Table 5.11: Comparison of the proposed method with existing methods

Methods	Ransomware samples	Total Class	Features (S/D)	Total Features	Accuracy (%)
[73]	1787	10	S	123	91.43
[74]	210	10	D	8	97.1
[76]	500	6	D	23	97.03
[77]	256	2	D	8	87
[75]	755	8	D	131	98
[78]	574	12	S/D		98.25
Proposed(ERAND)	2288 (Dataset 1)	2	S	15	98.2
	582 (Dataset 2)	12	D	Min: 4 Max: 42	98.8

5.6 Discussion

To effectively counter ransomware, a cost-effective method is proposed that leverages the dynamic features of ransomware. This method involves several key components to enhance its effectiveness and accuracy. Firstly, an ensemble approach is introduced to identify the most discriminative features of various ransomware families. This approach combines multiple feature selection techniques to ensure that the most relevant and distinguishing features are selected. Once the discriminative features are identified, a weighted majority-based combination function is proposed. This function integrates the outputs of multiple classifiers, assigning weights to each classifier based on their performance. This ensures that the final decision is influenced more by the classifiers that perform better, thereby improving the overall classification accuracy in ransomware detection. The proposed method, ERAND, has been tested on multiple variants of ransomware and has consistently shown superior performance. While ERAND has proven effective in controlled experimental settings, its performance in real-world enterprise networks remains to be validated. Future work involves deploying ERAND in actual enterprise environments to assess its effectiveness in detecting newer versions of ransomware. Such validation would provide valuable insights into its practical applicability and potential for real-time threat detection.

Chapter 6

An Ensemble Approach for Effective Malware Detection

6.1 Introduction

The security and integrity of computer systems, networks, and user data are seriously threatened by malware, often known as malicious software. Effective detection techniques are crucial for reducing these dangers as malware's sophistication and diversity keep growing. The identification of key characteristics that can precisely discriminate between dangerous and benign files is a fundamental component of creating effective malware detection systems. In general, the indicators of compromises (IoC) are categorized into system-level and network-level. Typically, system-level IoCs exist within distinct computer systems, servers, or endpoints. They provide information regarding potential compromise or malicious activity on a particular device. Network-level IoCs are observed in the network traffic and infrastructure, and they can help identify threats that traverse the network. A wide range of structural, behavioral, and statistical characteristics that are displayed by malware can be exploited as potential indicators for detection. The file format, code patterns, and header information are structural properties. Execution-related behaviors, network communication, and system interactions are behavioral attributes. Measures like entropy, frequency distribution, or coding complexity can all be considered statistical properties. From among the myriad options available, it is necessary to carefully analyze and investigate the relevant features.

In the fight against constantly evolving malware threats, machine learning

plays a crucial role in malware defense systems. By leveraging sophisticated algorithms and extensive datasets, machine learning models can rapidly detect and classify malicious software, including those with never-before-seen characteristics. Furthermore, the machine learning model can adapt and enhance its performance over time, acquiring knowledge from emerging threats and consistently improving its capacity to detect and combat malware. The process of selecting significant indicators of compromise (IoC) holds great significance in the context of machine learning-driven defenses against malware. IoCs play a crucial role in the process of efficiently identifying and mitigating possible malware threats. Machine learning models can achieve high levels of precision and efficiency by selecting the most relevant indicators. The curation of a comprehensive collection of IoCs serves the purpose of not only minimizing the occurrence of false positives but also maintaining the system's ability to promptly address newly emerging threats. Furthermore, the continuous refinement of IoCs in response to changing malware behaviors contributes to the development of effective malware defense solutions.

In this chapter, an ensemble-based feature selector is proposed to identify the most relevant set of features for effective malware detection. The performance of the method is found to be better with high accuracy.

