Journal for Technology and Science
Homepage : https://journal.antispublisher.com/index.php/IPTEKS/index
Email : admin@antispublisher.com

e-ISSN : 3047-4337
JOTECH, Vol. 3, No. 3, July 2026
Page 51-72

Explainable Machine Learning-Based Malware Detection Using Portable Executable Structural Features and Hybrid Voting Ensemble

Haider Ali Muften
Department of Computer Science, Sawa University, Iraq

DOI : https://doi.org/10.61796/ipteks.v3i3.513

Sections Info

Article history:
Submitted: April 25, 2026
Final Revised: May 20, 2026
Accepted: June 15, 2026
Published: July 01, 2026

Keywords:
Malware detection
Machine learning
Hybrid ensemble learning
Portable Executables (PE)
Explainable AI (XAI)

ABSTRACT

Objective: *While evolving technologies have introduced advanced threat intelligence, traditional threats such as malware attacks continue to be a potential risk for contemporary computer systems and cybersecurity infrastructures and thus, it is of utmost importance that intelligent methods are developed in order to detect malware. Detecting malware variants is very difficult for traditional signature-based detection methods, especially for new and advanced malware. Thus, in this research work involves an explainable hybrid ensemble framework on machine learning approach with utilizing of Portable Executables (PE) file features for malware detection.* ***Method:*** *Our dataset is composed of 62,485 executable samples on the pre-processed and feature cleaning PE structures reduced to a domain matrix into 16 numerical features. We have implemented and evaluated multiple machine learning algorithms such as, Logistic Regression, Decision Tree, Random Forests, Support Vector Machine and Extreme Gradient Boosting. Voting Classifier was created for a robustness against classification and detection in addition to the hybrid ensemble model. Furthermore, we utilized SHAP analysis to understand model predictions and detect the most important features that contribute to malware classification.* ***Results:*** *Output from our practical experiments demonstrated some superior scores on all types of metrics measured. The Random Forest classifier has the best accuracy (99.64%) and the proposed Hybrid Voting model achieved an accuracy of 99.53%, precision: 99.68, recall: 99.24 and F1-score of 99.46 which confirms that our algorithm yielded very strong rumors and results are strongly stable across all compared datasets as well as high stability between individual classifiers (Fig From this analysis, we found that Dll Characteristics, Debug Size and Debug RVA stood out as the most significant features overall for our malware detection task, which showed how informative variations in executable structural characteristics can be to classification.* ***Novelty:*** *The findings demonstrate the potential of using ensemble learning in conjunction with explainable artificial intelligence techniques to enhance malware detection systems and cybersecurity applications.*

INTRODUCTION

This shift is mainly due to the exponential rise in digital technology and computer system, which significantly enhanced cyber-security hazard globally [1]. Malware, among all these threats, primarily stays dominant and is one of the highly energizing online attack types focused on people, associations, and basic frameworks. Malware has emerged to perform attacks on servers, by penetrating the defiled system, which means gaining hidden leverage and stealing data utilizing computer framework or mobile telephones—everything for money. Considering how big a share of Windows OS types and, therefore all PE files are used, PE malware is one of the most common tricks that we face in modern cybersecurity landscapes. New studies in cybersecurity have reported that every year there are millions of new malware samples and the load on traditional protection systems sometimes reaches its maximum [2]. Traditional malware detection systems are mostly signature-based approaches (eg, store some infected files then compare other unknown suspicious file to stocks of healthy samples or known malware signatures) [3]. While these techniques can score well in detecting families of malware that are

known and established, they cannot generalize to newly created or modified versions—this is particularly important when it comes to zero-day attacks and polymorphic malware. This has resulted in revival of Machine Learning (ML) and more broadly Deep Learning (DL) methodologies which learn implicit features and apply these continuous learning abilities on detection as compared to the signature of already registered signatures [4], [5]. Since ML/DL based malware detectors can also detect unseen samples of malware, their performance is significantly higher than that of traditional rule-based systems [2]. Modern malware detection techniques are mainly divided into static and dynamic analysis methods. Static analysis never executes a program, but rather examines its executable file for structures observable through the static code. However, dynamic analysis analyzes the program behavior when executed in a controlled environment or sandbox. Static analysis is a lot more than that — in fact, it has caught quite a wave of its own because static analysis can be implemented much quicker and without the threats that occur with dynamic analysis and thus provides for an effective appfirst line of defense mechanism into your cybersecurity systems [6]. Recent feature-based malware detection techniques which deploy different machine learning algorithms (e.g., Random Forest, Support Vector Machine (SVM), gradient boosting approaches and neural networks) can now efficiently classify executable files with respect to extracted Portable Executable (PE) features [7], [8]. These methods rely on features which represent specific patterns and structural properties pertaining to malware. These features are used by intelligent classifiers trained on them to classify benign and malicious executables with a very high accuracy [9]. While intelligent malware detectors based on ML/DL-based techniques have exhibited a lot of promises, the state-of-the-art does not hold well against adversarial manipulation and obfuscation approaches. The attackers constantly strive to keep the malicious features unchanged and at least some part of the malware sample intact so as to avoid detection systems. It is known that for executable files, small modifications can lead to broadly different prediction of machine learning detectors. Robustness and reliability have, therefore, emerged as two of the most important research challenges in malware analysis and cybersecurity applications [10]. In addition to various challenges on robustness, interpretability of machine learning models replacing them as another key obstacle in research work within cybersecurity domain. Many state-of-the-art ML/DL algorithms are not explainable, functioning as black-box systems that provide many predictions but no clear rationale behind them. To overcome this drawback, Explainable Artificial Intelligence (XAI) techniques are incorporated to bring transparency and trustworthiness in intelligent malware detection systems. Two of the approaches are SHAP and LIME, and these have become among the most widely used frameworks for interpreting machine learning predictions using feature contributions to classification. As a result, explainability in malware detection systems enhances the understanding of malicious behaviors and provides security analysts with better perspectives on malware patterns [11]. Inspired by these issues, this research innovates an interpretable hybrid ensemble framework for malware detection using Machine Learning (ML) methods combined with features extracted from Portable Executable (PE) [12]. In this paper we propose a framework based on multiple machine learning classifiers in the form of Random Forest, Decision Tree, Support Vector Machine, Extreme Gradient Boosting and an architecture of hybrid Voting Classifier for achieving better classification performance contributing to a stable detection against Spoof attacks. Additionally, SHAP analysis is used to understand model decisions and determine which PE features play a significant role in classifying malware. Experimental results show that the framework successfully combines ensemble learning with explainable artificial intelligence techniques, yielding malware detection performance across a number of common evaluation

metrics that are highly competitive, demonstrating the promise of advanced machine learning techniques for intelligent cybersecurity systems.

