Cyber Security and Applications 3 (2025) 100076

Contents lists available at ScienceDirect

Cyber Security and Applications

journal homepage: http://www.keaipublishing.com/en/journals/cyber-security-and-applications/

Towards superior android ransomware detection: An ensemble machine learning perspective

Md. Alamgir Hossain[a,*], Tahmid Hasan[b], Fahad Ahmed[c,**], Sheikh Hasib Cheragee[a], Muntasir Hasan Kanchan[a], Md Alimul Haque[d]

[a] Department of Computer Science and Engineering, State University of Bangladesh, South Purbachal, Kanchan, Dhaka 1461, Bangladesh
[b] Department of Computer Science and Engineering, Prime University, Dhaka 1216, Bangladesh
[c] Department of Computer Science and Engineering, University of Asia Pacific, Dhaka 1205, Bangladesh
[d] Department of Computer Science, Veer Kunwar Singh University, Ara, Bihar, India

ARTICLE INFO

Keywords:
Android ransomware detection
Ensemble machine learning
Network traffic analysis
Mobile security solutions
Cybersecurity

ABSTRACT

Ransomware remains a pervasive threat to Android devices, with its ability to encrypt critical data and demand ransoms causing significant disruptions to users and organizations alike. This research proposes a novel ensemble-based machine learning approach for the detection of Android ransomware, leveraging the strengths of multiple classifiers to enhance detection accuracy and robustness. Utilizing a comprehensive dataset comprising 203,556 network traffic records across 10 distinct ransomware types and benign traffic, we meticulously preprocess and feature-engineer the data to ensure optimal model performance. The methodology integrates various ensemble classifiers, evaluating each through rigorous cross-validation. Feature importance analysis using Random Forest identifies key indicators of ransomware activity, enabling us to refine our models and focus on the most predictive features. The results demonstrate that the ensemble models, particularly Bagging, achieve near-perfect detection rates, with precision, recall, and F1 scores consistently exceeding 99% for different binary attacks and multi-class classification. Finally, in-depth statistical analysis further validates the superiority of our approach, showcasing significant improvements over traditional machine learning methods. This research sets a new benchmark for Android ransomware detection, offering a robust, scalable, and highly accurate solution that enhances the security and resilience of mobile networks against evolving cyber threats.

1. Introduction

The proliferation of ransomware targeting Android devices has emerged as a significant cybersecurity threat, with devastating consequences for individuals and organizations [1]. Specific types of Android ransomware, such as SVpeng, PornDroid, Koler, RansomBO, Charger, Simplocker, WannaLocker, Jisut, Lockerpin, and Pletor, encrypt sensitive data on infected devices and demand payment for its release. These ransomware variants operate by infiltrating the system, often through malicious downloads or compromised applications, and then encrypting user files, rendering them inaccessible. The negative impacts are multifaceted, including the loss of critical data, significant financial losses due to ransom payments, disruption of business operations, and erosion of user trust in digital security [2,3].

Addressing the threat of Android ransomware through dedicated research is imperative for several reasons. First, the widespread use of mobile devices for personal and professional purposes increases the potential impact of security breaches. By developing advanced detection mechanisms, it is possible to prevent data loss, protect financial assets, and maintain the integrity of business operations. Furthermore, an effective ransomware detection system enhances user trust in mobile technology and online security, encouraging the adoption of digital solutions [4,5]. Research-driven advancements in this field can also provide insights into evolving ransomware tactics, enabling the development of proactive measures to counteract future threats, and ultimately contributing to a safer digital environment [6].

Detecting and mitigating Android ransomware is particularly challenging due to several factors. First, ransomware continuously evolves,

Peer review under responsibility of KeAi Communications Co., Ltd.
* Corresponding author at: State University of Bangladesh, South Purbachal, Kanchan, Dhaka 1461, Bangladesh.
** Corresponding author at: Department of Computer Science and Engineering, University of Asia Pacific, 74/A, Green Road, Dhaka 1205, Bangladesh.
E-mail addresses: alamgir.cse14.just@gmail.com (Md.A. Hossain), tahmidhasan.cse@gmail.com (T. Hasan), fahadahmed@uap-bd.edu (F. Ahmed), hasibcheragi@gmail.com (S.H. Cheragee), muntasirhasank@gmail.com (M.H. Kanchan), shadvksu@gmail.com (M.A. Haque).

https://doi.org/10.1016/j.csa.2024.100076
Received 27 July 2024; Received in revised form 15 August 2024; Accepted 20 October 2024
Available online 21 October 2024

with new variants employing increasingly sophisticated evasion techniques that can bypass traditional security measures [7,8]. The sheer diversity of ransomware types, each with unique behaviors and encryption methodologies, complicates the creation of a one-size-fits-all detection solution. Additionally, the constrained resources and heterogeneous nature of mobile devices pose significant limitations on implementing comprehensive security measures. The dynamic nature of network traffic and the subtle indicators of ransomware activity further exacerbate the difficulty, requiring advanced analytical techniques to identify and respond to threats in real-time without impacting device performance [9,10].

Our approach leverages the strengths of ensemble-based machine learning to enhance the detection accuracy and robustness of Android ransomware detection, surpassing the capabilities of traditional methods. By applying ensemble classifiers with hyperparameter tuning, including Random Forest, Bagging, AdaBoost, Gradient Boosting, Extra Trees, XGBoost, and CatBoost, we achieve a comprehensive and adaptive detection mechanism. The proposed approach includes meticulous data preprocessing and feature engineering, handling missing values, normalizing numerical features, and encoding categorical variables. Rigorous feature importance analysis using Random Forest identifies the most predictive indicators of ransomware activity, refining our models and enhancing detection accuracy. By employing a variety of ensemble classifiers, our approach captures complex patterns and subtle anomalies in network traffic that single-model approaches often miss. Robust cross-validation techniques ensure the generalizability and reliability of our models, while in-depth statistical analysis, including paired *t*-tests, further validates the superiority of our approach. This combination of advanced preprocessing, feature selection, ensemble learning, and thorough validation provides a robust, scalable, and adaptive solution for detecting Android ransomware. Major contributions of this research are listed below:

- We study present and traditional approaches, identifying their limitations and strengths to inform and refine our methodology.
- We develop a novel ensemble-based machine learning approach, integrating multiple classifiers to enhance the detection of Android ransomware.
- We perform rigorous feature importance analysis using Random Forest to identify the most predictive indicators of ransomware activity.
- We conduct in-depth statistical analysis, including paired *t*-tests, to demonstrate the superiority of the Bagging ensemble over others.
- Our comprehensive framework provides a scalable and adaptive solution for real-time Android ransomware detection for both binary and multiclass classification with proper hyperparameter tuning.

The next section provides an analysis of related works, followed by a detailed description of the proposed methodology. Afterward, we present the results and discussion. Finally, the paper concludes with a summary and references.

2. Related works

In this section, the recent related papers are studied to provide a comprehensive overview of the current state of research in the field of Android ransomware detection using machine learning techniques.

With the exponential growth in Android device usage, the platform has become a lucrative target for cybercriminals. Among various malware, ransomware poses a significant threat by encrypting victim's data and demanding a ransom for its release [11]. Early detection and prevention are crucial to mitigate its impact. Existing detection approaches can be broadly classified into signature-based, behavior-based, and machine-learning-based methods. Signature-based detection relies on known malware signatures to identify threats. While effective against known ransomware, it fails against novel or polymorphic variants [12]. Novel variants refer to newly developed ransomware types that have not been previously encountered. Polymorphic variants are ransomware types that change their code slightly each time they infect a new system, making them difficult to detect with static signature-based methods [13]. Consequently, this approach often results in high false-negative rates, as it cannot recognize new or altered ransomware signatures. For instance, the studies by Singh and Tripathy [11] and Abiola and Marhusin [12] highlight the limitations of signature-based methods in detecting new and polymorphic ransomware variants. Zhou et al. [14] developed DroidRanger, which leverages permission-based heuristics and malware signatures to detect known malware, including ransomware. Rastogi et al. [15] introduced PERMIS, focusing on permission-based signature matching for malware detection. These approaches are efficient for known threats and have low computational overhead. But ineffective against new and evolving ransomware, also provides high false-negative rates.

Behavior-based detection analyzes the behavior of applications to identify anomalies indicative of ransomware. Canfora et al. [16] proposed a dynamic analysis approach using system call monitoring to detect ransomware behavior in real-time. Meng et al. [17] developed a hybrid method combining static and dynamic analysis, significantly improving detection accuracy for behavior-based threats. These approaches have high computational overhead and, the potential for high false-positive rates [18,19].