6.1.1 Motivation

The dynamic and ever-changing realm of malware demands novel strategies for the effective detection of malicious software. In this context, the motivation for developing an ensemble feature selection approach arises from the imperative to enhance the precision and dependability of identifying features crucial for distinguishing benign and malicious programs. Traditional feature selection methods may fall short in capturing the diverse and dynamic nature of malware, resulting in less-than-optimal outcomes. By integrating multiple feature selection techniques within an ensemble framework, this method aims to leverage the complementary strengths of various approaches, thus improving the robustness of the feature selection approach for the identification of the most discriminative features. This holistic strategy aims to equip malware detection models with a refined set of features, facilitating a more nuanced understanding of the unique characteristics present in malware. This approach strengthens the overall resilience of the malware detection approach against evolving malware threats.

6.1.2 Contribution

With the ever-evolving landscape of malicious software, developing effective detection methods is of utmost importance. This chapter focuses on the identification of important features for malware detection methods, aiming to enhance the accuracy and efficiency of such systems. In this work, we propose an ensemble approach called FRAMC to identify the key features that contribute significantly to the detection of malware. The effectiveness of FRAMC is assessed using different types of classifiers on a number of real-world malware datasets. The outcomes of our analysis demonstrate that the proposed approach excels in terms of performance when compared to other methods.

6.2 Background

6.2.1 Feature selection

In the realm of machine learning and data science, a dataset is composed of data points, and each data point is characterized by a collection of values. These values are referred to as variables, attributes, or features of the data point. The number of features in this set is also known as the dataset's dimensionality. The dimensionality of each data point holds significance when developing machine learning models. When the dimensionality of data becomes exceptionally large, it gives rise to the "curse of dimensionality," which has a detrimental impact on the performance of machine learning models [35]. Furthermore, the utilization of a high-dimensional feature space leads to overfitting of the learning model, as well as a substantial rise in memory needs and computing costs. In order to tackle these concerns, the utilization of feature selection is employed as a method of data preprocessing with the aim of decreasing the dimensionality [36] [37]. Not all features that exist in the real world has importance or relevance. The process of feature selection involves the identification and selection of a subset of pertinent features based on certain evaluation criteria [38] [39] [40] [41].

Based on the availability of class label information and selection tactics, feature selection algorithms are divided into different categories [42]. There are three types of these techniques, supervised, unsupervised, and semi-supervised, depending on the availability of ground truth knowledge. The supervised approach [43] attempts to identify a pertinent feature by considering the informa-

tion provided by the class label. The determination of feature significance is often accomplished through the use of different measures such as mutual information and correlation. The unsupervised approach [44] [45] [46] involves the selection of meaningful features from a set of unlabeled data. Given the absence of ground truth knowledge, the determination of feature relevance typically involves the utilization of various measures, such as data similarity and local discriminative information. The chosen subset of attributes demonstrates the capability to extract clusters from all instances. Semi-supervised approach [47] [48] [49] generates a relevant feature subset by taking into consideration of both labeled and unlabeled samples. It is similar to the supervised approach except it uses the partial label information.

The selection approach encompasses three distinct types of feature selection techniques: filter, wrapper, and embedded [37]. Filter methods [50] are employed to generate the most suitable subset of features by analyzing the attributes of data through some statistical criteria. The feature selection step of this method does not employ learning methods, resulting in improved computational efficiency. In contrast, wrapper methods [51] employ learning algorithms as a means of selecting criteria. The wrapper approach is a technique that aims to identify the most suitable subset of features by optimizing the predicted accuracy of a given classifier. When the dimension of the dataset is significantly large, it becomes computationally expensive. Additionally, there may be bias towards the provided classifier. Wrapper methods are generally regarded as more efficient than filter methods, although they do come with a higher computational cost. The embedded methods [52] are intermediate versions of wrappers and filters. It exploits the benefits of both methods. First, it selects subsets of candidate features using some statistical measures like filter methods and then it finally selects the optimal subset that ensures the best possible classification accuracy. It is computationally less expensive than wrapper methods.

