



AI-BASED RANSOMWARE DETECTION FRAMEWORK USING MACHINE LEARNING

Kajal Shildar Padvi¹, Dr. Rucha Mohite²

Savitribai Phule Pune University, Pune, Maharashtra, India^{1,2}

Abstract: Ransomware remains one of the major cybersecurity threats because modern ransomware variants continuously evolve and may differ considerably from previously identified samples. Traditional signature-based detection approaches are effective against known threats but may have limited capability against previously unseen variants. Machine learning provides an alternative approach by learning discriminative patterns from malicious and benign executable samples. However, high classification performance under a conventional random train-test split does not necessarily demonstrate generalization to ransomware families that were not represented during training.

This research proposes an **AI-Based Ransomware Detection Framework Using Machine Learning** based on static characteristics of Windows Portable Executable (PE) files. The framework extracts a compact set of eight static features consisting of API count, DLL count, unique API count, unique DLL count, file size, packed status, entropy, and PE timestamp year. Five machine learning algorithms—Logistic Regression, Decision Tree, Random Forest, XGBoost, and LightGBM—are trained and compared using accuracy, precision, recall, F1score, ROC-AUC, false positive rate, and false negative rate.

The experimental framework additionally incorporates validation-based probability threshold analysis, ransomware-family-disjoint evaluation, family-level generalization analysis, Isolation Forest-based novelty detection, and SHAP-based explainability. Random Forest achieved the best overall standard-test performance with **95.42% accuracy, 96.22% precision, 94.28% recall, 95.24% F1-score, and 98.98% ROC-AUC**. When evaluated on ransomware families excluded from training, the Random Forest model achieved **90.20% accuracy, 98.88% precision, 87.02% recall, 92.57% F1-score, and 97.72% ROC-AUC**. The family-disjoint experiment demonstrates that detection becomes more challenging when the model encounters ransomware families not represented during training. Isolation Forest additionally identified approximately **8.04%** of unseen ransomware samples as anomalous relative to the known ransomware distribution.

The results demonstrate that a compact static PE representation combined with ensemble machine learning can provide strong ransomware classification performance while enabling additional analysis of generalization, novelty, and interpretability.

Keywords: Ransomware Detection, Machine Learning, Static Analysis, Portable Executable, Random Forest, Explainable AI, SHAP, Unseen Ransomware Families, Anomaly Detection, Malware Detection.

1. INTRODUCTION

Ransomware is a type of malicious software designed to deny access to systems or data and commonly attempts to obtain financial benefit from victims. The continued development of ransomware has created significant challenges for cybersecurity systems because new variants can modify their implementation, packaging, and structural characteristics.

Traditional signature-based antivirus systems primarily depend on previously identified patterns. Although such approaches remain important for known threats, they may have difficulty recognizing previously unseen or modified ransomware variants. This limitation has motivated researchers to investigate machine learning and artificial intelligence techniques for malware and ransomware detection.

Machine learning-based detection systems attempt to learn relationships between executable characteristics and malicious labels from previously analyzed samples. Different algorithms, including Logistic Regression, Decision Trees, Random Forest, Support Vector Machines, gradient-boosting models, and deep-learning architectures, have been investigated for ransomware detection [1]–[26]. Existing studies demonstrate that machine learning can achieve high detection performance, but reported results are often dependent on the selected dataset, features, experimental protocol,



and ransomware families represented in the training data.

An important issue is therefore **generalization**. A model may achieve high performance when training and test samples are randomly distributed, while its performance may decrease when it encounters ransomware families that were completely excluded from training. Familydisjoint evaluation can provide a more challenging assessment of whether a model has learned general ransomware characteristics rather than family-specific patterns.

Another important issue is model interpretability. A classification result indicating that an executable is ransomware does not necessarily explain why the model reached that decision. Explainable Artificial Intelligence (XAI) techniques such as SHAP can provide information about the contribution of individual features to model predictions.

Static analysis provides an attractive approach because executable files can be examined without executing them. Static PE characteristics such as imported APIs, DLL information, entropy, file size, packing indicators, and PE metadata can provide useful information for machine learning-based classification.

Therefore, this research develops an **AI-Based Ransomware Detection Framework Using Machine Learning** based on static Windows PE characteristics. The framework compares five machine learning algorithms using a common feature representation and evaluates them through multiple performance measures. The framework further investigates threshold selection, unseen-family generalization, family-level performance, anomaly detection, and explainability.

The major objectives of this research are:

1. To develop a static-analysis ransomware detection framework for Windows PE files.
2. To extract a compact set of static executable characteristics.
3. To compare Logistic Regression, Decision Tree, Random Forest, XGBoost, and LightGBM.
4. To evaluate the models using multiple classification metrics.
5. To identify the best-performing model for the selected static feature representation.
6. To investigate model performance on ransomware families excluded from training.
7. To analyze family-level generalization.
8. To investigate novelty detection using Isolation Forest.
9. To improve model interpretability using SHAP.
10. To develop a practical prototype for static executable classification.

2. LITERATURE REVIEW

Ransomware detection has been investigated using static analysis, dynamic analysis, machine learning, deep learning, behavioral monitoring, rule-based systems, hybrid frameworks, anomaly detection, and explainable artificial intelligence.

The literature reviewed for this research includes 26 studies covering conventional machine learning, deep learning, static and dynamic analysis, behavioral detection, XAI, and ransomware-defense frameworks.

