STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–20.
Published online in International Academic Press (www.IAPress.org)

Wrapper-Based Categorical Feature Prioritization: Enhancing Ransomware Detection Agility for Resource-Constrained Infrastructure

Omar CHAIEB [1,*], Mohammed Berhili [1], Nabil Kannouf [2], Mohammed Benabdellah [1]

[1] *LIPIO Laboratory, SIOSI Team, Faculty of sciences, University Mohammed Premier,Oujda,Morocco*
[2]*LSA Laboratory, SOVAI Team, National School of Applied Sciences, Al Hoceima,Morocco*

Abstract Ransomware has become a growing menace to online economies and critical infrastructure in nearly every part of the world. Although useful in detecting new ransomware variants, behavioral detection techniques tend to produce high-dimensional feature spaces, which are computationally expensive and cannot be deployed in real time. To overcome these problems, we propose a new four-stage category-aware wrapper-based feature selection framework for binary ransomware classification. Unlike our previous two-stage approach that relied on a single evaluation algorithm, the proposed method analyzes individual categories of features with several learning algorithms, optimizes category sets, ranks features in the chosen category sets, and uses wrapper-based selection to generate small, informative sets. The proposed approach was able to produce high detection performance and achieve a dramatic dimensionality reduction using a dataset of 30,967 features structured into seven behavioral categories. The Logistic Regression wrapper combined with ensemble learning achieved 98.03% accuracy while an SVM wrapper reduced the feature space by 99.64%. In addition to increased accuracy, the selected subsets have substantial efficiency benefits. In this case, in the ensemble learning classifier the proposed method was 45-fold faster in training and 270-fold faster in testing than the full set of features. These results indicate that categorical feature prioritization enhances binary detection ability as well as computational efficiency, making this method especially applicable in resource-constrained settings.

Keywords Ransomware detection, Feature selection, Machine learning, High dimensional data, Computational Efficiency, Resource-Constrained Environments.

DOI: 10.19139/soic-2310-5070-4147

1. Introduction

Ransomware cases have been on the rise in the last few years and have become more complex and challenging to both individuals and businesses [1]. The average recovery spending in the world amounted to 2.73 million dollars in 2024 with a significant global impact. Ransomware is no longer a technical inconvenience but a huge crisis to organizational stability, with recent evaluations showing substantial losses exceeding 3$ billion to cybercrime since 2019, making such attacks one of the most predatory economic forces globally [2]. In addition, attackers are constantly improving their capabilities by spreading Ransomware-as-a-Service [3], thus allowing even amateur cybercriminals to use advanced ransomware software and launch successful extortion campaigns [4]. In addition, with the evolution of malware, such strains as LockBit 3.0 [5], BlackCat [6], and Royal have become harder to monitor and prevent as they can evade conventional detection systems and become threats to the very infrastructure, transforming digital vulnerability into a direct threat to the safety and economic stability of the population. In many global organizations, the changing nature of threats requires them to create efficient, computationally effective detection systems.

*Correspondence to: LIPIO Laboratory, SIOSI Team, Faculty of sciences, University Mohammed Premier,Oujda,Morocco

ISSN 2310-5070 (online) ISSN 2311-004X (print)

In order to combat the evolving and sophisticated nature of ransomware attacks, Artificial Intelligence (AI) offers a strong alternative to traditional detection strategies [7],[8],[9] demonstrating impressive effectiveness in detecting zero-day threats and minimizing the use of signature-based detection [10]. AI algorithms may make use of advanced analytical methods, such as the examination of bytecode and monitoring of API-calls [11] which are extracted during either static or dynamic malware analysis. Dynamic analysis, especially, proves to be better than the static and hybrid methods, but it produces large amounts of data that pose serious processing, storage, and analysis problems [12]. Therefore, automated feature-selection algorithms are needed to improve the accuracy of classification and decrease the computational complexity [13]. Although generic feature selection approach exists, not many are specifically designed for the inherently behavioral categories of ransomware.

The main goal of this study is to address the challenge of having to reconcile high dimensional behavioral telemetry with the limited resources of edge computing devices by taking advantage of the categorical nature of ransomware behaviors. To directly address these challenges, the main contributions of this paper are summarized as follows:

- **Novel Four-Stage Hierarchical Framework:** We introduce a novel Four-Stage Hierarchical Framework that systematically analyzes behavioral categories such as API calls and Registry modifications across multiple machine learning models to avoid the volumetric bias of traditional global feature selection methods.
- **Resource-Constrained Edge Optimization:** We demonstrate that our framework significantly decreases computational and memory consumption, making it an essential first line of defense for Internet of Things (IoT) and edge devices where real-time speed and energy efficiency are of utmost importance.
- **Extreme Dimensionality Reduction:** We empirically prove that the proposed SVM-based wrapper achieves a 99.64% reduction in feature space with only 110 out of 30,967 features and still has very competitive detection accuracy.
- **Statistically Validated High Performance:** We establish that Logistic Regression wrapper with ensemble voting classifier yields a detection rate of 98.03% which is statistically validated, with confidence intervals and operational error rate analysis (FPR/FNR).

The organizational pattern of this manuscript is the following. Section 2 discusses related literature and gives background of ransomware detection, behavioral analysis and feature selection. Section 3 describes our methodology, such as data set, feature-selection procedure, and evaluation measures. Section 4 reports our experimental results, analyzes the effect of the selected features on the accuracy of different classifiers, and discusses the limitations of this study along with future research directions. Lastly, Section 5 summarizes the key findings of the manuscript.

2. Related work and background

2.1. Ransomware Detection using dynamic analysis

Dynamic analysis involves the execution of programs on controlled sandbox environments [14], [15] to log behavioral signatures, including API calls, registry changes, and file activities [12]. Such an approach provides useful information for the identification of ransomware, especially by exposing abnormal behavioral patterns that are more resistant to variability than fixed characteristics [16]. Recent research has contributed to dynamic analysis methods significantly. Authors in [17] proposed a system called XRan, which combined explainable Artificial Intelligence (XAI) strategies and Convolutional Neural Network (CNN) models, and achieved a high detection accuracy with detailed explanatory results. However, the quality of extracted features determined the effectiveness of XRan, which in turn required automated feature selection mechanisms. Furthermore, while modern deep learning-based ransomware detection approaches like CNNs achieve high performance, they typically demand substantial computational overhead. Our proposed wrapper-based method is highly preferable in resource-constrained environments because it delivers comparable predictive power while offering superior computational efficiency and maintaining clear model interpretability.

[18] trained ensemble models to classify Android malware based on CIC-AndMal-2020 dataset [19], [20] using a combination of a Random Forest Regressor and Recursive Feature Elimination (RFE) to select features and Principal Component Analysis (PCA) to reduce the number of dimensions, achieving a 95.0% accuracy across twelve malware categories. Although classical feature selection baselines such as RFE and PCA are prevalent in the literature, they generally treat all features uniformly and do not capitalize on the inherent behavioral categorical structure of ransomware data, an aspect our category-aware framework explicitly exploits to optimize the feature subset.

2.2. Feature Selection in High Dimensional Data

Ransomware detection behavioral data are inherently high-dimensional, thus creating a severe bottleneck that is characterized by over-fitting, high computational cost, and reduced model accuracy, an effect often known as the curse of dimensionality. The previous feature-selection procedures have produced mixed results, in terms of addressing these issues. As an example, [21] used Mutual Information (MI) as a filter-based method [39], with promising results; however, this method was limited by the need to pre-define the number of features and depended on the whole dataset to initialize the method, reducing flexibility. Similarly, [22] proposed a group-based paradigm of high dimensionality processing, where MI was used to rank features, which were afterwards combined with a Logistic Regression (LR) and Particle Swarm Optimization (PSO) wrapper. Although this method was completely automated, which eliminated the manual intervention, it still retained 823 features, which is more than the 400 features that were chosen in [21], and thus increased the computational footprint and could potentially hamper real-time responsiveness in critical operational scenarios.

In recent years, the literature has been increasingly concerned with lightweight feature selection and optimization protocols to directly address the strict constraints of edge computing and IoT. For example, a new multi-class malware detection system used a Genetic Algorithm (GA) and correlation analysis to extract the most relevant features [36]. In spite of the slightly lower accuracy of the model compared to the Random Forest, the model size was reduced to 0.5 MB by using SMOTE and SOM-US data balancing techniques, which is very suitable for resource-constrained devices [36]. Likewise, another recent work proposed an effective method for the resource-constrained devices by using an Extra Tree Classifier (ETC) with Gini impurity scores [37]. This approach was able to reduce the 55 features to the 10 most influential features and a Random Forest model was able to achieve 99.39% accuracy with drastically lower memory and execution time [37]. Moreover, alternative methods have optimized the feature space itself, such as a recent work that showed the effectiveness of post-training quantization using histogram on XGBoost models for IoT botnet detection [38]. This approach achieved up to 3.7 times smaller model size and three times faster inference speed with near-perfect accuracy, demonstrating the feasibility of gradient-boosted models in edge scenarios [38]. Even with these recent advances, many cutting-edge techniques still do not fully leverage the granular semantic clustering of malware behaviors.

In our earlier study [23], we addressed these limitations by proposing a wrapper-based feature reduction algorithm to detect ransomware and it aimed to determine the impact of various categories of features on the performance and efficiency of detection algorithms. The critical features were selected in two steps: initially, the most salient feature categories were identified, and then critical features within the selected category were prioritized and selected. The methodology had a 98.73% feature reduction and a 97.37% accuracy, which is a significant enhancement in the computational efficiency and classification performance compared to the original feature set.

