Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Improving Machine-Learning Malware Detection Through IQR-Based Feature Reduction

Nurcahyo Fajar Setyanto [1)*], Rina Pramitasari [2)], Jeki Kuswanto [3)]
[1,2,3)] Prodi Teknik Komputer Fakultas Ilmu Komputer Universitas Amikom Yogyakarta, INDONESIA
[1)] nurcahyofajar@students.amikom.ac.id , [2)] rina.pramitasari@amikom.ac.id , [3)] jeki@amikom.ac.id

Submitted :Nov 27, 2025 | **Accepted** : Dec 5, 2025 | **Published** : Jan 02, 2026

Abstract: Malware detection is a significant challenge in cybersecurity due to the complex and evolving nature of threats. This study evaluates the effectiveness of machine learning algorithms, specifically XGBoost and LightGBM, in detecting malware. The approach includes data cleaning, normalization, feature selection, and the use of the Interquartile Range (IQR) technique to select relevant features. The initial dataset contained 21,752 files, evenly split between malicious and benign files. After data cleaning, the number of samples decreased to 19,256 files, with numerous features that were reduced after applying IQR. Results show that XGBoost outperforms other algorithms, achieving 99.20% accuracy, an improvement over the 98.99% accuracy without IQR. The IQR technique enhances data quality by filtering out features with significant differences between malware and benign files, improving model performance. Additionally, reducing the feature set helps prevent overfitting and strengthens the model's generalization ability. The study concludes that machine learning, particularly with algorithms like XGBoost and LightGBM, can effectively improve malware detection. By using IQR in feature selection, model performance is enhanced, leading to reduced false positives and increased detection efficiency. The research highlights the importance of feature selection techniques like IQR in boosting the predictive power of machine learning models, making them more efficient in identifying malware. Future work will explore additional feature selection methods to further improve malware detection accuracy.

Keywords: Malware, Machine Learning, Interquartile Range, XGBoost, Malware Detection

INTRODUCTION

Given the constantly changing nature of threats, malware detection is an essential component of cybersecurity maintenance. Malware has grown to be a serious concern that might jeopardize computer systems' confidentiality and integrity and result in large losses for both consumers and businesses. Conventional detection techniques, which depend on malware signatures, have become less and less successful against novel, before undiscovered variations (Jadhav et al., 2024). In this context, machine learning (ML) offers a potential solution, as it enables systems to identify malware by analyzing behavioral patterns and file characteristics that are otherwise undetectable by traditional methods. Various ML algorithms, such as Random Forest, Support Vector Machine (SVM), and XGBoost, have been tested for malware detection, and the results demonstrate their exceptional ability to recognize evolving threats (Patil et al., 2023).

With the ongoing creation of increasingly complex and challenging-to-detect malware varieties, the use of machine learning in malware detection is becoming more and more apparent. Recent research has shown that ML techniques, such as XGBoost and other ensemble algorithms, not only improve detection accuracy but also enable more efficient real-time detection compared to traditional methods (Malik, 2021; Varshney et al., 2023). The dataset we used included 26 malware families divided into four main categories, with a total of 21,752 samples, balanced between malicious files (10,876 samples) and benign files (10,876 samples) (Hussain, 2024)This demonstrates that the use of optimized XGBoost techniques can improve malware detection effectiveness on a wider range of platforms, accelerate threat response, and strengthen overall security systems. In order to combat the ever-increasing complexity of cyberthreats, it is crucial and essential to include machine learning into malware detection (Hilabi & Abu-Khadrah, 2024; Kumar et al., 2024).

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Feature selection for malware detection has been extensively researched, but most studies focus on feature selection using traditional statistical techniques or simpler feature importance-based methods, such as Random Forest or XGBoost. The specific IQR approach to tabular malware datasets has been rarely explored, even though IQR has been proven effective in identifying outliers and improving data quality in various types of datasets. This approach has not been thoroughly tested on malware datasets, which have different characteristics and require more robust feature selection techniques. In addition, the IQR method for feature selection has not been tested for multi-model benchmarking, where various machine learning models are tested with and without the application of IQR to compare the resulting performance improvements. This indicates a significant research gap in the application of IQR to malware datasets to improve detection effectiveness.

This study offers a novel contribution by implementing IQR as an outlier-based feature assessment mechanism in a large malware dataset consisting of 21,752 datasets. This technique allows for noise and irrelevant features reduction and improves the model's accuracy in detecting malware, providing a more effective approach than other feature selection techniques. Supported by the fact that the robust nature of IQR effectively stabilizes the feature distribution and prevents outliers from distorting the classification decision boundary, which directly improves the model's generalization. The improved model accuracy and efficiency (especially in minimizing False Negatives through improved Recall and F1-score) have great practical significance as they provide a computationally efficient, ready-to-implement solution for security analysts, while also reinforcing the theoretical significance of the need for robust feature engineering techniques in cybersecurity. However, this study acknowledges the methodological limitations that IQR, as a univariate method, fails to detect multivariate anomalies, and in highly skewed datasets, the use of a 1.5×IQR threshold risks swamping or masking, highlighting the critical trade-off between noise removal and retention of crucial anomaly signals. In addition, this study also conducts a comprehensive evaluation of seven machine learning models, both before and after training.

LITERATURE REVIEW

In the article Ensemble-Based Machine Learning Algorithm for Intelligent Network Security Threat Detection the author presents the algorithm he calls ‘IntelliGuard Threat Detector’ which is capable of identifying a network security breach and categorizing it using different classes of advanced machine learning (Fan & Chunsheng, 2025). This algorithm improves the network breach detection and the robustness of the network breach detection using the techniques of integrated robust scaler normalization, Composite Rank Ensemble (CORE) feature selection boosted with a trio of classifiers (TrioBoost). The results of the various tests achieved were 94 % accurate, 92 % precision, 95 % recall, 94 % F1-score, and a 93 % geometric mean and it surpassed other simple algorithms such as Decision Stump, Logistic Regression and SVM. The study also presents the IQR (Interquartile Range) as the 87% F1-score and 82% accuracy were only surpassed by Z-Score. The IQR approach calculates the first (Q1) and third quartiles (Q3) and defines the upper and lower bounds, and identifies as anomalies the data that falls outside the bounds. This reason presents the IQR approach as strong and effective for identifying anomalies that result from more intricate types of attacks.

Anomaly Detection Using Z-Score: Anomaly Detection Using Z-Score, Modified ZScore, IQR, Exponential, Moving Average, and Exponentially Weighted Moving Average, the author of the present research proposes an endpoint real-time anomaly detection framework built on the integration of the statistical methods Z, proposed Exponential, Modified Z, and Moving Average, and Exponentially Weighted Moving Average (EWMA). These methods are anomalies identified and telemetry data are parameters such as CPU Axis, network entries, disk transactions, and others. These are the anomalies of behaviorb (Azerbaijan Technical University, Baku, Azerbaijan & Asgarov, 2025). The experiments as a whole yielded positive, (0.91), was the Z-Score of the method best, (0.93), performance an exceptional. On the IQR anomaly of the undersigned, detection of the methods' effectiveness, as a result, was of the average, was as a result, as a result, was of the average, was of detection effective, as a result, the average of detection was such second this of was was of detection effective, as a result of detection lower result second than Z-Score. IQR method defines boundaries by way of the lower and upper by (Q1) and quartile (Q3) to detect anomalies by marking data to find dispose, having result of data disappearance from detection.

