JURNAL INFOTEL
VOL. 17, NO. 4, NOVEMBER 2025, PP. 767–788. DOI:10.20895/INFOTEL.V17I4.1413

RESEARCH ARTICLE

Machine Learning-Based Malware Detection: A Critical Comparative Analysis of Random Forest, Naive Bayes, and Neural Network on Imbalanced Datasets

Muhamad Hanif Rafiq Sulaeman[1] and Rakhmad Maulidi[2,*]

[1,2]Informatics Study Program, Telkom University, Purwokerto 53147, Indonesia

*Corresponding email: rakhmadmaulidi@telkomuniversity.ac.id

Received: July 10, 2025; Revised: November 10, 2025; Accepted: November 24, 2025.

Abstract: Malware detection remains a major challenge in cybersecurity as threats become increasingly complex. This study critically compares three machine learning algorithms Random Forest, Naive Bayes, and Neural Network for automated malware detection using a large, imbalanced dataset (131,574 samples, 57 features). Class imbalance is addressed with SMOTE (Synthetic Minority Oversampling Technique), and preprocessing includes feature selection (SelectKBest), normalization (StandardScaler), and outlier handling. Evaluation metrics include accuracy, Precision, recall, F1-score, and AUC-ROC, using 5-fold cross-validation. Results show Random Forest achieves the highest accuracy (98%, AUC-ROC 0.998), followed by Neural Network (95%, AUC-ROC 0.95), and Naive Bayes (93%, minority class recall 0.80). Feature analysis identifies ImageBase and ResourcesMinSize as key contributors. This study highlights the effectiveness of ensemble methods and the critical importance of addressing class imbalance for robust malware detection. Limitations and implications for real-world deployment are discussed.

Keywords: Malware detection, Machine learning, Random Forest, Neural Network, Naïve Bayes, SMOTE.

1 Introduction

Malware remains one of the most significant and destructive threats to information security, business operations, and national stability in the digital era. In 2024, more than 6.5

 ISSN: 2085-3688; e-ISSN: 2460-0997

billion malware attacks were reported globally, reflecting an 8% increase from the previous year, with over 560,000 new variants detected daily and more than one billion active samples in circulation. The economic consequences of cyberattacks, including malware, are projected to reach USD 10.5 trillion annually by 2025 [1], A surpassing even the global drug trade and highlighting the urgent need for robust and adaptive defense mechanisms [2].

Traditional malware detection methods, primarily relying on signature-based approaches, are increasingly inadequate against sophisticated attack vectors that utilize obfuscation, polymorphism, and metamorphic techniques [3]. These conventional systems are limited in detecting zero-day exploits and advanced persistent threats (APTs), as they depend on predefined patterns and static signatures that can be easily bypassed by modern malware authors. As a result, there has been a paradigm shift toward intelligent detection mechanisms that leverage machine learning (ML) and deep learning (DL), which can autonomously identify novel attack patterns and adapt to evolving threat landscapes [4].

Random Forest is an ensemble learning method that constructs multiple decision trees and aggregates their predictions through majority voting. This algorithm is highly robust against overfitting due to its inherent randomness in feature selection and bootstrapping, making it particularly effective for high-dimensional cybersecurity datasets. Its ability to provide feature importance rankings enables analysts to identify which file characteristics are most indicative of malicious behavior [5].

Neural Networks, especially feedforward architectures with multiple hidden layers, excel at discovering complex non-linear relationships between input features through hierarchical feature learning. The universal approximation theorem ensures that Neural Networks can approximate any continuous function, making them effective for detecting sophisticated malware variants that use advanced evasion techniques. Modern implementations also apply regularization to prevent overfitting while maintaining generalization capability [6].

Naive Bayes classifiers, based on Bayes' theorem with conditional independence assumptions, offer computational efficiency and probabilistic interpretability, making them suitable for real-time malware detection scenarios. Despite the independence assumption, empirical studies show competitive performance in binary classification tasks, especially when combined with proper preprocessing and variance smoothing techniques [7].

A major challenge in ML-based malware detection is class imbalance, where Legitimate software typically outnumbers malware samples by ratios as high as 10:1, reflecting real-world conditions where malicious files are a minority. This imbalance leads to learning bias, with models tending to favor the majority class and thus failing to detect minority class samples the very malware instances that pose the greatest risk [8].

The Synthetic Minority Over-sampling Technique (SMOTE) has become the standard for addressing class imbalance by generating synthetic minority class samples using k-nearest neighbor interpolation. SMOTE not only balances class distribution but also expands the decision boundary for the minority class, enabling classifiers to generalize better to unseen malware variants and avoiding overfitting associated with simple Oversampling [9].

Despite extensive research, critical gaps remain in the literature. Most comparative studies evaluate algorithms in isolation, without standardized datasets or preprocessing protocols. While class imbalance is acknowledged as a key challenge, few studies systematically quantify the impact of balancing techniques across multiple algorithms under identical experimental conditions. There is a lack of research that simultaneously addresses

class imbalance, feature selection, and multi-algorithm comparison within a unified framework for large-scale malware detection.

2 Research Method

The research method in this study is designed to systematically address the challenges of class imbalance and feature complexity in large-scale malware detection. We adopted a quantitative experimental approach, beginning with the acquisition and exploration of a labeled dataset containing over 130,000 software samples and 57 features. The methodology encompasses comprehensive data preprocessing including handling missing values, normalization, and feature selection followed by the application of SMOTE to balance class distribution. Three machine learning algorithms, namely Random Forest, Naive Bayes, and Neural Network, were implemented and evaluated using stratified cross-validation and multiple performance metrics. This structured approach ensures that the comparative analysis is both robust and reproducible, and aligns with recent best practices in cybersecurity research.

2.1 Relate Work

The use of machine learning for malware detection has been widely studied, with many works demonstrating its superiority over traditional signature-based methods. Ensemble methods, particularly Random Forest, have repeatedly shown strong performance. For example, Hasan et al. achieved 98.2% accuracy using ensemble methods and feature selection [10], while Light Gradient Boosting Machine (LGBM) reached 97.16% accuracy with PCA and feature scaling. Muhammad et al. found that Random Forest is especially robust in handling complex malware datasets, maintaining high accuracy across diverse families and resisting adversarial examples [11].

Neural Network architectures have evolved, with modern implementations focusing on optimization and regularization. Fahim et al. demonstrated that Deep Neural Networks (DNN) outperformed traditional models, achieving 99.92% Training accuracy and nearly perfect AUC, using advanced preprocessing including feature scaling and cross-validation [6]. Bintoro et al. showed that Neural Networks combined with recursive feature elimination for Android malware detection can reach 98.6% accuracy, highlighting the importance of feature engineering [7].