The above approach, however, had several limitations which inspired the current research. First, the use of one Logistic Regression algorithm to assess the evaluation might have limited the strength and external validity of the feature-selection process. Second, no systematic analysis of category-combination was provided, so there were opportunities to optimize the composition of feature subsets not exploited. Lastly, the approach lacked adequate investigation of computational-efficiency optimization strategies and did not provide a thorough analysis of algorithmic sensitivity to gain an insight into how various machine-learning algorithms react to the choice of feature categories. Such weaknesses reveal the necessity of a more robust structure that validates various algorithms, so that the system can be reliable in various security settings.

3. Methodology

3.1. Dataset description

The Resilient Information Systems Security (RISS) dataset, which was gathered by [21], consisted of 582 ransomware samples of 11 different families, such as CryptLocker, CryptoWall, TeslaCrypt, and Reveton, and 942 benign applications, including browsers, utilities, games, and office tools. This distribution presents a natural class imbalance, which we explicitly addressed to prevent model bias by using stratified data partitioning and macro-averaged performance metrics The authors used an isolated Cuckoo Sandbox environment [14] to perform dynamic analysis of the applications and obtain different features. The exact nature of the high dimensional feature space used in this study is very important to consider. The first 30,967 features are not artificially expanded from a small number of categorical features. Rather, each binary indicator is a distinct raw behavioral artifact seen during the dynamic analysis, like invoking a particular API function, modifying a targeted registry key, dropping a file with a particular extension, or dropping a particular embedded string.

This extreme dimensionality reduction directly translates to an operational monitoring overhead reduction that is tangible, as a result of our category-aware framework. In a real-world scenario, an endpoint agent would only need to hook, intercept and log 392 system events instead of almost 31,000. This radical reduction in the amount of telemetry required also directly reduces the CPU hooking overhead, memory usage and log storage needs, further proving the framework's suitability for lightweight and resource constrained detection environments.

In addition, the 30,967 features are categorized into seven categories described below:

File Extensions: Are important in windows operating systems to identify the files with the applications. Some ransomware families, including GrandCrap, use this design by targeting a specific file extension to determine and encrypt user files related to productivity, such as documents and databases, without impacting system-critical executables to maintain the functionality of the OS to pay the ransom [24].

API Invocations: The underlying technology of the windows operating system is called API calls, which allows the operating system to interact with applications, allowing it to perform functions like file access and memory management [25]. However, ransomware attackers use such calls during the cyber-kill chain: an example is that CryptGenKey creates encryption keys during the encryption stage, FindFirstFile searches directories to find targets during enumeration, and CreateRemoteThread injects malicious code into legitimate processes during execution[11]. Therefore, API call analysis is essential to detect and prevent ransomware attacks before it is too late to recover critical information [26].

Embedded strings: Ransomware code uses the connection with particular servers, including command-and-control (C&C) servers, to download malicious code [27]. To make this activity possible, ransomware actors place fixed commands and IP addresses into their malware, thus making sure that the persistent communication channels do not go down after the initial infection. The mechanism allows the latter to execute the follow-up attacks, including the data exfiltration or system disruption, as part of a larger malicious operation scheme [28].

Registry key operations: Windows Registry is one of the main weaknesses that ransomware attackers use [29]. These malicious codes use various tactics to attain their objectives, such as altering the registry keys to boot themselves on system startup, encrypting information and ransom to recover it, and interrogating the registry keys to determine specific system settings [30].

File operations: Ransomware uses several file operations when it attacks a system to encrypt and keep user data as a ransom. After getting access, the malware identifies valuable user files and begins the encryption process by creating different encrypted files and deleting the original files, transferring files to a temporary folder to encrypt and then restoring them with new extensions, or overwriting files after encrypting their contents. During the attack, ransomware uses read operations to retrieve original file data, write operations to save the encrypted data, rename operations to add new extensions, and delete operations to delete access to the original files [31].

File directory activity: Ransomware attackers use file directory activity to gain access into systems, spread laterally, and encrypt vital information; thus, it is crucial to learn how attackers use directories to compromise systems.

Extensions of dropped files: Ransomware attackers tactfully use the file extensions of dropped files to avoid detection, run malicious code, and enhance the effects of their attacks [32].

As illustrated in Table 1, the dimensional distribution of features across the seven behavioral categories is highly imbalanced. For instance, the Embedded strings (STR) category contains more than half of the total feature space with16,267 features while the highly critical categories, such as API invocations (API) and Extensions of dropped files (DROP), contain only 232 and 346 features, respectively. This imbalance in volume of structure is one of the main causes of the failure of traditional global feature selection methods. When the number of STR and REG features is large and compared to the less frequent but more discriminative features such as specific API calls, then the former features are statistically dominant and a biased and sub-optimal feature subset is selected. We suggest a category-aware hierarchical approach which naturally addresses this imbalance. Our framework will enforce semantic boundaries and rank features strictly in their respective isolated categories in the early stages to ensure that critical behavioral indicators from minority categories are prioritized and preserved. This prevents bias in the algorithm towards the volumetrically dominant categories and guarantees a balanced semantic representation of the behavior of the ransomware in the optimized subset at the end.

Table 1. Description of the behavioral feature categories and their dimensional distribution.

Category code	**Signification**	**Number of features**
API	API invocations	232
REG	Registry key operations	6622
STR	Embedded strings	16267
DROP	Extensions of the dropped files	346
DIR	File directory operations	2424
FILE	File operations	4141
FILES_EXT	File extensions involved in file operations	935
Total		**30,967**

3.2. Experimental setup

All experiments were performed on a dedicated workstation with an Intel Core i5-6200U processor (2.30 GHz) and 8 GB RAM to ensure the full reproducibility of our methodology and the accuracy of the computational overhead analysis, including model training, feature selection and runtime benchmarking. Python 3.9.7 was used for the software environment. Pandas 1.3.4 and scikit-learn 0.24.2 libraries were used for data manipulation and machine learning operations, respectively. To preserve similar conditions with the related works, we did not conduct hyperparameter optimization. Besides, to obtain reproducibility, we applied the default scikit-learn parameters and ensured that we explicitly mentioned non-default values. Our implementation utilizes specific scikit-learn modules including LogisticRegression, SVC, tree.DecisionTreeClassifier, and RandomForestClassifier. Furthermore, the ensemble learning technique was explicitly constructed using a hard VotingClassifier. To ensure a comprehensive and unbiased evaluation, the base estimators of this ensemble are dynamically configured to include the three algorithms not currently utilized as the feature selection wrapper. The class imbalance was addressed by using a strict Stratified Train-Test split, with 80% of the data used for training and 20% of the data used for an isolated hold-out test set. This stratification resulted in the exact proportion of ransomware to benign samples and the proportional distribution of the underlying ransomware families being perfectly preserved for both partitions. To

strictly prevent any risk of data leakage, all feature selection steps, including Mutual Information ranking and the wrapper-based evaluations, were conducted exclusively on the 80% training set [41]. On the training data, we performed stratified 4-fold cross-validation and reported the accuracy, macro-precision, macro-recall, and macro-F1 as the means and standard deviations of the folds. Lastly, we present single-evaluation scores on the withheld 20% test set. in addition, for all wrapper algorithms, we set a random_state equal to 42. In order to make the statistical results of our evaluations on the single hold-out test set reliable, we used McNemar's test, which is the most commonly used and most robust statistical comparison method [40] for comparing the classification results of two machine learning models tested on the same test set. A significance threshold of $p < 0.05$ was used to determine statistical significance.

3.3. The feature selection framework

The proposed methodology is an expansion of the two-stage methodology that was used before to a four-stage hierarchical framework of feature-selection. This development seeks to overcome the weaknesses that were found in the previous work without losing the strengths inherent in it. Table 2 compares the key aspects of our previous and enhanced methodologies.

Table 2. Comparison of our previous and improved methodologies.

Aspect	**Previous Method**	**Enhanced Method**
Selection Process	Two stages (category selection then individual feature selection)	Four stages with statistical validation at each step.
Algorithm Evaluation	Single algorithm (LR) evaluation	Multi-algorithm evaluation (LR, SVM, DT) with sensitivity analysis
Computational Analysis	Limited efficiency reporting	Comprehensive profiling across multiple metrics (processing time, accuracy, reduction rate, F1 score, precision)

3.3.1. Stage 1: Individual Category Assessment The proposed approach starts with a systematic assessment of every feature category. The seven feature categories (API, REG, STR, DROP, DIR, FILE, FILE_SEXT) are individually evaluated with the LR, SVM and DT algorithms. This step corrects a methodological shortcoming of the previous paper, in which category effectiveness was only proven using the LR algorithm but not empirically validated using various algorithmic paradigms. Comparative analysis using three different algorithms, which offer different analytical insights about the effectiveness of categories, is carried out: LR offers linear relationship modelling and probabilistic evaluation, SVM offers complex boundary recognition and margin optimization while DT offers rule-based feature-importance analysis and non-linear pattern recognition.

3.3.2. Stage 2: The Optimization of categories subset Based on the results of the individual category rankings of Stage 1, category combinations are evaluated systematically through an iterative method. This is a great improvement of the previous technique that depended on fixed combinations of categories. The iterative approach reveals sensitivity of the algorithm to category combinations, which is a major insight that has not been investigated before. The algorithm used to select categories in the iterative category combination procedure is highlighted in Algorithm 1. In our experiments, the stopping criteria parameter nb_group was explicitly set to 7, corresponding to the total number of behavioral feature categories available in the RISS dataset.