The author of Machine Learning in Cybersecurity (El Hajj Hassan & Duong-Trung, 2024): Advanced Detection and Classification Techniques for Network Traffic Environments studies the application of machine learning techniques corresponding to the enhancement of network security through the anomaly detection and the classification of network traffic. Leveraging techniques such as Logistic Regression, Decision Trees, Random Forest and XGBoost, the author thoroughly researches large-scale network traffic datasets, striving to identify malevolent activities and actively improve the performance of the network. This piece of research exemplifies the importance of diverse datasets, and the dataset provided by Kaggle, offers high relevance and features pertaining to the identification of patterns present in network traffic. The author illustrates the performance of the Random Forest and XGBoost as outsanding and in the upper echelon in the research under scrutiny, Random Forest being the best with 94.04% and 0.9845 and XGBoost with a good performance and having 0.9795 with 92.75%. Of the Isolation Forest and K-Means utilized in the segmentation of the anomaly detection the author attributes and

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

espouses the K-Means and IQR to possess valuable attributes in the identification of network deviations and K-Means of IQR does not rigitly conform to the notion of recessive. The author attributes and espouses the K-Means and IQR to possess valuable attributes in the identification of network deviations; and K-Means of IQR does not rigitly conform to the notion of recessive. The Random Forest and XGBoost machine learning models, with associated valuable attributes, are distinguished to achieved a high performance in the identification of network traffic and provide one with the attributes of enhanced security

Previous research has shown that machine learning has quite good performance in malware detection. One study used a dataset from Kaggle containing 15,036 applications (5,560 malware and 9,476 benign) (Nikam & Deshmuh, 2022). In the preprocessing process, missing data was addressed using the SimpleImputer technique from the ScikitLearn library, and relevant features were selected using the feature importance technique. Ten machine learning algorithms were tested, including XGBoost, Random Forest, and SVM. The results showed that XGBoost performed the best, with an accuracy of 98.72%, an AUC of 0.9900, an FPR of 0.008%, and an FNR of 0.015%. The confusion matrix used showed that XGBoost had highly accurate predictions with a low error rate in classifying applications as malware or benign.

Research (Poudyal et al., 2018) discusses the development of a framework for ransomware detection using reverse engineering and machine learning techniques. In the preprocessing stage, binary files are analyzed using a disassembler and a PE parser to convert the binary code into assembly instructions and DLLs. The resulting features are used to train various machine learning algorithms. Based on the experimental results, ransomware detection accuracy varies between 76% and 97%, with the highest accuracy obtained using Random Forest and AdaboostM1. The Random Forest model showed good performance in identifying ransomware with a low false positive rate (FPR), achieving an accuracy of 97.95%.

A study (Chen & Zhang, 2024) examined the use of machine learning algorithms to detect ransomware on the InterPlanetary File System (IPFS) network. The preprocessing method used involved collecting IPFS network traffic data, including both legitimate and suspicious activity. Features identified for ransomware detection included file access patterns, network request frequency, and file size changes. Logic Regression, Decision Trees, Random Forest, Gradient Boosting Machines (GBM), and Convolutional Neural Networks (CNN) were among the machine learning models that were tried. With an AUC-ROC of 0.95 and the greatest accuracy of 92.3%, the CNN model demonstrated outstanding detection skills even in hostile situations, according to the assessment data.

Another study (Akhtar & Feng, 2022) examined the use of various machine learning algorithms to detect malware. To increase accuracy and lower the chance of overfitting, data was retrieved and pertinent characteristics were chosen during the preprocessing step, with a focus on Decision Trees (DT), Convolutional Neural Networks (CNN), and Support Vector Machines (SVM). The evaluation results showed that the DT model had the highest accuracy of 99%, followed by CNN (98.76%) and SVM (96.41%). The confusion matrix analysis showed that DT had the lowest false positive rate (FPR) of 2.01%, indicating the model's effectiveness in detecting malware with a very low error rate.

A study (Aditya & Dash, 2024) discussed the use of LightGBM (Light Gradient Boosting Machine) to detect malware backdoors in SCADA networks and compared it with several other machine learning algorithms, such as Random Forest, SVM, XGBoost, and Neural Networks. In the preprocessing stage, the data used consisted of network traffic logs, system call records, and system file interactions taken from the Kaggle dataset. Feature extraction from the raw data, data cleaning, and normalization using the Z-score and Min-Max scaling techniques were all part of the preparation procedure. In order to enhance model performance, feature selection was then done using RFE and PCA. LightGBM got the greatest accuracy of 94.5%, precision of 92%, recall of 96%, and F1 score of 94%, according to the evaluation findings, demonstrating its capacity to reliably identify malware backdoors with a low false positive rate (FPR). This model is a great option for real-time virus detection in SCADA networks since it also had the quickest training time when compared to other models.

In previous studies, various machine learning algorithms have been applied to detect network security threats, with approaches that combine advanced techniques such as normalization, feature selection, and complex classification models. One of the methods introduced is Interquartile Range (IQR), which has proven effective in detecting anomalies by defining lower and upper limits based on Q1 and Q3. Although IQR shows fairly good results, my research offers advantages in terms of accuracy, with a model that successfully overcomes the limitations of IQR and achieves a higher level of accuracy (Fan & Chunsheng, 2025) (Azerbaijan Technical University, Baku, Azerbaijan & Asgarov, 2025) (El Hajj Hassan & Duong-Trung, 2024). This shows that the approach I use is able to detect threats more accurately and efficiently, even in more complex scenarios, making this research superior to previous studies.

METHOD

In the data flow shown in The process illustrated in figure one starts with Data Collection, then goes through the pre-processing pipeline in to analyze the data. The pre-processing pipeline is composed of a series of essential steps including Data Preparation, Data Cleansing, and Label Encoding to transform data into the right form. Next

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

is the Normalization step where the scales of the variables are made uniform, then Feature Engineering, then the data is split into Training and Testing sets. To further enhance the data quality, the processes of Binning and Feature Selection are also applied.

The Interquartile Range (IQR) is used in the gene selection process to identify outliers in the dataset. After data processing, classification is performed using the selected algorithm, and the results are evaluated to produce final conclusions. This process produces data ready for further analysis or data-driven decision-making.

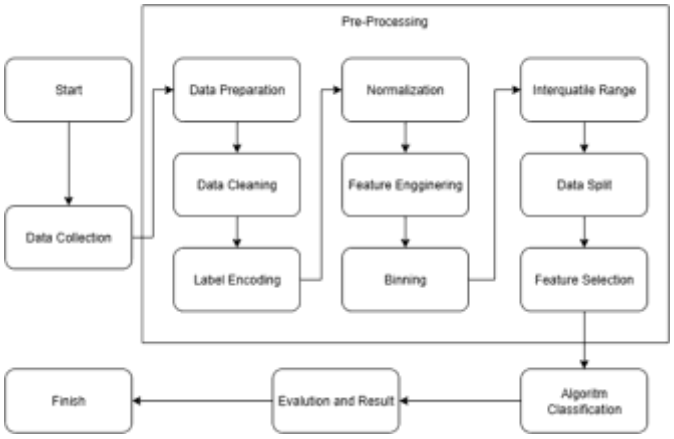


Fig. 1 Research Process

Dataset

The Ransomware Dataset 2024 is a dataset focused on malware analysis, specifically for ransomware detection and classification (Hussain, 2024). The dataset includes a total of 21,752 dataset, with a balanced split between 10,876 malicious and 10,876 benign files. The dataset focuses on 11 infamous ransomware families, like Cerber, DarkSide, WannaCry, and REvil, as well as 26 different malware families and their massive cyberattacks. The dataset has 4 main categorizations, although how it is categorized is not explained. The balanced split of malicious and benign files makes it highly useful for training machine learning models, as it aids in malware detection, ransomware mitigation, and even malware family classification. The dataset also supports a variety of applications in cybersecurity, including forensic analysis, training malware detection models, and research into developing countermeasures against the ever-evolving ransomware threat.

The original number of samples was 21,752 dataset, consisting of both benign and malicious files. After data cleaning, the number of samples decreased to 19,256 dataset. This reduction occurred due to the removal of incomplete, irrelevant, or corrupted files that did not meet the quality standards necessary for machine learning model training. These files may have contained missing or inconsistent feature data, making them unsuitable for reliable analysis. The dataset includes a variety of features, such as API calls, PE sections (Portable Executable sections), entropy (which measures the randomness or disorder of a file), and other metadata related to the executable file structure. These features are essential for distinguishing between benign and malicious files and are used to train machine learning models to recognize malware patterns and behaviors.