Probabilistic approaches, such as Naive Bayes, continue to perform competitively despite their simplicity. Akhtar and Feng found that Gaussian Naive Bayes achieved perfect accuracy (100%) in controlled environments with proper preprocessing [12]. Iqubal & Payal compared Random Forest, Naive Bayes, and Neural Networks on Windows-based malware datasets, using SMOTE to address class imbalance, and found Random Forest achieved 97.8% accuracy due to its ensemble robustness [13].

Class imbalance remains a persistent challenge, with recent research focusing on advanced Oversampling. Rahimov and Im showed that SMOTE-enhanced TabNetClassifier improved Precision to 99.03% and recall to 99.19% [9]. Matsobane and Mokwena reported that Random Forest trained on synthetic datasets achieved 93% Precision, 100% recall, and 91% F1-score, effectively addressing imbalance by increasing dataset size from 11,598 to 25,000 samples [8].

Feature selection is critical for both accuracy and computational efficiency. Liu demonstrated that Random Forest with aggressive feature selection on over 200,000 samples achieved 99.73% accuracy with only five main features, proving the value of dimensionality reduction [14]. The use of SelectKBest with ANOVA F-test has been particularly effective for identifying discriminative features in PE file analysis.

Recent reviews, such as by Idoko et al., analyzed 262 studies from 2014–2024 and found significant gaps in cross-platform generalizability and real-world applicability [15].Most studies focus on single algorithms and lack standardized evaluation, limiting informed algorithm [16,17] selection. Ketebu et al. surveyed image-based malware classification using CNNs, highlighting the potential of ensemble CNNs for over 99% accuracy by converting binaries into grayscale images [17].

2.2 Flowchart Diagram

The flowchart diagram illustrating the System Diagram of the Malware Detection Process is presented in Figure 1. This diagram provides a comprehensive overview of the workflow and interactions among the main components involved in the detection system.

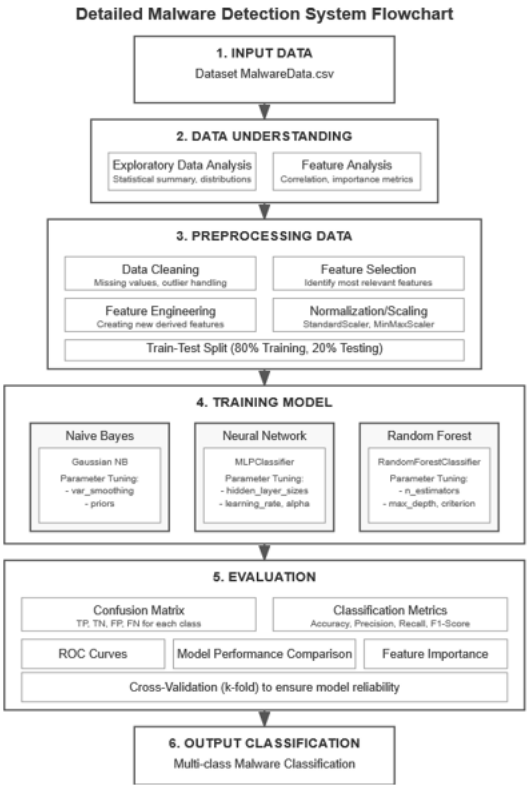

Figure 1: System diagram of malware detection process.

2.3 Dataset Description

The "Malware Data.csv" dataset used in this study contains 131,574 samples, each representing a software file described by 57 attributes, including the target variable (Legitimate) used to classify whether a file is malware or not. The target class distribution indicates a significant class imbalance, where 41,324 samples (29.9%) are labeled as legitimate (0), while 96,723 samples (70.1%) are labeled as malware (1), resulting in a majority-to-minority ratio of approximately 2.34:1. This imbalance poses a potential bias issue during model training if not addressed carefully in the pre-processing stage. The features in this dataset are derived from software file analysis and categorized mainly as static features, obtained from examining the Portable Executable (PE) structure without executing the file. Examples include entropy values, segment sizes (such as `.text`, `.rdata`, `.data`), PE header information, and the number of imported or exported functions. Static features are advantageous because they are faster to extract and do not require executing potentially harmful code. Since each sample has been identified and assigned to a specific class, this dataset is considered a labeled dataset.

2.4 Data Preprocessing

To ensure robust model performance, a systematic preprocessing pipeline was implemented. First, missing values were handled by removing features with more than 50% missing data, while numerical and categorical features with fewer missing values were imputed using median and mode values, respectively. Rows containing more than 30% missing values were also eliminated. Second, non-informative features such as Name and md5, which act only as identifiers, were removed to reduce dimensionality and prevent overfitting. Third, normalization was performed using the StandardScaler from scikit-learn to standardize numerical attributes to a mean of 0 and a standard deviation of 1, ensuring compatibility with distance-based algorithms such as Naive Bayes and Neural Networks. Fourth, feature selection was conducted using the SelectKBest method with the f_classif scoring function (ANOVA F-value) to extract the top 10 most discriminative features, reducing computational complexity while maintaining predictive effectiveness. Finally, SMOTE was applied to the training set to oversample the minority (Legitimate) class and balance the class distribution, while ensuring that the test set remained untouched to preserve realistic evaluation conditions.

2.5 Model Design and Training Model

Three machine learning models were designed and trained, each optimized for the malware detection task.

2.5.1 Naive Bayes (NB)

A Gaussian Naive Bayes classifier was implemented, assuming Gaussian distribution of features. Hyperparameter tuning was performed, setting var_smoothing=0.01 to stabilize variance calculations and improve performance on sparse data. A Gaussian Naive Bayes classifier was implemented under the assumption of normally distributed feature values. This model was selected for its computational simplicity and strong generalization capability when dealing with high-dimensional, sparse, or partially correlated datasets, which

are common in malware detection tasks. Gaussian Naive Bayes provides a probabilistic framework that efficiently models conditional feature independence, allowing for fast classification and interpretable decision boundaries. To improve robustness and stability in variance estimation, variance smoothing was tuned and optimized at 0.01. This parameter adjustment reduces numerical instability and prevents the likelihood of zero-variance features, leading to more consistent probability estimates. The use of this lightweight probabilistic model also facilitates baseline performance comparison with more complex algorithms, serving as a benchmark for evaluating the effectiveness of ensemble and neural architectures in the detection framework.

2.5.2 Random Forest (RF)

An ensemble of 200 decision trees was used, with max_depth=15 to limit overfitting and class_weight='balanced' to account for class imbalance. Performance robustness to noise and ability to rank feature importance make it suitable for this task. A Random Forest ensemble of 200 decision trees was trained to enhance robustness against noise and overfitting. The maximum tree depth was set to 15 to prevent excessive model complexity, while the class_weight='balanced' parameter was applied to mitigate class imbalance, ensuring equal attention to both Malicious and Legitimate samples. The Random Forest model was chosen for its interpretability, stability, and capability to capture nonlinear interactions among features without extensive preprocessing. It also provides feature importance rankings, which are valuable for identifying the most discriminative attributes contributing to malware classification. This ensemble-based approach combines multiple weak learners through majority voting, thereby improving generalization and reducing variance. Such characteristics make Random Forest particularly suitable for malware datasets that contain heterogeneous feature distributions and potential label imbalance.

