) and other methods like Chi-Squared and ANOVA (98.3%) appears significant. Even a slight increase in accuracy can have a substantial impact on threat identification. For example, a 0.4% difference in accuracy may seem minor, but it indicates that a portion of previously undetected data has been correctly classified.

After testing using a confusion matrix, it was shown that the XGBoost algorithm correctly predicted 5,465 data points in category P (Positive) and 5,439 data points in category N (Negative). However, there were misclassifications where 95 of P data were incorrectly classified as N, and 116 of N data were incorrectly classified as P, with a total of 11,115 data points tested (see Table VI). Error predictions raise serious concerns about the harmful impact of undetected malware. Nevertheless, XGBoost's performance remained the best compared to the other three algorithms.

TABLE VI. CONFUSION MATRIX OF XGBOOST ALGORITHM

		Predicted		
		P	N	Σ
Actual	P	546,560	95	**55**
	N	116	5,439	**5555**
	Σ	**5581**	**5534**	**11115**

Table VII presents the performance of the XGBoost algorithm with various feature selection methods, aimed at optimizing accuracy and computational efficiency. Without feature selection, XGBoost achieves an accuracy and F1 score of 98.7%, with a training time of 7.989 seconds. Using the Information Gain method, which selects only 147 features, the model maintains the same accuracy and F1 score of 98.7%, but significantly reduces training time to 5.068 seconds. This

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2294, Copyright ©2024
Submitted : October 9, 2024, Revised : November 6, 2024, Accepted : November 12, 2024, Published : November 22, 2024

highlights the advantage of Information Gain in improving computational efficiency without compromising model performance. In contrast, the Chi-Squared (X^2) and ANOVA methods, which select 148 and 149 features respectively, show a slight decrease in accuracy and F1 score to 98.3%, with training times of 5.068 and 5.939 seconds. In terms of efficiency and performance, Information Gain appears superior, as it optimally reduces training time without lowering accuracy or F1.

Table VII. Performance Analysis Result

Algorithme	**SF Methods**	**Number of Features**	**Accuracy**	**F1-Score**	**Training Time**	**Testing Time**
XGBoost	-	-	98.7%	98.7%	7.989	0.818
XGBoost (147)	Information Gain	147	98.7%	98.7%	5.068	0.748
XGBoost (148)	X2	148	98.3%	98.3%	5.068	0.605
XGBoost (149)	ANOVA	149	98.3%	98.3%	5.939	0.665
Stacking (DT, SVM, LR) [6]	SBFS	8	97.96%	97.0%	-	-

Additionally, comparison results from other studies show that the Stacking method (DT, SVM, LR) with SBFS feature selection achieves an accuracy of 97.96% with only 8 features, though with a slightly lower F1 score of 97%. Another approach, using XGBoost with LOFO, selects 30 features and achieves an accuracy of 95.6% and an F1 score of 93.9%. Unfortunately, information about the training and testing times for these models is unavailable. The illustration of comparison among algorithms are illustrated in Figure 2.

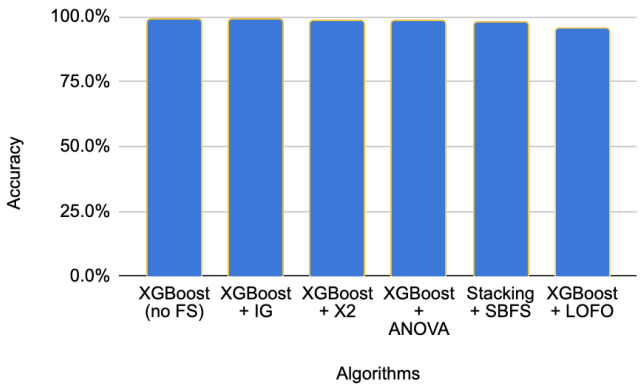

Fig. 2. Accuracy Comparison between six combinations

In the experiment, XGBoost achieved the highest performance in malware detection with an accuracy and F1-score of 98.7%. This performance was maintained using Information Gain with 147 features, which also resulted in faster processing times. In addition, there is the issue of false positives, where benign (safe) applications are mistakenly classified as malware, which can disrupt users and lower trust in the detection system. Conversely, false negatives, where malicious (malware) applications are mistakenly classified as benign, pose potential security threats to the system. To reduce these errors, the feature selection technique used must be capable of choosing the most relevant features that distinguish malware from safe applications. Due to the high risk of false positives and false negatives in malware detection, it is not recommended to sacrifice accuracy and F1 scores for processing speed. Therefore, using more features, such as the proposed 147. Recommended over the fewer features suggested by other researchers to minimize false classifications.