Pre-Processing

a. Data Cleaning
   The act of finding and fixing mistakes or inconsistencies in a dataset to enhance its quality and get it ready for analysis or the use of machine learning algorithms is known as "data cleaning." (Amato & Di Lecce, 2023). It involves handling missing data, correcting errors, eliminating duplication, and ensuring data is in a consistent format. Furthermore, data cleaning can include normalization or standardization to adjust the value scale, removing irrelevant features, and transforming data types if necessary. The goal is to ensure the dataset used is accurate, complete, and reliable, thereby improving the performance of the analysis or model training performed.
b. Label Endcoding
   Label encoding is a technique used in machine learning to convert categorical variables into a format that can be understood by machine learning algorithms. Specifically, this technique converts each category in a categorical feature into a unique numerical label. This process is important when dealing with categorical data in supervised learning, as most machine learning algorithms require numerical inputs. Label encoding assigns an integer number to each category, which is then used in the model for training. One of the simplest methods in label encoding is to assign labels based on the alphabetical order of the categories, but more sophisticated strategies may be used for specific purposes. The importance of label encoding in preprocessing is demonstrated in its ability to handle categorical features.

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

c. Normalization
Normalization is a process used to transform data into a uniform scale, so that each feature has a comparable contribution in a machine learning model (Wahid et al., 2018). In the context of EMG signals, normalization is done by taking the EMG feature value and dividing it by the area under the root mean square curve (AUC-RMS) of the EMG signal that has been computed beforehand. This is done to minimize the variability that is caused by difference in contraction levels of muscles of the subjects so that the machine learning techniques to model the hand movement patterns more accurately and consistently.

d. Feature Engineering
(Verdonck et al., 2024) define feature engineering as a process of preparing the data in a format consisting of features that can be understood by the machine learning algorithms. For this Walker (2022) states the feature engineering process consists of two parts. The first part involves taking raw data of any form and converting them into features that can be usable by the model. The second part of the process revolves around revising created features into new and more relevant features that fosters improvement of the model in terms of accuracy and interpretability. Using a concept and feature engineering process techniques the data can be represented in a more optimized way. The essence and characteristics of feature engineering techniques drawn the techniques importance since the features created tend to be the property of data that dictate the output of the machine learning models.

e. Binning
Binning is a data processing technique used to group continuous values into specific intervals, or "bins," (Arganda et al., 2023). The goal of binning is to reduce subtle variations in data to make it easier to analyze or apply to machine learning models. Binning is often used in the context of statistical analysis or in the creation of histograms to estimate probability distributions. In the context of machine learning, binning can be used to discretize the output of a classifier or to group data in a more structured way, thereby reducing noise and improving model performance. However, while binning methods can improve performance, there are also challenges in choosing the optimal bin size, which can affect model quality.

f. Interquartile Range
Interquartile range (IQR) is a statistical method used to identify variability or dispersion in a dataset by measuring the range between the first quartile (Q1) and the third quartile (Q3) (Khan et al., 2019). In the context of data analysis, the IQR can be used to select features that have significant differences between classes, such as in gene selection in genomics experiments. This is done by calculating the IQR for each feature and using this value to determine whether it is sufficiently far from the distribution of the other classes. Features with a larger IQR range, or whose expression values do not overlap between classes, are considered more informative and are selected as primary features. This technique guarantees that just the most pertinent characteristics are utilized in the analysis or prediction model while also assisting in the reduction of dataset redundancy. We first identify the first quartile (Q1), which is the value below which 25% of the data points fall, and the third quartile (Q3), which is the value below which 75% of the data points fall, in order to calculate the IQR for each feature. Next, Q1 is subtracted from Q3 to calculate the IQR (IQR = Q3 - Q1). This range represents the middle 50% of the data and is used to measure the dispersion. Features with a very small IQR, which indicates low variance or high noise, are discarded as they are unlikely to contribute useful information for distinguishing between classes. After applying this filtering process, the number of features in the dataset decreases, as features with minimal variability are removed, leaving only those with significant differences that can improve the prediction model's accuracy. Before filtering, the dataset contained X features, and after applying the IQR-based filtering, it was reduced to Y features. Here, X refers to the total number of features in the dataset before the filtering process, while Y represents the number of features remaining after the IQR filtering. Features with low variance or high noise (indicating they don't distinguish well between classes) are removed, reducing the feature set and leaving only those that are more informative and relevant for the predictive model.

g. Data Split
The process of dividing a dataset into two parts is known as data splitting. One part is used for model training (training set) and the other is used for model testing (Nguyen et al., 2021). This split is crucial for objectively assessing model performance because it ensures that the model remembers the training data (overfitting) and can generalize to new data.

h. Feature Selection
Feature selection is done to choose one subset of features from a set of featur3s of a particular dataset in order to improve the speed and performance of a machine learning model (Cai et al., 2018) Theaim of this process is to remove irrelevent and redundant features, also learning improves, and so the computational complexity is improved. For this, feature selection is done using a variety of methods such as statistical, information theor, and machine learning methods, more specifically filter, wrapper and embedded methods. In order to provide the features that explain the correlation, dependency and Euclidean distance, among enough features, the

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

selected features get evaluated for the model. The dimensionality of the data increases so that the selected model is a real model. The features selected for such a model is so that the model runs a low risk of generalizing.

Algorithm

a. Random Forest

Random Forest is a machine learning algorithm that applies ensemble methods to perform classification and regression (Jaiswal & Samikannu, 2017). The method establishes multiple decision trees randomly, wherefore each predicts a forecast. Based on Equation (1), the outcome is determined based on the plurality vote attained from each tree in the forest. Every tree is trained with a randomly chosen subset of the data, and during each tree's node, only a subset of variables is chosen at random with respect to the split. Besides its ability to manage big and complicated data, Random Forest is also popular due to its efficiency in managing missing data, noise, outliers, and other such problems. It may quantify feature relevance and offer internal estimates of generalization error by utilizing out-of-bag (OOB) error estimation techniques. Additionally, parameters such as n_estimators, which typically defaults to 100, control the number of trees in the forest, and max_depth, commonly set around 10, limits the depth of each tree, helping balance between complexity and model performance.

$$\hat{y} = mode\{h_b(x)\}_{b=1}^{B} \quad (1)$$

b. Support Vector Machine

A machine learning approach for classification and regression problems is called Support Vector Machine (SVM) (Cortes & Vapnik, 1995). As indicated by formula (2), SVM divides data into two unique groups by identifying the best hyperplane (separating line), with the objective of maximizing the margin between the two classes. In the case of non-linear data, SVM uses a kernel function to map the data to a higher dimension, allowing for more effective separation. SVM is very effective in recognizing complex patterns in data, especially in cases with many features and a limited number of samples. This algorithm can also be adapted for multi-class classification problems and has good capabilities in handling noisy and imbalanced data. This algorithm can also be adapted for multi-class classification problems and has good capabilities in handling noisy and imbalanced data. In practice, key parameters such as probability=True enable the model to predict class probabilities, which is useful for calculating metrics like AUC-ROC, while random_state=42 ensures reproducibility of the results. Furthermore, SVM is sensitive to feature scaling, and hence, StandardScaler is used to standardize the data, improving the model's performance

$$K_{RBF}(x, y) = \exp\left(-\gamma \|x - y\|^2\right) \quad (2)$$

c. XGBoost

XGBoost (eXtreme Gradient Boosting) is a highly efficient and powerful machine learning algorithm for classification and regression problems, which relies on ensemble techniques with decision trees (Cherif & Kortebi, 2019). This algorithm works by building a model incrementally by adding decision trees one by one, where each new tree focuses on the errors made by the previous tree. XGBoost is known for its speed and ability to handle large datasets with fewer resources, as well as regularization techniques that can prevent overfitting. In addition, XGBoost uses techniques such as shrinkage to improve the model incrementally and subsampling to prevent the model from overfitting to the data (3). Due to its superior performance and efficiency, XGBoost is often used in various machine learning competitions and industrial applications such as price prediction, medical diagnosis, and anomaly detection. Additionally, key parameters like random_state=42 ensure reproducibility during training, while use_label_encoder=False prevents warnings related to label encoding. The eval_metric='mlogloss' parameter is also used to assess the model's performance using log loss, which is particularly beneficial for multi-class classification tasks.