6.2.2 Ensemble Feature Selection

Ensemble feature selection approaches represent a sophisticated strategy aiming to enhance the robustness and reliability of feature identification processes. These methodologies leverage a combination of distinct feature selection techniques, operating on the premise that diverse methods can collectively contribute to a more comprehensive understanding of the dataset. The ensemble framework essentially

amalgamates the outputs of individual methods, mitigating the limitations of any single approach and producing a more accurate selection of crucial features.

Consensus building is a fundamental process in decision-making that involves achieving agreement or alignment among a group of individuals, often with diverse perspectives or interests. In the context of ensemble feature selection, consensus building is particularly noteworthy. Ensemble methods leverage the strengths of different feature selection techniques, and consensus building becomes pivotal in reconciling their outputs. Instead of relying on the findings of a single method, consensus is sought by integrating the results of various techniques to identify important features. Several voting mechanisms are employed to facilitate consensus building in ensemble feature selection. These mechanisms may include summing up individual feature ranks, calculating average ranks, or employing weighted voting schemes. The objective is to ensure that features identified as crucial are consistently recognized across multiple methodologies.

Ensemble feature selection methodologies can be broadly categorized into three main groups based on their approaches to aggregating the outputs of individual feature selection algorithms. Voting-based ensembles involve a democratic decision-making process, where each feature selection algorithm "votes" for the importance of features, and the final decision is made by the majority or through weighted voting. Ranking-based ensembles focus on consolidating feature rankings generated by individual algorithms to create an overall ranking, with features prioritized based on their positions. This category encompasses techniques such as summing ranks and Borda Count. On the other hand, model-based ensembles leverage the outputs of feature selection algorithms to train a predictive model, where the model's predictions dictate the final feature selection. Stacking and Random Forest Feature Importance are examples of model-based ensemble approaches. These categories provide a framework for understanding the diverse ways in which ensemble strategies can be employed to enhance the robustness and effectiveness of feature selection processes in various domains, including machine learning and data analytics. The choice of ensemble category depends on the specific characteristics of the dataset and the objectives of the analysis.

6.2.3 Markov Chain

A Markov chain is a mathematical framework utilized to depict a succession of events or states. In this model, the likelihood of transitioning from one state to another is solely determined by the present state, without any consideration

of the preceding ones. Each state in the chain represents a particular condition or configuration, and the transitions between states are governed by transition probabilities. These probabilities can be represented by a transition matrix, where each entry indicates the likelihood of moving from one state to another.

A Markov chain for a system is specified by a set of states S, the transition matrix M where each entry in the matrix represents the probability of transitioning from one state to another state. The ordering of those states based on their Markov chain analysis-obtained probabilities or scores is called Markov chain ordering. The system initiates some of the start states in S. It makes a jump from one state to another at each step. At each of the steps, the jump is dependent on M, if the system moves from state i to state j then M_{ij} is the probability associated with it. If the current state probability distribution is given, the product of M and the vector reflecting the current state distribution will be the next state probability distribution. Usually, the start state distribution is chosen as uniform distribution on M. For some conditions of the Markov chain, the system evaluates and reaches a certain point where stationary or constant state distribution is observed. This final distribution is known as stationary distribution.

6.3 Problem Statement

For a given dataset D of size $M \times N$ with M instances and each of N features, i.e., F $=\{f_1, f_2, f_3, ..., f_N\}$, the goal is to find an optimal feature subset F' such that $F' \subseteq F$ and for F', the learning model will give the highest level of generalization performance. The main objective is to decrease the dimensionality of dataset D by eliminating insignificant or unrelated features while maintaining the effectiveness of the learning model. The reduced feature set is obtained by minimizing the redundancies among features and maximizing the feature-class relevance.

6.4 Proposed Method

In this section, the basic structure and technical principles of the detailed modules in FRAMC are introduced, respectively.

The proposed novel approach comprised of developed an ensemble feature selector named FRAMC for effective malware detection. The proposed method is

comprised of three arrangement schemes: (1) extraction of indicator of comprises and (2) selection of key indicators, and (3) evaluation of the learning model.