IV. CONCLUSION

This research highlighted the effectiveness of machine learning, especially the XGBoost algorithm, for malware detection using the Drebin dataset. The study involved data collection, pre-processing with Random Under-Sampling (RUS) to balance the dataset, and applying feature selection methods like Information Gain, Chi-Squared, and ANOVA. XGBoost combined with Information Gain and 147 features achieved the highest accuracy and F1-score of 98.7% and faster processing speeds, outperforming other methods. The study underscores the importance of proper feature selection and balanced data to optimize machine learning performance in malware detection.

REFERENCES

[1] 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," *J. Inform. J. Pengemb. IT*, vol. 8, no. 2, pp. 113–118, May 2023, doi: 10.30591/jpit.v8i2.5207.

[2] Dr. Y. Perwej, S. Qamar Abbas, J. Pratap Dixit, Dr. N. Akhtar, and A. Kumar Jaiswal, "A Systematic Literature Review on the Cyber Security," *Int. J. Sci. Res. Manag.*, vol. 9, no. 12, pp. 669–710, Dec. 2021, doi: 10.18535/ijsrm/v9i12.ec04.

[3] C. Beaman, A. Barkworth, T. D. Akande, S. Hakak, and M. K. Khan, "Ransomware: Recent advances, analysis, challenges and future research directions," *Comput. Secur.*, vol. 111, p. 102490, Dec. 2021, doi: 10.1016/j.cose.2021.102490.

[4] L. Wang *et al.*, "MalRadar: Demystifying Android Malware in the New Era," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 2, pp. 1–27, May 2022, doi: 10.1145/3530906.

[5] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware Detection Issues, Challenges, and Future Directions: A Survey," *Appl. Sci.*, vol. 12, no. 17, p. 8482, Aug. 2022, doi: 10.3390/app12178482.

[6] Md. S. Rana and A. H. Sung, "Evaluation of Advanced Ensemble Learning Techniques for Android Malware Detection," *Vietnam J. Comput. Sci.*, vol. 07, no. 02, pp. 145–159, May 2020, doi: 10.1142/S2196888820500086.

[7] S. A. Roseline and S. Geetha, "Android Malware Detection and Classification using LOFO Feature Selection and Tree-based Models," *J. Phys. Conf. Ser.*, vol. 1911, no. 1, p. 012031, May 2021, doi:

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2294, Copyright ©2024
Submitted : October 9, 2024, Revised : November 6, 2024, Accepted : November 12, 2024, Published : November 22, 2024