$$L(\theta) = \sum_{i=1}^{n} l\left(y_i, \hat{y}_j\right) + \sum_{k=1}^{K} \Omega(\mathrm{f}_k) \quad (3)$$

d. Logistic Regression

Logistic Regression is a statistical method used to model the relationship between a binary dependent variable (e.g., 0 or 1) and one or more independent variables (Rymarczyk et al., 2019). In context, this model calculates the probability that an observation falls into a certain category based on the input variables. This process is carried out by calculating the log-odds or logarithm of the likelihood ratio, then

*name of corresponding author

using the sigmoid function to convert the resulting log-odds into a probability that lies between 0 and 1 in formula (4). Logistic regression is often used in various applications, such as classification, risk prediction, and diagnostic problems. In practical implementation, random_state=42 is used to ensure that the model's results are reproducible, while StandardScaler is applied to standardize the features, which is essential since Logistic Regression is sensitive to the scale of the input data.

$$p(\beta, X) = \frac{e^{X\beta}}{1 + e^{X\beta}} \quad (4)$$

e. Naive Bayes

Based on the idea that every characteristic in the dataset is independent of every other feature, Naive Bayes is a probability theory-based classification technique that applies Bayes' Theorem (Rahat et al., 2019). Based on feature data assigned probabilities for each class, the model chooses the class with the greatest probability as the expected class using formula (5). Despite the unrealistic unequally distributed features assumption, naive bayes classifiers are used on various texts classifying documents, sentiments, emails for spam, and more. Advantages of the algorithm include ability to parse and offer insights on large data sets in a timely and effective manner.

$$P(c|x_1, x_2, \cdots, x_n) = \frac{P(c)\prod_{i=1}^{n} P(x_i|c)}{P(x_1, x_2, \cdots, x_n)} \quad (5)$$

f. Decision Tree

A decision tree is one of the oldest and most popular classification and regressions in machine learning (Navada et al., 2011). The algorithm breaks down a dataset into smaller subsets a number of times based on certain conditions (rules/features) and these subsets are organized in a tree structure as seen in equation (6). Each node in the structure is a question/test and the leaves represent the classes or the expected outcomes of the dataset. Each branch of the tree is a feature value decision and the terminal is a leaf that provides the final prediction. Some of the most popular decision tree algorithms include ID3, C4.5, and CART (Classification and Regression Trees).

$$H(a) = -\sum [P(x_1|A)\log_2 P(x_1|A) - P(x_2|A)\log_2 P(x_2|A)] \quad (6)$$

g. Light GBM

LightGBM (Light Gradient Boosting Machine) is a Gradient Boosting Decision Tree (GBDT)-based machine learning algorithm designed to process big data more efficiently (Sun et al., 2020). Gradient-based one-side sampling and exclusive feature bundling in formula (7) are the algorithm's two primary methods. While the second strategy reduces the dimensionality of features by merging mutually incompatible characteristics, the first technique helps limit the quantity of data that must be processed at each iteration. LightGBM constructs decision trees vertically, in contrast to conventional GBDTs that construct trees horizontally. k This technique further enhances the efficiency of LightGBM on large datasets with complex attributes. What makes LightGBM well-known is its incredible speed and performance on large datasets with many attributes. LightGBM is also one of the most popular machine learning libraries for ranking, regression, and classification tasks. Additionally, in practice, random_state=42 is used to ensure reproducibility of results, while StandardScaler is applied to standardize the data, helping LightGBM perform optimally by reducing the effect of differing feature scales during training.

$$L(\theta) = \sum_i l\left(y_i, \hat{y}_j\right) + \sum_k \Omega\left(\mathrm{f}_k\right) \quad (7)$$

Evaluation Matrix

A confusion matrix is a performance metric used in machine learning to evaluate the accuracy of a classification model. (Heydarian et al., 2022). This matrix highlights the prediction results by comparing the anticipated classification with the actual findings. The four primary variables in this matrix are True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). These figures help calculate many important metrics, like as accuracy, precision, recall, and F1-score, that are used to assess the model's effectiveness. A confusion matrix is an excellent tool for understanding the kind of errors a model produces, which is crucial for improving its performance. In this study, the dataset was divided into training and testing sets in an 80:20 ratio, with 20% set aside for testing to assess the model's performance and 80% utilized for model training.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

- Accuracy measures how many predictions are correct using formula (8).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

- Precision shows how many positive predictions are correct using formula (9).

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

- Recall measures how much positive data was actually successfully predicted using formula (10).

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

- F1 Score is a measure that combining precision and recall to provide an overview of overall model performance using formula (11).

$$F1 = 2X \frac{Precision\ X\ Recall}{Precision\ X\ Recall} \quad (11)$$

RESULTS

In the results, we analyze the results of an experiment involving the Ransomware 2024 dataset, which contains various technical features related to executable files, with the aim of classifying files as benign or malware. The dataset includes various important information, such as file hash values (md5 and sha1), file types, and metadata related to the executable file structure such as EntryPoint, PEType, and MachineType. Furthermore, the data also includes information about file interactions with the system, such as registry activity, network activity, and running processes, all of which are important indicators in detecting potential malware threats. This study uses the Class column (Benign and Malware) to determine the final class used for the prediction results. Before further analysis, several preprocessing steps are applied, including data cleaning, handling of missing values, and feature transformation. Following this process, relevant features are selected and various machine learning algorithms are applied to test their ability to classify files as benign or malware. The results of the model evaluation will be discussed in detail, covering the accuracy and performance of each model, as well as the advantages and disadvantages of each approach used.

Dataset Distribution

The distribution shown in the graph Figure 2 Comparison of Class Distribution: Benign vs Malware (Training vs Test) shows that the data used in this study is quite balanced between the benign class (safe files) and the malware class (malicious files) in both data sets. In the training data (training set), although the number of benign files is slightly more (7,549 benign files and 7,855 malware files), the distribution remains in a relatively balanced proportion. Meanwhile in Figure 2, in the test data (test set), the class distribution is very balanced with 1,898 benign files and 1,954 malware files, which shows that the difference in the number of benign and malware files in the test data is very small, almost balanced.

With a more balanced distribution in the test data, the trained model has a better chance of predicting both classes fairly without favoring one class over the other. This helps the model avoid bias, ensuring that its malware detection capabilities on the test data are more representative and accurate, given the balanced number of classes.

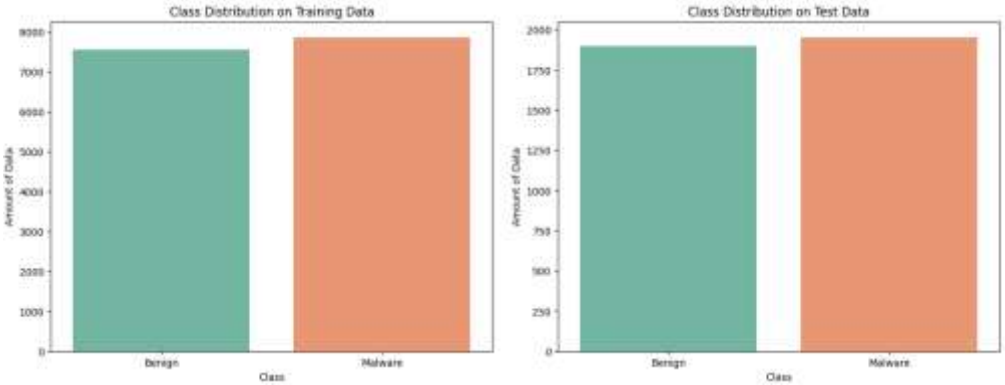


Fig. 2 Class Disibution in Training Data and Test Data

Machine Learning Model Performance

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

The performance of many machine learning methods for malware detection is compared in Table 1 without using the Interquartile Range (IQR) methodology for feature selection.. Based on accuracy, LightGBM performed best with 99.13%, followed by Random Forest with 99.03%. Excellent accuracy of over 98.8% was generated by the majority of methods, including XGBoost and SVM, suggesting that the models were successful in differentiating between malicious and benign data. With values as high as 0.99, other measures including Precision, Recall, and F1-Score demonstrated comparable outcomes among models. This shows that each method effectively struck a solid balance between lowering the likelihood of false positive detections (precision) and detecting malware files (recall).

Table 1. Performance Measure Model The algorithm does not use IQR.

