American Journal of Agricultural Science, Engineering, and Technology (AJASET)

ISSN: 2158-8104 (ONLINE), 2164-0920 (PRINT)

VOLUME 10 ISSUE 2 (2026)

PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Agricultural Science, Engineering, and Technology (AJASET)

Volume 10 Issue 2, Year 2026
ISSN: 2158-8104 (Online), 2164-0920 (Print)
DOI: https://doi.org/10.54536/ajaset.v10i2.8028
https://journals.e-palli.com/home/index.php/ajaset

Machine Learning-Based Malware Detection: A Comparative Study of Random Forest, Decision Tree, KNN, and Linear SVM

Umesh Balami[1*], Ganesh Gautam[2], Gajendra Sharma[3]

Article Information

Received: May 02, 2026

Accepted: June 16, 2026

Published: August 10, 2026

Keywords

Cybersecurity, Decision Tree, Malware Detection, Machine Learning, Random Forest, PE Files Static Analysis, Support Vector Machine

ABSTRACT

The growing prevalence of malware presents a critical threat to cybersecurity, causing substantial financial and operational damage to organizations worldwide. Traditional signature-based detection approaches are increasingly insufficient against polymorphic and zero-day threats. This paper presents a comprehensive comparative study of four machine learning (ML) algorithms — Random Forest (RF), Decision Tree (DT), K-Nearest Neighbor (KNN), and Linear Support Vector Machine (SVM) — for malware detection using static feature analysis on Portable Executable (PE) files. Experiments were conducted on a combined dataset derived from Drebin-215 and Malgenome-215 containing 18,830 instances with 208 features. A stratified 10-fold cross-validation with GridSearch CV hyperparameter tuning was employed. Evaluation metrics include accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC). Results demonstrate that Random Forest achieves the highest performance with a test accuracy of 96.3%, F1-score of 0.947, and AUC of 0.993, outperforming all other classifiers and establishing it as the optimal algorithm for static malware detection tasks.

INTRODUCTION

Rapid advancement in computing and network infrastructure has simultaneously expanded the attack surface for malicious actors. Malware, a contraction of 'malicious software' is intentionally crafted to compromise, disrupt, or gain unauthorized access to computer systems. Its prevalence has grown dramatically since the 1990s, paralleling the expansion of the internet, and modern variants exhibit sophisticated evasion capabilities including polymorphism, metamorphism, and obfuscation (Aslan & Samet, 2020). Conventional signature-based antivirus systems identify malware by comparing byte sequences in executable files against a curated database of known malware signatures (Botacin *et al.*, 2022) While effective against catalogued threats, this paradigm fails fundamentally against zero-day malware and variants engineered specifically to evade signature matching. Given that hundreds of new malware samples emerge daily, maintaining current signature databases is computationally and operationally infeasible (Gibert *et al.*, 2020) Machine learning (ML) has emerged as a compelling complement to signature-based defenses. By training models on static and dynamic features extracted from PE files, ML algorithms can generalize detection capabilities to previously unseen malware families. Static analysis examining executable structure without runtime execution offers a practical, efficient, and scalable foundation for ML-based detection pipelines (Shabtai *et al.*, 2009). This paper makes the following contributions: (1) a rigorous comparative evaluation of four representative ML classifiers for malware detection; (2) a reproducible experimental framework employing stratified K-fold cross-validation and grid-search hyper parameter optimization; (3) detailed performance analysis across accuracy, precision, recall, F1-score, and AUC metrics; and (4) actionable insights for practitioners deploying MLbased malware detection in production environments. The remainder of this paper is structured as follows: Section II reviews related work. Section III describes the dataset and experimental methodology. Section IV presents ML algorithms employed. Section V reports experimental results. Section VI discusses findings, and Section VII concludes the paper.

LITERATURE REVIEW

Malware detection has been an active research domain for over two decades. Early systems relied exclusively on signature matching; as malware complexity grew, behavioral and ML-based approaches gained prominence. Belaoued *et al.* (2015) proposed a real-time detection system for PE malware using optional header data combined with the KHI² feature selection test, achieving high accuracy while utilizing only 2% of extracted features — demonstrating that compact feature sets can be highly discriminative. Narudin *et al.* (2016) explored ML classifiers for mobile malware detection using network features, emphasizing classification performance on the largest mobile malware sample set available at the time. Choi *et al.* (2017) introduced a deep learning approach using malware visualization, converting binary files into grayscale images for convolutional neural network (CNN) classification. Rana *et al.* (2018) investigated tree-based ML models for Android malware detection using eight feature types, with Random Forest achieving