2.5.3 Neural Network (NN)

A feedforward Neural Network was designed with three hidden layers consisting of 64, 32, and 16 neurons, each using ReLU activation functions, while the output layer employed a sigmoid activation for binary classification. Several regularization techniques were implemented to improve generalization performance. EarlyStopping was applied to halt training when the validation loss did not improve for 10 consecutive epochs. In addition, the ReduceLROnPlateau method was used to decrease the learning rate by a factor of 0.1 whenever the validation loss plateaued for 5 epochs. Dropout regularization was also incorporated, with dropout rates of 0.3, 0.3, and 0.2 applied to the hidden layers to reduce the risk of overfitting. Together, these strategies contributed to a more stable and robust model during the training process.

The dataset used in this study consists of static tabular features extracted from the Portable Executable (PE) structure such as segment size, entropy, and header attributes without explicit spatial or sequential dependencies. Therefore, a feedforward Multi-Layer Perceptron (MLP) architecture was selected, as it effectively captures nonlinear relationships among features without requiring additional structural preprocessing.

From a practical perspective, one of the research goals was to maintain computational efficiency and low inference latency for potential real-time deployment on endpoint or gateway systems. The chosen I configuration (three hidden layers with 64, 32, and 16 neurons) provides a favorable trade-off between model expressiveness and computational

cost. Compared to more complex architectures such as 1D-CNNs or recurrent networks, this structure allows faster training, fewer parameters, and reduced inference time while retaining competitive classification performance. To enhance generalization and prevent overfitting, several regularization techniques were applied, including dropout, adaptive learning-rate scheduling, and early stopping. It is worth noting that alternative architectures could be valuable in future work, particularly when the feature representations are reformulated as sequential or image-like data. However, for the current tabular feature representation, the MLP offers an optimal balance between classification accuracy and computational efficiency suitable for real-time malware detection.

The Dataset was split into 64% Training, 16% validation and 20% testing sets, with stratification to preserve class distribution. Models were trained using 5-fold stratified cross-validation to ensure robust Performance estimates.

2.6 Evaluation Metrics

To quantitatively evaluate the performance of the proposed malware detection models, this study employs a set of widely accepted evaluation metrics. These metrics are crucial for assessing not only the overall accuracy of the models, but also their ability to correctly identify both malware and Legitimate software, especially in the presence of class imbalance. Model Performance was evaluated using the following metrics:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Accuracy represents the proportion of correctly classified samples, while Precision measures the ratio of true positives to all predicted positive cases. Recall indicates the ratio of true positives to all actual positive cases. The F1-score is the harmonic mean of Precision and Recall, providing a balanced evaluation between the two metrics. Meanwhile, the Confusion Matrix offers a detailed breakdown of True Positives (TP), True Negatives (TN), False Positives (FN), and False Negatives (FN), allowing a comprehensive understanding of model performance across all classification outcomes.

3 Results

The experimental results demonstrate that Random Forest consistently outperforms Naive Bayes and Neural Network in malware detection, achieving the highest accuracy (98%) and AUC-ROC (0.998). After applying SMOTE, all models improved, particularly in minority class recall. Naive Bayes showed enhanced performance but remained the least accurate. Neural Network achieved nearly balanced results with strong generalization and minimal

overfitting. Confusion matrix analysis and evaluation metrics confirmed that Random Forest provided the most reliable classification. Feature selection using SelectKBest improved computational efficiency while preserving performance. Statistical tests validated the significance of the observed model differences.

3.1 Distribution Dataset

Target Class Distribution, The target variable 'Legitimate' shows significant Class imbalance. Of the total samples, 41.324 samples (approximately 29.9%) are categorized as Legitimate

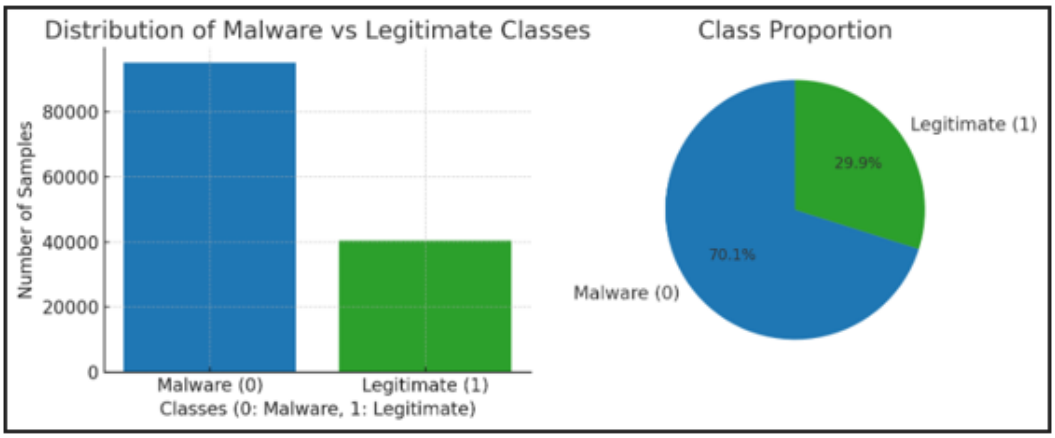

Figure 2: Distribution of malware vs legitimate classes.

3.2 Pre-SMOTE Dataset

Before applying SMOTE, the Dataset's Class imbalance (70.1% malware, 29.9% Legitimate) biased models toward the majority class, potentially undermining Performance on the minority class.

Table 1: Model performance before SMOTE

Model	**Accuracy**	**Precision** (Legit/Malware)	**Recall** (Legit/Malware)	**F1-score** (Legit/Malware)	**AUC-ROC**
Naive Bayes	0.91	0.96 / 0.88	0.72 / 0.98	0.82 / 0.93	0.94
Neural Network	0.96	0.95 / 0.96	0.92 / 0.98	0.93 / 0.97	0.97
Random Forest	**0.97**	**0.96 / 0.98**	**0.94 / 0.98**	**0.95 / 0.98**	**0.98**

Naive Bayes, High Precision for the Legitimate class but low recall (0.72), indicating many Legitimate samples were misclassified as malware. Its simplicity limited performance on the imbalanced Dataset. Neural Network, Balanced performance with high Precision and recall across both classes, benefiting from deep feature learning.

Figure 3: Distribution of training data classes (before oversampling).