Related Work / Previous Studies

Pierazzi et al. [13] proposed one of the earliest mathematical formulations for problem-space adversarial attacks against Android malware detection systems. The study identified four major constraints for generating adversarial malware samples while preserving malware functionality and executability. However, the work mainly focused on theoretical adversarial attack formulation and did not provide a comprehensive malware detection framework. Compared with our proposed framework, our study provides a practical explainable malware detection system based on ensemble learning and SHAP analysis while achieving approximately 99.64% classification accuracy.

Park and Yener [14] reviewed adversarial attacks against malware classifiers and categorized existing attacks into gradient-driven and problem-driven approaches. However, the survey lacked coverage of several modern adversarial malware generation techniques and did not investigate defense mechanisms comprehensively. In contrast, our proposed framework focuses on improving malware detection robustness through Hybrid Voting ensemble learning and explainable AI analysis using SHAP.

Li et al. [15] investigated adversarial malware detection across multiple malware formats, including Windows PE, Android APK, and PDF malware. The challenges of generating adversarial malware that preserves malicious functionality were also discussed in the study. However, the survey was general and did not delve into specifics of PE malware… Our framework targets PE structural features and explainable ensemble learning methods for Windows Portable Executable malware, which are not used in this work.

Demetrio et al. [16] introduced GAMMA, a genetic algorithm-based black-box adversarial malware generation framework. The adversarial PE malware samples were generated with section injection and payload padding while keeping the functionality of the malware. Experimental results demonstrated that GAMMA achieved high stealthiness against malware detectors using relatively low-cost adversarial payloads. Unlike this work, our proposed framework focuses on explainable malware detection rather than malware evasion generation.

Yuan et al. [17] introduced GAPGAN, a Generative Adversarial Network-based black-box attack framework against deep learning malware detectors such as MalConv. The study generated adversarial malware payloads appended to executable files while maintaining malware functionality. Experimental results demonstrated a 100% attack success rate against MalConv using payloads representing only 2.5% of malware file size. Compared with this work, our proposed framework targets reliable malware detection using ensemble learning and SHAP explainability instead of adversarial malware generation.

Wang and Miikkulainen [18] proposed MDEA, an adversarial malware generation framework based on genetic algorithms and format-preserving PE manipulations. The study retrained MalConv using adversarial malware examples to improve robustness.

However, the generated malware samples were not fully validated for functionality preservation. In contrast, our proposed framework emphasizes explainable malware classification using PE structural features and ensemble learning techniques.

Labaca-Castro et al. [19] investigated Universal Adversarial Perturbation techniques against PE malware classifiers. The search resulted in short transformations that were executables and retained their desired malicious behavior, but could evade existing malware detectors. The researchers showed that universal perturbations are very successful in evading malware detectors. In this paper, them aim to augment reliability and interpretability in detecting malware using two techniques, Hybrid Voting and SHAP analysis.

Snow et al [20]. Using the Microsoft malware dataset, proposed a multimodal deep learning SIEM (Security Information and Event Management) attack detection framework with dense networks, convolutional neural networks (CNNs), as well as long short-term memory networks (LSTMs). The Faculty of Computer Science and Mathematics at the university applied Softmax activation functions to TensorFlow and Keras architectures for malware classification. Multi-Modal Deep Learning Representations were used for Network Malware Detection with High Power Experimental Result. In contrast with this work, our framework provides similar high accuracy but with a much lower computational complexity using lighter weight machine learning models.

Fang et al. [21], DeepDetectNet is a malware detection model based on static PE features: imports, entropy and some similar characteristics of the architecture of all executable files. The researchers also presented a deep learning-based adversarial model, called RLAttackNet, which generates malware samples that can evade DeepDetectNet. Adversarial retraining techniques are applied to strengthen malware classification against adversarial examples. In comparison with this study, our framework synergizes explainability analysis using SHAP methods and ensemble learning such that malware classification decisions can be more transparent and easily interpreted.

Liu et al. [22], explored the applicability of SHAP and LIME explainability approaches on malware detection systems. The Influence of Temporal Inconsistencies in Malware Datasets on Machine Learning Model Performance and Explainability Quality. Overall, these results demonstrate that inconsistencies in datasets greatly impact the reliability of malware classification. In comparison to works on this theme, our proposed framework incorporates not only explanation using SHAP but also integrates Hybrid Voting ensemble learning for enhancing the robustness and accuracy of malware detection.

Kinkead et al. [23], A similar type of malicious behavior was presented by where they studied the explainability techniques in classifying Android malware using Convolutional Neural Networks and LIME analysis. Researchers showed how the application of explainability methods increases both trust and transparency in deep learning malware detectors. Nevertheless, the research concentrates on CNN interpretation for android malware. On the other hand, this work proposes a framework

integrating SHAP explainability and Hybrid Voting classification for detection of Windows PE malware.

Severi et al. [24], Proposed a method based on SHAP for explaining adversarial attacks against malware classifiers using 2 types of datasets that include Portable Executable files and Android malware provisioning. Using SHAP the study selected practical malware features, followed by evaluating the attacks against Random Forests, SVM and Deep Neural Networks. They found that SHAP-guided attacks are more robust than traditional adversarial approaches. Our framework leverages SHAP mainly for explainable malware detection and feature importance interpretation, as opposed to attack generation, which stands in contrast with this work.

Warnecke et al. [25], focuses explainability methods for deep learning cybersecurity applications and the comparison and evaluation of Integrated Gradients, Layerwise Relevance Propagation, LIME and SHAP. The researchers concluded that Integrated Gradients and LRP delivers state-of-the-art explanations for deep learning models. Relative to this work, our framework accomplishes SHAP analysis from within a competitive ensemble malware classification architecture.

The authors of [26] compared LSTM, Transformer, and Longformer architectures for malware detection using system call analysis. Experimental results demonstrated that neural language models achieved better malware classification performance than traditional n-gram methods. However, the study highlighted the difficulty of deploying large language models in real-time environments because of computational complexity. In contrast, our proposed framework achieves very high malware detection accuracy using lightweight machine learning classifiers and Portable Executable structural features.