Perfomance measure	Random forest	SVM	XGBoost	Logic regression	Naïve bayes	Decision tree	Light GBM
Accuracy	99.03	98.97	98.99	98.94	98.83	98.78	99.13
FPR	19	6	14	5	5	31	12
TPR	2166	2150	2159	2148	2143	2167	2163
FNR	23	39	30	41	46	22	26
TNR	2142	2155	2147	2156	2154	2130	2149
Precision	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Recal	0.99	0.99	0.99	0.99	0.99	0.99	0.99
F1_Score	0.99	0.99	0.99	0.99	0.99	0.99	0.99

However, differences between the algorithms can be seen in other metrics such as False Positive Rate (FPR) and False Negative Rate (FNR). For example, Decision Tree had the highest FPR (31) and Random Forest had a lower FNR (23), while Decision Tree had a significantly higher FPR (31) and a slightly higher FNR (22). TNR (True Negative Rate) and other metrics also showed small differences between the models, with LightGBM and Random Forest showing more stable and optimal performance in detecting malware. Overall, despite the small differences in these metrics, all tested algorithms performed very well in the classification task.

Table 2, which shows the performance results of machine learning algorithm models using the Interquartile Range (IQR) technique in feature selection, shows a slight improvement in several metrics compared to models that do not use IQR. For example, XGBoost achieved the highest accuracy with 99.20%, better than the 99.03% obtained without IQR. Other algorithms, such as Random Forest (99.09%) and LightGBM (99.17%), also showed higher accuracy than the model without IQR. The use of IQR in feature selection plays a significant role in reducing noise and improving data quality, which in turn improves the model's performance in better identifying malware threats.

Table 2. Performance Measure Model Algorithm Using IQR

Performance measure	Random forest	SVM	XGBoost	Logic regression	Naïve bayes	Decision tree	Light GBM
Accuracy	99.09	98.70	99.20	98.73	98.65	98.81	99.17
FPR	8	3	5	2	2	21	4
TPR	1871	1851	1872	1851	1848	1873	1870
FNR	27	47	26	47	50	25	28
TNR	1946	1951	1949	1952	1952	1933	1950
Precision	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Recal	0.99	0.99	0.99	0.99	0.99	0.99	0.99
F1_Score	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Other metrics, including as FPR (False Positive Rate), TPR (True Positive Rate), FNR (False Negative Rate), TNR (True Negative Rate), Precision, Recall, and F1-Score, likewise showed somewhat steady and consistent results across all models when IQR was installed. The use of IQR significantly reduced FPR in most models, such as Random Forest (FPR = 8) and XGBoost (FPR = 5), compared to models without IQR, which had higher FPRs. Furthermore, FNR was also recorded lower in many models, indicating that the models were more effective in avoiding false negative errors. Thus, the application of IQR not only increased accuracy but also improved the model's efficiency in detecting threats with fewer errors.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

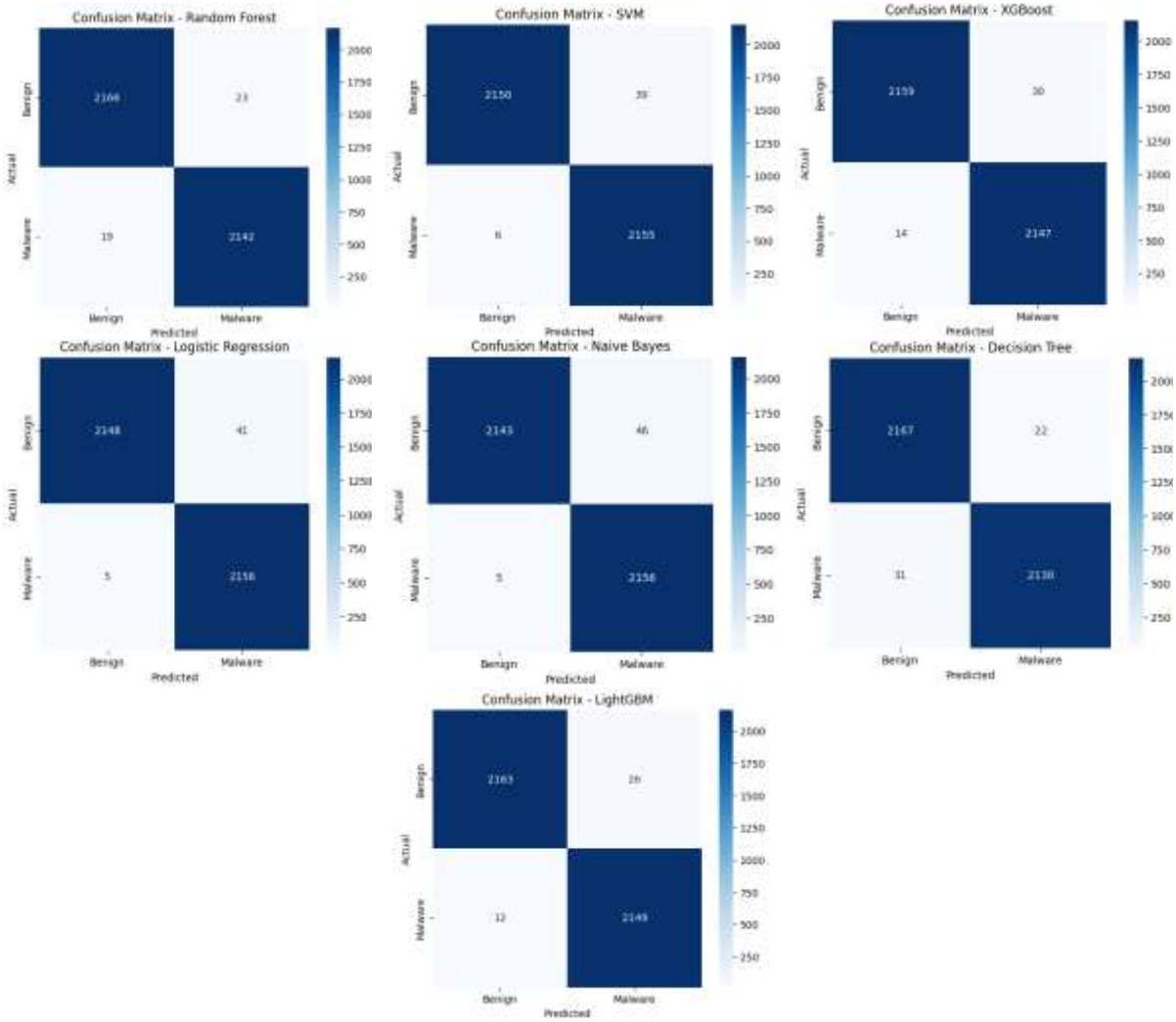
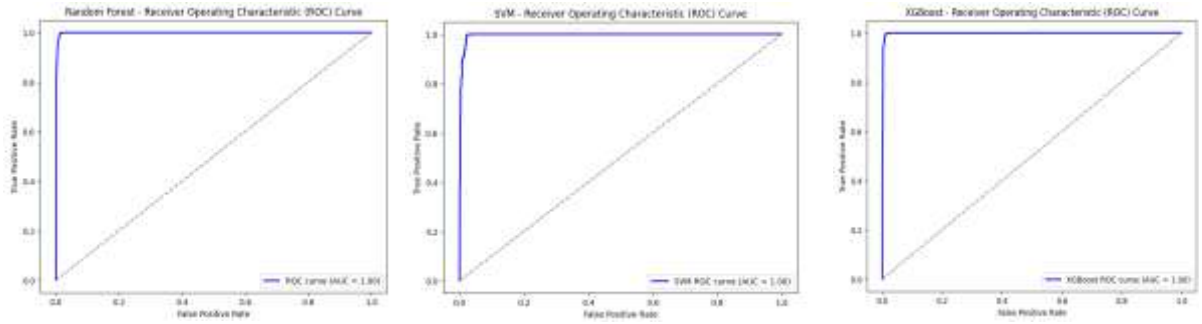


Fig. 3 Confusion Matrix comparison

Figure 3 shows the confusion matrix of several machine learning models for detecting malware, namely Random Forest, SVM, XGBoost, Logistic Regression, Naive Bayes, Decision Tree, and LightGBM. All models show good performance with a high number of True Positives (TP), such as Random Forest (2166 TP), SVM (2150 TP), and XGBoost (2159 TP). False Positives (FP) and False Negatives (FN) were relatively low, such as in Logistic Regression (2148 TP Benign, 2136 TP Malware) and Naive Bayes (2143 TP Benign, 2136 TP Malware), indicating that these models are quite accurate in distinguishing between Benign and Malware. Overall, all models demonstrate excellent accuracy.