2.1 Machine Learning-Based Ransomware Detection

Smith, Khorsandroo, and Roy [1] presented a comprehensive survey of machine learning algorithms and ransomware detection frameworks. Their work compared different machine learning approaches and discussed the datasets, algorithms, and challenges associated with ransomware detection. The study emphasized that ransomware continues to evolve and that no single detection approach completely addresses the problem.

Kritika [2] reviewed deep-learning-based ransomware detection research covering studies published between 2017 and 2024. CNN, RNN, LSTM, GAN, VAE, DBN, and hybrid architectures were discussed. The review reported high detection performance across many studies but identified challenges involving real-time detection, explainability, adaptability, and dataset quality.

Aggarwal [3] examined ransomware attacks, trends, attack vectors, detection approaches, and prevention mechanisms.



The study emphasized that ransomware has evolved from relatively simple attacks to increasingly sophisticated and targeted campaigns.

Ferdous et al. [4] presented a systematic review of AI-based ransomware detection. Their work analyzed datasets, preprocessing, feature engineering, model training, and evaluation metrics. The authors emphasized the need for more robust and systematic evaluation of AI-based ransomware detection systems.

These studies demonstrate the importance of machine learning for ransomware detection while also indicating that evaluation methodology and dataset characteristics strongly influence reported performance.

2.2 Advanced and Ensemble Machine Learning

Almotiri [5] investigated a double-tier AI framework using approaches including XGBoost, LightGBM, DNN, CNN, GAN, and Autoencoder models. The study reported strong classification performance but also identified computational and deployment limitations.

Rele et al. [6] investigated ransomware detection using system logs, network traffic, and file metadata collected through SIEM. Random Forest, Isolation Forest, Autoencoder, and SVM were considered. Their results demonstrated the potential of machine learning for ransomware detection but also highlighted the need for broader evaluation across different environments and ransomware families.

Singh, Khanna, and Verma [8] investigated feature engineering, PCA, Decision Tree, ANN, and XGBoost using a ransomware dataset. The reported results demonstrated the potential of ensemble learning. However, evaluation on a single dataset limits conclusions about crossdataset generalization.

These studies support the use of ensemble models while motivating further investigation into generalization beyond the dataset used for model development.

2.3 Static Analysis-Based Detection

Poudyal et al. [7] investigated ransomware detection using reverse engineering, feature extraction, and machine learning. Their approach examined executable characteristics and achieved high detection performance with several classifiers.

Poudyal et al. [11] later proposed a multi-level ransomware detection framework using reverse engineering, NLP, and machine learning. DLLs, function calls, and assembly instructions were converted into n-gram and TF-IDF representations. Logistic Regression achieved the best reported detection rate of 98.59% using combined multi-level TF-IDF trigrams.

Medhat, Gaber, and Abdelbaki [15], [17] investigated a static-based ransomware detection framework using YARA rules. Their work used static characteristics such as API functions, keywords, file extensions, and cryptographic signatures. The framework achieved strong detection performance and also evaluated ransomware families excluded from training.

These studies demonstrate that executable characteristics can provide useful information without requiring runtime execution.

2.4 Static and Dynamic Behavioral Analysis

Ahmadian and Shahriari [12] proposed 2entFOX, a framework for detecting high-survivable ransomware using static and dynamic behavioral analysis and a Bayesian Belief Network. The study demonstrated the potential of combining multiple behavioral characteristics but was evaluated on a relatively small number of samples.

Arabo et al. [16] investigated ransomware detection using process behavior analysis. The study monitored system-resource and process characteristics and used machine learning to distinguish ransomware from benign applications. The work demonstrated that process behavior can provide useful information for detection.

Poudyal and Dasgupta [18] proposed a hybrid static and dynamic ransomware detection framework using reverse engineering, sandboxing, NLP, association rule mining, and machine learning. The reported performance was high, but the framework requires significantly more computational and analytical processing than a compact static approach.



These studies show that dynamic analysis provides rich behavioral information but may increase computational requirements and operational complexity.

2.5 Deep Learning and Explainable AI

Zhang et al. [13] investigated opcode n-gram sequences with self-attention-based deep learning architectures for ransomware classification. Their approach demonstrated the potential of deep learning for long opcode sequences but required computationally intensive processing.

Gulmez, Kakisim, and Sogukpinar [9] proposed XRank, an explainable deep-learning-based ransomware detection system using dynamic API, DLL, and Mutex sequences. CNN and XAI methods including LIME and SHAP were used. The study reported high detection performance and demonstrated the value of explainability in ransomware detection.

Aslan and Yilmaz [26] investigated a hybrid deep-learning framework using malware images and transfer learning. Although strong classification results were reported, the approach requires relatively high computational resources and is not specifically designed around Windows PE static characteristics.

The literature therefore demonstrates the potential of deep learning and XAI while also motivating lightweight and interpretable static feature-based approaches.

2.6 Malware Detection Beyond Windows Ransomware

Shatnawi et al. [10] investigated static feature analysis and machine learning for Android malware detection. Although the study is not specifically focused on Windows ransomware, it demonstrates the general applicability of static characteristics for malicious-software classification.

Ahmed et al. [24] investigated Android ransomware using supervised machine learning and deep learning. Their work demonstrates the importance of feature selection and model comparison but operates in a different platform and feature environment.

The difference between Android and Windows PE environments highlights the importance of developing platform-specific ransomware detection approaches.

2.7 Ransomware Defense and Generalization

Kapoor et al. [14] proposed the Detection-Avoidance-Mitigation (DAM) framework as a theoretical approach for organizing ransomware defense strategies. The work emphasizes that ransomware protection requires multiple complementary mechanisms rather than a single detection technique.