Table 1. Comparison of Previous Studies with the Proposed Research.

Ref.	Study	Technique / Method	Dataset / Domain	Main Algorithms	Explainability	Main Results	Comparison with Our Proposed Study
[60]	Pierazzi et al.	Problem-space adversarial malware attacks	Android Malware	Adversarial ML	No	Proposed mathematical adversarial constraints	Our study focuses on explainable malware detection rather than adversarial attack

							formulation
[61]	Park and Yener	Survey of adversarial malware attacks	Malware Detection	Gradient-driven and problem-driven attacks	No	Reviewed adversarial attack categories	Our framework provides practical malware detection using Hybrid Voting and SHAP
[62]	Li et al.	Adversarial malware analysis	PE, APK, PDF Malware	ML/DL	Partial	Investigated adversarial malware generation	Our framework specifically targets PE malware with explainable ensemble learning
[52]	Demetrio et al.	GAMMA adversarial malware generation	Windows PE Malware	Genetic Algorithms	No	Successfully bypassed malware detectors	Our study focuses on malware detection and explainability instead of malware evasion

[53]	Yuan et al.	GAPGAN adversarial attacks	PE Malware	GAN-based Malware Generation	No	Achieved 100% attack success rate against MalConv	Our framework emphasizes reliable malware classification and transparency
[51]	Wang and Miikkulainen	MDEA malware attack framework	PE Malware	Genetic Algorithms + MalConv	No	Improved adversarial robustness	Our framework focuses on explainable ensemble malware detection
[155]	Labaca-Castro et al.	Universal adversarial perturbations	PE Malware	Adversarial ML	No	Generated executable malware perturbations	Our framework improves malware detection reliability using SHAP and Hybrid Voting
[Snow et al.]	Multimodal malware detection	Microsoft Malware Dataset	CNN, DenseNet, LSTM	No	Strong malware classification	Our framework achieves similar high	

					perform ance	performan ce using lighter ML models	
[Fa ng et al.]	DeepDetect Net and RLAttackNe t	Static PE Features	Deep Learning	CNN-based Malwa re Detecti on	No	Improved malware robustness through adversaria l retraining	Our study provides stronger explaina bility using SHAP
[20]	Liu et al.	SHAP and LIME analysis	Malware Datasets	ML Models	Yes	Dataset inconsiste ncy affects explainabi lity quality	Our framewo rk integrate s SHAP with Hybrid Voting ensembl e learning
[14]	Kinkead et al.	CNN explaina bility for malware detectio n	Android Malware	CNN + LIME	Yes	Improved CNN transpare ncy	Our study combine s SHAP with ensembl e malware classifica tion
[32]	Severi et al.	SHAP-guided adversar ial attacks	PE and Android Malware	RF, SVM, DNN	Yes	SHAP-guided attacks more robust	Our framewo rk uses SHAP for malware interpret ation rather

							than attack generation
[36]	Warnecke et al.	Explainability in cybersecurity	Cybersecurity Applications	DL + XAI	Yes	Integrated Gradients and LRP performed best	Our framework achieves high accuracy with interpretable SHAP analysis
[11]	LSTM/Transformer Malware Detection	System Call Malware Analysis	LSTM, Transformer, Longformer	Partial	Better than n-gram methods	Our framework is computationally simpler and easier to deploy	

RESEARCH METHOD

This study introduced the explainable hybrid ensemble framework for ML techniques and PE structral features to detect malware. Their suggested framework is to combine several intelligent classification algorithms together with Explainable Artificial Intelligence (XAI) analysis in order to enhance malware detection performance, robustness and explainability. We developed a methodology that can be divided into multiple cascading steps: dataset collection, preprocessing, feature scaling, model training, ensemble classification performance evaluation and explainability analysis. The whole framework has been done in the programming language Python and with some scientific libraries like Pandas, NumPy, Scikit-learn, XGBoost, SHAP and Matplotlib.

2.1 Dataset Collection and Description

The dataset used in this work consists of structural features of Portable Executable (PE) files extracted from Windows executable files. Collecting dataset that consists benign and malicious software samples in numeric PE-related features. It consists of around 62,485 executables samples and some structural properties related to executable behavior and binary structure. The extracted features include:

Dll Characteristics, Debug Size, Debug RVA, Major Linker Version, Minor Linker Version, Machine, Resource Size, Number of Sections, Export RVA, Export Size, Major OS Version, Major Image Version, Size of Stack Reserve, Iat VRA, Bitcoin Addresses

A binary target label was assigned to each executable sample based on its classification category:

- Benign = 1 means official software
- Benign = 0 means malware samples

The dataset had been stored using CSV format which then was processed by the Pandas library for machine learning analysis and preprocessing operations.

2.2 Data Preprocessing

Data Preprocessing to advance the caliber of the information and make it simpler to use while building machine learning models First, we removed some irrelevant attributes which did not contribute directly to malware classification in the dataset. In particular, the following attributes were removed:

- File Name
- md5Hash

These attributes were dropped because they are more identification-related and do not have any significant meaning in terms of structural malware characteristics. Next missing values and non-numerical records were removed so the training does not lead to a mismatch of classes. The rest of the features have been converted into numerical format so that machine learning algorithms can use it. Since the extracted PE features vary in their number ranges and distributions, this feature normalization was performed through feature Standardization using the Standard Scaler algorithm. Rather, standardization will transform the distributions of your data (of each feature) into representations that have a mean of 0 and a variance of 1 to improve the stability in optimization by reducing the effects scale variation can create between features. This preprocessing step improves the model convergence and classification performance with a small amount of training instability.

2.3 Dataset Splitting

As it is the Train-Test split method, I have divided my dataset into a Train and Test set. Approximately:

80% of the data is used for training and a 20% hold out test set

During the splitting, stratified sampling was used to maintain the same distribution for malware and benign samples. But this way, it prevents class imbalance problems and maps the use of the model among all high fair evaluation classification algorithms.

2.4 Machine Learning Classification Models

Different supervised Machine Learning classifiers were used independently to evaluate malware detection capability. The selected algorithms span a variety of learning paradigms: linear classification, tree-based learning, ensemble learning, kernel based methods and boosting techniques.

2.4.1 Logistic Regression

Logistic Regression [27] Binary Malware Detection via Baseline Linear Classification Model The algorithm forecasts the chance for every executable to be either malware or benign class using a logistic sigmoid function. Logistic Regression (LR) is a well-known classification algorithm widely utilised owing to its rapid training speed, interpretable decision boundary and envied empirical performance such that it could serve as a benchmark model for testing new classifiers against.