Machine learning (ML) has emerged as a powerful tool for detecting ransomware by learning patterns from large datasets. Especially the ensemble-based ML approaches provide high accuracy, adaptability to new threats, and the ability to handle large datasets [20]. An analysis of the recent machine learning-based approaches in Android ransomware detection is presented in Table 1, highlighting key aspects such as major contributions, classification methods, types of ransomware, datasets used, methodologies employed, results achieved, strengths, and limitations of each approach.

The literature reveals significant advancements in the detection of Android ransomware through various machine learning and deep learning techniques. Despite these advancements, several limitations persist across the studies. Most classifiers provide better results in binary classification but struggle with multi-class classification, which is crucial for distinguishing between different ransomware types. Furthermore, the true positive rate for Android ransomware classification should be improved to ensure more reliable detection. Our approach addresses these limitations by implementing an ensemble-based machine learning framework that excels in both binary and multi-class classification. By integrating advanced feature selection, utilizing multiple classifiers, and leveraging comprehensive datasets, our method enhances detection accuracy and robustness. This strategy not only improves detection rates but also reduces false positives and negatives, ensuring a more reliable and efficient ransomware detection system.

3. Methodology

This section provides the detailed methodology, covering everything from the dataset to the approach of statistical analysis. Fig. 1 shows the complete pipeline of the proposed approach.

3.1. Dataset

The dataset [31] used in this research comprises 203,556 network traffic records from Android devices, including both benign traffic and ten distinct types of ransomware: SVpeng, PornDroid, Koler, RansomBO, Charger, Simplocker, WannaLocker, Jisut, Lockerpin, and Pletor. Each record consists of 85 features, capturing various aspects of network behavior, such as flow duration, packet length, and inter-arrival times. Table 2 details the amount of data and the percentage representation for each class, highlighting the distribution of benign and ransomware traffic within the dataset.

Table 1
Evaluating cutting-edge machine learning techniques for android ransomware detection.

Reference	Major Contribution	Classification	Ransomware Types	Dataset	Method	Results	Strengths	Limitations
Farhan and Salman [21]	Deep learning for Android ransomware detection with a focus on Feedforward Neural Networks (FNNs)	Binary	Simplocker, LockerPin, DoubleLocker, Police-themed ransomware, Fusob	AndroZoo (benign), RansomProber (malicious)	FNN using Keras Sequential with 3 densely connected layers	Achieved 98.9% accurac, F1-score of 0.662, precision of 0.5, and recall of 1.0	Demonstrated high accuracy and recall using deep learning; automated feature extraction and learning from raw data;	Limited to a simple FNN; low precision indicating high false positives; lack of exploration into more advanced architectures like CNNs or RNNs;
Ahmed et al. [22]	Utilized ML and DL techniques for efficient, precise, and robust models for binary classification of ransomware.	Binary	10 different Android ransomware types	Android_Ransomware	Decision Tree, SVM, KNN, ensemble (DT, SVM, KNN), FNN, TabNet.	DT achieved 97.24% accuracy, 98.50% precision, 98.45% F1-score;	Comprehensive analysis using both ML and DL techniques; Use of recent dataset;	Ensemble and DL models not sufficiently explored, lacking comparison with other advanced DL models; limited to binary classification, which does not address multi-class detection complexities;
Sharma et al. [23]	Ensemble-based supervised machine learning utilizing novel features, dimensionality reduction, and comparative analysis on GPU and CPU for computational time.	Binary	Locker and Crypto Ransomware	RansomProber (2721 ransomware samples) and AndroZoo (2000 benign samples)	PCA for dimensionality reduction and Random Forest	99.67% accuracy	Comprehensive feature extraction, use of PCA for dimensionality reduction, comparative analysis of machine learning models on both GPU and CPU, and high accuracy in detection.	Limited to binary classification (ransomware vs. benign), does not address real-time detection, potential overfitting due to high accuracy claims, and the computational time analysis might not be generalizable to all real-world scenarios.
Oneil Victoriano [24]	ML approach using various classifiers	Binary and Multiclass	HelDroid (various types)	HelDroid	Decision Tree, Random Forest, Gradient Boosting, and AdaBoost; evaluated using cross-validation on training and test sets.	Accuracy of 98.05% on both training and test sets; 99.08% accuracy with Decision Tree on transformed dataset.	Utilized multiple classifiers for a comprehensive evaluation; effective feature selection and transformation; high accuracy achieved; provided insights into feature importance.	Overfitting observed with Naive Bayes classifiers; did not address scalability issues; limited to the dataset used; did not explore other types of classifiers or hybrid approaches for comparison.
Faris et al. [25]	Introduced an optimized extreme learning machine using Salp Swarm for ransomware detection.	Binary classification (benign vs. ransomware)	CryptoLocker, WannaCry, Petya, etc.	1000 Android apps with both benign and ransomware samples	Combined Salp Swarm optimization with Kernel Extreme Learning Machine	Achieved 98.0% accuracy, 98.0% precision, and 98.0% recall	High accuracy and low false positive rate	Requires careful tuning of hyperparameters and may be computationally intensive.
Hiba Zuhair [26]	Hybrid cluster traits for smartphone security.	Multi-class, Binary	Genric.17.1762, LockDroid, Koler, Pletor, Simplocker,	Hel-Droid, Virus Total, Google Play Store, APK Tool	Hybrid analysis combining static and dynamic analysis using ML and DL.	96.50% accuracy for DNN.	Holistic evaluation, hybrid trait clustering.	High resource usage, time-sensitive, complexity in managing diverse ransomware traits, limited to known traits, high FPR.

(*continued on next page*)

Table 1 (*continued*)

Reference	Major Contribution	Classification	Ransomware Types	Dataset	Method	Results	Strengths	Limitations
Bagui and Woods [27]	Improved detection of Android ransomware using network traffic data.	Binary	Charger, Jisut, Koler, Lockerpin, Pletor, PornDroid, RansomBo, SVPeng, Simplocker, WannaLocker	CICAndMal2017	ML classifiers (DT, NB, OneR) and Information Gain for feature selection.	DT: Accuracy 99.67%, Precision 99.68%, Recall 99.67%, F1-Score 99.67%;	Effective feature selection; Improved classification results;	Limited to Binary Classification; May not generalize well to new or evolving ransomware families; Does not incorporate recent advancements in ransomware detection techniques.
Samah and Iman [28]	Proposed API-RDS, a static analysis system for Android ransomware detection using API calls.	Binary	Not specified	Custom dataset with 2959 ransomware and 500 benign samples.	API-based static analysis	Accuracy: 97%, Complexity reduction: 26%	Feature reduction improves performance;	Limited to static analysis, might miss dynamic behaviors;
Masum et al. [29]	Developed a feature selection-based framework using multiple ML classifiers for efficient and effective detection of Android ransomware.	Binary	Locker, Crypto	Custom dataset with 138,047 samples.	DT, RF, NB, LR, NN	RF: Accuracy: 99±0.01, F-beta: 0.97±0.03, Recall: 0.97±0.03, Precision: 0.99±0.00	Comprehensive feature selection process, robust evaluation of multiple classifiers, high accuracy and precision achieved with RF.	Potential overfitting to a specific dataset; computationally intensive; may not adapt well to new ransomware variants.
Arabo et al. [30]	Developed a detection mechanism using process behavior analysis to identify ransomware.	Binary	ViraLock, WannaCry, Cerber, WinLocker	Custom dataset with 7 ransomware.	Process behavior analysis, machine learning.	Accuracy: 98.9%, False Positive Rate: 1.5%, False Negative Rate: 2.6%	High accuracy in distinguishing between ransomware and benign applications.	Very limited sample size, focuses on specific ransomware types, potentially high computational cost due to detailed process monitorin.

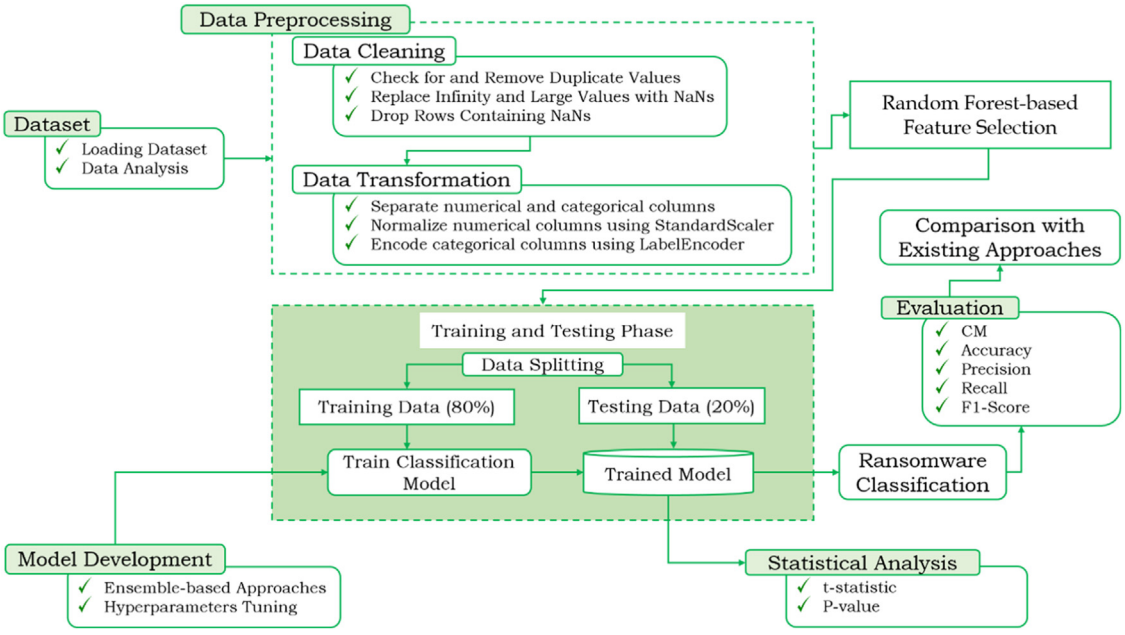

Fig. 1. Pipeline of the proposed approach.