Fernando, Komninos, and Chen [19] reviewed machine-learning and deep-learning approaches while considering malware evolution and concept drift. Their work emphasizes that detection performance may decrease as ransomware characteristics change over time.

Wahab et al. [20] investigated an AI/ML-based hybrid framework using behavioral features, supervised learning, explainability, and anomaly detection. Their work demonstrates the growing importance of combining classification, anomaly detection, and interpretability.

Subha et al. [21] investigated AI/ML-based ransomware prediction and detection in healthcare environments and highlighted the need for proactive approaches against evolving ransomware.

Delshadi et al. [22] reviewed AI-based ransomware detection across machine learning, deep learning, anomaly detection, static analysis, dynamic analysis, and XAI.

Qasim and Lu [23] reviewed ransomware detection approaches and emphasized the importance of diverse datasets, robust evaluation, explainability, and adaptation to evolving attacks.

2.8 RESEARCH GAP



The literature review identifies several important gaps.

Gap 1: Limited family-disjoint evaluation

Many studies report performance using conventional train-test splits. Such evaluation may not adequately demonstrate performance on ransomware families that were completely absent during training.

Gap 2: Dependence on complex architectures

Several existing approaches use deep learning, dynamic analysis, sandboxing, or hybrid processing pipelines. Although powerful, these methods may require substantial computational resources.

Gap 3: Limited explainability

High classification performance does not necessarily explain why an executable was classified as ransomware. Explainability is important for security analysts.

Gap 4: Limited novelty analysis

Supervised models generally classify samples according to known class distributions. Samples with substantially different characteristics may require an additional novelty or anomaly detection mechanism.

Gap 5: Dataset-dependent performance

Reported results can vary significantly depending on the dataset, feature representation, family distribution, and experimental protocol.

Gap 6: Family-level variation

Overall accuracy can hide poor performance on smaller ransomware families. Family-level analysis is therefore necessary for understanding generalization.

2.9 RESEARCH CONTRIBUTIONS

Based on the identified gaps, this research makes the following contributions:

1. A static Windows PE ransomware detection framework based on machine learning.
2. A compact eight-feature representation for executable classification.
3. Comparative evaluation of five machine learning algorithms.
4. Validation-based probability threshold analysis.
5. Family-disjoint evaluation using previously excluded ransomware families.
6. Family-level generalization analysis.
7. Isolation Forest-based novelty detection for unseen ransomware samples.
8. SHAP-based model explainability.
9. A practical static executable detection prototype.

The main contribution is therefore not a new machine learning algorithm. Instead, the contribution lies in the **integrated evaluation framework combining classification, threshold analysis, unseen-family testing, anomaly detection, and explainability using a compact static PE representation.**



3. PROPOSED METHODOLOGY

3.1 Overall Framework

The proposed framework follows the pipeline:

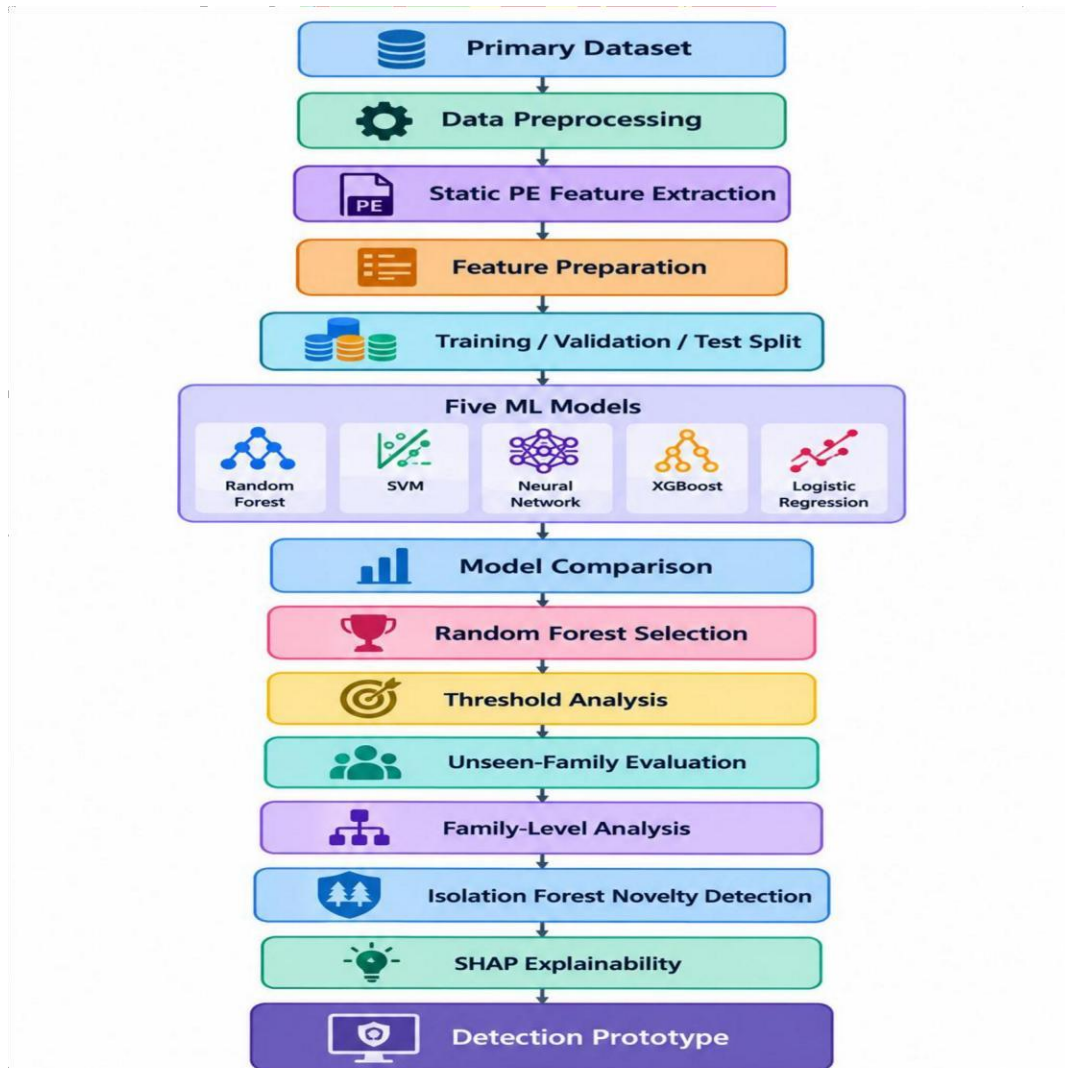


Figure 1. Overall workflow of the proposed AI-based ransomware detection framework.

The framework consists of sequential stages including dataset preprocessing, static PE feature extraction, machine learning model comparison, Random Forest selection, threshold analysis, unseen-family evaluation, family-level analysis, novelty detection, SHAP-based explainability, and prototype development.

3.2 Primary Dataset

The primary dataset used in this research contains Windows PE executable samples belonging to two classes:

- Benign
- Ransomware

The dataset contains approximately **215,404 samples**, consisting of approximately:

- **110,788 benign samples**
- **104,616 ransomware samples**

The dataset also provides ransomware-family information that enables family-disjoint evaluation.



3.3 Data Preprocessing

The preprocessing stage includes:

1. Dataset inspection.
2. Selection of relevant records.
3. Cleaning of inconsistent records.
4. Label preparation.
5. Feature type verification.
6. Static feature extraction.
7. Generation of machine-learning-ready datasets.

The final target label is:

0 = Benign

1 = Ransomware

No claim is made that a specific oversampling or undersampling algorithm was applied unless explicitly implemented in the experimental code.

3.4 Static PE Feature Extraction

The framework uses static analysis, meaning that executable files are analyzed without executing them. The final feature representation consists of eight features:

Feature	Description
API_Count	Total number of imported API functions
DLL_Count	Number of imported DLLs
Unique_API_Count	Number of unique imported APIs
Unique_DLL_Count	Number of unique imported DLLs
Size	Executable file size
Packed	Packing indicator
Entropy	Entropy of executable content
Year	PE timestamp year

The feature vector is:

$X = [\text{API_Count}, \text{DLL_Count}, \text{Unique_API_Count}, \text{Unique_DLL_Count}, \text{Size}, \text{Packed}, \text{Entropy}, \text{Year}]$
 $X = [\text{API_Count}, \backslash \text{DLL_Count}, \backslash \text{Unique_API_Count}, \backslash \text{Unique_DLL_Count}, \backslash \text{Size}, \backslash \text{Packed}, \backslash \text{Entropy}, \backslash \text{Year}]$

The target variable is: $Y \in \{0,1\}$ where 0 represents benign and 1 represents ransomware.

3.5 Standard Train-Validation-Test Strategy

The standard dataset is divided into:

Dataset	Samples	Validation	32,311
Training	150,782	Test	32,311



The training set is used for model fitting.

The validation set is used for model-related decisions, particularly probability threshold selection. The independent test set is reserved for final evaluation.

The standard test set contains:

- 16,619 benign samples
- 15,692 ransomware samples

This separation prevents the final test set from being directly used for threshold selection.

3.6 Unseen-Family Evaluation

A family-disjoint evaluation is performed to determine how the selected model behaves when ransomware families are completely excluded from training.

The experiment contains:

- **427 ransomware families represented in training**
- **106 ransomware families reserved as unseen families**

No ransomware-family overlap is permitted between the corresponding training and unseenfamily test groups. This experiment provides a stronger generalization test than a conventional random split.

3.7 Machine Learning Models

Five supervised machine learning algorithms are evaluated.

Logistic Regression

Logistic Regression provides a baseline linear classification model.

Decision Tree

Decision Tree provides an interpretable nonlinear classification model.

Random Forest

Random Forest combines multiple decision trees and is capable of modeling nonlinear relationships between static PE features.

XGBoost

XGBoost uses gradient boosting to construct an ensemble of decision trees sequentially.

LightGBM

LightGBM provides an efficient gradient-boosting-based tree ensemble for comparison with Random Forest and XGBoost.

All models are evaluated using the same final feature representation.

3.8 Probability Threshold Selection

The probability threshold determines the probability above which a sample is classified as ransomware. Thresholds between 0.10 and 0.90 were examined on the validation set.

A lower threshold generally increases recall but can increase false positives. A higher threshold can reduce false positives but increase false negatives.

The validation analysis identified **0.50** as the threshold producing the highest validation F1score. The selected threshold is therefore:

$$T=0.50 \quad T = 0.50$$

The threshold was selected using the validation data before final independent test evaluation.

3.9 Evaluation Metrics

The following metrics are used.

**Accuracy**

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

Precision

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

Recall

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

F1-score

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

False Positive Rate

$$\text{FPR} = \frac{FP}{FP + TN}$$

False Negative Rate

$$\text{FNR} = \frac{FN}{FN + TP}$$

ROC-AUC

ROC-AUC measures the ability of the model to distinguish benign and ransomware samples across different classification thresholds.