3.3.3. Stage 3: Ranking of Individual Features The third phase focuses on the most appropriate characteristics in each group that was selected. The rankings of features in each of the chosen groups are computed separately using only the training dataset by applying Mutual Information (MI) that measures the statistical dependence of each of the features on the target variable, hence providing a measure of feature importance.

Algorithme 1 : Stage 2: feature category Selection

Data :
top_rank - Set of top-ranked groups evaluated and sorted in Stage 1
sel_func - Selection function (LR, SVM, DT)
nb_group - Number of groups (stopping criteria)
Result :
Best_subset - Optimal selected subset of category groups

Function `Group_Selection(`*top_rank, sel_func, nb_group*`)` **:**

```
Best_subset ← [top_rank[0], top_rank[1]]          // Initialize with top 2 groups
Best_score ← sel_func(Best_subset)                 // Initial best score
for i ← 2 to nb_group − 1 do
                                                   // Start from 3rd group
    Eval_subset ← Best_subset.copy()
    Eval_subset.append(top_rank[i])                // Add next-ranked group
    Eval_score ← sel_func(Eval_subset)
    if Eval_score > Best_score then
        Best_score ← Eval_score
        Best_subset ← Eval_subset.copy()
return Best_subset
```

3.3.4. Stage 4: Selecting Individual Features The last step uses wrapper-based feature selection, where each of the three algorithms (LR, SVM, DT) is used as the wrapper engine. This enhancement of the former wrapper strategy offers an overall assessment of generated feature subsets.

- **Subset Generation:** The subset generation begins with an empty set, and one feature of each chosen group is added to the set in succession.
- **Subset Evaluation:** 4-fold stratified cross-validation on the isolated training set was used to evaluate each candidate subset and the performance of the subset is defined as the mean accuracy in across all folds.
- **Stopping Criteria:** The process continues until the first 100 iterations are done per feature category, which is justified in section Discussion. Because this 100-feature threshold is applied individually to each chosen category from Stage 2, the final global feature count represents the aggregate sum of these category-specific subsets.
- **Choosing the Best Subset:** we choose the subset that performs the best in the evaluation process.

After the wrapper assessment, the effectiveness of the chosen characteristics was assessed with the help of other classification algorithms other than the wrapper technique to determine the generalizability of the methodology. Such assessments showed better performance compared to previous feature-selection techniques [21], [22], [23]. The four-step detection procedure is integrated to result in the workflow presented in Figure 1 whereby the identified features are given into machine-learning classifiers to identify the ransomware.

4. Results and discussion

4.1. Results

The first phase of our approach is the assessment of every category through the comparison of the performance of the LR, SVM, DT algorithms. All these algorithms are then applied to determine the most effective set of features

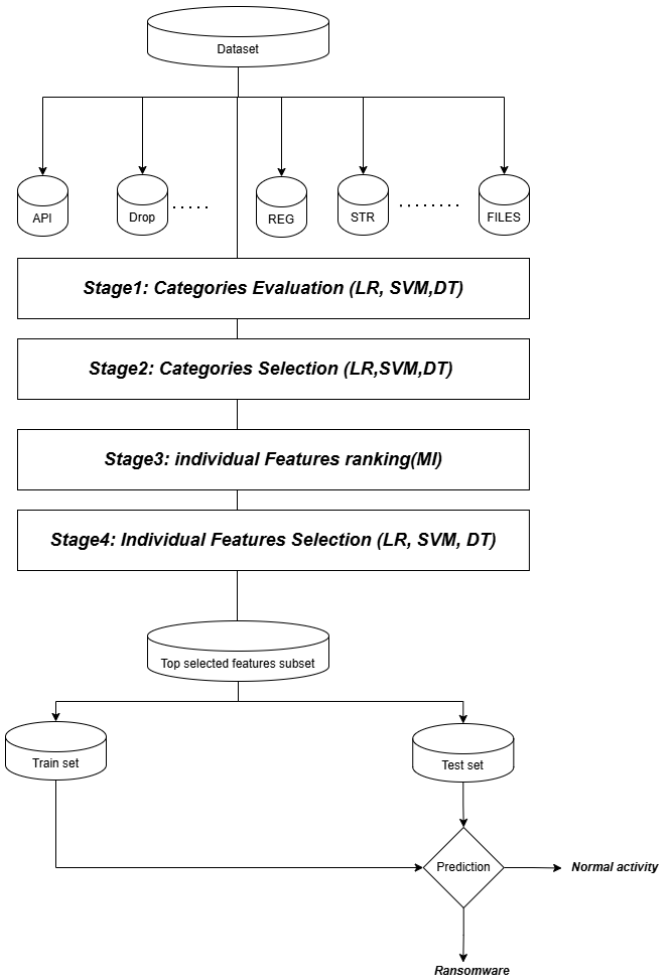

Figure 1. The general scheme of the intrusion detection process.

and are used as a wrapper algorithm in the last phase of the process of selecting the most relevant features. Figure 2 illustrates the accuracy of each evaluation algorithm when trained on each of the single feature categories. As depicted, behavioral characteristics associated with direct system interactions exhibit the highest predictive power. The API features showed better performance in all the algorithms with an accuracy of 93.60%, 94.25% and 94.42% respectively with DT, SVM, and LR. This was followed by features of STR (91.13% with DT, 90.72% with SVM, and 90.56% with LR) and REG features (87.69% with DT and 88.51% with LR). These findings form a definite order of feature categories in terms of their respective prediction ability and form the foundation of our feature-selection approach.

The second stage involved the repeated combination of feature categories to find the best combinations to use with the DT, SVM and LR algorithms. In the case of the DT algorithm, API + STR + FILES_EXT + FILES gave the highest accuracy of 96.22%. Notably, however, the addition of DROP, REG, or DIR categories did not lead to significant changes in the scores, which means that the valuable information is adequately represented once the FILES category has been added. Figure 2 shows the trends in accuracy as the categories were added incrementally with the help of the DT model. In the case of LR, the most suitable feature categories were API, STR, REG, and FILES_EXT, with an accuracy of 97.78%. The inclusion of other variables like DROP, FILES or DIR led to a minor reduction in accuracy as it dropped to 97.45%, as compared to 97.70%. Figure 4 shows these tendencies.

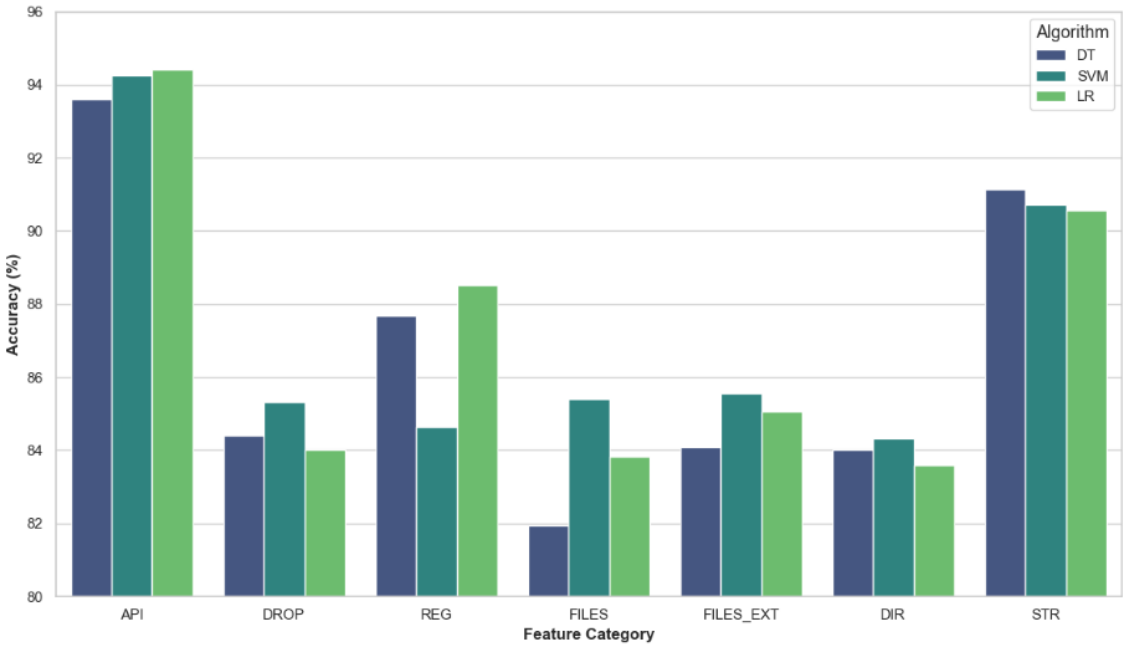

Figure 2. Classification accuracy of ML algorithms by individual feature category

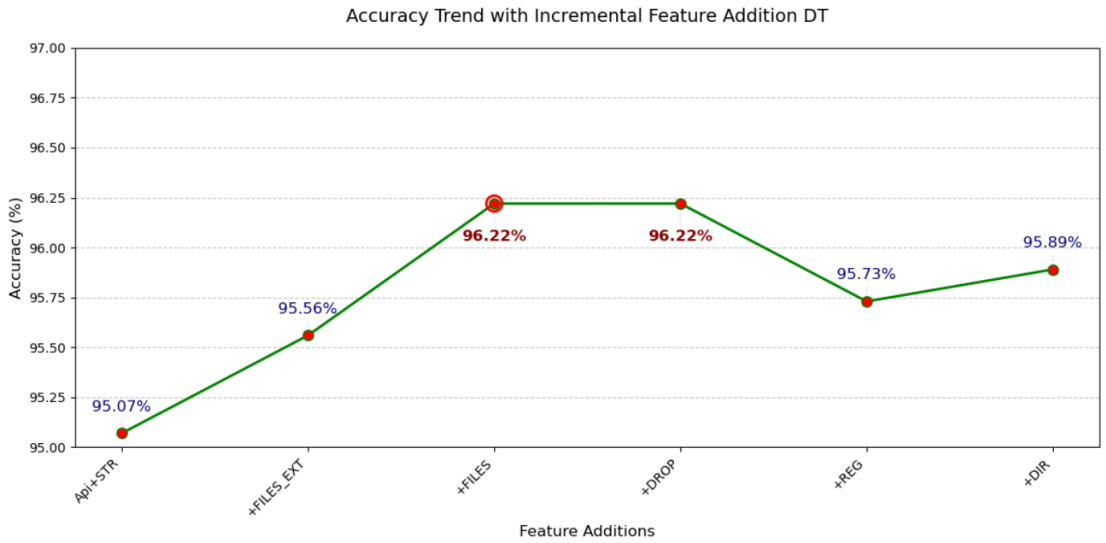

Figure 3. Accuracy trend with the addition of incremental features with DT model.