Table 2
Amount of data for each class.

Ransomware Category	Amount of Data	Percentage of each Class
Benign	43,091	11.0%
Charger	39,551	10.1%
Jisut	25,672	6.5%
Koler	44,555	11.4%
Lockerpin	25,307	6.5%
Pletor	4715	1.2%
PornDroid	46,082	11.8%
RansomBo	39,859	10.2%
SVpeng	54,161	13.8%
Simplocker	36,340	9.3%
WannaLocker	32,701	8.3%

3.2. Data preprocessing

The data preprocessing steps are crucial for preparing the dataset for effective machine learning model training [32]. The following steps outline the preprocessing techniques applied to the dataset:

1. Duplicate entries in the dataset were identified using the *'duplicated()'* method and subsequently removed to ensure data integrity.
2. Infinite values and excessively large numeric values were replaced with NaNs to maintain data consistency. Rows containing NaN values were dropped to ensure the dataset's completeness and reliability.
3. The dataset is then divided into numerical and categorical columns for appropriate preprocessing. Numerical features were normalized using the *StandardScaler* given in Eq. (1) to ensure that all features contribute equally to the model.

$$X_{scaled} = \frac{X - \mu}{\sigma} \tag{1}$$

Where μ is the mean and σ is the standard deviation of the numerical features.

4. Categorical features are encoded using *LabelEncoder* to convert them into numerical format suitable for machine learning algorithms.

These preprocessing steps ensure that the dataset is clean, consistent, and ready for the subsequent stages of feature selection and model training.

3.3. Feature selection

Feature selection is a critical step in this research, aimed at identifying the most predictive features to enhance the accuracy and efficiency of the machine learning models. By focusing on the most relevant features, we can reduce the complexity of the model, improve performance, and avoid overfitting [33].

We employed the Random Forest classifier for feature selection due to its effectiveness in estimating the importance of various features in the dataset. Random Forest was chosen over other methods because it is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes or mean prediction of the individual trees. This approach is highly effective in handling large datasets with high-dimensional feature spaces. Additionally, Random Forest inherently provides measures of feature importance based on the contribution of each feature to the accuracy of the model, making it particularly suitable for identifying the most significant features in our dataset. At first, the Random Forest classifier is trained using the dataset to evaluate the importance of each feature. The classifier consists of 100 decision trees (n_estimators = 100), each trained on random subsets of the data and features, with a fixed random state (random_state = 42). Next, feature importance is computed based on the average decrease in impurity across all trees. The importance score for a feature f_j is given by Eq. (2).

$$I(f_j) = \frac{1}{T}\sum_{t=1}^{T} \Delta Impurity(f_j,\ t) \tag{2}$$

Where, $I(f_j)$ is the importance of feature f_j, T is the total number of trees, and $\Delta Impurity(f_j,\ t)$ is the decrease in impurity in tree t due to feature f_j.

Then, the features are ranked based on their importance scores. The top k features with the highest scores are selected. The selection criterion is: Top k features $= \{f_j : I(f_j) \geq threshold\}$

Finally, the names and importance scores of the selected features are extracted for further analysis mentioned in **Table 3**. This step ensures that only the most significant features are used in model training, thereby optimizing the model's performance.

3.4. Classifiers

The ensemble-based classifiers used in this research are Bagging, ExtraTrees, RandomForest, AdaBoost, GradientBoosting, XGBoost, and CatBoost. The working processes of these classifiers, including their training and prediction mechanisms, are described below.

3.4.1. Bagging

Bagging, or Bootstrap Aggregating, is an ensemble method designed to improve the stability and accuracy of machine learning algorithms by training multiple models on different subsets of the training data [34]. It reduces variance and helps avoid overfitting. The classification and prediction process of Bagging is displayed in Fig. 2.

Bagging involves creating multiple bootstrap samples from the original training set. A bootstrap sample is a randomly selected subset of the data, created by sampling with replacement. This means that some data points may appear multiple times in a single bootstrap sample, while others may not appear at all. Each bootstrap sample is then used to train a base estimator, like a decision tree, on each subset of the data. It creates multiple bootstrap samples D_i from the original training set D. A base estimator h_i is then trained on each bootstrap sample D_i like the below Eq. (3).

$$h_i = Train\ (D_i) \tag{3}$$

Where, i = 1, 2, …, M. For prediction, the outputs from all base estimators are aggregated by majority voting for classification:

$$\hat{y} = majority\_vote\{h_i(x)\}_{i=1}^{M} \tag{4}$$

Where, $\hat{y}$ is the final prediction, M is the number of base estimators, and $h_i(x)$ is the prediction from the i^{th} estimator.

By training each model on a different subset of the data and averaging their predictions, Bagging reduces the overall variance and improves the robustness and accuracy of the model, making it highly effective for the classification task of Android Ransomware Detection.

3.4.2. Extra trees

Extra Trees (Extremely Randomized Trees) classifier builds multiple decision trees with random splits for each node, enhancing the diversity of the ensemble [35]. In the training phase, for each tree, a random subset of features is chosen, and each node is split using the best random split among these features: h_i = Train (D, random_splits), i = 1, 2, …, M.

For prediction, the final output is obtained by averaging the predictions of all trees:

$$\hat{y} = \frac{1}{M}\sum_{i=1}^{M} h_i(x) \tag{5}$$

Where, $\hat{y}$ is the final prediction, M is the number of base estimators, and $h_i(x)$ is the prediction from the i^{th} tree.

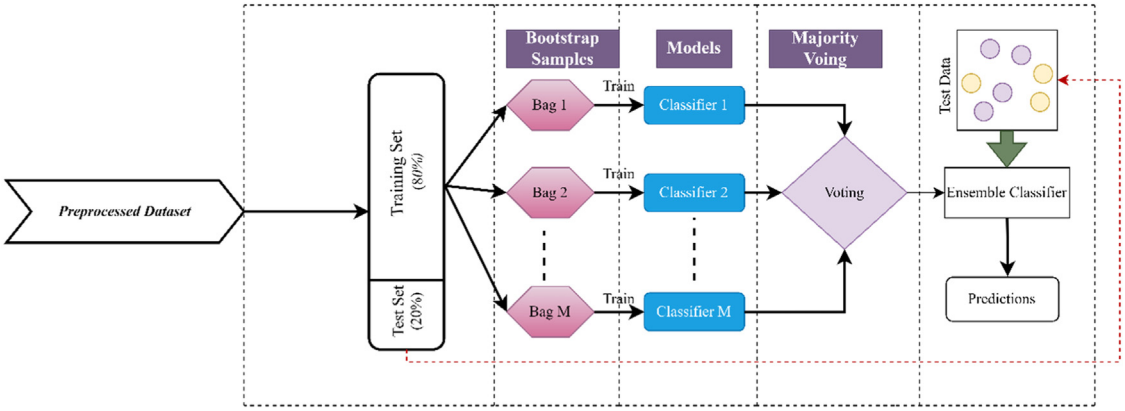

Fig. 2. Classification and prediction process of bagging ensemble.

3.4.3. XGBoost

XGBoost (Extreme Gradient Boosting) is an advanced implementation of gradient boosting that includes regularization to avoid overfitting. It is highly efficient and scalable, making it suitable for large datasets [36]. XGBoost involves training a sequence of models to correct the errors of the combined model. Additionally, XGBoost incorporates regularization terms to penalize model complexity using the Eq. (6).

$$L(\theta) = \sum_{i=1}^{n} l(y_i,\ \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{6}$$

Where, $L(\theta)$ is the regularized objective, $l(y_i,\ \hat{y}_i)$ is the loss function, and $\Omega(\mathrm{f_k})$ is the regularization term defined as $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|w\|^2$.