3.10 Isolation Forest Novelty Detection

An Isolation Forest component is used to investigate whether some unseen ransomware samples have characteristics substantially different from the known ransomware distribution.

The model is trained using known ransomware samples and then applied to unseen-family ransomware samples. The unseen evaluation contains:

- 52,169 unseen ransomware samples.
- 4,192 samples identified as anomalous.
- 47,977 samples identified as non-anomalous. The anomaly detection rate is:

$$\frac{4192}{52169} \times 100 \approx 8.04\%$$

This **8.04% value is an anomaly/novelty detection rate, not classification accuracy.**

A complete anomaly-detection performance evaluation would require additional control experiments; therefore, this study reports the result conservatively as an anomaly rate.

3.11 SHAP Explainability

SHAP (SHapley Additive exPlanations) is incorporated to investigate the contribution of individual static features to Random Forest predictions.

The analysis supports:

- Global feature importance.
- Feature contribution analysis.
- Individual prediction explanation.
- Identification of characteristics associated with model decisions.

SHAP results are interpreted based on the actual generated SHAP output. Specific feature rankings should not be claimed unless supported by the final SHAP results.

3.12 Detection Prototype

A prototype application was developed to demonstrate the practical use of the trained model.

The prototype accepts a Windows PE executable as input and extracts the required static features. The trained Random Forest model then produces a ransomware probability and classification result using the selected threshold.

The system can additionally use the Isolation Forest result to indicate whether the sample appears unusual relative to the known ransomware distribution.

The prototype is intended as a research demonstration of the proposed framework and should not be interpreted as a production-grade endpoint security system.

4. EXPERIMENTAL SETUP



4.1 Software Environment

The experiments were implemented using:

- Python 3.11.9
- Pandas
- NumPy
- Scikit-learn
- XGBoost
- LightGBM
- SHAP
- Matplotlib
- Seaborn
- Imbalanced-learn where required by the project environment

The project was implemented in a Windows-based Python virtual environment.

4.2 Dataset Processing Files

The major processed datasets include:

- benign_clean.csv
- ransomware_clean.csv
- api_features.csv
- ml_features.csv
- standard_train.csv
- standard_validation.csv
- standard_test.csv
- train_unseen_family.csv
- test_unseen_family.csv

The final machine-learning representation consists of eight static features and one binary target variable.

4.3 Model Configuration

The Random Forest configuration includes:

- Number of estimators: 200 Random state: 42
- Parallel processing: enabled

The unseen-family split uses a fixed random seed of 42.

Approximately 20% of ransomware families were reserved for unseen-family testing.

5. RESULTS AND DISCUSSION

5.1 Standard Model Comparison

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression	79.26%	77.65%	80.47%	79.03%	85.85%
Decision Tree	93.22%	92.95%	93.10%	93.03%	98.37%
Random Forest	95.42%	96.22%	94.28%	95.24%	98.98%
XGBoost	93.70%	94.18%	92.77%	93.47%	98.36%
LightGBM	93.75%	94.28%	92.77%	93.52%	93.22%

Random Forest achieved the strongest overall performance among the five evaluated algorithms.

Its F1-score was higher than the other models, while its ROC-AUC of 98.98% indicates strong class-separation capability under the standard test setting.

5.2 Random Forest Performance



The Random Forest model achieved:

- Accuracy: **95.42%**
- Precision: **96.22%**
- Recall: **94.28%**
- F1-score: **95.24%**
- ROC-AUC: **98.98%**

The high precision indicates that a relatively small proportion of files classified as ransomware were actually benign. The recall of 94.28% indicates that most ransomware samples in the independent test set were correctly detected. However, these results represent performance under the primary experimental dataset and should not be interpreted as guaranteed performance against future ransomware.

5.3 Threshold Analysis

Threshold	Accuracy	Precision	Recall	F1-score	FPR	FNR
0.10	89.02%	82.22%	98.74%	89.73%	20.16%	1.26%
0.20	92.85%	88.72%	97.70%	92.99%	11.73%	2.30%
0.30	94.36%	92.11%	96.67%	94.34%	7.82%	3.33%
0.40	95.02%	94.28%	95.54%	94.90%	5.48%	4.46%
0.50	95.21%	95.66%	94.41%	95.03%	4.04%	5.59%
0.60	94.96%	96.76%	92.73%	94.70%	2.93%	7.27%
0.70	94.35%	97.73%	90.48%	93.96%	1.99%	9.52%
0.80	93.41%	98.57%	87.71%	92.82%	1.20%	12.29%
0.90	91.36%	99.24%	82.85%	90.31%	0.60%	17.15%

The threshold of **0.50** produced the highest validation F1-score of approximately **95.03%**. Therefore, 0.50 was selected without using the final test set for threshold optimization.

5.4 Unseen-Family Evaluation

The family-disjoint evaluation produced:

Metric	Standard Test	Unseen-Family Test
Accuracy	95.42%	90.20%
Precision	96.22%	98.88%
Recall	94.28%	87.02%
F1-score	95.24%	92.57%
ROC-AUC	98.98%	97.72%
FPR	3.50%	2.32%
FNR	5.72%	12.98%

The most notable change is the reduction in recall from **94.28% to 87.02%**.

This indicates that previously unseen ransomware families are more difficult to detect than samples evaluated under the standard test setting.

At the same time, precision increases to **98.88%**, indicating that the model produces relatively few false ransomware predictions in the unseen-family test.