The optimal performance of SVM was 96.47% with a minimum API + STR combination with no extra features. SVM algorithm was also sensitive to the addition of features. In fact, addition of FILES_EXT, FILES or DROP slightly decreases the performance and addition of REG or DIR dramatically decreases accuracy to 95.57%. Figure 2 shows the trends in accuracy as the categories were added to the SVM model.

The third and fourth steps entailed the selection of individual features in the best combination of categories that were selected in stage two. We have used the wrapper-based feature-selection method and obtained significant dimensionality reduction with a high classification accuracy. The DT wrapper algorithm used 316 features, which

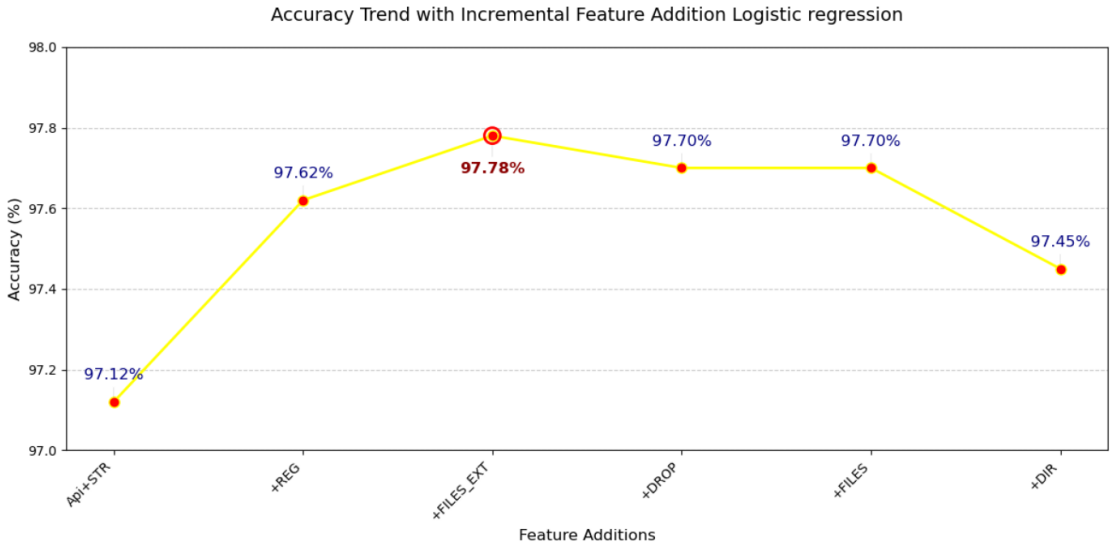

Figure 4. Accuracy trend with the addition of incremental features with LR model

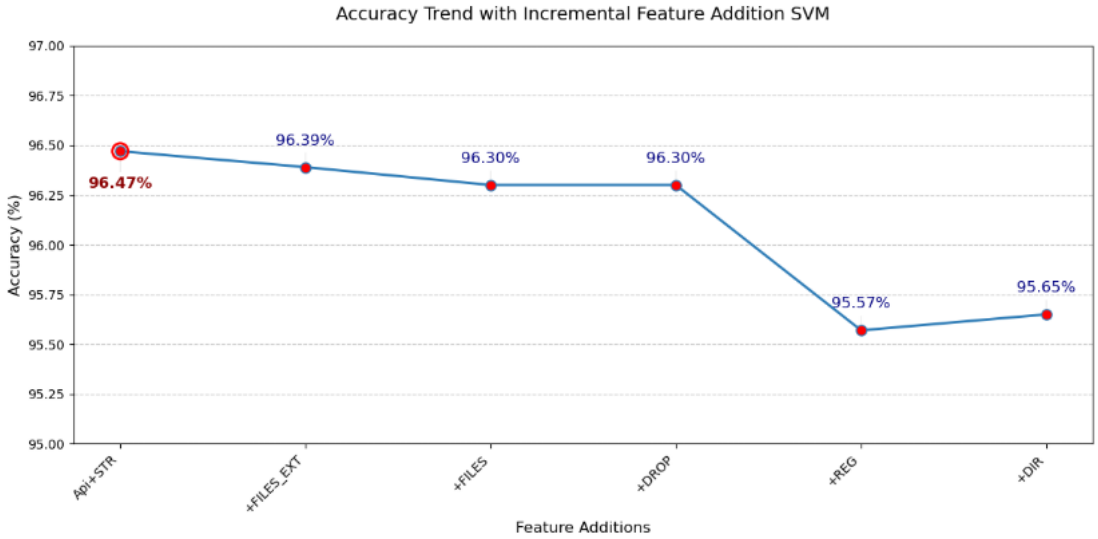

Figure 5. Accuracy trend with the addition of incremental features with SVM model.

is a 98.97% reduction, and 97.78% accuracy with the combination of Random Forest classifier. The SVM wrapper only selected 110 features, which represents a 99.64% reduction, but still had 97.29% accuracy with Random Forest. LR wrapper algorithm identified 392 features which is a 98.73% decrease and reached the overall accuracy of 98.03% when combined with a Voting Classifier ensemble learning.

Table 3 indicates that there is an evident accuracy-efficiency trade-off between wrapper strategies. To clarify our evaluation metrics, the reported accuracies in this paragraph represent the single evaluation scores on the withheld 20% test set. Although the LR Wrapper had the highest test accuracy of 98.03%, the proposed DT Wrapper was also competitive achieving 97.78% accuracy with a smaller number of features. In addition, it is interesting to note that the proposed SVM Wrapper has the highest reduction achieving 99.64%, with only 110 features and, at the same time, a competitive accuracy of 97.29%, which means that the majority of the discriminative information

is concentrated in a very small set of behavioral features. Comparatively, [22] retained 823 features with 97.33% reduction and lower accuracy with 97.06% ±0.21, whereas [21] used a filter method and only reported one accuracy score of 97.48% without providing precision, recall, or F1 measures. The Voting Classifier ensemble learning combined with the suggested feature-selection strategy has transformative benefits over the use of the entire dataset. Assessing the ensemble classifier, the entire set of 30,967 features has a prohibitive computational cost that takes 22.53 s to train and 8.12 s to test. The proposed categorical prioritization, which enabled 45-fold and 270-fold improvements without any detection integrity loss, brings these latencies down to 0.49 s and 0.03 s respectively.

Conversely, the hierarchical wrapper method suggested, was demonstrated to prove that speed need not be at the cost of accuracy. As an example, the LR wrapper in fact improves the performance to 98.03% when using only 392 features, the DT and SVM wrappers maintain high scores of 97.78% and 97.29% respectively, with significantly smaller sets of 316 and 110 features.

To ensure the statistical reliability of these results on the single hold-out test set, we used the standard binomial proportion formula, which resulted in a 95% Confidence Interval (CI) of $\pm$ 1.56% for the peak accuracy of our optimal ensemble of 98.03%.

Baseline Comparison using Identical Protocols

To provide a comprehensive evaluation of the performance of our method compared to standard feature selection methods, we used two baseline methods with our exact data split and validation protocol: a global Mutual Information (MI) filter and Recursive Feature Elimination (RFE). For a fair comparison, the 392 features chosen by both baselines were tested with our optimal Voting Classifier ensemble, and the final results were obtained by applying a single evaluation to the isolated hold-out test set. The global MI filter achieved an accuracy of 94.34%, showing that global methods are severely affected by volumetric feature bias. The RFE baseline achieved a higher accuracy of 97.78%. The highest empirical accuracy of 98.03% was obtained by our proposed category-aware LR wrapper ensemble. Most importantly, our framework is statistically equivalent to the intensive RFE baseline in terms of predictive performance, as confirmed by the McNemar's test with a p-value of 0.617.

In addition to predictive performance, standard RFE and MI are black-box approaches lacking categorical context and need to be manually preset with a target feature count before being applied. In contrast to these baselines, our proposed method does not need to have the number of features predetermined. We address these limitations with our category-aware four-stage framework, which maintains semantic interpretability, reduces volumetric bias across different behavioral threat vectors, and automatically selects subsets dynamically through performance saturation.

Supplementary Analysis: Hyperparameter Optimization To further validate the robustness of our framework and investigate its potential in the optimized conditions, we performed a supplementary experiment using Grid Search in a nested 4-fold cross-validation framework on our training partition alone. For our optimal LR Wrapper subset with the 392 features and the Voting Classifier ensemble, we systematically tuned key parameters across the base estimators, namely the regularization parameter (C) for SVM, the maximum depth for the Decision Tree and the number of estimators for the Random Forest.