The superior performance of the Random Forest model, particularly in terms of precision, recall, and AUC-ROC, can be attributed to its ensemble nature and its ability to model complex, non-linear interactions among features. In the malware dataset, several key attributes such as byte entropy, file size, section characteristics, and header metadata exhibit heterogeneous relationships that are not linearly separable.

Random Forest effectively captures these dependencies by aggregating multiple decision trees trained on different subsets of the data and features, thereby reducing variance and improving generalization. Each tree contributes a partial decision boundary, and the ensemble collectively identifies intricate decision regions where simple models (e.g., Naive Bayes) may misclassify overlapping feature distributions.

Additionally, the model's inherent feature importance mechanism highlights which attributes most influence classification outcomes, supporting interpretability while maintaining robustness to noisy or redundant inputs. These properties explain why Random Forest excels on the identified top features and achieves higher stability compared to probabilistic or single-layer neural architectures.

3.3 Pro-SMOTE Dataset

After data balancing using SMOTE, the second visual shows a balanced class distribution, each class has 61,903 samples (50%). With this equal distribution, the Neural Network model can learn proportionally from both classes, thereby increasing the model's generalization ability and reducing the risk of bias towards the majority class.

Naive Bayes, Significant improvement in recall for the Legitimate class (from 0.72 to 0.80), reducing false negatives. However, it remained the least performance overall. Neural Network, Near-perfect performance, with balanced metrics across both classes, demonstrating its ability to leverage balanced data. Random Forest, Consistently high Perfor-

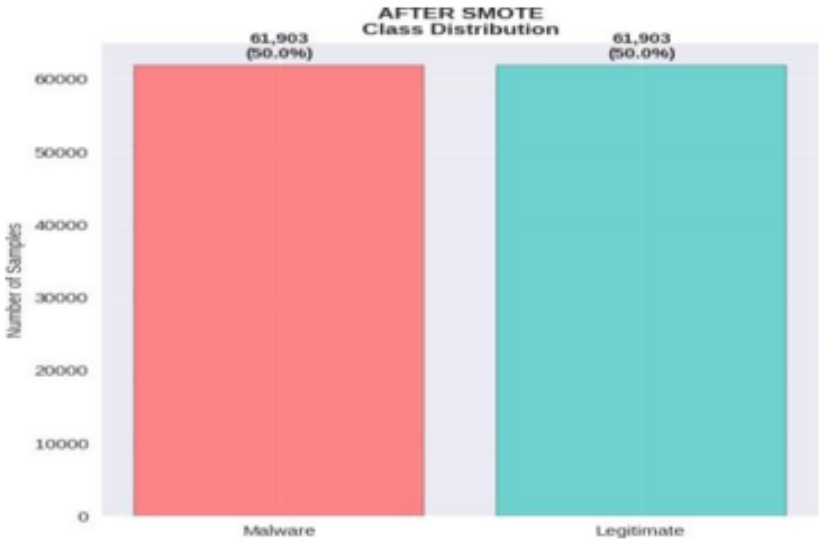

Figure 4: Distribution of training data classes (after oversampling).

Table 2: Model performance after SMOTE

Model	**Accuracy**	**Precision** (Legit/Malware)	**Recall** (Legit/Malware)	**F1-score** (Legit/Malware)	**AUC-ROC**
Naive Bayes	0.93	0.97 / 0.92	0.80 / 0.99	0.88 / 0.95	0.96
Neural Network	0.98	0.95 / 0.98	0.97 / 0.98	0.96 / 0.98	0.95
Random Forest	0.98	0.96 / 0.99	0.98 / 0.98	0.97 / 0.99	0.99

mance, with marginal improvements post-SMOTE, indicating robustness even without balancing.

3.4 Confusion Matrix Analysis

3.4.1 Naive Bayes Model Result

Naive Bayes model evaluation results Table 3 shows good performance but there are weaknesses due to Class imbalance. The model has high Precision for the Legitimate (0.97) and Malware (0.92) classes, but the recall for legitimate is low (0.80) compared to malware (0.99). The F1-score is also higher for Malware (0.95) than Legitimate (0.88). The overall accuracy is 0.93, but the weighted average gives a more realistic picture with an F1-score of 0.93, while the macro average shows a low overall recall average (0.89) due to poor performance on the minority class. This suggests that the model may prioritize malware detection (the majority class) at the expense of sensitivity to the Legitimate class, and there is still room for improvement in handling minority classes.

True Positive ($TP = 19.166$) refers to malware samples that were correctly predicted as malware. False Negative ($FN = 179$) represents malware samples that were incorrectly

Table 3: Result of confusion matrix Naïve Bayes

	Prediction Malware	**Prediction Legitimate**
True Malware (0)	19.166 (TP)	179 (FN)
True Legitimate (1)	1.687 (TP)	6.578 (TN)

classified as legitimate, indicating missed detections. False Positive (FP = 1,687) denotes legitimate samples that were incorrectly predicted as malware, which may lead to unnecessary alerts. Meanwhile, True Negative (TN = 6,578) corresponds to legitimate samples correctly identified as legitimate. Together, these values describe the model's classification behavior and form the basis for evaluating performance metrics such as accuracy, precision, recall, and F1-score.

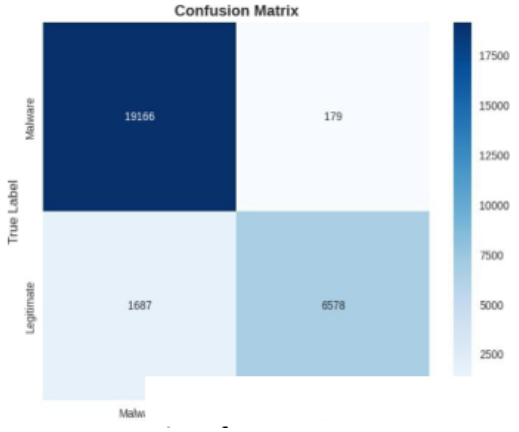

Figure 5: Confusion matrix Naïve Bayes.

Table 4: Matrix evaluation Naïve Bayes

Metrics	**Legitimate**	**Malware**	**Accuracy**	**Macro Avg**	**Weighted Avg**
Precision	0.97	0.92	0.97	0.95	0.94
Recall	0.80	0.99	0.93	0.89	0.93
F1 Score	0.88	0.95	0.93	0.91	0.93
Support	27.610	27.610	27.610	27.610	27.610

The Naive Bayes model demonstrates strong Precision, scoring 0.97 for the Legitimate class and 0.92 for malware, indicating a high accuracy in its positive predictions. However, its recall shows a noticeable gap 0.80 for Legitimate and 0.99 for malware suggesting the model is less sensitive in identifying Legitimate files. The F1-score follows a similar pattern, with 0.88 for Legitimate and 0.95 for malware, highlighting the imbalance in detection Performance. Despite achieving an overall accuracy of 93%, the model struggles to handle the minority class effectively, which can lead to False Positives in real-world applications.