The final prediction is the sum of the predictions from all trees:

$$\hat{y} = \sum_{m=0}^{M} \eta h_m(x) \tag{7}$$

Where, $\hat{y}$ s the final prediction, M is the number of trees, and $h_m(x)$ is the prediction from the m^{th} tree. XGBoost builds a robust model that minimizes the loss function and reduces overfitting, making it highly effective for the classification task of Android Ransomware Detection.

3.4.4. Random forest

The Random Forest is an ensemble-based machine learning method that combines multiple decision trees to improve classification performance. By training each tree on a different subset of the data using bootstrap sampling (random sampling with replacement), and combining their predictions, the Random Forest reduces overfitting and increases accuracy [37,38]. During the training process, multiple subsets D_i are created from the training set D using bootstrap sampling, and each subset D_i is used to train a decision tree T_i. At each split in a tree, a random subset of features F_i is selected, and the best feature from this subset is used for splitting. For prediction, each trained tree T_i in the forest provides a classification $h_i(x)$ for a new input x in Eq. (8). The final prediction is obtained by majority voting:

$$\hat{y}\cdot = mode\left(\{h_1(x), h_2(x), \ldots, h_M(x)\}\right) \tag{8}$$

Where, M is the total number of trees in the forest. This approach is particularly effective for Android Ransomware Detection, as it leverages the robustness of ensemble methods to improve detection accuracy.

3.4.5. AdaBoost

The AdaBoost (Adaptive Boosting) classifier is an ensemble method that combines the predictions of several weak learners, typically decision stumps, to create a strong classifier. AdaBoost works by sequentially training these weak learners, where each subsequent learner focuses more on the errors made by the previous learners [39].

During the training process, AdaBoost begins by assigning equal weights $w_i = \frac{1}{N}$ to each training sample $(x_i,\ y_i)$, where N is the total number of samples. For each iteration t a weak learner $h_t(x)$ is trained on the weighted training data, and the weighted error ϵ_t is calculated using Eq. (9).

$$\epsilon_t = \frac{\sum_{i=1}^{N} w_i.1\left(h_t(x_i)y_i\right)}{\sum_{i=1}^{N} w_i} \tag{9}$$

The weight of the learner α_t is computed with Eq. (10).

$$\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{10}$$

Subsequently, the weights of the training samples are updated according to Eq. (11) and the weights are normalized.

$$w_i \leftarrow w_i.exp\left(\alpha_t.1\left(h_t(x_i)y_i\right)\right) \tag{11}$$

For prediction, each trained weak learner h_t provides a classification $h_t(x)$ for a new input x. The final prediction $\hat{y}$ is obtained by a weighted majority vote as shown in Eq. (12).

$$\hat{y} = sign\left(\sum_{t=1}^{T} \alpha_t.h_t(x)\right) \tag{12}$$

Where, T is the total number of weak learners. This iterative process of focusing more on difficult cases allows AdaBoost to improve the overall accuracy and robustness of the model, making it suitable for the classification task of Android Ransomware Detection.

3.4.6. Gradient boost

The Gradient Boosting classifier is an ensemble method that builds a strong classifier by sequentially adding weak learners, typically decision trees, in a stage-wise manner [40]. During the training process, it starts with an initial model $F_0(x)$ that predicts a constant value, usually the mean of the target variable. For each iteration t, the negative gradient (pseudo-residuals) for the current model $F_{t-1}(x)$ is computed as shown in Eq. (13).

$$r_i^{(t)} = -\frac{\partial L\left(y_i,\ F(x_i)\right)}{\partial F(x_i)}|_{F(x)=F_{t-1}(x)} \tag{13}$$

A weak learner $h_t(x)$ is then trained to predict these residuals $r_i^{(t)}$. The optimal step size γ_t is determined by minimizing the loss function according to Eq. (14).

$$\gamma_t = arg\ \min_{\gamma} \sum_{i=1}^{N} L\left(y_i,\ F_{t-1}(x_i) + \gamma h_t(x_i)\right) \tag{14}$$

The model is updated as shown in Eq. (15).

$$F_t(x) = F_{t-1}(x) + \gamma_t h_t(x) \tag{15}$$

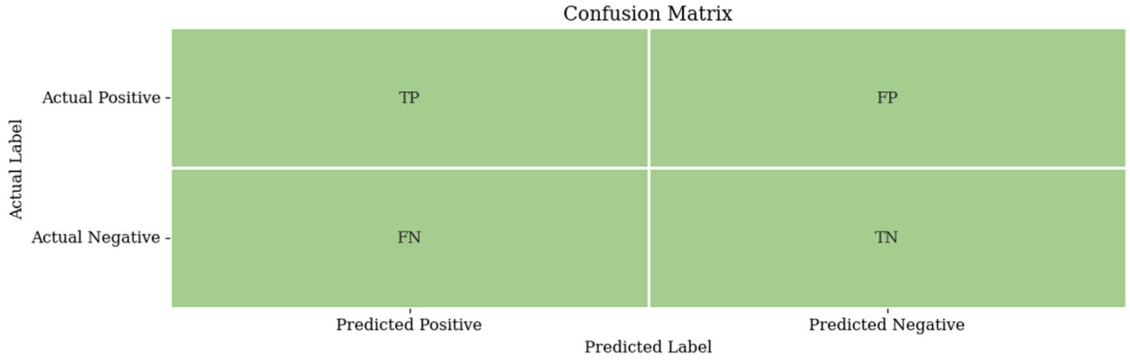

Fig. 3. Confusion matrix.

For prediction, the final model $F_T(x)$ is used to predict the output for a new input x as shown in Eq. (16).

$$\hat{y} = F_T(x) \tag{16}$$

Where, T is the total number of iterations. By focusing on the residuals (errors) at each stage, Gradient Boosting builds a robust model that minimizes the loss function, making it effective for the classification task of Android Ransomware Detection.

3.4.7. CatBoost

CatBoost (Categorical Boosting) is a gradient boosting algorithm that handles categorical features efficiently, reducing the need for extensive preprocessing. It is known for its robustness and accuracy in handling categorical data [41].

Similar to other gradient boosting methods, CatBoost trains a sequence of models, but it uses a more sophisticated way to process categorical features, improving accuracy and reducing overfitting using the Eq. (17).

$$h_t = Train\big(D,\ r_{t-1},\ categorical\_encoding\big) \tag{17}$$

The final prediction is the sum of the predictions from all trees:

$$\hat{y} = \sum_{t=0}^{T} \eta h_t(x) \tag{18}$$

Where, $\hat{y}$ is the final prediction, T is the number of trees, η is the learning rate, and $h_i(x)$ is the prediction from the i^{th} tree. By efficiently handling categorical features, CatBoost builds a robust model, making it effective for the classification task of Android Ransomware Detection.

3.5. Evaluation metrics

Evaluation metrics are essential for assessing the performance of a machine learning model, providing quantitative measures to compare models and determine their effectiveness. These metrics help in understanding how well the model generalizes to new, unseen data and its ability to make accurate predictions [42]. One key evaluation metric used in this research is the Confusion Matrix shown in Fig. 3. A Confusion Matrix is a table that is often used to describe the performance of a classification model by comparing actual labels to predicted labels. It consists of four types of values: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

True Positives and True Negatives are the correctly predicted positive and negative cases, respectively, while False Positives and False Negatives are the incorrectly predicted cases. The Confusion Matrix provides a detailed breakdown of the model's performance, highlighting the number of correct and incorrect predictions for each class. The other evaluation metrics generated from the Confusion Matrix used in this research, along with their brief descriptions and related equations, are presented in Table 3.

3.6. Statistical analysis for best classifier

Statistical analysis is an essential component of this research, providing a robust framework for evaluating the performance and significance of the proposed ensemble-based classifiers [43]. By employing statistical techniques, we ensure the reliability and validity of our findings, allowing us to draw meaningful conclusions about the effectiveness of our models in detecting Android ransomware.

The paired t-test is used to compare the performance of two classifiers across multiple folds of cross-validation [44]. It evaluates whether the mean difference in performance metrics (such as accuracy) between two classifiers is significantly different from zero. The Stratified K-Fold cross-validation with n_splits=5 has been used to ensure that each fold has the same proportion of classes as the original dataset. The performance metrics are calculated for each classifier with the below equation.

$$Metric = \frac{1}{K}\sum_{k=1}^{K} Metric_k \tag{19}$$

Where, K is the number of folds.

The paired t-test compares the mean performance of two classifiers. The t-statistic is calculated as:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \tag{20}$$

Where, $\bar{d}$ is the mean of the differences between paired observations. s_d is the standard deviation of the differences. n is the number of observations (folds). The *p-value* indicates the probability of obtaining the observed results under the null hypothesis (i.e., no difference between the classifiers).