The best model found in this inner cross validation was then tested on the single test of the isolated hold-out test set. The optimized model obtained an empirical accuracy of 97.64% at the end. Interestingly, this is a strong test score that is statistically equal to the best empirical accuracy of 98.03% obtained with default parameters. This is very important for the main goal of our paper, which is deployments in resource-constrained edge environments. It empirically shows that our feature subsets selected categorically are very discriminative. This means that security practitioners can get near-optimal, state-of-the-art ransomware detection with default algorithmic configurations, without incurring the computational burden and time requirements usually required for hyperparameter tuning.

One of the most noticeable results of the applied methodology is a significant reduction of processing overhead with testing latency decreasing by over 99.98%. The SVM wrapper, specifically, proved to be the most robust option, capable of maintaining a high level of detection accuracy, as well as significantly reducing the computational tax, which is normally linked with high-dimensional analyses. The results showed that the choice of strategic features serves as a filter to the digital noise, thus simplifying the system in resource-constrained settings and essentially enhancing the intelligence of the model. The correlation between efficiency and accuracy is shown in Figure 6, where the scores of the improved group accuracy are compared with the high-dimensional baseline.

Table 3. Performance Comparison of Feature Selection Methods and Best Classifiers

Classification Model	**Best Classifier**	**Accuracy (95% CI)**	**Precision**	**Recall**	**F1-Score**	**FPR**	**FNR**
Global MI (Filter Baseline)	Ensemble Learning	94.34% (± 2.59%)	94.18%	93.71%	93.92%	2.81%	6.30%
RFE (Embedded Baseline)	Ensemble Learning	97.78% (± 1.65%)	97.84%	97.43%	97.62%	1.69%	1.57%
DT Wrapper	Random Forest	97.78% (± 1.65%)	97.79%	97.48%	97.62%	2.25%	0.79%
SVM Wrapper	Random Forest	97.29% (± 1.82%)	97.17%	97.04%	97.10%	1.69%	2.36%
Proposed LR Wrapper	**Ensemble Learning**	**98.03% (± 1.56%)**	**98.03%**	**97.76%**	**97.86%**	**1.69%**	**1.57%**
[21]	RF	97.48%	-	-	-	-	-
[22]	RF	97.06%	-	-	-	-	-

This figure explicitly highlights that while the entire dataset offers high baseline accuracy, our optimized subsets achieve statistically equivalent predictive performance, with drastically reduced computational demands.

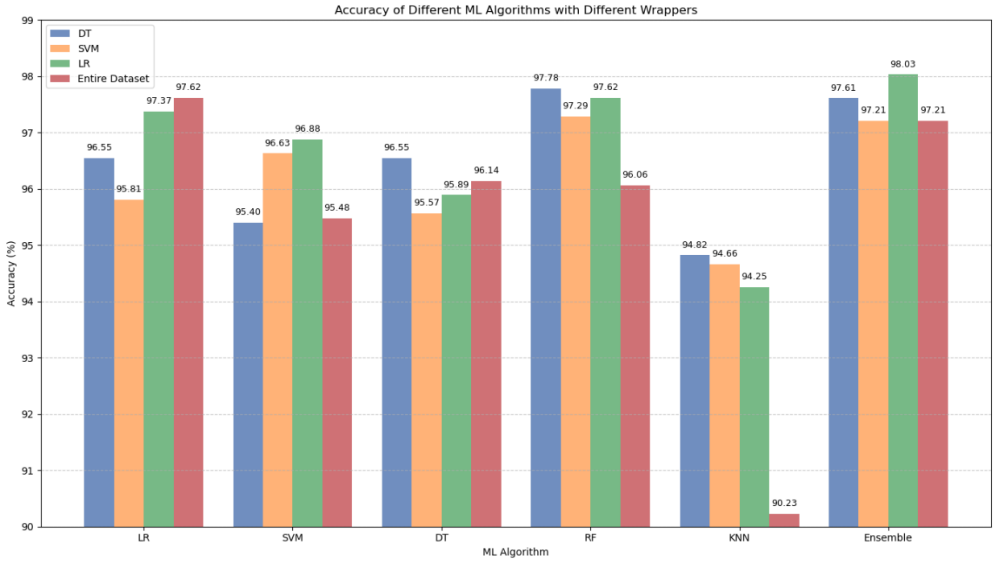

Figure 6. Performance Comparison of Wrapper Methods and the original dataset Across ML Classifiers

Besides the identified improvement in predictive accuracy, the feature selection process also resulted in a significant rise in computational efficiency. Figure 7 shows (a) the training time and (b) the testing time of selected features and the full dataset. In both cases, the subset of features chosen through the suggested category-based wrappers show significant improvements in efficiency as compared to the full dataset of 30967 features. Focusing on the high-performance ensemble learning classifier, the SVM wrapper offered a 45x speedup in training by reducing latency from 22.53s to 0.49s and a 270x speedup in testing by reducing latency from 8.12s to 0.03s. These findings confirm the hypothesis that the selection of specific categories of features can greatly decrease feature dimension without affecting detection accuracy.

4.2. Discussion

To be deployed in practice, the SVM Wrapper is most effective when minimizing memory consumption and maximizing real-time detection speed are the most important factors, since it requires only 110 features and still has a high accuracy near optimal. On the other hand, the LR Wrapper with the Voting Classifier ensemble learning is the most effective configuration in Table 3 when the main goal is to achieve the highest detection accuracy.

Operational Implications of Error Rates.

In the real world, it is not enough to measure classification performance based on overall accuracy. We have explicitly considered the False Positive Rate (FPR) and False Negative Rate (FNR) of our proposed framework as they have different and important operational implications. A high FPR can significantly impact the usability of the system, resulting in false-positive alerts and alert fatigue for security operations teams, which can lead to disruptions in business continuity. On the other hand, a high FNR can put infrastructure at risk of catastrophic data loss, because one variant of ransomware can infect the entire network. The empirical results show that our optimized ensemble model has an extremely low FPR of 1.69% and an FNR of 1.57%. We have chosen our feature subset to ensure that legitimate system workflows are not hampered while the edge nodes are protected from threats without increasing administrative burden.

Furthermore, the dominance of API-related characteristics in all the algorithms is in line with previous literature that recognizes API call patterns as one of the most indicative of ransomware activity [11], [33]. Ransomware normally leverages targeted API calls to file encryption, system manipulation and persistence controls that are

significantly different than those used by legitimate applications. Similarly, the high performance of STR features suggests that embedded strings capture useful semantic data related to malware functionality, which supports the previous findings that string features can be used to differentiate ransomware families [28]. As a result, the current results support the prioritization of API and STR categories in any feature-selection protocol aimed at training ransomware detection models.

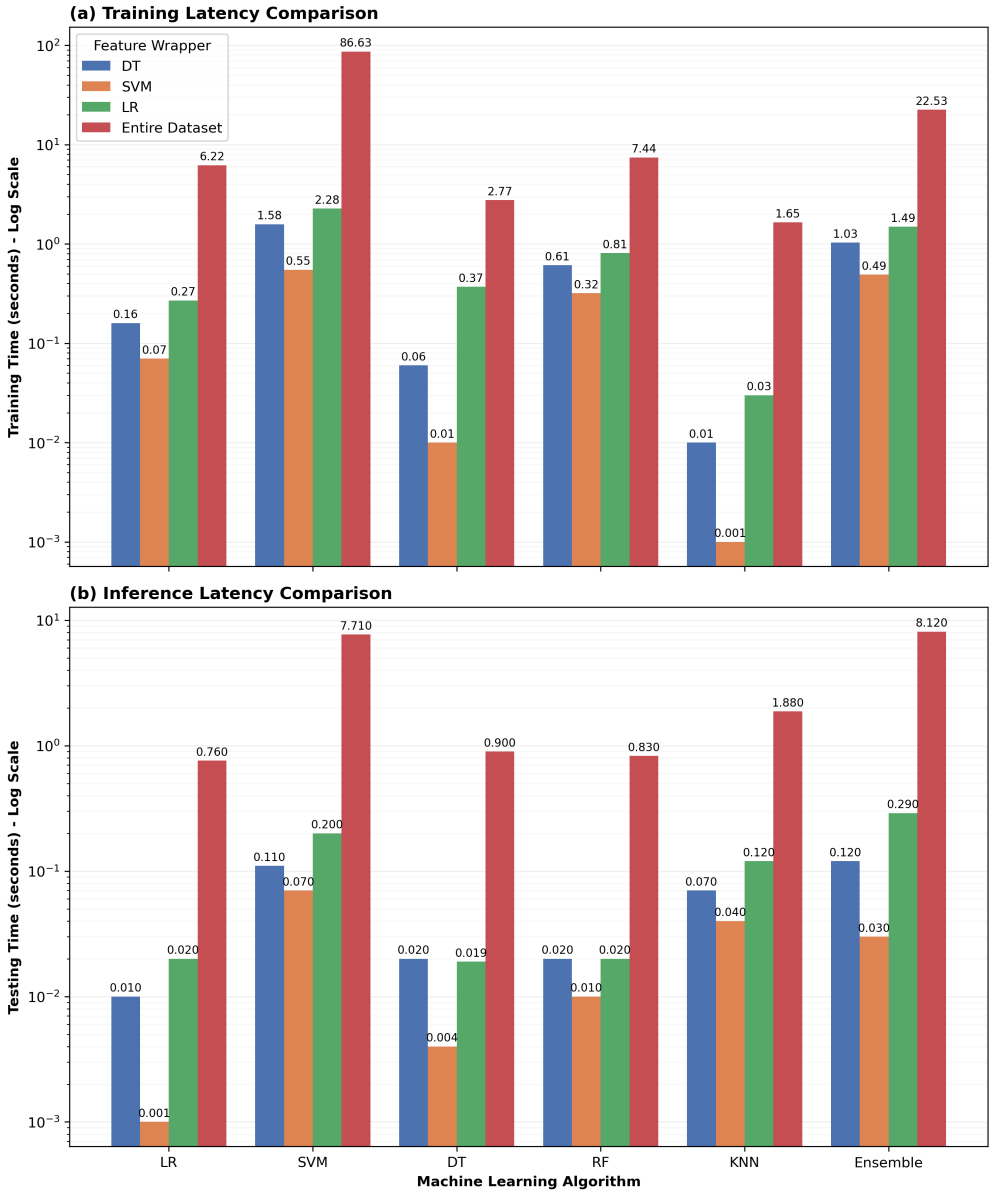

Figure 7. Comparison of computational latency between selected features and full dataset