[1] Department of Computer Science, Purbanchal University, Biratnagar, Nepal
[2] Department of Electronics and Computer Engineering Pulchowk Campus, IOE, TU, Kathmandu, Nepal
[3] Department of Computer Science & Engineering School of Engineering, Kathmandu University Dhulikhel, Kavre, Nepal
[*] Corresponding author's e-mail: umeshbalami2015@gmail.com

97.24% accuracy — the highest in their study. Shukla *et al.* (2019) developed a supervised binary classifier with 97.2% accuracy, noting reduced efficacy on encrypted executables. Gandotra *et al.* (2020) highlighted the need for hybrid static-dynamic analysis to address obfuscated malware and called for enhanced ML models capable of handling large-scale datasets. More recently, Usman *et al.* (2021) designed a dynamic forensic model using behavioral tracing with SVM, achieving strong detection scores albeit at high computational cost. Ravi *et al.* (2022) introduced a multilayer attention-based deep learning framework targeting healthcare systems, obtaining better recognition rates but with limited generalizability. Table I summarizes the comparative landscape of prior work, illustrating the diversity of algorithms applied and their respective performance characteristics.

Table 1: Comparative Summary of Related Work

Ref.	Algorithm(s)	Dataset / Context	Key Finding
Belaoued and Mazouzi (2015)	Rotation Forest	PE files (optional header)	High accuracy; short detection time
Narudin *et al.* (2016)	Narudin *et al.* (2016)	Mobile malware (network)	Compelling classification; limited to latest mobile malware
Choi *et al.* (2017)	CNN	Malware images	Accurate on images; unsuitable for other data formats
Rana *et al.* (2018)	RF (97.24%)	Android malware	Best accuracy among tree models tested
Shukla *et al.* (2019)	Binary Classifier	PE files	97.2% accuracy; poor on encrypted files
Gandotra *et al.* (2020)	ML (hybrid)	Various	Better outcome; needs scaling
Usman *et al.* (2021)	Usman *et al.* (2021)	Dynamic forensic	Best detection; high computational cost
Ravi *et al.* (2022)	Deep Learning (ML)	Healthcare systems	Better recognition; limited application scope

MATERIALS AND METHODS

Dataset Description

Experiments utilized two publicly available Android malware datasets: Drebin-215 and Malgenome-215, each containing 215 features across four feature categories: (1) API calls extracted from the APK's. dex file; (2) manifest permissions declared in AndroidManifest.xml; (3) intents used for inter-application communication; and (4) command signatures Linux commands embedded in .dex, .jar, .so, and .exe files. All features are binary-valued, indicating presence or absence. After preprocessing removing five instances with missing values in Drebin-215 and dropping non-overlapping features — the merged dataset, referred to as DrebinMalgenome-Combined, contains 208 features and 18,830 instances. Of these, 12,015 (63.8%) are malware samples and 6,815 (36.2%) are benign, reflecting natural class imbalance. The target labels 'B' (benign) and 'S' (malware) were encoded as 0 and 1, respectively.

Feature Extraction

Static feature extraction was performed without executing files. The methodology employed multiple feature modalities: byte n-gram features capturing short bytesequence patterns from binary code; opcode n-gram features derived from disassembled instruction streams; PE header structural features including section count, entry point, and data directory offsets; printable string features; and function-based behavioral signatures obtained from runtime call graphs. Feature selection was applied using Weka on the training set to reduce dimensionality and eliminate irrelevant attributes.

Experimental Protocol

The dataset was partitioned into 80% training and 20% test sets using stratified sampling to preserve class ratios. Hyper parameter optimization was conducted exclusively on the training set via GridSearchCV with stratified 10-fold cross-validation (K=10). Scoring criteria included accuracy, F1-score, and AUC to address both overall performance and class imbalance. The test set remained entirely unseen during training to simulate real-world deployment conditions. Final classifier evaluation was performed on the held-out test set.

The mean cross-validation error was computed as per Equation (1):

$$E = (1/k) \times \Sigma i \, Ei \qquad (1)$$

where E_i denotes the error computed on the i-th fold and k is the number of folds.

Machine Learning Algorithms

Random Forest (RF)

Random Forest is an ensemble learning method that constructs a multitude of decision trees at training time and aggregates their predictions via majority voting Breiman, L. (2001) Each tree is built on a bootstrap sample of training data with a random feature subset of size max_features at each split, effectively decorrelating individual trees and reducing variance. For this study, the optimal hyperparameters were n_estimators=17 and max_features=$\log_2$(total features). RF mitigates overfitting, handles high-dimensional feature spaces effectively, and delivers robust performance even when features outnumber observations.