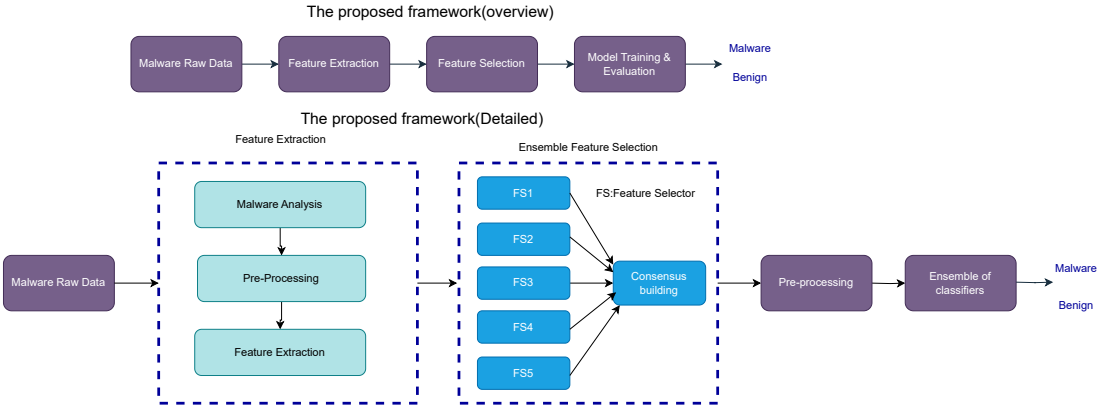


Figure 6-1: Overview of the proposed method

6.4.1 Extraction of indicator of compromises

Indicators of Compromise (IoCs) refer to specific features or characteristics derived from malware data that are considered pertinent and relevant to the particular security issue or threat under investigation. These features serve as valuable indicators or evidence in the identification and resolution of security incidents or possible breaches. Features or attributes or characteristics have a significant role to play because they affect the data's complexity, readability, and functionality. The feature extraction process goes through three stages namely data(both malware and goodware) collection, data analysis, and feature extraction. During the data collection phase, the acquisition of malware can occur via honeynet systems and public repositories[34]. Subsequently, in the second stage, comprehensive malware analysis is performed for additional preprocessing and the extraction of features. Finally, the last stage involves the extraction of features from the analysis reports of the malware. For a comprehensive understanding of the feature extraction process, please refer to our earlier work, where we provide detailed information on this procedure [79]. In our case, we have used features of both Windows and Android malware binaries.

6.4.2 Selection of key indicators

Feature selection is an important process that involves selecting the most relevant and informative subset of features from a high-dimensional dataset. By selecting

a subset of relevant features, it reduces overfitting, enhances model performance and interpretability, improves computational efficiency, and facilitates data comprehension. Therefore, it is necessary to select an optimal subset of features with the highest degree of relevance and the least amount of redundancy. Since our datasets contain a large number of features, we use feature selection to identify the optimal subset of features. Specifically, we employ an ensemble feature selection strategy that aims to leverage the benefits of multiple feature selection algorithms to improve the robustness and efficacy of feature selection processes. By combining the outputs of individual base feature selection methods, ensemble approaches can identify a subset of features whose collective performance is preferable to that of any individual method. The framework of the ensemble feature selection process is depicted in Figure 6-2 .