Besides, achieving a high level of dimensionality reduction of features, between 98.73% and 99.64%, while preserving or even enhancing the accuracy of classification is a major advancement in real-time ransomware detection. Although the dimensionality reductions of 97.33% [22] and 98.70% [21] have been reported in previous studies, the proposed SVM-wrapper selection process sets a new benchmark in terms of efficiency with a 99.64% reduction in features. Most importantly, the testing time of the high-accuracy ensemble classifier is also cut down

to only 0.03 s against a baseline of 8.12 s. that enables the earlier detection of threats, preventing the execution of ransomware encoder loads, and is especially useful in resource-constrained environments. Furthermore, the tested algorithms had different preferences to combinations of features categories. The highest accuracy was reached with LR when using the API, STR, REG, and FILES_EXT categories, which made 97.78%. Conversely, SVM reached its peak empirical accuracy of 96.47% using a small subset of only API and STR categories. Our algorithmic sensitivity to specific feature categories is the basis of our approach of simultaneously testing multiple classifiers in feature-category assessment, hence allowing a more refined and effective feature-selection procedure. In addition, the approach proved to be highly generalized by several machine learning models. Compared to other classifiers, the LR wrapper consistently yielded the highest predictive accuracy of $98.03\%(\pm 1.56)\%$, which demonstrates that LR-identified features can be particularly valuable in the context of a situation where the highest detection accuracy is needed. At the same time, the RF classifier reported a consistent accuracy of 97.29% and 97.78% across all feature-selection methods, which is also consistent with the previous literature indicating that ensemble tree-based methods are more effective in intrusion detection because they can discover non-linear relationships in the data[34], [35].

Robustness Across Ransomware Families. The ability of the proposed category-aware feature selection framework to generalize its results is important to verify, as the model should not overfit on a particular ransomware strain, but rather learn the invariant behavioral features of ransomware. For this purpose, we tested the detection accuracy of the optimal Category-Aware LR Wrapper, with the Voting Classifier ensemble, on the 11 different ransomware families in the test split of the RISS dataset. As shown in Table 4, the optimized ensemble model displays very high consistency between different threat variants. It was able to detect all of the 11 evaluated families with a perfect 100% detection rate. The model achieved extremely high detection rates for the rest of the families . The results also validate the robustness of our model to a wide and constantly changing threat landscape by showing that the 392-feature subset we extracted through our hierarchical approach is able to capture fundamental, cross-family malicious behaviors such as universal encryption routines through API calls and specific registry persistence mechanisms.

Table 4. Performance Breakdown per Ransomware Family

Ransomware Family	Detection Rate	Test Samples
Critroni	100.00%	9
CryptLocker	100.00%	20
CryptoWall	100.00%	7
KOLLAH	100.00%	6
Kovter	100.00%	14
Locker	85.00%	20
MATSNU	100.00%	13
PGPCODER	100.00%	1
Reveton	94.74%	19
TeslaCrypt	100.00%	2
Trojan-Ransom	100.00%	5

To make our framework more robust, we must address theoretical computational complexity. Our four-stage methodology mitigates the $\mathbf{O}\left(\mathbf{2}^{N}\right)$ complexity typically associated with exhaustive wrappers. By evaluating categories individually in Stage 1 and 2, and filtering features using Mutual Information in Stage 3, we restrict the final wrapper evaluation in Stage 4 to a predefined maximum of 100 features per category. Consequently, the search space complexity is bounded by $\mathbf{O}\left(K \times C\right)$, where K is the subset limit and C is the number of optimized categories, ensuring consistent scalability irrespective of hardware infrastructure.

Feature Stability and Semantic Interpretability. One of the major problems in high-dimensional malware analysis is feature instability: the subsets generated by the traditional global wrapper methods tend to be highly unstable and hard to interpret if they are trained on data that changes over time. This is automatically addressed by our 4-fold cross validation splits and the concept of "semantic stability" that we introduce. Features are ranked initially in an isolated way within their behavioral categories, so any differences between training splits (such as swapping the exact ranking of two highly correlated Registry keys) are strictly intra-categorical.

In addition, the iterative evaluation by the wrapper algorithm ensures deterministic structural consistency. No matter the data split, the distribution of the subset being evaluated (the percentage of API calls, Registry operations, Strings, etc.) is mathematically limited by the algorithm. Thus, while the local MI of individual features may change from fold to fold, the overall meaning of the final optimized subset of features is fully retained. This structural consistency ensures reliability against ransomware variations and provides security practitioners with highly interpretable, domain-relevant insights into the specific attack vectors being targeted.

4.3. Logic behind the Stopping Criteria of Wrapper Feature Selection.

During the third stage of our methodology, we used a wrapper-based feature selection scheme, whereby features were selected from each category until a limit of 100 features per category was achieved. In the following section we shall discuss the rationale behind the selection of 100 feature threshold as stopping criteria.

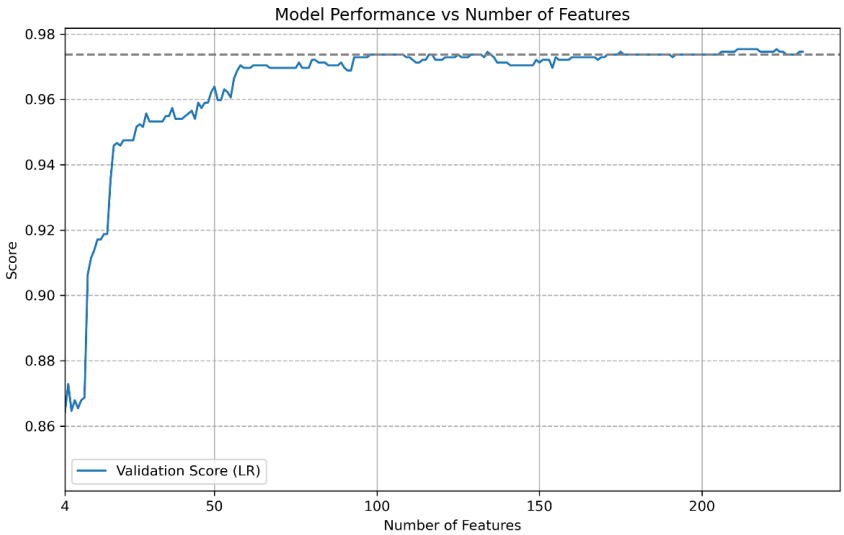

Figure 8. Justifying the stopping criterion: LR performance in terms of features

Figure 8 shows the correlation between the number of selected features and the validation accuracy when LR was used as the wrapper algorithm. The curve Shows that when approaching hundred-feature mark, the validation accuracy remained at 97%. In addition, the first 50-75 features provided significant predictive advantages, however, further addition of features after 100 did not provide significant predictive advantages in validation accuracy. The 100-feature threshold is a good trade-off between predictive accuracy and computational efficiency since the resultant flat performance curve means that additional feature addition would only increase the computational costs without significant performance improvements.

Figure 9 shows that the SVM validation score also flattens in the 75-100 feature range. Beyond this cutoff does not improve the discriminating power of the model, thus supporting the 100 features-mark as the best compromise between performance and agility.

Lastly, the decision-tree algorithm performance, as revealed in blue Figure 10, the validation measure levels off at around 96% when 75-100 features are included. In addition, further addition of features after 100 features does not significantly change the validation score, thus giving strong reasons to use 100 features as the stopping point. The addition of more features does not justify the computational cost in terms of any meaningful improvement in performance.

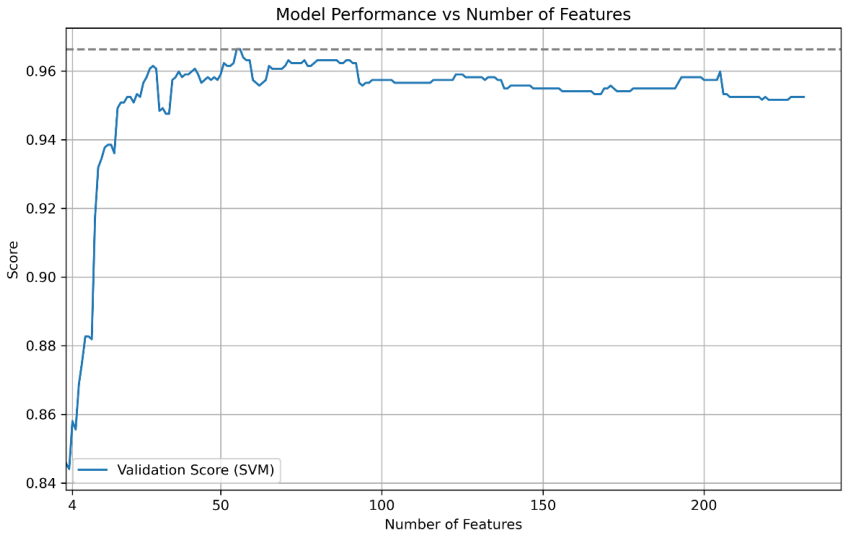

Figure 9. The justification of stopping criterion based on SVM performance based on the number of features.