We first perform cross-validation using the selected features for each classifier. Then, we compute the mean and standard deviation of each metric for each classifier. Finally, we conduct paired t-tests between each pair of classifiers for accuracy and other metrics. By rigorously applying these statistical methods, we can robustly evaluate and compare the performance of our ensemble-based classifiers, ensuring that our conclusions are statistically sound and reliable.

4. Results and analysis

This section presents the experimental results with proper figures and tables, showcasing the performance of the proposed ensemble-based classifiers in detecting Android ransomware.

The experiments for this research were conducted using Google Colab, leveraging its powerful cloud-based environment for machine learning and data analysis. We utilized several essential Python libraries: Pandas for data manipulation and preprocessing, NumPy for numerical operations, scikit-learn for implementing machine learning models and evaluation metrics, XGBoost for the XGBoost classifier, CatBoost for the CatBoost classifier, and Matplotlib for plotting and visualizing results. The process began with data loading and preprocessing, including

Table 3
Evaluation metrics of the proposed approach.

Metric Name	Description	Related Syntax/Equation
Accuracy	The ratio of correctly predicted instances to the total instances.	$\frac{TP+TN}{TP+FN+FP+TN}$
Precision	The ratio of correctly predicted positive observations to the total predicted positives.	$\frac{TP}{TP+FP}$
Recall	The ratio of correctly predicted positive observations to all the observations in the actual class.	$\frac{TP}{TP+FN}$
F1-Score	The weighted average of Precision and Recall, providing a balance between the two metrics.	$\frac{2*(recall*precision)}{recall+precsion}$

Table 4
Selected features and their importance scores with random forest.

Feature Name	Importance Scores	Description
Flow Packets/s	0.011622	Measures the number of packets transmitted per second within a network flow. Abnormal packet rates can indicate the presence of ransomware attempting to communicate with a command-and-control server.
Flow IAT Max	0.014946	The maximum inter-arrival time between packets can reveal significant delays that might occur during ransomware encryption or communication processes.
Flow Duration	0.015056	The total duration of a network flow can be indicative of ransomware activity, especially if the duration is unusually long or short compared to normal traffic.
Flow IAT Mean	0.015314	Average Inter-Arrival Time between packets, used to detect irregular traffic behavior.
Flow IAT Min	0.016908	Minimum Inter-Arrival Time, showing the shortest delay between packets which may reveal anomalies.
Source Port	0.020188	Port number of the source, with certain ports being more commonly associated with malicious traffic.
Destination IP	0.032051	IP address of the destination, which can identify known malicious servers.
Flow ID	0.072034	Unique identifier for each network flow, used to track specific flows that may be suspicious.
Source IP	0.141092	IP address of the source, important for identifying the origin of potential ransomware attacks.
Timestamp	0.427583	Timestamps are critical for correlating network events over time, helping to identify patterns and timelines that are characteristic of ransomware attacks.

Table 5
Evaluation metrics for various ensemble classifiers for multi-class classification both for before and after feature selection.

Classifier	Accuracy		Precision Macro		Recall Macro		F1 Macro	
	Before	After	Before	After	Before	After	Before	After
Bagging	0.99808 ± 0.00012	0.99818 ± 0.00018	0.99700 ± 0.00022	0.99722 ± 0.00029	0.99687 ± 0.00029	0.99744 ± 0.00047	0.99694 ± 0.00021	0.99733 ± 0.00035
XGBoost	0.97023 ± 0.00353	0.99468 ± 0.00031	0.97031 ± 0.00431	0.99315 ± 0.00082	0.94155 ± 0.00392	0.99179 ± 0.00065	0.95258 ± 0.00435	0.99246 ± 0.00059
Extra Trees	0.87664 ± 0.00344	0.99405 ± 0.00034	0.86899 ± 0.00371	0.99270 ± 0.00051	0.85905 ± 0.00389	0.99066 ± 0.00069	0.86315 ± 0.00380	0.99166 ± 0.00049
Random Forest	0.94786 ± 0.00357	0.99756 ± 0.00018	0.94192 ± 0.00466	0.99668 ± 0.00018	0.92924 ± 0.00491	0.99531 ± 0.00067	0.93479 ± 0.00487	0.99598 ± 0.00043
AdaBoost	0.57985 ± 0.00054	0.54897 ± 0.00058	0.35187 ± 0.00057	0.37624 ± 0.00028	0.46239 ± 0.00040	0.46069 ± 0.00057	0.38810 ± 0.00039	0.39699 ± 0.00037
Gradient Boost	0.93353 ± 0.00124	0.99769 ± 0.00016	0.91755 ± 0.01029	0.99686 ± 0.00048	0.90389 ± 0.00311	0.99642 ± 0.00024	0.90732 ± 0.00361	0.99664 ± 0.00032
CatBoost	0.98240 ± 0.00270	0.99461 ± 0.00040	0.97827 ± 0.00238	0.99364 ± 0.00037	0.97383 ± 0.00370	0.99202 ± 0.00114	0.97589 ± 0.00294	0.99281 ± 0.00066

handling missing values, normalizing numerical features with StandardScaler, and encoding categorical features with LabelEncoder. Feature selection was performed using a RandomForestClassifier to compute feature importances, selecting the top 10 features. Multiple ensemble classifiers—Bagging, ExtraTrees, RandomForest, AdaBoost, GradientBoosting, XGBoost, and CatBoost—were trained and evaluated using Stratified K-Fold cross-validation, with performance metrics (accuracy, precision, recall, and F1-score) calculated for each. Finally, paired *t*-tests were conducted to compare the performance of different classifiers, ensuring the statistical significance of the results. This rigorous and systematic evaluation, facilitated by Google Colab and the extensive capabilities of the utilized libraries, ensured the robustness and reliability of our findings.

Table 4 presents the selected features and their importance scores as identified by the Random Forest classifier. These features were chosen based on their ability to significantly contribute to the detection of Android ransomware. The importance scores indicate the relative contribution of each feature to the model's predictions.

Table 5 summarizes the performance metrics for each classifier, both before and after feature selection, using five-fold cross-validation. The results indicate that Bagging shows the best results in both scenarios, demonstrating its robustness and high performance with both the original 85 features and the selected 10 features. Each metric is presented with its mean and standard deviation to provide a comprehensive measure of the classifiers' performance, showcasing not only their average effectiveness but also the consistency and reliability of their predictions across multiple cross-validation folds.

Fig. 4 illustrates the confusion matrix for the Bagging ensemble classifier applied to multi-class classification of Android ransomware. The confusion matrix provides a detailed breakdown of the classifier's performance by showing the number of true positive, false positive, and false negative predictions for each class. The high number of true positives for each class (diagonal elements) reflects the model's high accuracy in detecting various ransomware types and benign traffic. The minimal number of off-diagonal elements signifies that the classifier maintains low misclassification rates, further indicating its robustness and reliability in real-world scenarios.

Overall, the confusion matrix highlights the Bagging ensemble's effectiveness in accurately detecting multiple types of Android ransomware, showcasing its robustness, high accuracy, and low misclassification rates. This level of performance is crucial for deploying reliable cybersecurity measures against the diverse and evolving threat landscape of Android ransomware.

Table 6 presents the detection results for each type of ransomware, as well as benign traffic, using the Bagging ensemble classifier. The near-perfect scores across all evaluation metrics for most classes highlight the Bagging ensemble's exceptional ability to detect and classify different types of Android ransomware with high accuracy and reliability. The

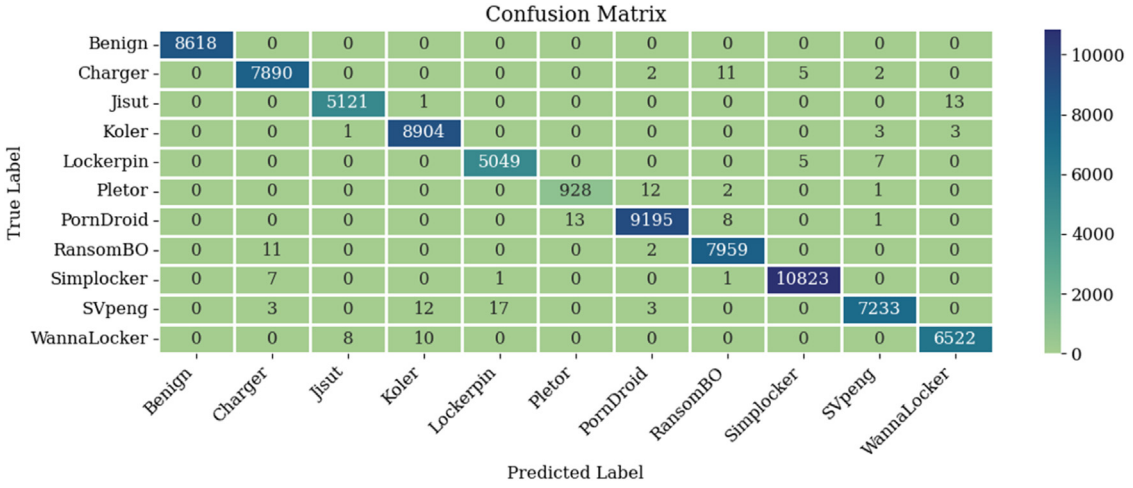

Fig. 4. Confusion Matrix of Bagging Ensemble for Multi-class Classification.