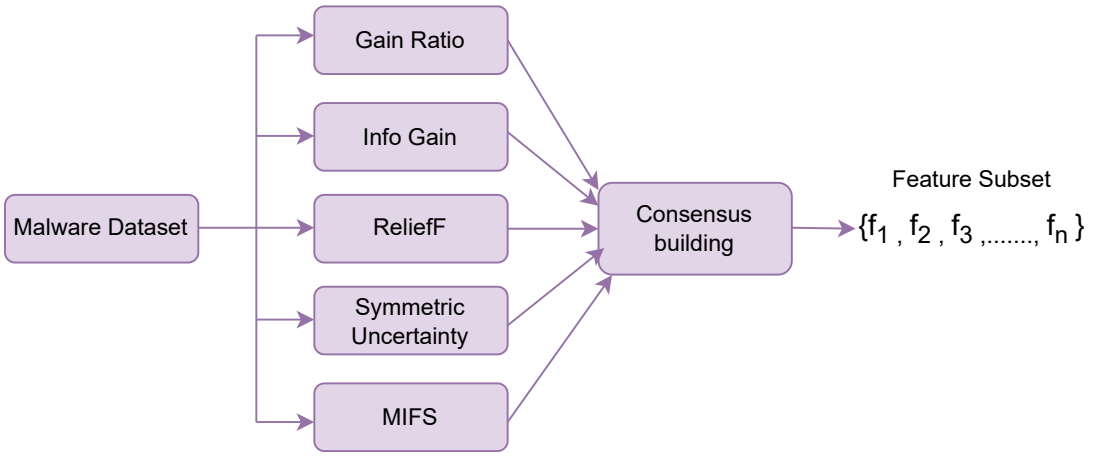


Figure 6-2: Ensemble feature selection framework

6.4.2.1 Base Feature Selection Algorithm

Ensemble feature selection involves two steps, just like other ensemble methods. Initially, we need a set of proficient base feature selectors, each of which provides a subset of features identified as significant. Second, use an appropriate consensus function to provide the optimal subset of relevant features based on the results of each ranker's or feature selection process. Guided by our experimental results, the following five base feature selectors, namely, Symmetric Uncertainty[80], MIFS[32], ReliefF[33], Gain Ratio [81], and Information Gain [82] are used.

6.4.2.2 Consensus Building

Ensemble feature selection eliminates the biases of individual participating feature selection methods to yield the best possible output using an appropriate consensus function. In our work, a Markov chain model is used for consensus building which is described next.

Let's assume we have "N" feature ranking lists given by N base feature selectors, denoted as $f_1, f_2, ..., f_N$, each of which ranks "M" features. These rankings can be represented as matrices where each row corresponds to a feature, and the values in the matrices represent their respective ranks. For example, $f_i[j]$ represents the rank of feature "j" in the i^th ranking list. Now, the goal is to aggregate those N feature lists to generate a final feature list.

The proposed method employs a consensus-building procedure in order to determine an optimal subset of features. It accepts the individual feature lists from the base feature selection algorithms and generates the final aggregated feature list. Below are the steps to generate the aggregated feature list. Firstly, it constructs the set U that consists of all feature lists provided by each base feature selector. Next, the transition matrix is constructed and normalized. The resulting matrix captures the transition probabilities between features based on the provided feature lists. Then, convert the above-normalized matrix into an ergodic matrix U_m by using the equation 6.1.

$$((normalized\_matrix) * (1 - ergodic\_value)) + (ergodic\_value/\|U\|) \quad (6.1)$$

Next, the stationary distribution matrix is calculated as a principal left eigenvector y associated with the ergodic matrix U_m to get the aggregated rank list.

$$yU_m = \alpha y \quad (6.2)$$

Where α is the eigenvalue of U_m. Finally, the importance or relevance of each feature in the aggregated list is assessed using a set of classifiers.

Algorithm 2: FRAMC

Input: C=$\{f_1, f_2, ...f_n\}$ is the set of feature lists

Output: y=the aggregated feature subset

initialization;

U=$\emptyset$;

for *i=1 to n* **do**

| U.add()

end

U_T=$transition\_matrix(U)$;

U_N=$normalize\_matrix(U_T)$;

U_m=$ergodic\_matrix(U_N)$;

yU_m=αy;

Return y;