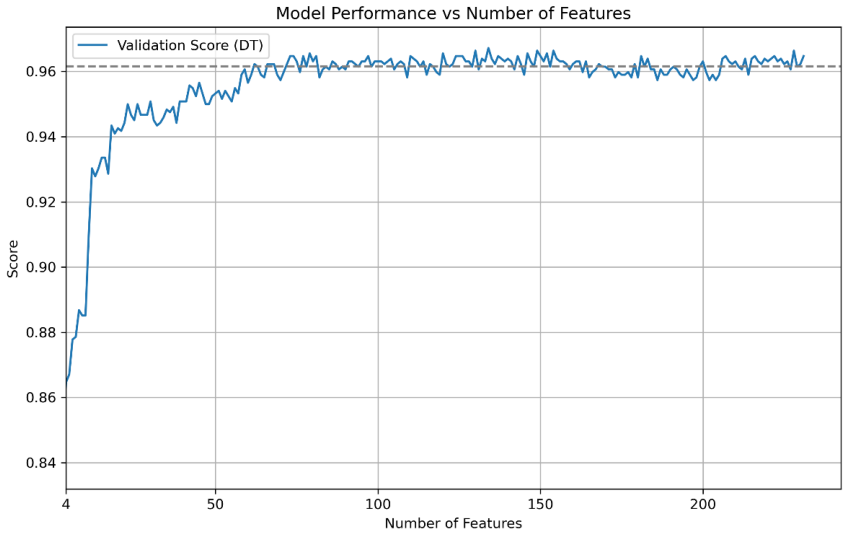

Figure 10. Justifying the stopping criterion: DT performance in terms of features.

4.4. Computational Overhead and Scalability Analysis

To give a full picture of the computational needs of our framework, we empirically profiled the time needed to run the feature selection process. The stage-wise computational breakdown of our proposed methodology is given in Table 5. The overall execution time from analyzing the raw categories to the final optimal subset of 392 features is 857.06 seconds. This execution overhead is very efficient, because the initial data set is very high-dimensional.

The most computationally intensive stage is Stage 3 during the Mutual Information Ranking with 626.89 seconds as seen in the breakdown. This is natural, since it requires a thorough statistical analysis of the whole feature

Table 5. Stage-wise computational execution time of the proposed feature selection framework.

Framework Stage	Process Description	Execution Time (Seconds)
Stage 1	Single Category Evaluation	123.64
Stage 2	Incremental Category Combination	81.07
Stage 3	Mutual Information Ranking	626.89
Stage 4	Category-Aware Wrapper Selection	25.46
Total	**Complete Feature Optimization**	**857.06**

space. However, by carefully choosing these features and categorizing them in our hierarchical order, the final combinatorial search in Stage 4 takes only 25.46 seconds, completely avoiding the slowdown that is common with traditional wrapper methods. Most importantly, this computational cost is an offline one-time training cost. The online operational detection phase is extremely light after the framework has identified the optimal set of 392 features.

The complexity metrics of the final optimized model are significant to evaluate the proposed framework's suitability for resource constrained environments like IoT gateways and edge devices, from offline training to online deployment. The standard high dimensional models with 30,967 features would need a lot of memory allocation and a high CPU cycle cost for matrix multiplication in the inference phase. Our optimized Logistic Regression classifier, however, only uses 392 features. The deployed inference model in this case is simply a 392 dimensional weight vector and one intercept value. The entire predictive model requires only 4 Kilobytes (KB) of memory, in standard 64-bit floating-point precision. The tiny footprint of this memory enables the model to be completely stored in the fast SRAM of typical microcontroller devices, eliminating the need for expensive storage read/write operations. Moreover, the inference time complexity is lowered to $\mathcal{O}(k)$, where $k = 392$, which results in prediction latency of microseconds per sample. The energy consumption per prediction is extremely low, reducing the number of CPU cycles needed for vector multiplication by 98.7% compared to the original feature space, and satisfying the constraints of edge-based ransomware detection.

4.5. Evasion Resistance and Adversarial Vulnerabilities

The proposed category-aware feature selection framework has shown exceptional predictive accuracy, but it is important to recognize that behavioral features may be susceptible to adversarial evasion methods. New ransomware strains are now using advanced techniques to evade machine learning classifiers, including time-delaying execution, API hooking, or API mimicry (injecting benign API call patterns to reduce the number of detections). A detection model that heavily or solely depends on a single behavioral vector, like API calls, is very vulnerable to these mimicry attacks. The key benefit of our categorical feature selection approach, however, is the built-in 'defense-in-depth' feature level approach. The framework ensures that the final optimized subset covers highly discriminative variables across multiple distinct categories such as Registry Key Operations, File System Modifications, Embedded Strings and Dropped Files, making it much harder and more expensive for an attacker. For a ransomware strain to be able to evade detection it would have to be able to seamlessly assume a non-malicious behavior on all of these categoric dimensions. Even if adversarial techniques are effective at obfuscating malicious API calls, the fast traversal and modification of the file system and persistent registry changes that would be needed to encrypt the payload will trigger a positive detection.

However, we recognize that the adversarial machine learning threat is ongoing. The model may lose its effectiveness over time if advanced evasion techniques are used that dynamically change the importance of the features during runtime. A thorough quantitative investigation of adversarial robustness and the incorporation of adversarial training into the proposed ensemble model are important directions for future research in this area.

4.6. Limitations and Future Work

While the proposed hierarchical feature selection framework demonstrates exceptional performance and computational efficiency, the current evaluation is inherently tied to the structural properties of the RISS dataset.

The core requirement of our framework is a high-dimensional feature space that is naturally segmentable into distinct semantic or behavioral categories namely, API calls, Registry operations and File system events. Consequently, the direct application of this framework to datasets with monolithic, unsegmented feature vectors, or purely static features such as raw byte n-grams, would require a preliminary feature clustering phase to reconstruct a categorical hierarchy.

Despite these encouraging outcomes, there are several limitations regarding the specific data source that should be mentioned. The experimental evaluation was performed with the help of one ransomware dataset, and the applicability of these findings to new ransomware datasets is yet to be confirmed. While the RISS dataset provides a comprehensive foundational benchmark, its samples were collected around 2016, which means there is a risk of obsolescence when facing fast-changing modern ransomware families such as LockBit 3.0, BlackCat and Royal. In addition, dynamic analysis is inherently prone to weaknesses such as possible evasion techniques by sophisticated malware that can identify sandboxes or virtualized environments. Therefore, we frame our four-stage categorical prioritization primarily as a methodological contribution.

Establishing the broader generalizability of this categorical prioritization strategy remains an important direction for future research. The practical effectiveness of the selected behavioral indicators against contemporary threats requires further empirical validation using recent datasets, such as the CIC-MalMem-2022 dataset. To maintain relevance and overcome this generalization problem, a continuous re-training strategy is recommended, wherein the model periodically ingests dynamic behavioral logs from emerging ransomware families, re-evaluating the categorical weights. Furthermore, future work will focus on adapting the framework to other domains of cybersecurity that generate high-dimensional, multi-vector logs—such as network intrusion detection systems (NIDS) spanning multiple protocols, or IoT malware analysis—to confirm the transferability of our hierarchical subset optimization across different threat landscapes.

While this study concentrates on ransomware detection, the proposed hierarchical feature selection framework is basically ransomware agnostic. Malicious activities of all types, including rootkits that set up system persistence, trojans that start unauthorized network communications, and spyware that performs keylogging operations, all use common operating system interactions. These interactions are naturally classified into the universal behavioral categories that our framework uses, such as API invocations, registry changes, file system traversals, and embedded strings. The approach used in structuring the features of our methodology around these universal categorical boundaries, as opposed to ransomware-specific heuristics, makes the framework highly transferable to general malware detection. The key for this transferability is that high dimensional, categorically structured dynamic analysis logs are available. The application of this category-aware wrapper framework to a variety of malware families, including Advanced Persistent Threats (APTs) or polymorphic viruses, is a very promising direction for future research.

5. Conclusion

This manuscript presents a robust four-level ransomware detection hierarchical architecture that achieves optimal and statistically validated balance between high-level security and high computational agility, reducing the dimensionality by up to 99.64% when using SVM as wrapper algorithm . The proposed solution demonstrates that category-based feature selection enables the ensemble learning classifier to achieve a significant 270x testing speedup, reaching a detection rate of 98.03% while remaining computationally lightweight enough for real-time applications. Furthermore, this efficiency responds to the unique requirements of digital environment in which vital infrastructures frequently have to perform in resource-limited environments that require high-performance security on constrained hardware. In contrast to deep learning models, which can be viewed as black boxes, the proposed approach is interpretable. It provides security practitioners with actionable information, which is an essential roadmap to secure growing digital economies worldwide against advanced extortion. This framework provides a mathematically based and scalable set of conditions to ensure the protection of different organizational structure against the ever-evolving ransomware threat by isolating highly discriminatory behavioral indicators of digital noise.