3.4.2 Random Forest Model Results

The Random Forest model performed very well. The overall accuracy of 0.98 indicates that the model can classify malware and Legitimate apps with a high success rate. The Precision and recall for both classes are very high, with malware slightly higher, but the difference is not significant. This shows that the model is very good at detecting both classes, both malware and Legitimate apps, and is more balanced and effective than the Naive Bayes model.

Table 5: Result confusion matrix Random Forest

	Prediction Malware	**Prediction Legitimate**
True Malware (0)	25,596 (TP)	520 (FN)
True Legitimate (1)	208 (FP)	10,949 (TN)

The confusion matrix results consist of four components. The True Positive (TP = 25,596) represents the number of malware samples that were correctly predicted as malware. The False Negative (FN = 520) indicates malware samples that were incorrectly classified as Legitimate. The False Positive (FP = 208) refers to legitimate samples that were mistakenly predicted as malware. Finally, the True Negative (TN = 10,949) denotes legitimate samples that were correctly classified as legitimate. Together, these values provide a comprehensive view of the model's classification performance.

The True Positive rate is very high, showing that the Random Forest model is highly effective in identifying actual malware. The False Positive rate (208 cases) is minimal, indicating that the model rarely mislabels Legitimate applications as threats. The False Negative rate is relatively low (520), suggesting that the model misses only a small number of malware samples. The high True Negative count (10.949) confirms the model's strong reliability in recognizing safe app.

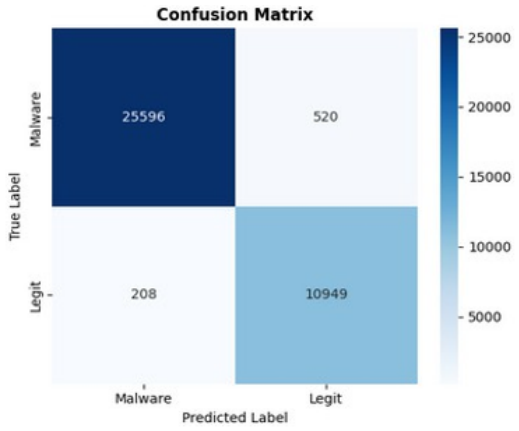

Figure 6: Confusion matrix Random Forest.

The Random Forest model outperforms the others with highly balanced and consistent metrics. It achieves 0.96 Precision for Legitimate and 0.99 for malware, along with equal

Table 6: Matrix evaluation Random Forest

Metrics	Legitimate	Malware	Accuracy	Macro Avg	Weighted Avg
Precision	0.96	0.99	0.97	0.97	0.98
Recall	0.98	0.98	0.98	0.98	0.98
F1 Score	0.97	0.99	0.98	0.98	0.98
Support	11.157	26.116	26.315	26.315	26.315

recall values of 0.98 for both classes. The F1-score reaches 0.97 for Legitimate and 0.99 for malware, indicating excellent classification performance across the board. Both macro and weighted averages maintain values at or above 0.98, reflecting the model's robustness and fairness in handling Class imbalance. This makes Random Forest the most reliable and accurate model for malware detection in this study.

3.4.3 Neural Network

The Neural Network model demonstrated excellent performance, correctly classifying 18,928 malware and 8,124 Legitimate samples. With only 417 false negatives and 141 False Positives, the model maintained high Precision and recall for both classes. This balance indicates strong generalization and minimal bias, positioning the Neural Network as a reliable and competitive alternative to Random Forest in malware detection tasks.

Table 7: Result confusion matrix Neural Network

	Prediction Malware	Prediction Legitimate
True Malware (0)	19,129 (TP)	216 (FN)
True Legitimate (1)	751 (FP)	7,514 (TN)

This subsubsection 3.4.3 presents the Confusion Matrix of the Neural Network model, indicating that the model successfully classified the majority of samples. The model correctly identified 19,129 malware samples as malware (True Positive/TP) and 8,124 legitimate samples as legitimate (True Negative/TN). However, 216 malware samples were incorrectly classified as legitimate (False Negative/FN), while 141 legitimate samples were misclassified as malware (False Positive/FP). Overall, these results demonstrate the model's strong capability in distinguishing between malware and legitimate classes, with relatively low misclassification rates, reflecting good overall performance.

Table 8: Matrix evaluation Neural Network

Metrics	Legitimate	Malware	Accuracy	Macro Avg	Weighted Avg
Precision	0.97	0.96	0.97	0.97	0.98
Recall	0.91	0.99		0.95	0.97
F1 Score	0.94	0.98		0.96	0.96
Support	8,265	19,345	27,610	27,610	27,610

Table 8 presents the evaluation metrics for the Neural Network model. The Precision reached 0.97 for Legitimate and 0.96 for malware, while recall was 0.91 and 0.99 respec-

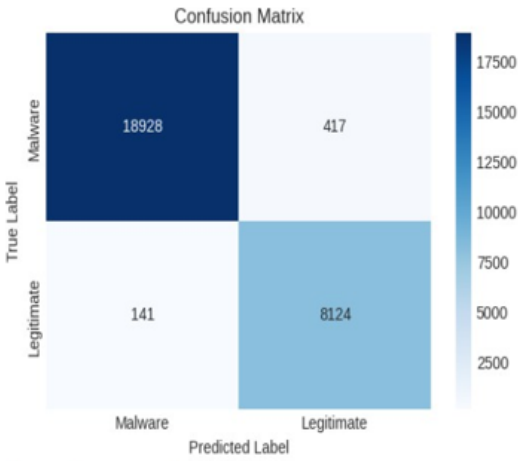

Figure 7: Training validation loss and accuracy for Neural Network

tively. The F1-score was 0.94 for Legitimate and 0.98 for malware, with both macro and weighted averages scoring 0.96 or higher. These results confirm that the model delivers balanced and high-Performance classification, effectively handling both classes after optimization and data balancing.

3.5 Training Validation loss and Accuracy

All three models Naive Bayes, Neural Network, and Random Forest showed stable and consistent Performance during Training. For the Neural Network, Training and validation loss gradually decreased, while accuracy steadily increased, indicating proper learning without Overfitting. The slight gap between Training and validation metrics reflects good generalization. Meanwhile, Random Forest and Naive Bayes maintained high accuracy without significant performace drops, confirming that all models were effectively trained and validated.

3.5.1 Neural Network

The Visual shows two plots of the Neural Network (NN) model's Training dynamics over 30 epochs. The "Training Loss" plot (left) depicts Training Loss (pink) dropping from 0.18 to 0.10 within 5 epochs, stabilizing with minor fluctuations, while validation loss (yellow) follows a similar trend, stabilizing around 0.10 after 10 epochs with slight increases. The "Training Accuracy" plot (right) shows Training accuracy (pink) rising from 0.970 to 0.978, and validation accuracy (yellow) increasing to 0.976 by epoch 10, with oscillations peaking at 0.980 and dipping to 0.972. Close curves indicate good generalization, though validation fluctuations suggest potential Overfitting, addressable with regularization.