These results demonstrate why family-disjoint evaluation is important for ransomware research.

5.5 Family-Level Generalization



The unseen-family experiment contains **106 ransomware families**. The mean family-level recall is approximately: **70.29%**

The median family-level recall is approximately: **80.73%**

The difference between the mean and median indicates substantial variation among ransomware families. Some families achieve high detection rates, while some small families exhibit very low recall.

Family-level results must therefore be interpreted carefully, especially for families containing only a small number of test samples.

This analysis demonstrates that sample-level aggregate metrics can hide substantial variation in performance between ransomware families.

5.6 Novelty Detection Results

Isolation Forest was applied to 52,169 unseen ransomware samples. The result was:

- Anomalous: **4,192**
- Non-anomalous: **47,977**
- Anomaly rate: **8.04%**

The result indicates that approximately 8.04% of unseen ransomware samples were sufficiently different from the known ransomware distribution to be flagged as anomalous.

This result should not be interpreted as anomaly-detection accuracy because the experiment does not provide a complete set of independently labelled anomaly and non-anomaly controls.

5.7 SHAP Explainability

SHAP analysis was performed on the Random Forest model to improve interpretability.

The SHAP analysis provides an explanation of how the eight static features contribute to the model's output. The analysis can be presented using:

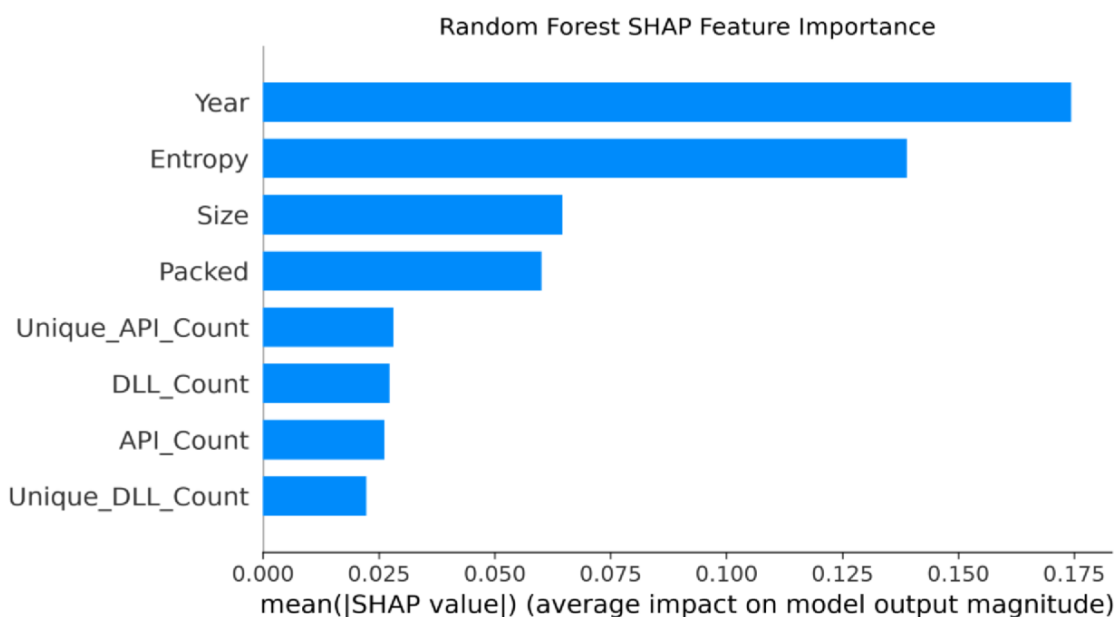


Figure 1. SHAP Global Feature Importance

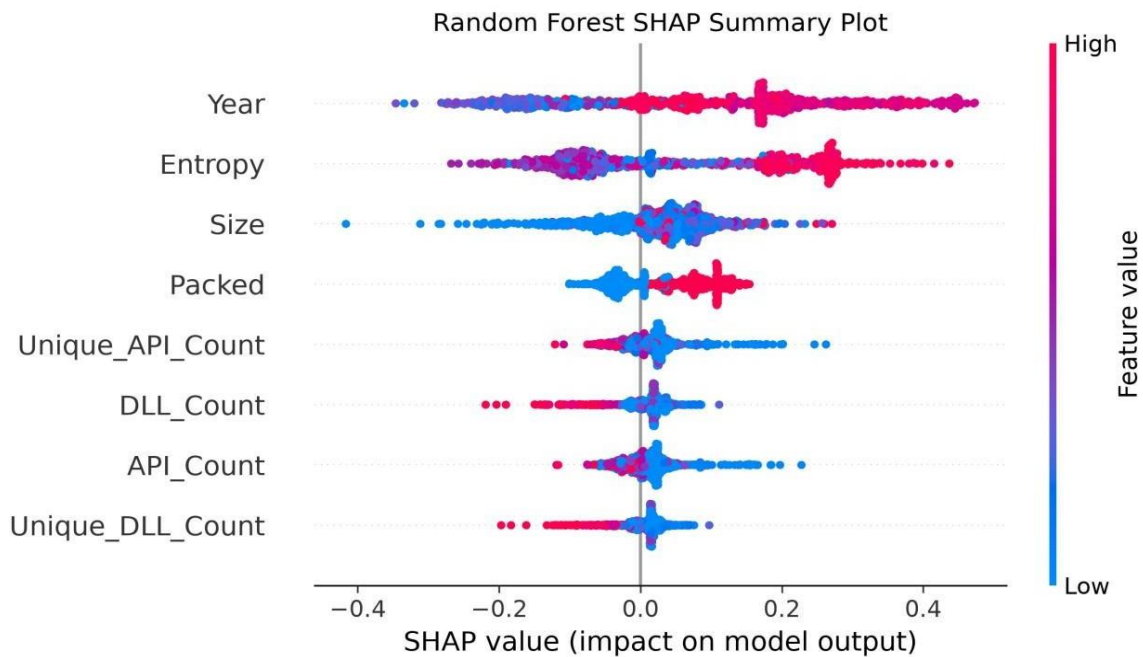


Figure 2. SHAP Summary Plot

The final discussion of feature importance should be based directly on the generated SHAP output. This approach allows the system to provide more information than a simple binary classification result and can support further investigation of suspicious executable characteristics.