Table 6
Detection results for each type of ransomware with the bagging ensemble.

Category with Encoding Value	Precision	Recall	F1-Score	Support
SVpeng (8)	1.00	1.00	1.00	10,832
PornDroid (6)	1.00	1.00	1.00	9217
Koler (3)	1.00	1.00	1.00	8911
Benign (0)	1.00	1.00	1.00	8618
RansomBO (7)	1.00	1.00	1.00	7972
Charger (1)	1.00	1.00	1.00	7910
Simplocker (9)	1.00	1.00	1.00	7268
WannaLocker (10)	1.00	1.00	1.00	6540
Jisut (2)	1.00	1.00	1.00	5135
Lockerpin (4)	1.00	1.00	1.00	5061
Pletor (5)	0.99	0.98	0.99	943

Fig. 5. ROC curve for multiclass ransomware classification.

slight variation in the Pletor class performance does not significantly detract from the overall robustness of the model, making it highly effective for practical applications in ransomware detection. This consistent performance across various types of ransomware and benign traffic further validates the efficacy of the proposed approach in real-world cybersecurity scenarios.

Fig. 5 illustrates the ROC curves for the multiclass classification task, showcasing the performance of the proposed approach across various classes. Each curve represents a class, with the Area Under the Curve (AUC) values provided for each. The AUC values are exceptionally high, with most classes achieving an AUC close to 1.0000, indicating near-perfect classification performance. The curves show that the true positive rates are high and false positive rates are low, further confirming the model's effectiveness.

The near-perfect AUC values across all classes demonstrate the proposed model's exceptional ability to distinguish between different types of ransomware and benign traffic.

Table 7 presents the evaluation metrics for a Bagging Ensemble classifier applied to detect various types of Android ransomware, alongside benign samples. The metrics reported are Accuracy, Precision,

Table 7
Evaluation metrics for bagging ensemble for binary classification with the selected features.

Type	Accuracy	Precision	Recall	F1 -Score
SVpeng	0.99998	0.99998	0.99999	0.99998
PornDroid	0.99998	0.99998	0.99998	0.99998
Koler	1.00000	1.00000	1.00000	1.00000
RansomBO	1.00000	1.00000	1.00000	1.00000
Charger	0.99998	0.99998	0.99998	0.99998
Simplocker	1.00000	1.00000	1.00000	1.00000
WannaLocker	1.00000	1.00000	1.00000	1.00000
Jisut	1.00000	1.00000	1.00000	1.00000
Lockerpin	0.99998	0.99998	0.99999	0.99998
Pletor	0.99995	0.99997	0.99978	0.99988

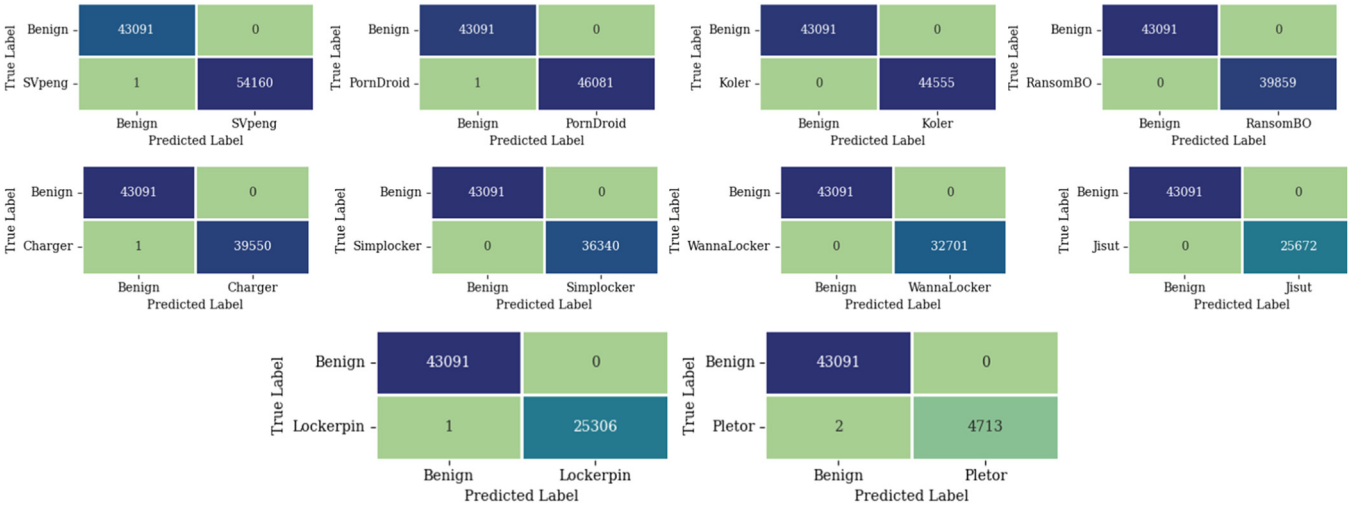

Fig. 6. Confusion matrix of bagging ensemble for binary classification.

Recall, and F1-Score for each ransomware type. The proposed approach of using a Bagging Ensemble classifier for binary classification of ransomware detection exhibits exceptional performance across all metrics. This demonstrates the classifier's high effectiveness and reliability in identifying ransomware and benign samples with minimal errors.

Fig. 6 displays the confusion matrices for the Bagging Ensemble classifier applied to binary classification of various Android ransomware types. The confusion matrices indicate near-perfect classification performance, with almost all benign and ransomware samples correctly identified. The minimal misclassification errors, shown as small numbers of false positives and false negatives, confirm the high effectiveness and reliability of the Bagging Ensemble classifier in detecting Android ransomware.

Table 8 presents the statistical analysis results for various ensemble classifiers, comparing pairs based on accuracy, precision macro, recall macro, and F1 macro using paired *t*-tests. The table lists the t-statistics and p-values for each metric, indicating the significance of differences between classifiers. For example, the comparison between Bagging and XGBoost shows a significant difference in all metrics (accuracy, precision, recall, and F1-score) with p-values close to 0, indicating that Bagging performs significantly better. Similarly, Bagging outperforms Extra Trees, Random Forest, AdaBoost, Gradient Boosting, and CatBoost with highly significant t-statistics and p-values. The consistent and superior t-statistics for Bagging across all comparisons confirm its robustness and efficacy in the classification task.

The statistical analysis highlights that Bagging provides better results than other classifiers due to its ability to aggregate multiple models to reduce variance and avoid overfitting. The significant t-statistics and low p-values across all performance metrics demonstrate that Bagging consistently outperforms other classifiers. This robust performance can be attributed to Bagging's ensemble method, which enhances model stability and accuracy by training multiple base models on different subsets of the data and combining their predictions. Consequently, Bagging proves to be the most effective classifier for the detection of Android ransomware, as evidenced by its superior statistical metrics.

Table 9 compares the performance of the proposed approach with an existing approach [22] in terms of accuracy, precision, recall, and F1-score. The proposed approach, evaluated on both binary classification and 11-class classification, demonstrates superior performance with perfect scores (100.00%) in all metrics for binary classification and very high scores for the 11-class classification (accuracy of 99.82%, precision of 99.72%, recall of 99.74%, and F1-score of 99.73%). In contrast, the existing approach shows lower performance metrics, with an accuracy of 97.24%, precision of 98.50%, recall of 98.40%, and F1-score of 98.45% for binary classification. This comparison highlights the significant improvements offered by the proposed approach in accurately detecting and classifying ransomware.

The proposed method using Bagging ensemble classifier demonstrates substantial improvements over existing methods. For instance, the accuracy, precision, recall, and F1 scores for Bagging consistently exceed 99%, highlighting its exceptional performance. These metrics indicate near-perfect detection rates for both binary and multi-class classification, which is a marked improvement compared to traditional machine learning approaches. The statistical analysis, including paired *t*-tests, further validates these findings by showing significant differences in performance metrics between Bagging and other classifiers, underscoring the effectiveness of the proposed method.

Table 8
Statistical analysis of ensemble classifiers.