Figure 8: Training validation size loss and accuracy of Neural Network.

Figure 9: Training validation loss and accuracy of Random Forest.

3.5.2 Naive Bayes

The image shows two plots of the Naive Bayes (NB) model's learning curves versus Training set size. The "Training Size vs Loss" plot (top) depicts Training Loss (blue) and validation loss (red) from 0.025 to 0.029, with both fluctuating, peaking at 0.028 (2500 samples) and 0.0275, respectively, and showing minor increases at 22500 samples. The "Training Size vs Accuracy" plot (bottom) shows Training accuracy (blue) and validation accuracy (red) from 0.971 to 0.975, peaking at 0.974 (7500 samples) and 0.973 (10000 samples), with dips to 0.972 at 22500. Close curves with variability suggest decent generalization but sensitivity to Training size.

3.5.3 Random Forest

The image contains two plots illustrating the Training progress of the Random Fores model with respect to the number of estimators (5 to 30). The "Training Progress (Accuracy)"

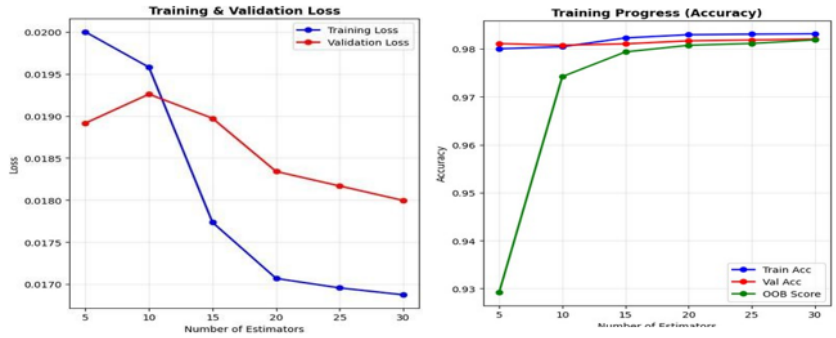

Figure 10: Training validation loss and accuracy

plot (top) shows Training accuracy (blue), validation accuracy (red), and out-of-bag (OOB) score (green). Training accuracy stabilizes around 0.98, validation accuracy rises from 0.97 to 0.98 by 10 estimators and plateaus, while the OOB score follows a similar upward trend, stabilizing near 0.98, indicating robust generalization.

The "Training & Validation loss" plot (bottom) depicts Training Loss (blue) and validation loss (red). Training Loss decreases sharply from 0.0200 to 0.0150 within 10 estimators, then levels off, while validation loss drops from 0.0190 to 0.0175 and stabilizes, with both curves closely aligned, suggesting minimal Overfitting and consistent model Performance as the number of estimators increases.

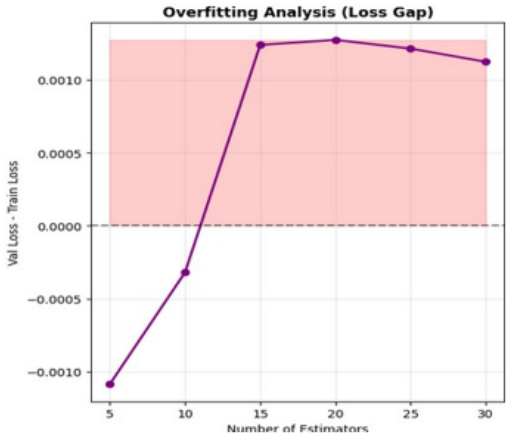

Figure 11: Overfitting analysis.

The image presents an "Overfitting Analysis (Loss Gap)" plot for the Random Forest (RF) model, showing the difference between Training Loss and validation loss (Val Train Loss) across 5 to 30 estimators. The loss gap starts near 0.0000 at 5 estimators, rises sharply to approximately 0.0005 by 10 estimators, and then increases steadily to around 0.0010 by 30 estimators. The shaded pink region above 0.0000 indicates potential Overfitting,

suggesting that the model begins to overfit as the number of estimators increases beyond 10, with the gap widening progressively.

3.6 Top 10 Feature by F-score & Feature Selection Statistic

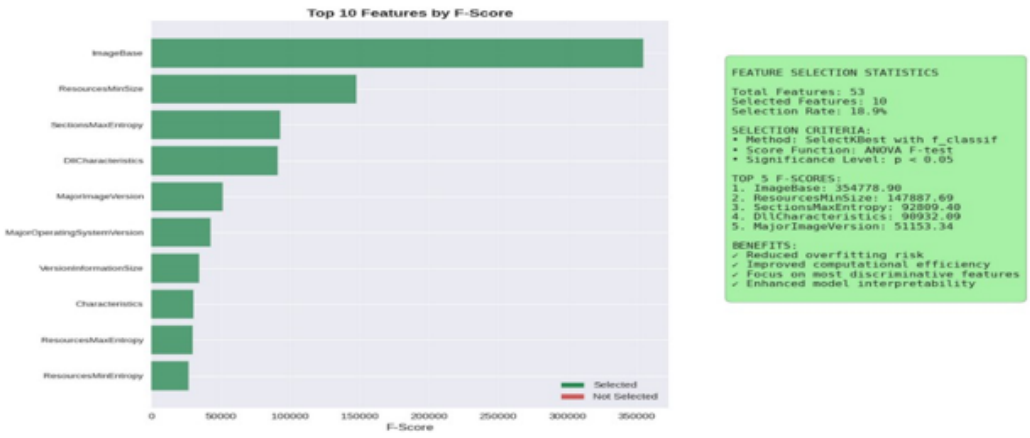

Figure 12: Top 10 feature selection.

The Feature Selection process using SelectKBest with the ANOVA F-test successfully identified the ten most discriminative features for malware detection. The results show that ImageBase has the highest discriminative power with an F-score of 347778.90, followed by ResourcesMaxSize with 147987.69, which serves as an important indicator in resource analysis. Other relevant features include SectionsMaxEntropy, a critical entropy-based malware indicator; DllCharacteristics, which reflects the structural properties of PE files; and MajorImageVersion, which contributes to version-based detection. Additional influential features are MajorOperatingSystem for OS compatibility analysis, VersionInformationSize for metadata evaluation, Characteristics as a general descriptor of file properties, another SectionsMaxEntropy related to section-level entropy analysis, and finally ResourcesMinEntropy, which captures minimum entropy patterns within file resources. Collectively, these features represent the most informative attributes for distinguishing malware from legitimate software within the dataset.