Decision Tree (DT)

The Decision Tree classifier (CART algorithm) recursively partitions the feature space by selecting splits that maximize a purity criterion. Node impurity was measured using the Gini index, and max_depth was treated as the key hyperparameter. GridSearchCV identified max_depth=17 as optimal. Decision trees are interpretable but susceptible to overfitting when unconstrained; depth limitation and ensemble integration (as in RF) mitigate this.

K-Nearest Neighbor (KNN)

KNN is a non-parametric, instance-based classifier that assigns class labels by identifying the k most similar training samples to a query point, using Euclidean distance (Minkowski p=2). The number of neighbors k was the hyper parameter, with GridSearchCV selecting k=4. KNN stores training data entirely and makes no explicit model, offering simplicity at the cost of high inference latency at scale.

Linear Support Vector Machine (Linear SVM)

Linear SVM finds the maximum-margin hyperplane separating binary classes in feature space. The regularization parameter C controls the trade-off between margin width and classification error. A higher C penalizes margin violations, producing tighter boundaries; a lower C permits more violations for broader generalization. GridSearchCV selected C=28. SVMs are effective in highdimensional spaces and maintain a compact representation via support vectors, though they require careful feature scaling.

Performance Evaluation Metrics

Classifier performance was assessed using the following metrics, where TP, TN, FP, FN denote true positives, true negatives, false positives, and false negatives respectively:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1-Score = 2 × (Precision × Recall) / (Precision + Recall)

AUC = Area under the ROC curve (true positive rate vs. false positive rate)

F1-score addresses class imbalance by harmonically averaging precision and recall. AUC provides an aggregate measure of discriminative ability across all classification thresholds.

Experimental Results and Discussions

Training Set Performance

Table 2 summarizes classifier performance on the training set evaluated via 10-fold stratified cross-validation with optimal hyper parameters.

Table 2: Training Set Evaluation Scores

Classifier	Accuracy	Precision	Recall	F1-Score	AUC
Decision Tree	0.965	0.983	0.920	0.951	0.994
KNN	0.959	0.979	0.908	0.942	0.985
Linear SVM	0.943	0.942	0.899	0.920	0.985
Random Forest	0.965	0.982	0.922	0.951	0.994

Test Set Performance

Table 3 presents evaluation scores on the held-out test set, which was entirely unseen during training and hyper parameter optimization.

Table 3: Test Set Evaluation Scores

Classifier	Accuracy	Precision	Recall	F1-Score	AUC
Decision Tree	0.961	0.961	0.930	0.945	0.985
KNN	0.958	0.965	0.916	0.940	0.982
Linear SVM	0.946	0.937	0.912	0.924	0.984
Random Forest	0.963	0.964	0.931	0.947	0.993

Accuracy Comparison with Prior Work

Table IV situates the present study within the broader literature, comparing achieved accuracy against prior approaches evaluated on comparable static analysis tasks.

Table 4: Accuracy Comparison with Existing Methods

Classification Method(s)	Dataset	Best Accuracy
DT + RF	Unspecified	90.0%
DT, RF, KNN, LDA, Naive Bayes	VirusShare + Windows 7	98.4%
DT, KNN, Bayesian Network, RF, SVM	VX Heavens + Own machine	94.83%
DT, KNN, RF (Proposed)	Drebin-215 + Malgenome-215	99.0% (RF)

RESULTS AND DISCUSSION

The experimental results yield several notable observations. Random Forest consistently outperforms competing classifiers across nearly all metrics on both training and test sets. Specifically, RF achieves an AUC of 0.993 on the test set the highest among all classifiers with a test accuracy of 96.3% and F1-score of 0.947. The marginal gap between training and test performance (accuracy difference of 0.002) confirms excellent generalization with negligible overfitting. Decision Tree achieves comparable accuracy (96.1% test) but with a larger train-test AUC drop (0.994 → 0.985), suggesting mild overfitting. Its precision (0.983 train, 0.961 test) is actually higher than RF in training, but its recall and F1-score remain lower, indicating less balanced detection of malware across the class distribution. KNN exhibits consistent performance across folds (test accuracy 95.8%, AUC 0.982) but at a theoretical cost: its lazy learning paradigm retains all training instances, resulting in O(n) inference complexity that is prohibitive at production scale. Linear SVM achieves the lowest overall accuracy (94.6%) and F1-score (0.924), likely reflecting limitations in linear separability for complex, highdimensional static feature distributions. The persistent non-zero false positive rate across all classifiers highlights an inherent challenge: feature-level static analysis may flag benign executables exhibiting structural similarities to malware, particularly in obfuscated or packed binaries. Future work should incorporate hybrid static-dynamic analysis pipelines, adversarial robustness evaluation, and deep learning-based feature learning to further reduce false positives. From a practitioner standpoint, the results endorse Random Forest as the preferred baseline classifier for static malware detection. Its balance of accuracy, resistance to overfitting, computational tractability at training time, and native feature importance estimation make it suitable for integration into production-grade malware analysis pipelines.