5.8 Discussion

The experimental results demonstrate several important findings. First, Random Forest achieved the strongest performance among the evaluated models. This suggests that nonlinear ensemble learning is suitable for the selected compact static PE feature representation. Second, the difference between standard and unseen-family performance demonstrates that conventional random train-test evaluation may overestimate generalization capability. The reduction in recall from 94.28% to 87.02% indicates that previously unseen ransomware families remain challenging. Third, the high precision of the unseen-family experiment indicates that the model remains conservative when classifying samples as ransomware. Fourth, family-level analysis demonstrates that overall metrics do not fully describe model behavior across ransomware families. Finally, the Isolation Forest experiment demonstrates that supervised classification can be complemented with a novelty-detection component. However, the current anomaly experiment should be considered exploratory rather than a complete anomaly-detection benchmark.

6. APPLICATION PROTOTYPE

A prototype static ransomware detection application was developed as part of the framework. The application follows:

Upload PE File



Static Feature Extraction



Random Forest Prediction



Probability Threshold = 0.50



Benign / Ransomware Classification



Isolation Forest Novelty Analysis



Detection Result



The application provides the predicted ransomware probability and classification. The novelty-detection component can additionally indicate whether the sample exhibits characteristics that are unusual relative to the known ransomware distribution. The prototype demonstrates how the research model can be integrated into an executable analysis workflow. Because the framework is based on static analysis, the executable is analyzed without requiring normal execution of the potentially malicious file.

7. LIMITATIONS

The proposed framework has several limitations.

7.1 Static Analysis Limitation

The framework relies on static executable characteristics and therefore does not directly observe runtime behavior.

7.2 Compact Feature Representation

Only eight static features are used in the final machine-learning representation. Other PE characteristics may provide additional information.

7.3 Family Variation

Detection performance varies between ransomware families, especially families represented by small numbers of samples.

7.4 Novelty Detection Evaluation

The current Isolation Forest experiment reports an anomaly rate rather than complete anomaly detection accuracy.

7.5 Dataset Dependency

Although the primary dataset is large, the reported performance remains dependent on the dataset distribution and feature representation.

7.6 Independent External Validation

An independent external dataset was not included in the current experimental results because a suitable independently sourced dataset compatible with the exact eight-feature representation was not established.

This limitation is explicitly acknowledged rather than reporting an unsupported secondary validation result.

8. FUTURE WORK

Future research can extend the proposed framework in several directions.

1. **Independent cross-dataset validation** using compatible Windows PE datasets.
2. **Additional static PE features**, including section characteristics, header information, imports, and structural properties.
3. **Dynamic analysis** to incorporate runtime behavioral characteristics.
4. **Improved unknown-ransomware detection** through hybrid supervised and unsupervised learning.
5. **Temporal evaluation** using ransomware samples from different time periods.
6. **Adversarial robustness testing** against modified and obfuscated ransomware.
7. **Family-specific modeling** for difficult or poorly detected ransomware families.
8. **Lightweight deployment** for endpoint environments.
9. **Extended XAI analysis** using SHAP and complementary explainability techniques.
10. **Independent secondary validation** using a suitable external dataset.
11. **Continuous model updating** to address ransomware evolution and concept drift.

9. CONCLUSION

This research presented an **AI-Based Ransomware Detection Framework Using Machine Learning** for identifying ransomware from static Windows Portable Executable characteristics. The framework compared Logistic Regression, Decision Tree, Random Forest, XGBoost, and LightGBM using a common eight-feature representation consisting of API count, DLL count, unique API count, unique DLL count, file size, packing status, entropy, and PE timestamp year. Among the evaluated models, Random Forest achieved the strongest standard-test performance, obtaining **95.42% accuracy, 96.22% precision, 94.28% recall, 95.24% F1score, and 98.98% ROC-AUC**. The study also demonstrated the importance of family-disjoint evaluation. When tested on 106 ransomware families excluded from training, Random Forest achieved **90.20% accuracy, 98.88% precision, 87.02% recall, 92.57% F1-score, and 97.72% ROC-AUC**.



The decrease in recall under the unseen-family experiment demonstrates that strong conventional test performance does not necessarily imply equivalent performance on previously unseen ransomware families. Family-level analysis further showed substantial variation between ransomware families. Isolation Forest identified approximately **8.04% of unseen ransomware samples as anomalous**, providing an additional novelty-analysis perspective.

SHAP explainability was incorporated to improve interpretation of model predictions and provide insight into the contribution of static PE characteristics. Overall, the results demonstrate that a compact static PE representation combined with ensemble machine learning can provide strong ransomware classification performance while supporting additional evaluation of generalization, novelty detection, and interpretability.