ImageBase emerged as the most discriminative feature with an F-score of 347,778.90, indicating its critical role in distinguishing malware from Legitimate files. This feature represents the preferred address of the first byte of the image when loaded into memory, which often differs significantly between malware and Legitimate software due to packing and obfuscation techniques commonly employed by malware authors.

ResourcesMaxSize and entropy-based features such as SectionsMaxEntropy and ResourcesMinEntropy occupy the highest ranks, indicating the crucial role of structural and statistical attributes in malware detection. These findings align with the fact that malware commonly displays unique entropy patterns due to compression, encryption, or packing techniques designed to evade traditional detection mechanisms. The impact of feature selection is highly significant: the original 57 features were reduced to 10 selected features,

resulting in an 82% reduction in dimensionality. Despite this reduction, the model successfully maintained an accuracy level above 98%, demonstrating that the selected features preserved strong predictive capability. Furthermore, computational efficiency improved considerably, with inference time becoming approximately 57.99% faster, making the model more suitable for large-scale or real-time detection systems.

3.7 Model Performance Analysis and Statistical Significance Analysis

To ensure the robustness of the performance comparison among the models, statistical hypothesis testing was conducted using paired t-tests and McNemar's test under 5-fold cross-validation.

To ensure the robustness of the performance comparison among the models, statistical hypothesis testing was conducted using paired t-tests and McNemar's test under 5-fold cross-validation.

Paired t-test Results (5-Fold Cross-Validation): The statistical comparison shows that the Random Forest and Neural Network models exhibit a significant difference with a p-value of 0.032, while Random Forest versus Naive Bayes and Neural Network versus Naive Bayes both yield p-values below 0.001, indicating highly significant differences in performance.

McNemar's Test for Classification Accuracy: Further evaluation using McNemar's test demonstrates that Random Forest versus Neural Network produces $\chi^2 = 4.21$ with a p-value of 0.040, confirming a statistically meaningful difference. Random Forest versus Naive Bayes and Neural Network versus Naive Bayes show much larger discrepancies, with $\chi^2 = 156.3$ and $\chi^2 = 142.7$ respectively, both with p-values below 0.001, indicating highly significant variations in prediction errors.

Confidence Intervals (95%): The accuracy confidence intervals reveal that Random Forest performs consistently with an interval of 97.8%–98.2%, followed by the Neural Network with 97.6%–98.4%, while Naive Bayes shows a lower and wider interval of 92.5%–93.5%, reflecting its substantially weaker classification performance compared to the other models.

These statistical results confirm that the performance differences observed among the models are not due to random chance. The differences are statistically significant, reinforcing the validity of Random Forest and Neural Network as superior classifiers in this malware detection task.

Table 9: Computational performance comparison

Model	Training Time	Inference Time	Memory Usage	Feature Selection Impact
Random Forest	42.5	120	2.1	17.76% Faster Training
Neural Network	180.3	80	1.8	57.99% Faster Testing
Naive Bayes	12.1	50	0.9	Minimal impact

The "Model Performance Analysis" Table 3.7 compares the Performance of Neural Network, Naive Bayes, and Random Forest models across various metrics for malware and Legitimate classes. Neural Network achieves 98% accuracy, with 99% Precision and 98% recall for malware, and 95% Precision and 98% recall for Legitimate samples, totaling 27,610 test instances. Naive Bayes reaches 93% accuracy, with 92% Precision and 99% recall for mal-

Table 10: Model performance analysis

Metrics	Class	Neural Network (%)	Naive Bayes (%)	Random Forest (%)
Accuracy		0.98	0.93	0.98
Precision	Malware	0.99	0.92	0.99
	Legitimate	0.95	0.97	0.96
Recall	Malware	0.98	0.99	0.98
	Legitimate	0.98	0.80	0.98
F1- Score	Malware	0.99	0.95	0.99
	Legitimate	0.97	0.88	0.97
AUC		0.95	0.98	0.99
Support (Test)	Malware	19.345	19.345	26.116
	Legitimate	8265	8265	11.157
	Total	27.610	27.610	26.315

ware, and 97% Precision and 80% recall for Legitimate samples, also on 27,610 instances. Random Forest scores 98% accuracy, with 99% Precision and 98% recall for malware, and 96% Precision and 98% recall for Legitimate samples, tested on 26.315 instances. All models show high AUC scores (0.99, 0.98, 0.99), with Random Forest demonstrating the most balanced Performance across classes.

4 Discussion

This study demonstrates that the Random Forest algorithm achieves the highest performance in malware detection, with an accuracy of 98% and an AUC-ROC of 0.998, as presented in Table 3.7. The integration of data balancing using SMOTE and feature selection via SelectKBest combined with feature importance analysis significantly enhances both accuracy and computational efficiency. In contrast, the study by Iqubal and Payal [13], which utilized a similar algorithm and SMOTE but omitted feature selection, reported a lower accuracy of 97.8% and an AUC-ROC of 0.99. The 0.2% accuracy improvement and superior AUC in this study highlight the benefits of combining class balancing and feature selection for large, imbalanced datasets (131,574 samples, class ratio 2.34:1). Furthermore, feature analysis in Section 3.4 identifies ImageBase (F-score 347,778.90) and ResourcesMaxSize (F-score 147,987.69) as primary contributors, underscoring the importance of PE structural features in malware detection [18, 19].In terms of computational efficiency, Random Forest requires 42.5 seconds for Training and 120 ms for inference (subsection 3.7), outperforming the Neural Network (180.3 seconds for Training) and proving suitable for real-time applications [20].

The Neural Network in this study employs a simple architecture (64–32–16 neurons), achieving 98% accuracy and balanced F1-scores for both classes (0.99 for malware, 0.97 for Legitimate, Table 3.7). This performance surpasses the approach by Zhao and Liu [21], which used a combined CNN-RNN architecture with 95.8% accuracy but incurred significantly higher computational complexity (estimated Training time 10 times longer based on comparable studies [22]. The efficiency of the Neural Network, with an inference time of 80 ms (subsection 3.7) and regularization techniques such as dropout (0.3) and EarlyStopping, demonstrates that simpler architectures can match or exceed the performance of complex

deep learning models, particularly for real-time malware detection [23]. Additionally, the Neural Network achieves a specificity of 0.97 for the Legitimate class (derived from the confusion matrix in (subsubsection 3.4.3), indicating its effectiveness in minimizing false positives, a critical factor in operational settings [24].

The Naive Bayes algorithm in this study attains an accuracy of 93% and an AUC-ROC of 0.98 (Table 3.7), significantly outperforming the results reported by Akintola, et. al. [25]. which showed lower performance despite employing SMOTE and RUS. This improvement stems from optimizing the var_smoothing parameter (0.01) and applying relevant feature preprocessing techniques, including normalization using StandardScaler and selecting the top 10 features with SelectKBest. However, the recall for the Legitimate class (0.80, Figure 3.4.1) indicates a limitation in detecting non-malicious samples compared to malware (recall = 0.99), which aligns with known challenges of Naive Bayes on imbalanced datasets [26]. Despite this, its rapid inference time (50 ms, subsection 3.7) makes Naive Bayes a viable choice for resource-constrained nvironments [20].