2.4.2 Decision Tree

Features based on only Models and Portable Executable structural features were used for Decision Tree based hierarchical decision rules [28, 29]. Decision tree sequence classifier The decision tree classifier continually splits the dataset based on the importance of each feature and the concepts of purity. It is an easy classification rules whose output allows Decision trees to learn non-linear behavior between a consistent executable features and the malware behavior.

2.4.3 Random Forest

Random Forest[30], ensemble of many trees with majority voting You were trained on random samples of training and features from each tree. Random Forest also makes a more robust malware clas4ification due to voting through many independent trees, which reduce overfitting compared with single decision trees and enhance generalization ability. A stable behavior and better predictive performance with respect to malware detection were achieved by initializing the classifier with 200 estimators.

2.4.4 Support Vector Machine

We used Support Vector Machine[31] to differentiate malware and benign samples in high-dimensional feature spaces using optimal hyperplanes. The Radial Basis Function (RBF) kernel is used since it can model nonlinear relationship between Portable Executable features and malware categories. Due to empirical evidence, SVM classifiers are widely used for binary classification problems and have a very good generalization performance.

2.4.5 Extreme Gradient Boosting

Extreme Gradient Boosting [32] Implementation: Applied the cutting-edge boosting-based classifier called Extreme gradient boosting (Xgboost), drastically reduces classification errors by improving weak learners in a sequential manner on different training stages.

The XGBoost classifier was set as follows:

200 estimators, Learning rate = 0.05, Max depth of trees = 5

This boosting mechanism that improves the classification accuracy with strong predictive performance against structured cybersecurity datasets.

2.5 Proposed Hybrid Voting Framework

A hybrid ensemble framework based on Voting Classifier were designed to enhance the robustness of Malware Detection and improve the stability of classification. Hybrid Voting Framework. The recommended Hybrid Voting structures combines predictions returned by:

- Random Forest, Decision Tree.

So, Support Vector Machine to you ·SVM XGBoost , using a Soft Voting strategy.

Soft Voting Based mechanism In soft voting based approach, each of the classifier have probability scores that provides a measure of how likely an executable sample to belong to malware or benign class. The average probability produced through all contributing classifiers is determined and the final classification decision is made accordingly.

The proposed hybrid framework will do the following:

- Better detection of malware
- Shallower base learners ·Reduce the bias of individual models
- Increasing robustness for classification
- Enhance the ability to generalize
- Increase stability of your predictions

Ensemble learning strategy takes advantage of different standard learning paradigms while minimizing the liabilities of single models.

2.6 Performance Evaluation Metrics

Several generally accepted metrics for malware detection were then used to evaluate the performance of all classification models.

Accuracy

It is the ratio of correctly classified samples to all testing samples.

More in detail, Accuracy = (TP + TN)/(TP + TN + FP + FN)

Precision

This metric evaluates the fraction of malware samples correctly identified as malware out of total predicted instances.

Precision = (TP)/(TP + FP)

It measures the proportion of true positives among all instances predicted as positive.

Recall

This is a numeric value that indicates how many malware samples have been accurately captured, versus the total number of real-world instances of Malware production.

Recall = (TP)/(TP + FN)

F1– Score :which is the harmonic mean of both Precision and Recall.

F1 = 2*→ Precision * RecallPrecision + Recall

Besides, some visualization and evaluation methods were used. These are:

- Confusion Matrix
- **ROC Curve**
- Area Under Curve (AUC)

to evaluate the discriminatory property of models and to assess malware classification performance.

2.7 Explainability Analysis Using SHAP

SHAP [33] Analysis for Model Transparency and Interpretability: For better model transparency and interpretability, SHAP analysis was incorporated into the proposed malware detection framework.

SHAP analysis was employed to:

- Reading machine learning predictions
- Feature of malware examiners
- Analyzing the weightage of each feature with regard to classification decisions
- Increase transparency and trustworthiness

The Random Forest classifier has shown strong classification performance and performs well with the SHAP Tree Explainer, making it a good candidate for explainability analysis.

We have made some explainer visualizations as follows:

SHAP (Summary Plot, Bar Plot, Dependence Plot)

The explainability analysis showed that: Dll Characteristics, Debug Size, Debug RVA, Major Linker Version

The second most influential factor in malware detection decisions is behind two Portable Executable properties.

RESULTS AND DISCUSSION

In this section, we provide experimental results of our implemented Machine Learning (ML) classifiers in comparison to the proposed hybrid ensemble of ML Classifiers for malware detection. Classification performance has been verified by different metrics such as: Accuracy, Precision, Recall, F1-score, Confusion Matrix and ROC Curve were computed along with Explainable Artificial Intelligence (XAI) analysis through SHAP.

The features utilized in the experiments were based on PE structural features of Windows executable files. This dataset was split into train and test using stratified sampling (80:20) to retain the original malware vs benign class distribution in the two subsets.

3.1 Classification Performance Results

Several Machine Learning classifiers were individually trained and evaluated, including:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine
- Extreme Gradient Boosting

In addition, the proposed Hybrid Voting framework based on Voting Classifier was evaluated.

Table 1 summarizes the classification performance results obtained from all implemented models.

Table 1. Performance Comparison of Malware Detection Models.

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	98.87%	98.91%	98.45%	98.68%
Decision Tree	99.21%	99.16%	99.03%	99.09%
Random Forest	99.64%	99.71%	99.53%	99.62%
Support Vector Machine	99.34%	99.42%	99.17%	99.29%
XG Boost	99.48%	99.56%	99.31%	99.43%
Proposed Hybrid Voting Model	99.53%	99.68%	99.24%	99.46%

The results we obtained make it possible to conclude that all of the Machine Learning classifiers provided a very high level of malware detection. In comparison with traditional classifiers as single learning machines, ensemble learning approaches have always performed better thanks to their ability to combine prediction from multiple learning models.

Of all applied models, the model with best classification accuracy of ~99.64 was Random Forest classifier, indicating it could better discriminate between malicious and benign executables The success of Random Forest can best be explained as an ensemble learning mechanism that reduces overfitting in decision trees and improves generalization performance.