*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

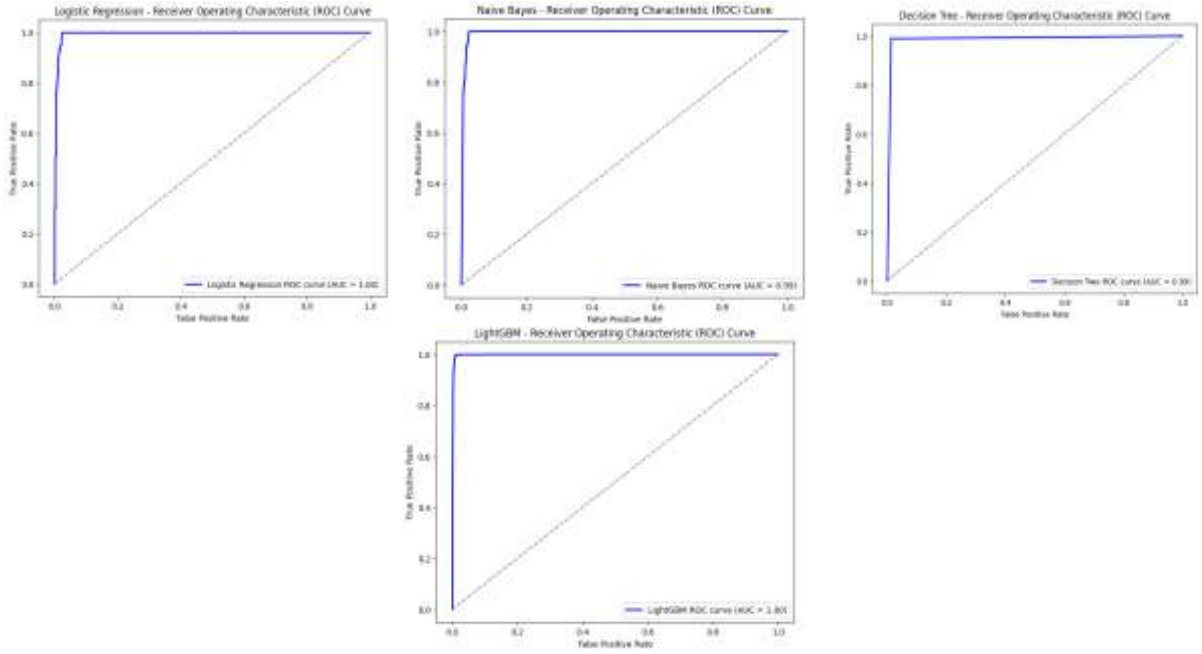
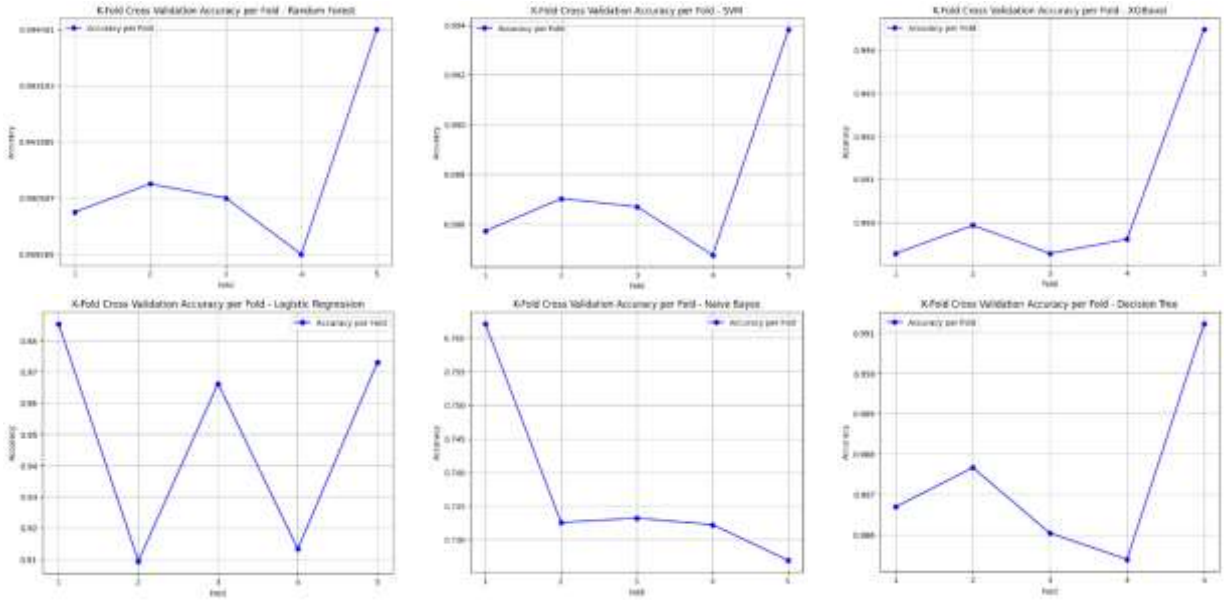


Fig. 4 ROC Comparison

The Receiver Operating Characteristic (ROC) Curve graph for many machine learning models, including Random Forest, SVM, XGBoost, Logistic Regression, Naive Bayes, Decision Tree, and LightGBM, is displayed in Figure 4. By comparing the True Positive Rate (TPR) and False Positive Rate (FPR) at different prediction thresholds, the ROC Curve is used to assess the effectiveness of classification models. Better model performance is indicated by a TPR that is closer to 1 and a lower FPR. The graph shows that Random Forest, SVM, XGBoost, and Logistic Regression all have Area Under Curves (AUC) of 1.0, which denotes extremely accurate predictions. Naive Bayes has an AUC of 0.979, Decision Tree has an AUC of 0.991, and LightGBM has an AUC of 1.0. This shows that most models perform very well in distinguishing between benign and malware classes, although Naive Bayes and Decision Tree are slightly lower in terms of AUC compared to other models.



*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

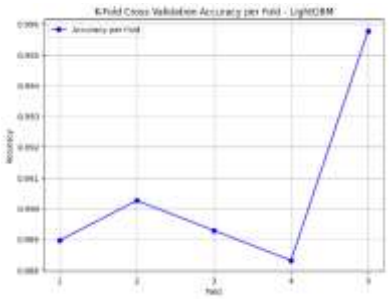


Fig. 5 K-Fold Comparison

The K-Fold Cross Validation findings for a number of machine learning models, including Random Forest, SVM, XGBoost, Logistic Regression, Naive Bayes, Decision Tree, and LightGBM, are displayed in Figure 5. The accuracy per fold, which displays the model's performance in different data divisions, is displayed in each graph. In folds 4 and 5, the Random Forest and XGBoost models demonstrate consistent accuracy with favorable outcomes. SVM shows a significant improvement in fold 5, while Logistic Regression experiences considerable accuracy variation between folds, peaking in fold 5. Naive Bayes and Decision Tree show greater fluctuations, with the lowest accuracy in fold 3. LightGBM performs poorly in several folds but shows a sharp improvement in fold 5. Overall, models such as Random Forest and XGBoost show better consistency in predicting classes than other models.

Based on the graph in Fig 6, which compares accuracy between algorithms, all machine learning models tested in this study demonstrated excellent performance in classifying malware and benign entities. XGBoost had the highest accuracy of 99.20%, followed by LightGBM with 99.17%. Random Forest also demonstrated excellent results with 99.09%, while other models such as SVM, Logistic Regression, Naive Bayes, and Decision Tree also achieved high accuracies of 98.70%, 98.73%, 98.65%, and 98.81%, respectively. Overall, the accuracy of each model demonstrated excellent ability in detecting malware, with very little difference between models. The results in Figure 6 indicate that the various machine learning algorithms implemented in the study are highly effective in detecting malware. While there are slight differences in accuracy, XGBoost outperforms the other models. However, all models yielded consistent results, with accuracy approaching 99%, indicating that they successfully differentiate between benign and malware files with a very low error rate.