Classifier Pair	Accuracy		Precision_macro		Recall_macro		F1_macro	
	t-statistic	p-value	t-statistic	p-value	t-statistic	p-value	t-statistic	p-value
Bagging vs. XGBoost	15.19410	0.00011	11.48246	0.00033	27.69626	0.00001	19.87206	0.00004
Bagging vs. Extra Trees	72.69173	0.00000	69.81524	0.00000	74.79013	0.00000	72.78774	0.00000
Bagging vs. Random Forest	27.86368	0.00001	23.30237	0.00002	28.38528	0.00001	25.71129	0.00001
Bagging vs. AdaBoost	1892.15766	0.00000	3011.4633	0.00000	2499.5281	0.00000	5078.08537	0.00000
Bagging vs. Gradient Boost	112.01805	0.00000	15.50348	0.00010	60.08481	0.00000	49.93018	0.00000
Bagging vs. CatBoost	10.98740	0.00039	14.22507	0.00014	12.20850	0.00026	13.72470	0.00016
XGBoost vs. Extra Trees	29.66029	0.00001	28.92002	0.00001	22.67995	0.00002	23.93566	0.00002
XGBoost vs. Random Forest	10.83439	0.00041	10.22535	0.00052	4.46675	0.01110	6.39875	0.00306
XGBoost vs. AdaBoost	202.62748	0.00000	266.24037	0.00000	231.93536	0.00000	247.72796	0.00000
XGBoost vs. Gradient Boost	19.36422	0.00004	11.08948	0.00038	12.07294	0.00027	20.65903	0.00003
XGBoost vs. CatBoost	−15.85258	0.00009	−6.55270	0.00280	−13.9900	0.00015	−13.50269	0.00017
Extra Trees vs. Random Forest	−38.13817	0.00000	−35.4719	0.00000	−25.1831	0.00001	−28.81042	0.00001
Extra Trees vs. AdaBoost	200.95152	0.00000	317.54660	0.00000	221.26745	0.00000	271.84466	0.00000
Extra Trees vs. Gradient Boost	−32.06159	0.00001	−8.03150	0.00130	−22.7937	0.00002	−14.81867	0.00012
Extra Trees vs. CatBoost	−38.80198	0.00000	−40.7572	0.00000	−37.9272	0.00000	−40.31267	0.00000
Random Forest vs. AdaBoost	222.83494	0.00000	269.16198	0.00000	197.56379	0.00000	234.11116	0.00000
Random Forest vs. Gradient Boost	7.87634	0.00140	4.48017	0.01099	6.69082	0.00260	8.05075	0.00129
Random Forest vs. CatBoost	−21.27504	0.00003	−16.9888	0.00007	−27.6412	0.00001	−22.74566	0.00002
AdaBoost vs. Gradient Boost	−615.5588	0.00000	−110.784	0.00000	−277.508	0.00000	−283.0329	0.00000
AdaBoost vs. CatBoost	−273.5954	0.00000	−466.189	0.00000	−278.211	0.00000	−390.4297	0.00000
Gradient Boost vs. CatBoost	−33.34912	0.00000	−12.5683	0.00023	−21.0724	0.00003	−25.22449	0.00001

Table 9
Comparison of Proposed Approach with Existing Approach for the Used Dataset.

Approach with Reference	Category	No. of Selected Feature	Records	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Proposed	Binary	10	392,035	100.00	100.00	100.00	100.00
	11 Classes			99.82	99.72	99.74	99.73
DT [22]	Binary	19	392,035	97.24	98.50	98.40	98.45

5. Conclusion

In this research, we investigated an ensemble-based machine learning approach for detecting Android ransomware, aiming to enhance accuracy and robustness compared to traditional methods. Our methodology involved comprehensive data preprocessing, feature selection using Random Forest, and the application of various ensemble classifiers, including Bagging, XGBoost, and CatBoost. The results demonstrated that the proposed approach significantly outperforms existing methods, achieving near-perfect accuracy, precision, recall, and F1-scores in both binary and multi-class classification tasks. These findings underscore the effectiveness and reliability of our approach, providing a robust solution for real-world cybersecurity applications. However, it is important to acknowledge the limitations and potential challenges of our approach in real-world implementation. One limitation is the reliance on a comprehensive and up-to-date dataset for training the models, which may not always be available. Additionally, the computational resources required for training and deploying ensemble classifiers can be substantial, posing challenges for implementation on devices with limited processing power and memory. Another challenge is the continuous evolution of ransomware, which may require frequent updates to the detection models to maintain their effectiveness. Finally, real-time detection in a mobile environment necessitates efficient algorithms that can operate with minimal latency and resource consumption.

Future research directions include developing methods for continuously updating the training dataset with the latest ransomware variants and investigating more resource-efficient algorithms suitable for a wider range of mobile devices. Additionally, creating adaptive models that automatically update based on new data and optimizing real-time detection for lower latency and minimal resource consumption will enhance practical applicability.

Accessibility of datasets and resources

The dataset used in this research is publicly available and properly cited in the dataset section of the paper. If required, the coding and implementation details will also be made available upon request.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. The study was conducted independently by the authors without any external financial support.

Declaration of competing interest

The authors declare that there is no conflict of interest regarding the publication of this research paper. The research was conducted without any commercial or financial relationships that could be construed as a potential conflict of interest. All authors have approved the manuscript and agree with its submission to the Cyber Security and Applications journal. Furthermore, the authors have no affiliations with any organization or entity with a direct financial interest in the subject matter or materials discussed in this manuscript.

CRediT authorship contribution statement

Md. Alamgir Hossain: Writing – original draft, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Tahmid Hasan:** Writing – review & editing, Methodology, Investigation, Conceptualization, Visualization. **Fahad Ahmed:** Data curation, Formal analysis, Investigation, Methodology, Resources.

Sheikh Hasib Cheragee: Conceptualization, Formal analysis, Software, Visualization, Writing – original draft. **Muntasir Hasan Kanchan:** Conceptualization, Methodology, Resources, Supervision, Validation, Writing – review & editing. **Md Alimul Haque:** Formal analysis, Investigation, Project administration, Resources, Supervision, Visualization, Writing – review & editing.

References

[1] M. Benmalek, Ransomware on cyber-physical systems: taxonomies, case studies, security gaps, and open challenges, Internet Things Cyber-Phys. Syst. 4 (2024) 186–202, doi:10.1016/j.iotcps.2023.12.001.

[2] M. Cen, F. Jiang, X. Qin, Q. Jiang, R. Doss, Ransomware early detection: a survey, Comput. Netw. 239 (2024) 110138 Feb, doi:10.1016/j.comnet.2023.110138.

[3] A.S.M. Al-Ruwili, A.M. Mostafa, Analysis of ransomware impact on android systems using Machine Learning techniques, Int. J. Adv. Comput. Sci. Appl. 14 (11) (2023), doi:10.14569/IJACSA.2023.0141178.

[4] A. Kapoor, A. Gupta, R. Gupta, S. Tanwar, G. Sharma, I.E. Davidson, Ransomware detection, avoidance, and mitigation scheme: a review and future directions, Sustainability. 14 (1) (2021) 8 Dec, doi:10.3390/su14010008.

[5] Md.A. Hossain, Md.S. Islam, A novel hybrid feature selection and ensemble-based machine learning approach for botnet detection, Sci. Rep. 13 (1) (2023) 21207 Dec, doi:10.1038/s41598-023-48230-1.

[6] J.W. Hu, Y. Zhang, Y.P. Cui, Research on android ransomware protection technology, J. Phys. Conf. Ser. 1584 (1) (2020) 012004 Jul, doi:10.1088/1742-6596/1584/1/012004.

[7] A. Pagan, K. Elleithy, A multi-layered defense approach to safeguard against ransomware, in: 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, NV, USA, 2021, pp. 0942–0947, doi:10.1109/CCWC51732.2021.9375988. Jan.

[8] M.A. Hossain, M.S. Islam, Ensuring network security with a robust intrusion detection system using ensemble-based machine learning, Array (2023) 100306 Jul, doi:10.1016/j.array.2023.100306.

[9] A. Vehabovic, N. Ghani, E. Bou-Harb, J. Crichigno, A. Yayimli, Ransomware detection and classification strategies, in: 2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 2022, pp. 316–324, doi:10.1109/BlackSeaCom54372.2022.9858296. Jun.

[10] B. Yamany, M.S. Elsayed, A.D. Jurcut, N. Abdelbaki, M.A. Azer, A holistic approach to ransomware classification: leveraging static and dynamic analysis with visualization, Information 15 (1) (2024) 46 Jan, doi:10.3390/info15010046.

[11] N. Singh, S. Tripathy, It's too late if exfiltrate: early stage Android ransomware detection, Comput. Secur. 141 (2024) 103819 Jun, doi:10.1016/j.cose.2024.103819.

[12] A. Moshood Abiola, M. Fadzli Marhusin, Signature-based malware detection using sequences of N-grams, Int. J. Eng. Technol. 7 (4) (2018) 4 .15Art.15Oct, doi:10.14419/ijet.v7i4.15.21432.