10.1088/1742-6596/1911/1/012031.
[8] Y. Yin *et al.*, "IGRF-RFE: a hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset," *J. Big Data*, vol. 10, no. 1, p. 15, Feb. 2023, doi: 10.1186/s40537-023-00694-8.
[9] R. Sihwail, K. Omar, and K. Akram Zainol Ariffin, "An Effective Memory Analysis for Malware Detection and Classification," *Comput. Mater. Contin.*, vol. 67, no. 2, pp. 2301–2320, 2021, doi: 10.32604/cmc.2021.014510.
[10] A. G. Baydin *et al.*, "Toward Machine Learning Optimization of Experimental Design," *Nucl. Phys. News*, vol. 31, no. 1, pp. 25–28, Jan. 2021, doi: 10.1080/10619127.2021.1881364.
[11] V. Çetin and O. Yıldız, "A comprehensive review on data preprocessing techniques in data analysis," *Pamukkale Univ. J. Eng. Sci.*, vol. 28, no. 2, pp. 299–312, 2022, doi: 10.5505/pajes.2021.62687.
[12] K. Hwang, W. Kang, and Y. Jung, "Application of the class-balancing strategies with bootstrapping for fitting logistic regression models for post-fire tree mortality in South Korea," *Environ. Ecol. Stat.*, vol. 30, no. 3, pp. 575–598, Sep. 2023, doi: 10.1007/s10651-023-00573-8.
[13] Z. Abedjan *et al.*, "Detecting data errors: where are we and what needs to be done?," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 993–1004, Aug. 2016, doi: 10.14778/2994509.2994518.
[14] STMIK Lombok, S. Saikin, S. Fadli, STMIK Lombok, M. Ashari, and STMIK Lombok, "Optimization of Support Vector Machine Method Using Feature Selection to Improve Classification Results," *JISAJurnal Inform. Dan Sains*, vol. 4, no. 1, pp. 22–27, Jun. 2021, doi: 10.31326/jisa.v4i1.881.
[15] M. Al-Omari and Q. A. Al-Haija, "Towards Robust IDSs: An Integrated Approach of Hybrid Feature Selection and Machine Learning," *J. Internet Serv. Inf. Secur.*, vol. 14, no. 3, pp. 47–67, Aug. 2024, doi: 10.58346/JISIS.2024.I2.004.
[16] S. Tangirala, "Evaluating the Impact of GINI Index and Information Gain on Classification using Decision Tree Classifier Algorithm*," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 2, 2020, doi: 10.14569/IJACSA.2020.0110277.
[17] N. Wijaya, "Evaluation of Naïve Bayes and Chi-Square performance for Classification of Occupancy House," *Int. J. Inform. Comput.*, vol. 1, no. 2, p. 46, Feb. 2020, doi: 10.35842/ijicom.v1i2.20.
[18] K. Dissanayake and M. G. Md Johar, "Comparative Study on Heart Disease Prediction Using Feature Selection Techniques on Classification Algorithms," *Appl. Comput. Intell. Soft Comput.*, vol. 2021, pp. 1–17, Nov. 2021, doi: 10.1155/2021/5581806.
[19] U. Moorthy and U. D. Gandhi, "A novel optimal feature selection technique for medical data classification using ANOVA based whale optimization," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 3, pp. 3527–3538, Mar. 2021, doi: 10.1007/s12652-020-02592-w.
[20] S. Chehreh Chelgani, H. Nasiri, and A. Tohry, "Modeling of particle sizes for industrial HPGR products by a unique explainable AI tool- A 'Conscious Lab' development," *Adv. Powder Technol.*, vol. 32, no. 11, pp. 4141–4148, Nov. 2021, doi: 10.1016/j.apt.2021.09.020.
[21] O. Uludağ and A. Gürsoy, "On the Financial Situation Analysis with KNN and Naive Bayes Classification Algorithms," *Iğdır Üniversitesi Fen Bilim. Enstitüsü Derg.*, vol. 10, no. 4, pp. 2881–2888, Dec. 2020, doi: 10.21597/jist.703004.
[22] Z. Lubis, P. Sihombing, and H. Mawengkang, "Optimization of K Value at the K-NN algorithm in clustering using the expectation maximization algorithm," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 725, no. 1, p. 012133, Jan. 2020, doi: 10.1088/1757-899X/725/1/012133.
[23] "Prediction of Heart Disease Using Feature Selection and Random Forest Ensemble Method," *Int. J. Pharm. Res.*, vol. 12, no. 04, Jun. 2020, doi: 10.31838/ijpr/2020.12.04.013.
[24] F. Mohr and J. N. Van Rijn, "Fast and Informative Model Selection Using Learning Curve Cross-Validation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 9669–9680, Aug. 2023, doi: 10.1109/TPAMI.2023.3251957.
[25] S. Farahdiba, D. Kartini, R. A. Nugroho, R. Herteno, and T. H. Saragih, "Backward Elimination for Feature Selection on Breast Cancer Classification Using Logistic Regression and Support Vector Machine Algorithms," *IJCCS Indones. J. Comput. Cybern. Syst.*, vol. 17, no. 4, p. 429, Oct. 2023, doi: 10.22146/ijccs.88926.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v13i3.2294, Copyright ©2024
Submitted : October 9, 2024, Revised : November 6, 2024, Accepted : November 12, 2024, Published : November 22, 2024