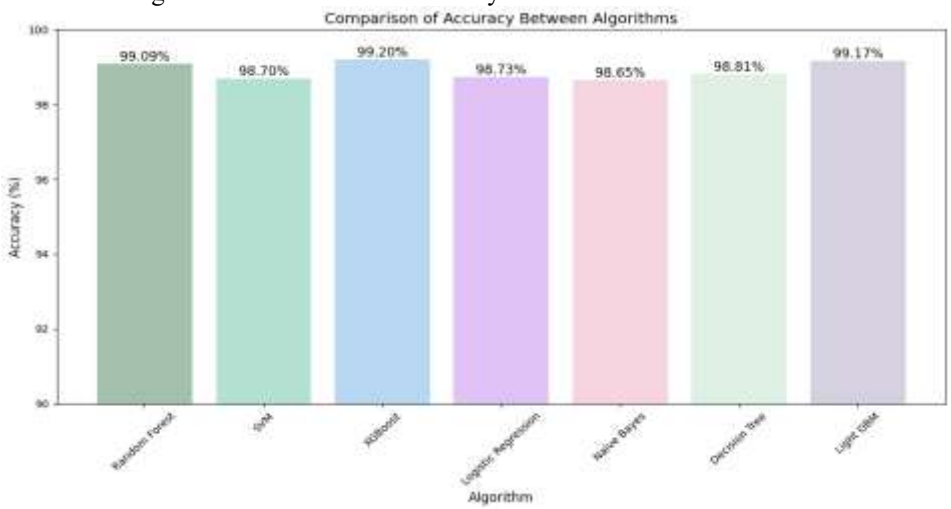


Fig. 6 Comparison of Algorithm Accuracy Results with the IQR Technique

DISCUSSIONS

The effectiveness of several machine learning algorithms in identifying malware is clearly compared in Tables 1 and 2, both before to and following the use of feature selection methods utilizing Interquartile Range (IQR). All algorithms exhibit good accuracy in Table 1, which details the performance of models without IQR. LightGBM recorded the best accuracy of 99.13%, followed by Random Forest with 99.03% and XGBoost with 98.99%. While overall accuracy is relatively good, some measures like False Positive Rate (FPR) and False Negative Rate (FNR) show rather considerable variances. For example, Decision Tree shows the highest FPR of 31, which indicates that this model tends to produce high false alarms, which can reduce user confidence in the malware detection system.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 10, Issue 1, Januari 2026
DOI : https://doi.org/10.33395/sinkron.v10i1.15634
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Conversely, the Random Forest algorithm has a lower FNR (23) compared to Decision Tree (22), indicating that Random Forest is better at detecting actual malware without missing any.

In Table 2, which shows the results of the model with IQR application, we see a significant improvement in performance in several algorithms, especially XGBoost, which achieved the highest accuracy of 99.20%, better than 99.03% in the model without IQR. The application of IQR proved effective in reducing FPR and FNR in most models, with XGBoost successfully lowering FPR to 5, much lower than 14 in the model without IQR, indicating that feature selection with IQR makes the model more efficient in avoiding false positive predictions. Similarly, Random Forest applying IQR showed an increase in accuracy to 99.09%, with a lower FPR compared to without IQR (FPR = 8). Overall, although the differences in accuracy between models are not too significant, the application of IQR has a clear positive impact in reducing detection errors (both FPR and FNR), as well as increasing the stability and efficiency of the model in detecting malware. This indicates that although all tested models demonstrate excellent performance, IQR as a feature selection technique plays a crucial role in minimizing classification errors, thereby making a significant contribution to improving malware detection quality in this study.

In this study, the application of machine learning techniques for malware detection with a focus on feature selection using the Interquartile Range (IQR) method has shown promising results. The experimental results show that the use of IQR in feature selection can significantly improve data quality, which ultimately strengthens the model's ability to distinguish between malware and benign classes. One important finding is that the XGBoost algorithm used in this study achieved the highest accuracy of 99.20%, slightly better than the model without IQR, which only achieved an accuracy of 99.03% in Fig. 6. This improvement shows that the IQR technique is very effective in reducing interference from irrelevant or redundant features, thereby improving data quality and allowing machine learning models to focus more on features that are truly relevant in the malware detection process. This proves that good feature selection is important in improving model performance, especially when dealing with large and complex datasets, such as the one used in this study, which included more than 21,000 malware and benign files, and after pre-processing resulted in 19,256 malware and benign files.

XGBoost is superior in this study due to its ability to utilize effective ensemble learning techniques to improve prediction accuracy and reduce classification errors. As a gradient boosting algorithm, XGBoost builds models incrementally by adding new decision trees to correct errors made by previous trees, allowing the model to better handle complex and unbalanced data. In addition, XGBoost is equipped with powerful regularization techniques to prevent overfitting and is capable of handling large datasets more efficiently, making it faster in the training process and more accurate in classification. Another advantage lies in its ability to handle various data types and optimize models with various parameters, resulting in more stable and higher performance compared to other algorithms such as LightGBM and Random Forest, especially when using feature selection techniques such as IQR that improve data quality and reduce noise. With an accuracy of 99.20% as shown in Table 2, XGBoost has proven to be a highly effective model for detecting malware, demonstrating that this algorithm has advantages in handling the increasingly complex and dynamic challenges of malware detection.

The use of IQR also shows a positive impact in reducing the False Positive Rate (FPR) and False Negative Rate (FNR) in various machine learning algorithms that were tested. For example, after applying IQR, the XGBoost algorithm successfully reduced the FPR to 5, which previously reached 14 in models without IQR. This is very important in the context of malware detection, because a high False Positive rate can cause false alerts, which can potentially disrupt system operations or cause users to lose confidence in the detection system. This reduction in FPR also reflects an increase in the efficiency of the detection system, where the model produces fewer false alarms and focuses more on accurate predictions. On the other hand, the decrease in FNR also shows that with IQR, the model becomes better at recognizing malware without missing existing threats, which could be fatal to the security system if ignored. Thus, these results emphasize that IQR not only improves accuracy but also optimizes the model's performance in reducing detection errors, both those that are too sensitive (false positives) and those that are less sensitive (false negatives).

However, even though the application of IQR in feature selection shows very positive results, it is important to note that there is no significant difference between models that use IQR and those that do not use IQR in terms of Precision, Recall, and F1-Score, all of which remain at very good levels (around 0.99) in almost all algorithms. This shows that although IQR contributes to improving accuracy, the quality of malware detection remains high even without IQR. The excellent Precision, Recall, and F1-Score values in all models show that each algorithm, whether using IQR or not, has almost equal capabilities in recognizing malware and benign files with a very low error rate. However, models that use IQR still show an improvement in detection efficiency, due to lower FPR and FNR, which ultimately leads to a reduction in errors in malware identification. Overall, the results of this study underscore the importance of applying feature selection techniques such as IQR to improve the performance of machine learning models in malware detection, while also showing that choosing the right feature selection technique can bring significant changes in the effectiveness and efficiency of the model.

*name of corresponding author

CONCLUSION

This study has successfully identified the effectiveness of several machine learning algorithms in detecting malware, with a focus on models such as XGBoost which demonstrated very high accuracy, reaching 99.20%. The pre-processing methods applied in this study, including data cleaning, normalization, feature selection, and the use of Interquartile Range (IQR) to select the most relevant features, have been shown to improve data quality and help the model recognize malware patterns more accurately. IQR, as a pre-processing technique, plays a crucial role in filtering features that have significant differences between benign and malware classes, thus strengthening detection capabilities. However, the study also encountered challenges such as dataset imbalance, where the number of benign files may overshadow malware files, potentially affecting model performance. Furthermore, there is a potential for overfitting due to the complexity of models like XGBoost, which may learn patterns too specific to the training data. This underscores the necessity of cross-validation to assess the model's generalization ability and ensure its robustness on unseen data. Additionally, to improve interpretability and understand the model's decision-making process, incorporating methods like SHAP or LIME could provide valuable insights into how features impact the prediction of malware versus benign files.