[13] C. Beaman, A. Barkworth, T.D. Akande, S. Hakak, M.K. Khan, Ransomware: recent advances, analysis, challenges and future research directions, Comput. Secur. 111 (2021) 102490 Dec, doi:10.1016/j.cose.2021.102490.

[14] C. Zheng, et al., SmartDroid: an automatic system for revealing UI-based trigger conditions in android applications, in: Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices, Association for Computing Machinery, New York, NY, USA, 2012, pp. 93–104, doi:10.1145/2381934.2381950. SPSM '12Oct.

[15] V. Rastogi, Y. Chen, X. Jiang, Catch me if you can: evaluating android anti-malware against transformation attacks, IEEE Trans. Inf. Forensics Secur. 9 (1) (2014) 99–108 Jan, doi:10.1109/TIFS.2013.2290431.

[16] G. Canfora, E. Medvet, F. Mercaldo, C.A. Visaggio, Detecting Android malware using sequences of system calls, in: Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile, ACM, Bergamo Italy, 2015, pp. 13–20, doi:10.1145/2804345.2804349. Aug.

[17] M. Bi, J. Xu, M. Wang, F. Zhou, Anomaly detection model of user behavior based on principal component analysis, J. Ambient Intell. Humaniz. Comput. 7 (4) (2016) 547–554 Aug, doi:10.1007/s12652-015-0341-4.

[18] Y. Kaya et al., "Demystifying behavior-based malware detection at endpoints," May 09, 2024, *arXiv*: arXiv:2405.06124. Accessed: Jul. 27, 2024. [Online]. Available: http://arxiv.org/abs/2405.06124

[19] J. Yang, Z. Zhang, H. Zhang, J. Fan, Android malware detection method based on highly distinguishable static features and DenseNet, PLoS. One 17 (11) (2022) e0276332 Nov, doi:10.1371/journal.pone.0276332.

[20] Md.A. Hossain, Md.S. Islam, Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity, Cybersecur. (Singap) 7 (1) (2024) 16 Jan, doi:10.1186/s42400-024-00205-z.

[21] R. Ismael Farhan, An approach to android ransomware detection using deep learning, Wasit J. Pure Sci. 3 (1) (2024) 90–94 Mar, doi:10.31185/wjps.325.

[22] A. Albin Ahmed, A. Shaahid, F. Alnasser, S. Alfaddagh, S. Binagag, D. Alqahtani, Android ransomware detection using supervised machine learning techniques based on traffic analysis, Sensors 24 (1) (2023) 189 Dec, doi:10.3390/s24010189.

[23] S. Sharma, R. Krishna, R. Kumar, An ensemble-based supervised machine learning framework for android ransomware detection, Int. Arab J. Inf. Technol. 18 (3A) (2021), doi:10.34028/iajit/18/3A/5.

[24] I. Alazzam, K.M.O. Nahar, Combined source code approach for test case prioritization, in: Proceedings of the 2018 International Conference on Information Science and System, ACM, Jeju Republic of Korea, 2018, pp. 12–15, doi:10.1145/3209914.3209936. Apr.

[25] H. Faris, M. Habib, I. Almomani, M. Eshtay, I. Aljarah, Optimizing extreme learning machines using chains of salps for efficient android ransomware detection, Appl. Sci. 10 (11) (2020) 3706 May, doi:10.3390/app10113706.

[26] H. Zuhair, "A panoramic evaluation of machine learning and deep learning-aided ransomware detection tools using a hybrid cluster of rich smartphone traits," in *Advances On Smart and Soft Computing*, vol. 1399, F. Saeed, T. Al-Hadhrami, E. Mohammed, and M. Al-Sarem, Eds., in Advances in Intelligent Systems and Computing, vol. 1399., Singapore: Springer Singapore, 2022, pp. 387–408. doi: 10.1007/978-981-16-5559-3_32.

[27] Sikha Bagui and T. Woods, "Machine learning for android ransomware detection," Feb. 2021, doi: 10.5281/ZENODO.4533395.

[28] S. Alsoghyer, I. Almomani, Ransomware detection system for android applications, Electronics. (Basel) 8 (8) (2019) 868 Aug, doi:10.3390/electronics8080868.

[29] M. Masum, M.J. Hossain Faruk, H. Shahriar, K. Qian, D. Lo, M.I. Adnan, Ransomware classification and detection with machine learning algorithms, in: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, Las Vegas, NV, USA, 2022, pp. 0316–0322, doi:10.1109/CCWC54503.2022.9720869. Jan.

[30] A. Arabo, R. Dijoux, T. Poulain, G. Chevalier, Detecting ransomware using process behavior analysis, Procedia Comput. Sci. 168 (2020) 289–296, doi:10.1016/j.procs.2020.02.249.

[31] "Android ransomware detection." Accessed: Jul. 25, 2024. [Online]. Available: https://www.kaggle.com/datasets/subhajournal/android-ransomware-detection

[32] Ms. Bithi, Md.A. Hossain, Md.K. Ahmed, R. Sultana, I. Ahammad, M.S. Islam, Enhanced DDoS detection in software defined networking using ensemble-based machine learning, in: 2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT), IEEE, Dhaka, Bangladesh, 2024, pp. 1032–1037, doi:10.1109/ICEEICT62016.2024.10534483. May.

[33] Md.A. Hossain, Enhanced ensemble-based distributed denial-of-service (DDoS) attack detection with novel feature selection: a robust cybersecurity approach, Artif. Intell. Evol. 4 (2) (2023) 165–186 Aug, doi:10.37256/aie.4220233337.

[34] G. Ngo, R. Beard, R. Chandra, Evolutionary bagging for ensemble learning, Neurocomputing. 510 (2022) 1–14 Oct, doi:10.1016/j.neucom.2022.08.055.

[35] B.S. Bhati and C.S. Rai, "Ensemble based approach for intrusion detection using extra tree classifier," in *Intelligent Computing in Engineering*, vol. 1125, V. K. Solanki, M. K. Hoang, Z. (Joan) Lu, and P. K. Pattnaik, Eds., in Advances in Intelligent Systems and Computing, vol. 1125., Singapore: Springer Singapore, 2020, pp. 213–220. doi: 10.1007/978-981-15-2780-7_25.

[36] Z. Chen, X. Ren, An efficient boosting-based windows malware family classification system using multi-features fusion, Appl. Sci. 13 (6) (2023) 4060 Mar, doi:10.3390/app13064060.

[37] Md.A. Hossain, et al., AI-enabled approach for enhancing obfuscated malware detection: a hybrid ensemble learning with combined feature selection techniques, Int. J. Syst. Assur. Eng. Manag. (2024) Mar, doi:10.1007/s13198-024-02294-y.

[38] Md.A. Hossain, Md.Saiful Islam, An ensemble-based machine learning approach for botnet-based DDoS attack detection, in: 2023 IEEE International Conference on Telecommunications and Photonics (ICTP), IEEE, Dhaka, Bangladesh, 2023, pp. 1–5, doi:10.1109/ICTP60248.2023.10490528. Dec.

[39] A. Rehman Javed, Z. Jalil, S. Atif Moqurrab, S. Abbas, X. Liu, Ensemble Adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles, Trans. Emerg. Telecommun. Technol. 33 (10) (2022) Oct, doi:10.1002/ett.4088.

[40] K. Thosar, P. Tiwari, R. Jyothula, D. Ambawade, Effective malware detection using gradient boosting and convolutional neural network, in: 2021 IEEE Bombay Section Signature Conference (IBSSC), IEEE, Gwalior, India, 2021, pp. 1–4, doi:10.1109/IBSSC53889.2021.9673266. Nov.

[41] N.S. Bhati and M. Khari, "A new intrusion detection scheme using CatBoost classifier," in *Forthcoming Networks and Sustainability in the IoT Era*, vol. 353, E. Ever and F. Al-Turjman, Eds., in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 353., Cham: Springer International Publishing, 2021, pp. 169–176. doi: 10.1007/978-3-030-69431-9_13.

[42] Md.A. Hossain, Md.S. Islam, Enhancing DDoS attack detection with hybrid feature selection and ensemble-based classifier: a promising solution for robust cybersecurity, Meas. Sens. 32 (2024) 101037 Aug, doi:10.1016/j.measen.2024.101037.

[43] H.H. Rashidi, S. Albahra, S. Robertson, N.K. Tran, B. Hu, Common statistical concepts in the supervised Machine Learning arena, Front. Oncol. 13 (2023) 1130229 Feb, doi:10.3389/fonc.2023.1130229.

[44] O. Rainio, J. Teuho, R. Klén, Evaluation metrics and statistical tests for machine learning, Sci. Rep. 14 (1) (2024) 6086 Mar, doi:10.1038/s41598-024-56706-x.