6.4.3 Complexity Analysis

The ensemble method involves using multiple feature selectors, including Gain Ratio, Info Gain, ReliefF, Symmetric Uncertainty, and MIFS, followed by a consensus-building step to determine the optimal feature subset from a malware dataset. Each feature selection algorithm contributes to the overall complexity: Gain Ratio and Info Gain each have a complexity of $O(n \cdot \log n)$, as they require sorting feature values. ReliefF has a complexity of $O(m \cdot n)$, where m is the number of instances and n is the number of features, due to the need to compute distances between instances. Symmetric Uncertainty has a complexity of $O(n \cdot m)$, involving mutual information calculations between features. MIFS is the most computationally intensive, with a complexity of $O(n^2 \cdot m)$, as it evaluates pairwise feature interactions. The consensus-building step aggregates the results from k feature selectors, each producing a ranking of n features, with a complexity of $O(k \cdot n)$. Therefore, the overall complexity of the ensemble feature selection method is $O(n \cdot \log n + m \cdot n + n^2 \cdot m + k \cdot n)$.

6.4.4 Optimal Features

The ensemble feature selector FRAMC has effectively identified a subset of features that significantly contribute to the detection of malware. The selected features represent key characteristics within the datasets that are most informative for distinguishing between malware and goodware. FRAMC assigns a rank to each

feature within a dataset. The top-ranked features are considered the optimal future set, providing the best possible classification accuracy. FRAMC minimizes bias and variance found in each base feature selection algorithm. These chosen features serve as the foundation for the predictive model, enabling it to effectively distinguish malware and goodware. In the Windows malware dataset, Table 6.1 displays the top optimal features selected by FRAMC. Similarly, for the Android malware dataset, Table 6.2 lists the top optimal features chosen by FRAMC.

Table 6.1: Top 10 Features Selected by FRAMC for Windows Dataset

Sl No.	Feature
1	Subsystem
2	SizeOfOptionalHeader
3	SizeOfStackReserve
4	VersionInformationSize
5	MajorSubsystemVersion
6	DllCharacteristics
7	MajorImageVersion
8	SizeOfHeapCommit
9	SizeOfHeapReserve
10	ResourcesMinSize

Table 6.2: Top 10 Features Selected by FRAMC for Android Dataset

Sl No.	Feature
1	com.android.vending.BILLING
2	android.permission.SEND_SMS
3	android.permission.READ_PHONE_STATE
4	com.google.android.c2dm.permission.RECEIVE
5	android.permission.READ_EXTERNAL_STORAGE
6	com.google.android.c2dm.intent.RECEIVE
7	android.permission.CHANGE_WIFI_MULTICAST_STATE
8	android.intent.action.BOOT_COMPLETED
9	android.intent.action.DATA_SMS_RECEIVED
10	android.permission.WRITE_EXTERNAL_STORAGE

6.5 Performance Analysis

FRAMC has been implemented in Python using a Dell Precision 7810 workstation with 2x Intel Xeon (R) W-2145 comprising 8 cores, 64GB RAM, NVIDIA Tesla K80 GPU with 12GB VRAM, and Ubuntu OS. Materials used, preprocessing carried out, and performance achieved are discussed next.