The research does not claim that the proposed framework completely solves ransomware detection. Instead, it provides an experimentally evaluated framework that combines conventional classification with family-disjoint testing, novelty analysis, and explainability. Independent external validation, dynamic analysis, adversarial robustness, and temporal evaluation remain important directions for future research.

REFERENCES

- [1] D. Smith, S. Khorsandroo, and K. Roy, "Machine Learning Algorithms and Frameworks in Ransomware Detection," *IEEE Access*, vol. 10, pp. 117597–117610, 2022, doi: **10.1109/ACCESS.2022.3218779**.
- [2] Kritika, "Comprehensive Literature Review of Deep Learning-based Ransomware Detection," 2025.
- [3] M. Aggarwal, "Ransomware Attacks, Trends, Detection Techniques, Attack Vectors and Prevention Strategies," 2023.
- [4] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "AI-Based Ransomware Detection: A Comprehensive Review," *IEEE Access*, vol. 12, pp. 136666–136695, 2024, doi: **10.1109/ACCESS.2024.3461965**.
- [5] S. H. Almotiri, "AI-Based Malware/Ransomware Detection Using Machine Learning and Deep Learning Approaches," 2025.
- [6] M. Rele, J. Samuel, D. Patil, and U. Krishnan, "AI and Machine Learning-Based Ransomware Detection Using System and Security Data," 2025.
- [7] S. Poudyal et al., "Ransomware Detection Using Reverse Engineering, Feature Extraction and Machine Learning," 2018.
- [8] S. Singh, T. Khanna, and D. K. Verma, "Ransomware Detection Using Feature Engineering, PCA and Hybrid Machine Learning," 2025.
- [9] S. Gülmez, A. Gorgulu Kakisim, and I. Sogukpinar, "XRan: Explainable Deep LearningBased Ransomware Detection Using Dynamic Analysis," *Computers & Security*, vol. 139, 103703, 2024, doi: **10.1016/j.cose.2024.103703**.
- [10] A. S. Shatnawi, A. Yateem, et al., "Static Feature Analysis and Machine Learning for Android Malware Detection," 2022.
- [11] S. Poudyal, D. Dasgupta, Z. Akhtar, and K. Datta Gupta, "A Multi-Level Ransomware Detection Framework Using Natural Language Processing and Machine Learning," *Proceedings of the International Conference on Malicious and Unwanted Software (MALCON)*, 2019.
- [12] M. M. Ahmadian and H. R. Shahriari, "2entFOX: A Framework for High Survivable Ransoms Detection," *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, 2016, doi: **10.1109/ISCISC.2016.7736455**.
- [13] B. Zhang, W. Xiao, X. Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Static Analysis of N-Gram Opcode Sequences for Ransomware Detection Using Self-Attention-Based Deep Learning," 2020.
- [14] A. Kapoor, A. Gupta, R. Gupta, S. Tanwar, G. Sharma, and I. E. Davidson, "Ransomware Detection, Avoidance, and Mitigation Scheme: A Review and Future Directions," *Sustainability*, vol. 14, no. 1, p. 8, 2022, doi: **10.3390/su14010008**.
- [15] M. Medhat, S. Gaber, and N. Abdelbaki, "A New Static-Based Framework for Ransomware Detection," 2018



IEEE 16th International Conference on Dependable, Autonomic and Secure Computing / Pervasive Intelligence and Computing / Big Data Intelligence and Computing / Cyber Science and Technology Congress, 2018, doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00124.

- [16] A. Arabo, R. Dijoux, T. Poulain, and G. Chevalier, "Detecting Ransomware Using Process Behavior Analysis," *Procedia Computer Science*, vol. 168, pp. 289–296, 2020, doi: **10.1016/j.procs.2020.02.249**.
- [17] M. Medhat, S. Gaber, and N. Abdelbaki, "Static-Based Ransomware Detection Using YARA Rules," 2018.
- [18] S. Poudyal and D. Dasgupta, "AI-Powered Hybrid Static and Dynamic Analysis for Ransomware Detection," 2020.
- [19] D. W. Fernando, N. Komninos, and T. Chen, "A Study of Machine Learning and Deep Learning Approaches for Ransomware Detection and Concept Drift," 2020.
- [20] A. Wahab, Z. Sajid, H. Bux, E. Ahmed, A. M. Brohi, M. Tahir, A. K. K. Baig, and S. Ahmed, "AI/ML-Based Hybrid Framework for Ransomware Detection Using Behavioral Features, Explainability and Anomaly Detection," 2025.
- [21] D. T. Subha, S. B. Anwar, R. H. Rimom, M. J. Hoque, S. M. Zubair, and T. Fatema, "AI/ML-Based Ransomware Prediction and Detection in Healthcare Environments," 2026.
- [22] A. M. Delshadi, O. Ullah, Y. Khan, M. W. Iqbal, H. A. B. Muhammad, K. Hamid, F. Abbas, and M. Ibrar, "Artificial Intelligence-Based Ransomware Detection: A Review," 2025.
- [23] H. Qasim and Y. Lu, "Systematic Evaluation of Ransomware Detection Approaches Using Machine Learning and Deep Learning," 2026.
- [24] A. A. Ahmed, A. Shaahid, F. Alnasser, S. Alfaddagh, S. Binagag, and D. Alqahtani, "Machine Learning and Deep Learning-Based Android Ransomware Detection," *Sensors*, vol. 24, 189, 2024.
- [25] A. Banu, "Artificial Intelligence and Machine Learning for Ransomware Detection and Prevention," 2024.
- [26] Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning and Transfer Learning," 2021.