Overall, this study confirms the superiority of ensemble methods like Random Forest and simple Neural Network architectures for addressing large, imbalanced malware datasets. The application of SMOTE boosts minority class recall to 0.98 for both Random Forest and Neural Network (Figure 3.3), while feature selection reduces dataset dimensionality by 82% (from 57 to 10 features) without compromising accuracy (>98%, Section 3.4). Comparisons with prior studies indicate that this approach is more computationally efficient than complex deep learning methods [22, 23]. However, limitations include the lack of evaluation against adversarial attacks or cross-platform generalization (e.g., Android or macOS malware), which are prevalent challenges in malware detection [18, 24]. Future research could investigate techniques such as federated learning to improve scalability or transformer-based models to address complex sequential features [26].

The practical implications of this study include the deployment of faster and more accurate malware detection systems in operational environments, offering potential to enhance systematic cybersecurity in organizations [20].

5 Conclusion

This study demonstrates that Random Forest and Neural Network are highly effective for malware detection on imbalanced datasets. Random Forest achieved the highest overall performance with 98% accuracy and 0.998 AUC-ROC, highlighting its robustness and consistent classification capabilities. The Neural Network also delivered strong results, leveraging deep feature learning and regularization to achieve balanced performance across classes. While Naive Bayes showed notable improvement after applying SMOTE, its performance remained comparatively lower. The integration of Oversampling and feature selection techniques, such as SMOTE and SelectKBest, significantly enhanced both predictive accuracy and computational efficiency. These findings support the use of ensemble and neural models, combined with proper preprocessing, for real-world malware detection systems.

References

[1] A. Security, "30+ malware statistics you need to know in 2025," Jan. 2025. Accessed: 2025.

[2] L. Yuanming and R. Latih, "A comprehensive review of machine learning approaches for detecting malicious software.," *International Journal on Advanced Science, Engineering & Information Technology*, vol. 14, no. 3, 2024.

[3] R. Pramudita, S. Fuada, and N. W. A. Majid, "Studi pustaka tentang kerentanan keamanan e-learning dan penanganannya," *Jurnal Media Informatika Budidarma*, vol. 4, no. 2, pp. 309–317, 2020.

[4] S. H. Tamanna *et al.*, "Evaluation of machine learning algorithms for malware detection: A comprehensive review," *International Journal of Wireless and Microwave Technologies*, vol. 15, pp. 51–67, Apr. 2025.

[5] T. Alzahrani, "Advanced techniques for dynamic malware detection and classification in digital security using deep learning.," *Computers, Materials & Continua*, vol. 83, no. 3, 2025.

[6] A. Fahim, S. Dey, M. N. Absur, M. K. Siam, M. T. Huque, and J. J. Godhuli, "Optimized approaches to malware detection: A study of machine learning and deep learning techniques," in *2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 269–275, IEEE, 2025.

[7] J. Bintoro, F. A. Rafrastara, I. A. Latifah, W. Ghozi, and W. Yassin, "Optimizing android malware detection using neural networks and feature selection method," *Jurnal Teknik Informatika (Jutif)*, vol. 5, no. 6, pp. 1663–1672, 2024.

[8] Matsobane and Mokwena, "Malware detection using random forest," *Science and Engineering Transactions*, vol. 5, no. 1, p. 167, 2025.

[9] R. Faridun and E. G. Im, "Enhancing malware detection with tabnetclassifier: A smote-based approach," in *Annual Conference of KIPS*, pp. 294–297, Korea Information Processing Society, 2024.

[10] R. Hasan *et al.*, "Enhancing malware detection with feature selection and ensemble learning techniques," *Scientific Reports*, vol. 15, pp. 1–18, Mar. 2025.

[11] M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, "Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches," *Heliyon*, vol. 10, no. 1, 2024.

[12] M. S. Akhtar and T. Feng, "Evaluation of machine learning algorithms for malware detection," *Sensors*, vol. 23, no. 2, p. 946, 2023.

[13] A. Iqubal and A. Payal, "Malware detection technique for android devices using machine learning algorithms," in *2024 International Conference on Computing, Sciences and Communications (ICCSC)*, pp. 1–6, IEEE, 2024.

[14] Q. Liu, "Feature selection with random forest for ransomware detection," tech. rep., Institute for Homeland Security, 2025.

[15] B. Idoko, F. Ogwueleka, and S. Bassey, "Systematic literature review on malware detection and machine learning algorithms: Identifying gaps for possible remedies.," *International Journal of Computers*, vol. 10, 2025.

[16] N. Alyemni and M. Frikha, "Exploring machine learning in malware analysis: Current trends and future perspectives.," *International Journal of Advanced Computer Science & Applications*, vol. 16, no. 1, 2025.

[17] K. E. Ketebu *et al.*, "A recent survey of image-based malware classification using convolution neural network," *Journal of Autonomous Intelligence*, vol. 7, no. 5, pp. 1–12, 2024.

[18] A. Kumar, S. Gupta, and R. Sharma, "A structured framework for reproducible cybersecurity research," *IEEE Transactions on Information Forensics and Security*, vol. 20, no. 1, pp. 50–62, 2025.

[19] M. Zhang, Y. Chen, and L. Wang, "Static feature analysis for portable executable files in malware detection," *Computers & Security*, vol. 142, p. 103115, 2025.

[20] J. Lee and H. Kim, "Addressing class imbalance in malware detection: A comprehensive survey," *Journal of Cybersecurity*, vol. 11, no. 2, pp. 15–28, 2025.

[21] Y. Zhao and Y. Liu, "Malware detection based on optimized deep learning in data-driven mode," 2024.

[22] S. Patel and R. Jain, "Advanced techniques for handling missing data in cybersecurity datasets," *IEEE Access*, vol. 13, pp. 1000–1012, 2025.

[23] F. Ahmad and N. Singh, "Normalization strategies for machine learning-based malware detection," *Journal of Machine Learning Research*, vol. 26, no. 3, pp. 200–215, 2025.

[24] K. Liu, T. Zhao, and Y. Li, "Feature selection for high-dimensional malware datasets using anova f-test," *Information Sciences*, vol. 650, pp. 119–130, 2025.

[25] A. G. Akintola, A. O. Balogun, H. A. Mojeed, F. Usman-Hamza, S. A. Salihu, K. S. Adewole, G. B. Balogun, and P. O. Sadiku, "Performance analysis of machine learning methods with class imbalance problem in android malware detection," *International journal of interactive mobile technologies*, vol. 16, pp. 140–162, 2022.

[26] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE communications surveys & tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.