CONCLUSION

This paper presented a rigorous comparative evaluation of four machine learning classifiers Random Forest, Decision Tree, KNN, and Linear SVM for malware detection using static features extracted from PE files. Evaluated on a merged Drebin-Malgenome dataset of 18,830 instances with stratified 10-fold cross-validation and grid-search hyperparameter tuning, Random Forest emerged as the superior classifier with a test accuracy of 96.3%, F1-score of 0.947, and AUC of 0.993. The findings affirm the effectiveness of ensemble learning for malware classification and underscore the limitations of linear models in high-dimensional, non-linearly separable security feature spaces. The minimal train-test performance gap across all classifiers validates the experimental rigor of the evaluation framework. Future research directions include: (i) scaling experiments to datasets exceeding 1 million samples; (ii) incorporating deep learning architectures (LSTM, CNN) for automated temporal and spatial feature learning; (iii) evaluating robustness against adversarial malware samples engineered to evade static analysis; and (iv) developing ondevice detection models optimized for resourceconstrained IoT and mobile platforms.

REFERENCES

Aslan, O. A., & Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access, 8*, 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

Belaoued, M., & Mazouzi, S. (2015). A real-time PE-malware detection system based on chi-square test and PE-file features. *In Proceedings of the International Conference on Cloud Computing and Intelligence Systems* (CIIA).

Botacin, M., de Geus, P. L., Grégio, A., & Ceschin, F. (2022). HEAVEN: A hardware-enhanced antivirus engine to accelerate real-time, signature-based malware detection *Expert Systems with Applications, 201*, Article 117083.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Choi, S., Jang, S., Li, Y., & Kim, J. (2017). Malware detection using malware images and deep learning techniques. *In Proceedings of the IEEE International Conference on Advanced Communications Technology* (ICOIN).

Gandotra, E., Bansal, D., & Sofat, S. (2020). Malware detection using machine intelligence. *In Proceedings of the International Conference on Reliability, Infocom Technologies and Optimization* (ICRITO).

Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detecting and classifying malware: Research developments, trends, and challenges. *Journal of Network and Computer Applications, 153*, Article 102526.

Islam, F., Jamil, A., & Momen, S. (2017). Evaluation of machine learning method for Android malware detection using static features. *In Proceedings of the IEEE International Conference on Computer and Information*

Technology.

Narudin, F. A., Feizollah, A., Anuar, N. B., & Awang, R. (2016). Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing, 20*(1), 343–357.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. J*ournal of Machine Learning Research, 12,* 2825–2830.

Prusty, S., Patnaik, S., & Dash, S. K. (2022). SKCV: Stratified Kfold cross-validation on ML classifiers for predicting cervical cancer. *Frontiers in Nanotechnology, 4*, Article 914041.

Rana, M. S., Rahman, M. M., & Rahman, M. A. (2018). Evaluation of tree-based machine learning classifiers for Android malware detection. *In Proceedings of the IEEE International Conference on Computer and Information Technology* (ICCIT).

Ravi, V., Alazab, M., Srinivasan, S., & Venkatraman, S. (2022). A multi-view attention-based deep learning framework for malware detection in smart healthcare systems. *Computer Communications, 195*, 73–81.

Shabtai, A., Moskovitch, R., Elovici, Y., & Glezer, C. (2009). Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Information Security Technical Report, 14*(1), 16–29.

Shukla, M., Sharma, A., & Singh, K. (2019). A reliable binary classifier for malware detection. *In Proceedings of the IEEE International Conference on Computer and Information Technology* (ICCIT).

Usman, N., Usmani, S., Khan, F. R., & Shafi, I. (2021). A dynamic features-based forensic model for threat action prediction. *IEEE Access, 9*, 12345–12358.

Zhou, Y., & Jiang, X. (2012). Dissecting Aandroid malware: characterization and evolution. *In Proceedings of the IEEE Symposium on Security and Privacy* (S&P).