REFERENCES

1. E. Kalaimannan, S. K. John, T. DuBose, and A. Pinto, "Influences on ransomware's evolution and predictions for the future challenges," J. Cyber Secur. Technol., vol. 1, no. 1, pp. 23–31, Jan. 2017, doi: 10.1080/23742917.2016.1252191.
2. S. Adam, "The State of Ransomware 2024," Sophos News.
3. A. A. M. A. Alwashali et al., "A Survey of Ransomware as a Service (RaaS) and Methods to Mitigate the Attack," in 2021 14th International Conference on Developments in eSystems Engineering (DeSE), Dec. 2021, pp. 92–96. doi: 10.1109/DeSE54285.2021.9719456.
4. P. H. Meland, et al., "The Ransomware-as-a-Service economy within the darknet," Comput. Secur., vol. 92, p. 101762, May 2020, doi: 10.1016/j.cose.2020.101762.
5. O. Akinyemi, et al., "Analysis of the LockBit 3.0 and its infiltration into Advanced's infrastructure crippling NHS services," Aug. 10, 2023, arXiv: arXiv:2308.05565. doi: 10.48550/arXiv.2308.05565.
6. K. Kim et al, "Cryptocurrency-driven ransomware syndicates operating on the darknet: A focused examination of the Arab world," Egypt. Inform. J., vol. 30, p. 100665, Jun. 2025, doi: 10.1016/j.eij.2025.100665.
7. M. U. Ilyas et al., "Machine learning approaches to network intrusion detection for contemporary internet traffic," Computing, vol. 104, no. 5, pp. 1061–1076, May 2022, doi: 10.1007/s00607-021-01050-5.
8. M. Abid, et al., "Decentralized Anomaly Detection in Electric Vehicle Supply Equipment via Federated Learning and Blockchain Integration," Computing, vol. 108, no. 3, p. 41, Feb. 2026, doi: 10.1007/s00607-026-01632-1.
9. G. Sri vidhya et al., "A novel bidirectional LSTM model for network intrusion detection in SDN-IoT network," Computing, vol. 106, no. 8, pp. 2613–2642, Aug. 2024, doi: 10.1007/s00607-024-01295-w.
10. O. Chaieb,et al., "Machine Learning-Based Intrusion Detection System: Review and Taxonomy," in Proceedings of the 6th International Conference on Big Data and Internet of Things, in Lecture Notes in Networks and Systems. Springer International Publishing, 2023, pp. 10–21. doi: 10.1007/978-3-031-28387-1_2.
11. P. Bajpai et al., "An Empirical Study of API Calls in Ransomware," in 2020 IEEE International Conference on Electro Information Technology (EIT), Jul. 2020, pp. 443–448. doi: 10.1109/EIT48999.2020.9208284.
12. A. Damodaran, et al., "A comparison of static, dynamic, and hybrid analysis for malware detection," J. Comput. Virol. Hacking Tech., vol. 13, no. 1, pp. 1–12, Feb. 2017, doi: 10.1007/s11416-015-0261-z.
13. D. Escudero García et al., "Optimal feature configuration for dynamic malware detection," Comput. Secur., vol. 105, p. 102250, Jun. 2021, doi: 10.1016/j.cose.2021.102250.
14. F. H. da Costa et al., "Exploring the use of static and dynamic analysis to improve the performance of the mining sandbox approach for android malware identification," J. Syst. Softw., vol. 183, p. 111092, Jan. 2022, doi: 10.1016/j.jss.2021.111092.
15. C. Omar, et al., "A Systematic Review and Taxonomy of Ransomware Detection Based on Artificial Intelligence Algorithms," in 2024 3rd International Conference on Embedded Systems and Artificial Intelligence (ESAI), Dec. 2024, pp. 1–11. doi: 10.1109/ESAI62891.2024.10913856.
16. U. Ahmed, et al., "Mitigating adversarial evasion attacks of ransomware using ensemble learning," Comput. Electr. Eng., vol. 100, p. 107903, May 2022, doi: 10.1016/j.compeleceng.2022.107903.
17. S. Gulmez, et al., "XRan: Explainable deep learning-based ransomware detection using dynamic analysis," Comput. Secur., vol. 139, p. 103703, Apr. 2024, doi: 10.1016/j.cose.2024.103703.
18. R. Islam, et al., "Android malware classification using optimum feature selection and ensemble machine learning," Internet Things Cyber-Phys. Syst., vol. 3, pp. 100–111, Jan. 2023, doi: 10.1016/j.iotcps.2023.03.001.
19. D. S. Keyes, et al., "EntropLyzer: Android Malware Classification and Characterization Using Entropy Analysis of Dynamic Characteristics," in 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge , May 2021, pp. 1–12. doi: 10.1109/RDAAPS48126.2021.9452002.
20. A. Rahali, et al., "DIDroid: Android Malware Classification and Characterization Using Deep Image Learning," in Proceedings of the 2020 10th International Conference on Communication and Network Security, in ICCNS '20. New York, NY, USA: Association for Computing Machinery, Mar. 2021, pp. 70–82. doi: 10.1145/3442520.3442522.
21. D. Sgandurra, et al., "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection," Sep. 10, 2016, arXiv: arXiv:1609.03020. doi: 10.48550/arXiv.1609.03020.
22. M. S. Abbasi, et al., "Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection," Appl. Soft Comput., vol. 121, p. 108744, May 2022, doi: 10.1016/j.asoc.2022.108744.
23. O. Chaieb, et al., "A new wrapper feature selection approach for binary ransomware detection," IAES Int. J. Artif. Intell. IJ-AI, vol. 14, no. 3, Art. no. 3, Jun. 2025, doi: 10.11591/ijai.v14.i3.pp2104-2112.
24. S. Lee, et al., "Ransomware protection using the moving target defense perspective," Comput. Electr. Eng., vol. 78, pp. 288–299, Sep. 2019, doi: 10.1016/j.compeleceng.2019.07.014.
25. P. Maniriho, et al., "API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques," J. Netw. Comput. Appl., vol. 218, p. 103704, Sep. 2023, doi: 10.1016/j.jnca.2023.103704.
26. B. Zhou, et al. "A novel malware detection method based on API embedding and API parameters," J. Supercomput., vol. 80, no. 2, pp. 2748–2766, Jan. 2024, doi: 10.1007/s11227-023-05556-x.
27. K. Begovic, et al. "Cryptographic ransomware encryption detection: Survey," Comput. Secur., vol. 132, p. 103349, Sep. 2023, doi: 10.1016/j.cose.2023.103349.
28. S. Torabi, et al., "A Strings-Based Similarity Analysis Approach for Characterizing IoT Malware and Inferring Their Underlying Relationships," IEEE Netw. Lett., vol. 3, no. 3, pp. 161–165, Sep. 2021, doi: 10.1109/LNET.2021.3076600.
29. N. A. Rosli, et al., "Clustering Analysis for Malware Behavior Detection using Registry Data," Int. J. Adv. Comput. Sci. Appl. IJACSA, vol. 10, no. 12, Art. no. 12, Jun. 2019, doi: 10.14569/IJACSA.2019.0101213.
30. K. Zirari, et al., "Enhancing Ransomware Detection: A Registry Analysis-Based Approach," in 2023 10th International Conference on Future Internet of Things and Cloud (FiCloud), Aug. 2023, pp. 65–70. doi: 10.1109/FiCloud58648.2023.00018.

31. N. Scaife, et al., “CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data,” in 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), Jun. 2016, pp. 303–312. doi: 10.1109/ICDCS.2016.46.
32. A. Liska et al., Ransomware[Book], November 2016. O’Reilly Media, Inc.
33. M. Scalas, et al., “On the effectiveness of system API-related information for Android ransomware detection,” Comput. Secur., vol. 86, pp. 168–182, Sep. 2019, doi: 10.1016/j.cose.2019.06.004.
34. L. Zou, et al., “HC-DTTSVM: A Network Intrusion Detection Method Based on Decision Tree Twin Support Vector Machine and Hierarchical Clustering,” IEEE Access, vol. 11, pp. 21404–21416, 2023, doi: 10.1109/ACCESS.2023.3251354.
35. M. Rashid, et al., “A tree-based stacking ensemble technique with feature selection for network intrusion detection,” Appl. Intell., vol. 52, no. 9, pp. 9768–9781, Jul. 2022, doi: 10.1007/s10489-021-02968-1.
36. Alif Rahman, et al. 2025. Enhancing Multi-class Malware Detection in Resource-constrained Environments. In Proceedings of the 12th International Conference on Next Generation Computing, Communication, Systems and Security (NSysS ’25). Association for Computing Machinery, New York, NY, USA, 73–78. https://doi.org/10.1145/3777555.3777568
37. S. Panja, et al, "An Efficient Malware Detection Approach Based on Machine Learning Feature Influence Techniques for Resource-Constrained Devices," in IEEE Access, vol. 13, pp. 12647-12665, 2025, doi: 10.1109/ACCESS.2025.3526878.
38. Rauf Ali Khan,et al. (2025). Lightweight Quantized XGBoost for Botnet Detection in Resource-Constrained IoT Networks. IoT, 6(4), 70. https://doi.org/10.3390/iot6040070
39. B. Bataineh and G. Shakah, “Information-Theoretic Feature Selection via Variational Autoencoders for Network Intrusion Detection: A Mutual Information Maximization Approach,” Stat., Optim. Inf. Comput., 2026.
40. Y. Boulkhiout, S. Balbal, K. Nasri, and A. Moussaoui, “An Empirical Comparative Review of Classifiers for Real-Time IoT Cyber Attack Detection,” Stat., Optim. Inf. Comput., 2026.
41. H. Elgohari, M. Z. Fouda, M. Asim, and O. M. Elzeki, “A Statistical Evaluation of Feature Selection Methods for Breast Cancer Classification: A Simulation and Real-Data Study,” Stat., Optim. Inf. Comput., vol. 16, pp. 2826–2845, 2026.