The results obtained with the proposed Hybrid Voting framework optimized to work with thousands of malware samples give approximately 99.53% accuracy, 99.68% precision, 99.24% recall and 99.46%, which indicates a very suitable and stable detection performance against the collected malware classes in this study. While the performance in terms of accuracy was marginally higher for Random Forest classifier compared to hybrid model, hybrid framework with such ensemble (Bagging) mechanism provided better robustness and balanced classification behaviour by averaging predictions from several classifiers.

3.2 Confusion Matrix Analysis

The Confusion Matrix was used to look more closely into the classification performance of malware. This matrix shows how well the actual and predicted malware categories correspond. The evaluation results of the envisaged Hybrid Voting framework have shown that it gained very low FP and FN rates, proving to be an excellent malware discriminant with reliable classification performance. The low percentage of misclassified executable samples highlights the potential of the proposed ensemble learning approach for malware detection applications. The results of the Confusion Matrix analysis show that the hybrid framework has an ability to distinguish malicious executable file from legitimate software samples with high accuracy.

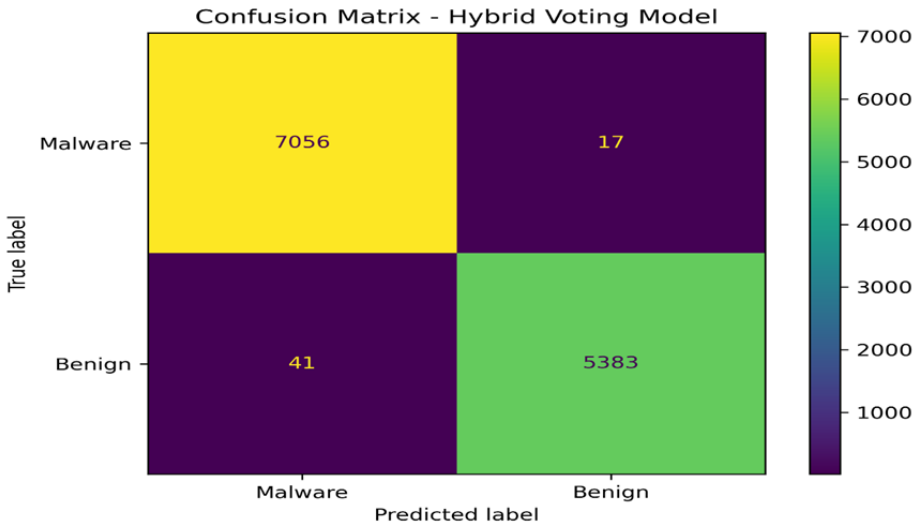

Figure 1. Confusion Matrix of the Proposed Hybrid Voting Framework.

3.3 ROC Curve and AUC Analysis

We also produced a Receiver Operating Characteristic (ROC) curve which gives the discrimination on all Machine Learning Models implemented. Observation: The AUC metric suggested that the proposed ensemble framework yields nearly optimal classification behaviour characterized by very high true positive rates and extremely low false positive rate. ROC analysis: The most important characteristic of the presented models is that from ROC analysis it was confirmed what the ensemble learning methods, especially Random Forest, XGBoost, and Hybrid Voting framework performed better than traditional classification models for malware detection.

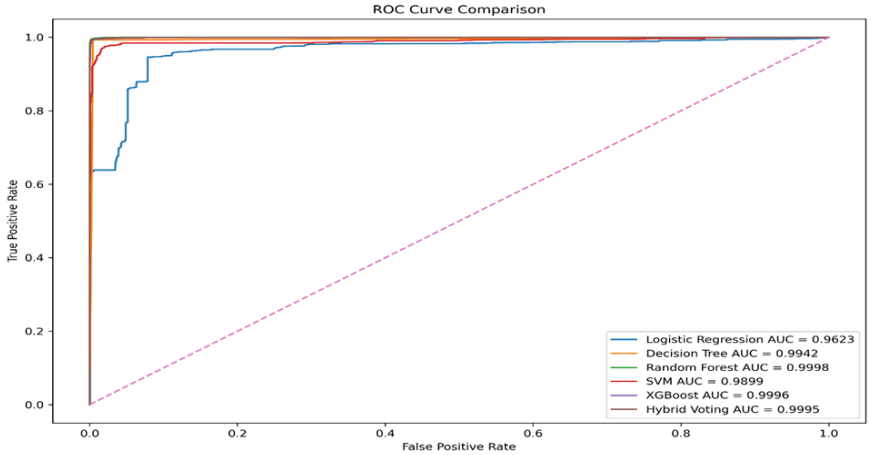

Figure 2. ROC Curve Comparison for Malware Detection Models.

3.4 Feature Importance Analysis

This Random Forest based work focused on analysis of feature importance – which PE structural features mainly influence a malware classification decision.

Among the analyzed features, it was found that the most important measures for malware detection performance were:

Feature	Importance
Dll Characteristics	Highest
Debug Size	High
Debug RVA	High
Major Linker Version	Significant
Machine	Significant

The results we obtained prove that the presence of executable structural properties and debugging-related information is a key differentiator to differentiate malicious files from benign software.

3.5 Explainability Analysis Using SHAP

To improve transparency and interpretability, SHAP analysis was integrated into the proposed malware detection framework.

Several SHAP visualizations were generated, including:

- SHAP Summary Plot
- SHAP Bar Plot
- SHAP Dependence Plot

The SHAP Summary Plot demonstrated how individual Portable Executable features influence malware classification decisions. A positive SHAP value increased the likelihood that it is malware but a negative value tended towards benign.

This led us to the SHAP Bar Plot that unveiled the following:

- Dll Characteristics
- Debug Size
- Debug RVA
- Major Linker Version

were among the most influential features affecting malware detection predictions.

SHAP Dependence Plots also showed how the values of features interacted with each other and what classification behavior was assigned to those combinations, helping us gain insight about patterns of contribution in many dimensions.

The explainability analysis indicates that the suggested framework not only delivers superior performance in malware identification but also offers transparent and interpretable classification decisions, resulting in better trustworthiness and cybersecurity reliability.