To address dataset imbalance, future studies could explore techniques such as oversampling, undersampling, or using cost-sensitive learning to balance the classes, improving model fairness and detection performance. To mitigate the potential for overfitting, cross-validation should be employed, ensuring that the model's performance is assessed across multiple subsets of data, thereby improving its ability to generalize. Additionally, for better interpretability, using SHAP (Shapley Additive Explanations) or LIME (Local Interpretable Model-Agnostic Explanations) would allow researchers to gain deeper insights into the model's decision-making process, highlighting which features are driving the predictions. This would enhance trust in the model's outputs and help refine the features used for malware detection.

REFERENCES

Aditya, & Dash, C. S. (2024). LightGBM-Powered Solutions for Backdoor Malware Detection in SCADA Networks. *2024 International Conference on Communication, Computing and Energy Efficient Technologies (I3CEET)*, 1765–1771. https://doi.org/10.1109/I3CEET61722.2024.10994149

Akhtar, M. S., & Feng, T. (2022). Malware Analysis and Detection Using Machine Learning Algorithms. *Symmetry*, *14*(11), 2304. https://doi.org/10.3390/sym14112304

Amato, A., & Di Lecce, V. (2023). Data preprocessing impact on machine learning algorithm performance. *Open Computer Science*, *13*(1), 20220278. https://doi.org/10.1515/comp-2022-0278

Arganda, E., Perez, A. D., De Los Rios, M., & Sandá Seoane, R. M. (2023). Machine-learned exclusion limits without binning. *The European Physical Journal C*, *83*(12), 1158. https://doi.org/10.1140/epjc/s10052-023-12314-z

Azerbaijan Technical University, Baku, Azerbaijan, & Asgarov, K. (2025). REAL-TIME ENDPOINT ANOMALY DETECTION USING ADAPTIVE STATISTICAL METHODS FOR BASELINE DEVIATIONS. *Problems of Information Technology*, *16*(1), 11–17. https://doi.org/10.25045/jpit.v16.i1.02

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing*, *300*, 70–79. https://doi.org/10.1016/j.neucom.2017.11.077

Chen, J., & Zhang, G. (2024). *Detecting Stealthy Ransomware in IPFS Networks Using Machine Learning*. Open Science Framework. https://doi.org/10.31219/osf.io/38ex9

Cherif, I. L., & Kortebi, A. (2019). On using eXtreme Gradient Boosting (XGBoost) Machine Learning algorithm for Home Network Traffic Classification. *2019 Wireless Days (WD)*, 1–6. https://doi.org/10.1109/WD.2019.8734193

El Hajj Hassan, S., & Duong-Trung, N. (2024). Machine Learning in Cybersecurity: Advanced Detection and Classification Techniques for Network Traffic Environments. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, *11*(3). https://doi.org/10.4108/eetinis.v11i3.5237

Fan, M., & Chunsheng, Y. (2025). Ensemble-Based Machine Learning Algorithm for Intelligent Network Security Threat Detection. *Informatica*, *49*(7). https://doi.org/10.31449/inf.v49i7.6640

Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: Multi-Label Confusion Matrix. *IEEE Access*, *10*, 19083–19095. https://doi.org/10.1109/ACCESS.2022.3151048

Hilabi, R., & Abu-Khadrah, A. (2024). Windows operating system malware detection using machine learning. *Bulletin of Electrical Engineering and Informatics*, *13*(5), 3401–3410. https://doi.org/10.11591/eei.v13i5.8018

Hussain, A. (2024). *Ransomware Dataset 2024 (1.0)* [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.13890887

*name of corresponding author

Jadhav, P., Bhavsar, P., Deore, S., Kuyate, K., & Bendale, Miss. G. (2024). Detecting Malware Activity Using Machine Learning. *IJARCCE*, *13*(4). https://doi.org/10.17148/IJARCCE.2024.13453

Jaiswal, J. K., & Samikannu, R. (2017). Application of Random Forest Algorithm on Feature Subset Selection and Classification and Regression. *2017 World Congress on Computing and Communication Technologies (WCCCT)*, 65–68. https://doi.org/10.1109/WCCCT.2016.25

Khan, Z., Naeem, M., Khalil, U., Khan, D. M., Aldahmani, S., & Hamraz, M. (2019). Feature Selection for Binary Classification Within Functional Genomics Experiments via Interquartile Range and Clustering. *IEEE Access*, *7*, 78159–78169. https://doi.org/10.1109/ACCESS.2019.2922432

Kumar, K., Parveen, A., Hasan, F., Kumar, A., Jain, A., & Kumar, V. (2024). Malware Attack Detection Using Machine Learning Techniques. *2024 4th Asian Conference on Innovation in Technology (ASIANCON)*, 1–4. https://doi.org/10.1109/ASIANCON62057.2024.10838032

Malik, S. (2021). The Machine Learning in Malware Detection: Muhammad Shairoze Malik. *International Journal for Electronic Crime Investigation*, *5*, 29–36. https://doi.org/10.54692/ijeci.2022.050387

Navada, A., Ansari, A. N., Patil, S., & Sonkamble, B. A. (2011). Overview of use of decision tree algorithms in machine learning. *2011 IEEE Control and System Graduate Research Colloquium*, 37–42. https://doi.org/10.1109/ICSGRC.2011.5991826

Nguyen, Q. H., Ly, H.-B., Ho, L. S., Al-Ansari, N., Le, H. V., Tran, V. Q., Prakash, I., & Pham, B. T. (2021). Influence of Data Splitting on Performance of Machine Learning Models in Prediction of Shear Strength of Soil. *Mathematical Problems in Engineering*, *2021*, 1–15. https://doi.org/10.1155/2021/4832864

Nikam, U. V., & Deshmuh, V. M. (2022). Performance Evaluation of Machine Learning Classifiers in Malware Detection. *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, 1–5. https://doi.org/10.1109/ICDCECE53908.2022.9793102

Patil, B. A., S, S. A., & M G, Dr. J. (2023). Detection of Malware using Machine Learning Approach. *International Journal for Research in Applied Science and Engineering Technology*, *11*(8), 736–741. https://doi.org/10.22214/ijraset.2023.55233

Poudyal, S., Subedi, K. P., & Dasgupta, D. (2018). A Framework for Analyzing Ransomware using Machine Learning. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1692–1699. https://doi.org/10.1109/SSCI.2018.8628743

Rahat, A. M., Kahir, A., & Masum, A. K. M. (2019). Comparison of Naive Bayes and SVM Algorithm based on Sentiment Analysis Using Review Dataset. *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, 266–270. https://doi.org/10.1109/SMART46866.2019.9117512

Rymarczyk, T., Kozłowski, E., Kłosowski, G., & Niderla, K. (2019). Logistic Regression for Machine Learning in Process Tomography. *Sensors*, *19*(15), 3400. https://doi.org/10.3390/s19153400

Sun, X., Liu, M., & Sima, Z. (2020). A novel cryptocurrency price trend forecasting model based on LightGBM. *Finance Research Letters*, *32*, 101084. https://doi.org/10.1016/j.frl.2018.12.032

Varshney, G., Varshney, S., Suman, A., Chouhan, K., & Suman, P. (2023). Machine Learning Based Malware Detection System. *2023 3rd International Conference on Advancement in Electronics & Communication Engineering (AECE)*, 559–563. https://doi.org/10.1109/AECE59614.2023.10428565

Verdonck, T., Baesens, B., Óskarsdóttir, M., & Vanden Broucke, S. (2024). Special issue on feature engineering editorial. *Machine Learning*, *113*(7), 3917–3928. https://doi.org/10.1007/s10994-021-06042-2

Wahid, M. F., Tafreshi, R., Al-Sowaidi, M., & Langari, R. (2018). Subject-independent hand gesture recognition using normalization and machine learning algorithms. *Journal of Computational Science*, *27*, 69–76. https://doi.org/10.1016/j.jocs.2018.04.019

*name of corresponding author