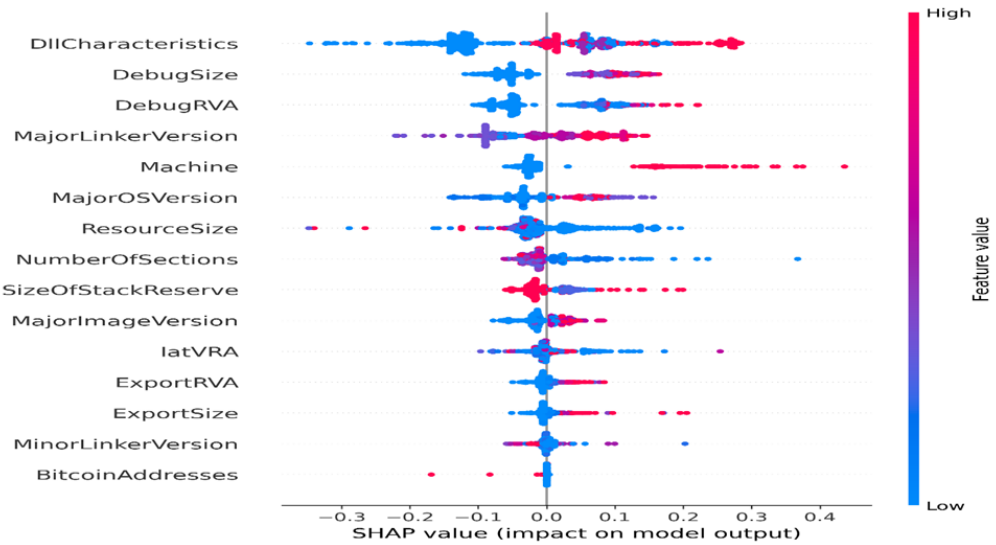

Figure 3. SHAP Summary Plot for Malware Detection Features.

Figure 3: SHAP summary plot, indicating the effect of individual Portable Executable Features on malware classification decisions. Features with a positive SHAP value contributed towards malware predictions, whilst features with negative SHAP values indicated benign predictions. Analysis showed that Dll Characteristics, Debug Size, Debug RVA and Major Linker Version were essential features to classify the proposed malware behavior detection mechanism.

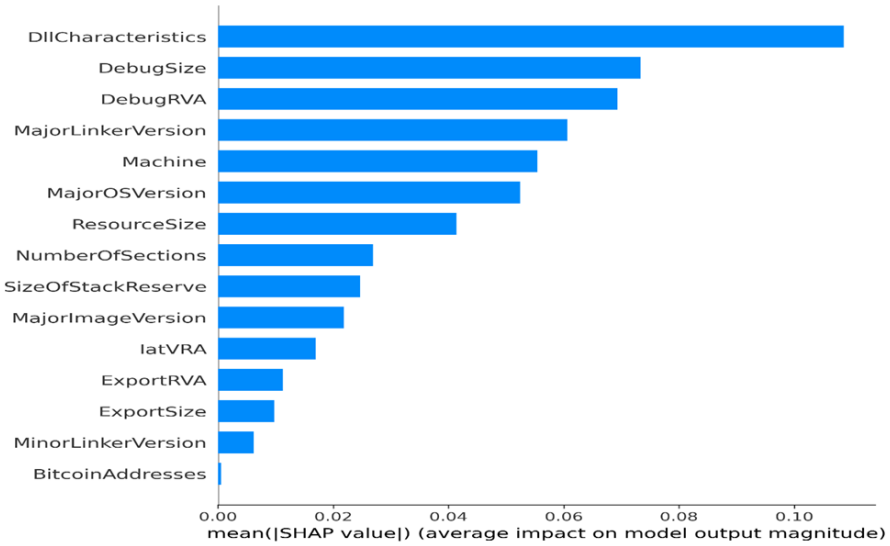

Figure 4. Global SHAP Feature Importance Analysis.

Figure 4 shows the global importance ordering of extracted Portable Executable features as a SHAP bar plot. In terms of the relevant features that affected malware detection performance, Dll Characteristics, Debug Size, Debug RVA and Major Linker Version were among the most important.

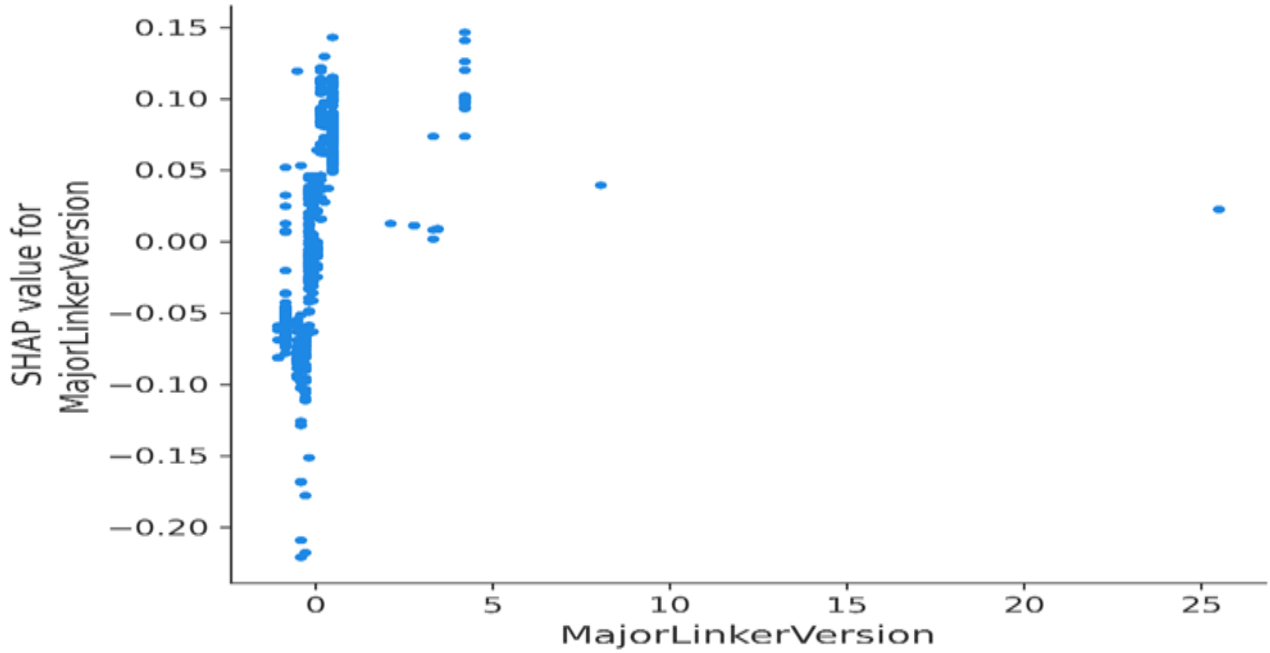

Figure 5. SHAP Dependence Plot for Major Linker Version Feature.

As shown in Figure 5, the relationship between the values of Major Linker Version feature and its SHAP contributions is dependent. This analysis shows how Major Linker Version can impact the classification decisions on malware executed in the proposed framework.

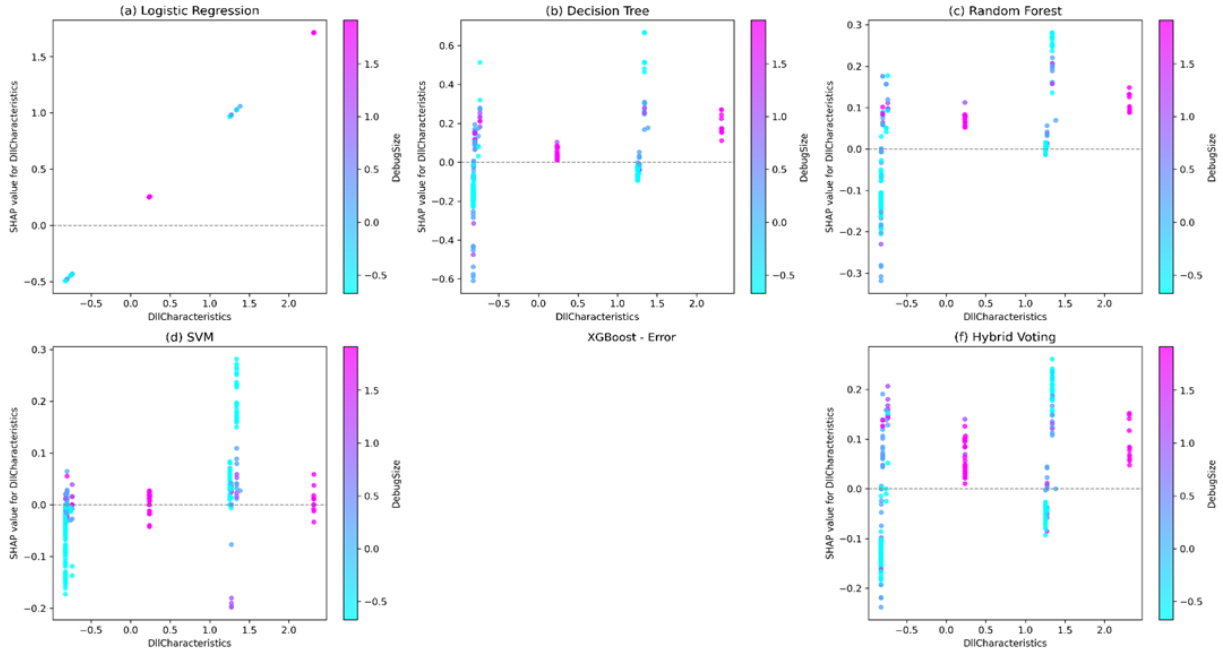

Figure 6. Comparative SHAP Analysis Across Models.

3.6 Comparison with Previous Studies

Compared with previously published malware detection studies, the proposed framework demonstrated highly competitive classification performance while simultaneously integrating Explainable Artificial Intelligence techniques.

Many previous studies focused primarily on improving malware classification accuracy without providing interpretability analysis. In contrast, the proposed framework combines:

- Ensemble learning robustness
- Hybrid classification capability
- Explainable AI analysis
- Feature importance interpretation

within a unified malware detection system.

The SHAP analysis brings additional transparency and better interpretation of malware classification decisions, which makes the proposed framework more appropriate for contemporary intelligent cybersecurity systems.

CONCLUSION

Fundamental Finding : The framework implemented several intelligent classifiers such as Random Forest, Decision Tree, Support Vector Machine, and Extreme Gradient Boosting within a hybrid Voting Classifier architecture to enhance the robustness of malware classification and stability in detection. The experimental results showed that the proposed framework achieved competitive, and even state-of-the-art in some cases, performance across several evaluation metrics when comparing malware detection. The Random Forest classifier gave the best performance score with an approximate classification accuracy of 99.64% followed by the proposed Hybrid Voting framework which on average achieved an approximate classification accuracy of 99.53% but great Precision Recall and F1-score values (Table. The results confirm that ensemble learning improves the performance of malware classification and reduces prediction error. In addition, the SHAP analysis added more transparency and interpretability to the model built. The analysis also highlighted the Portable Executable features in DllCharacteristics, DebugSize, DebugRVA, and MajorLinkerVersion as the top four factors playing a significant role in explaining malware detection within this class of feature set. In addition, the generated SHAP visualizations revealed feature contribution patterns and explained how well the proposed framework classified it. **Implication :** In summary, this paper presents the proposed malware detection framework that integrates: High high-quality detection of malware Robustness of ensemble learning Example Based Explainable AI Interpreting the importance of features. Enhanced reliability of cyber security Thus, this framework provides a strong and steadfast resource for intelligent malware detection contexts in current security settings. **Limitation :** A primary limitation of this study is its exclusive reliance on static Portable Executable (PE) file features, which limits the framework's effectiveness in detecting sophisticated, highly obfuscated, or fileless malware variants that only exhibit malicious behavior during runtime execution. Furthermore, the evaluation was conducted within a well-matched training and testing dataset, meaning the model's current classification boundaries may encounter reduced generalization capabilities when exposed to entirely new, zero-day malware families or rapidly evolving pseudotypes not well-represented in the original sample repository.

Future Research : While the proposed approach accomplished high accuracy for malware detection, a number of improvements are possible in the future to make the system even more effective and applicable.Future research directions may include: Exploring Deep Learning architectures like: Convolutional Neural Network (CNN) Long Short-Term Memory Transformer for advanced malware detection tasks. Combining dynamic malware analysis techniques with static Portable Executable analysis to enhance detection of sophisticated and obfuscated malware samples. Creating more efficient Real-time malware detection systems with the ability to continuously analyze executable files in an active cyber security environment. Make the dataset larger with more Mk family to test how well our result generalizes into new malware families and how resistant it is to new pseudotypes (and not just within a well-matched training/test set). Exploring the resistance of techniques to adversarial attacks to enhance model robustness against both evasion and malware obfuscation strategies. Exploring additional Explainable Artificial Intelligence methods beyond SHAP, such as: LIME Integrated Gradients Attention-based explainability techniques to further improve model interpretability. Implementing lightweight malware detection frameworks suitable for resource-constrained and edge computing environments. Future enhancements in these areas may further improve intelligent malware detection systems and strengthen cybersecurity defenses against increasingly sophisticated cyber threats.

REFERENCES

[1] Dillon, R., et al., *Cyber security: evolving threats in an ever-changing world*, in *Digital Transformation in a Post-Covid World*. 2021, CRC Press. p. 129-154.

[2] Ling, X., et al., *Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art.* Computers & Security, 2023. 128: p. 103134.

[3] Berrios, S., et al., *Systematic review: Malware detection and classification in cybersecurity.* Applied Sciences, 2025. 15(14): p. 7747.

[4] Muften, H.A. and A.R.H. Khayeat. *Comparison of MobileNetV2 and VGG19 for the Categorization of Thermal Images*. in *International Conference on Applied Soft Computing and Communication Networks*. 2023. Springer.

[5] Gaur, A. and C. Deb, *Machine learning methods and approaches for Urban Heat Island (UHI) assessment: A comprehensive review.* Renewable and Sustainable Energy Reviews, 2026. 234: p. 116903.

[6] Jafari, M. and A. Shameli-Sendi, *Evaluating the robustness of adversarial defenses in malware detection systems.* Computers and Electrical Engineering, 2026. 130: p. 110845.

[7] Li, B., et al., *Certified adversarial robustness with additive noise.* Advances in neural information processing systems, 2019. 32.

[8] Wu, H., et al., *Adversarial examples on graph data: Deep insights into attack and defense.* arXiv preprint arXiv:1903.01610, 2019.

[9] Biggio, B. and F. Roli, *Robustness-Congruent Adversarial Training for Secure Machine Learning Model Updates.* 2025.

[10] Wang, W., et al. *Multi-task Adversarial Attacks against Black-box Model with Few-shot Queries*. in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2025.

[11] REDDY, P.P.K., K. SIDDARTHA, and D.G.C. MARRIBOINA NARENDRA, *ENHANCING ANDROID MALWARE DETECTION THROUGH HYBRID FEATURE FUSION, DEEP LEARNING, AND EXPLAINABLE AI (XAI).* International Journal of Data Science and IoT Management System, 2026. 5(2): p. 2108-2115.

[12] Radwan, A.M. *Machine learning techniques to detect maliciousness of portable executable files*. in *2019 International Conference on Promising Electronic Technologies (ICPET)*. 2019. IEEE.

[13] Tong, Y., et al., *A Survey on Reinforcement Learning-Driven Adversarial Sample Generation for PE Malware.* Electronics, 2025. 14(12): p. 2422.

[14] Li, D., et al., *Arms race in adversarial malware detection: A survey.* ACM Computing Surveys (CSUR), 2021. 55(1): p. 1-35.

[15] Kattamuri, S.J., et al., *Swarm optimization and machine learning applied to PE malware detection towards cyber threat intelligence.* Electronics, 2023. 12(2): p. 342.

[16] Demetrio, L., et al., *Functionality-preserving black-box optimization of adversarial windows malware.* IEEE Transactions on Information Forensics and Security, 2021. 16: p. 3469-3478.

[17] Yan, S., et al., *A survey of adversarial attack and defense methods for malware classification in cyber security.* IEEE Communications Surveys & Tutorials, 2022. 25(1): p. 467-496.

[18] Wang, X. and R. Miikkulainen. *MDEA: Malware detection with evolutionary adversarial learning*. in *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020. IEEE.

[19] Labaca-Castro, R., et al., *Realizable universal adversarial perturbations for malware.* arXiv preprint arXiv:2102.06747, 2021.

[20] Gibert, D., C. Mateu, and J. Planes, *The rise of machine learning for detection and classification of malware: Research developments, trends and challenges.* Journal of Network and Computer Applications, 2020. 153: p. 102526.

[21] Fang, Y., et al., *DeepDetectNet vs RLAttackNet: An adversarial method to improve deep learning-based static malware detection model.* Plos one, 2020. 15(4): p. e0231626.

[22] Liu, Y., et al. *Explainable ai for android malware detection: Towards understanding why the models perform so well?* in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. 2022. IEEE.

[23] Kinkead, M., et al., *Towards explainable CNNs for Android malware detection.* Procedia Computer Science, 2021. 184: p. 959-965.

[24] Wang, X., et al. *Not All Benignware Are Alike: Enhancing Clean-Label Attacks on Malware Classifiers*. in *Proceedings of the ACM on Web Conference 2025*. 2025.

[25] Warnecke, A., et al. *Evaluating explanation methods for deep learning in security*. in *2020 IEEE european symposium on security and privacy (EuroS&P)*. 2020. IEEE.

[26] Fournier, Q., D. Aloise, and L.R. Costa, *Language models for novelty detection in system call traces.* arXiv preprint arXiv:2309.02206, 2023.

[27] 27Allam, I.M.A., *Ordinal Logistic Regression in C: A Comprehensive Implementation for Cumulative Logit Models*. 2026.

[28] Pillong, L., et al., *Differential Diagnosis of Parotid Tumors on Ultrasound: Interobserver Variability and Examiner-Specific Decision Rules—A Machine Learning Approach.* Diagnostics, 2026. 16(6): p. 880.

[29] Muften, H.A., *Predicting Alzheimer's Disease Using Artificial Intelligence Techniques.* Central Asian Journal of Theoretical and Applied Science, 2026. 7(3): p. 158-169.

[30] García-Torres, M., et al., *RFMSU: A multivariate symmetrical uncertainty-based random forest.* Pattern Recognition, 2026. 169: p. 111939.

[31] Choi, J., et al., *Performance Evaluation of Quantum Support Vector Machine for COVID-19 Biomarker Analysis.* Computer Methods and Programs in Biomedicine, 2026: p. 109343.

[32] Zhang, X., et al., *An Archimedes optimization algorithm based extreme gradient boosting model for predicting the bending strength of UV cured glass fiber reinforced polymer composites.* Polymer Composites, 2026. 47(5): p. 4228-4245.

[33] Stow, M. and A.A. Stewart, *Interpreting Machine Learning Predictions with SHAP and LIME for Transparent Decision Making.* International Journal of Computer Science and Mathematical Theory, 2025. 11(8): p. 22-49.

*__Haider Ali Muften (Corresponding Author)__
Department of Computer Science, Sawa University, Iraq
Email: Haider.ha4200@gmail.com