6.5.1 Datasets and preprocessing

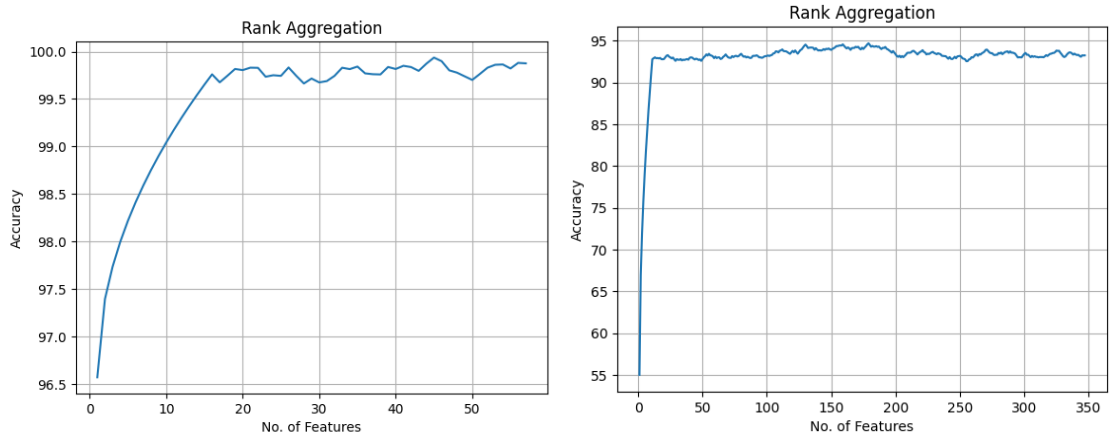
To understand the effectiveness of FRAMC, it has been tested on two datasets of malware[83] [84]. The Windows dataset comprises 41324 categories of benign software and 96724 types of malicious software. The dataset pertaining to Android comprises 3074 distinct categories of benign software and 1926 distinct categories of malicious software. The attributes of these datasets include both numeric and categorical values. The detailed description of these datasets is given in Table 6.3.

Table 6.3: Dataset details

Dataset	# Instances	# Features	# Classes
Windows	138047	57	2
Android	5000	347	2

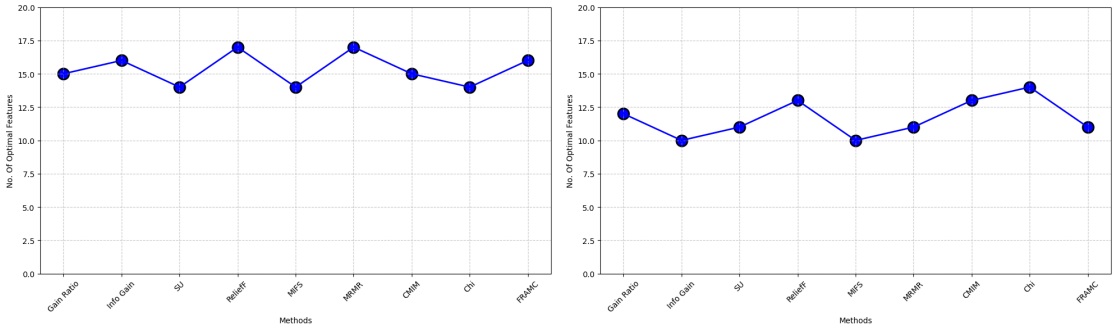
6.5.2 Classification Performance Analysis and Comparative Evaluation

The performance of our method is assessed based on classification accuracy. For this, we have used six different classifiers, namely, Random Forest, Decision Tree, AdaBoost, Gradient Boost, SVM, and Naive Bayes. The 10-fold cross-validation is used to estimate the performance of the classifier on the selected features. In addition, our approach is benchmarked against various feature selection techniques, including Symmetric Uncertainty (SU) [80], Mutual Information Feature Selection (MIFS) [32], ReliefF [33], Gain Ratio [81], Information Gain (IG) [82], Minimum Redundancy Maximum Relevance (MRMR) [85], Conditional Mutual Information Maximization (CMIM) [86], and Chi-Square (ChiSqaure) [87]. Tables 6.4 through 6.11 illustrate the performance of the proposed method on both datasets, encompassing evaluations conducted across the entire feature space as well as the optimized feature space. These tables detail the classification accuracy, precision,



(a) Accuracy on Windows malware dataset **(b)** Accuracy on Android malware dataset

Figure 6-3: Performance of FRAMC in terms of accuracy



(a) Optimal no. of features on Windows malware dataset **(b)** Optimal no. of features on Android malware dataset

Figure 6-4: Optimal range of the size of feature subsets

recall, and F1 score outcomes. From the experimental results, it can be observed that the proposed method is a top performer on both datasets and at least at par with the other competing methods. It is also apparent that the average accuracy achieved by FRAMC on the Windows malware dataset exceeds that of other methods and is comparable to Gain Ratio and Information Gain. Similarly, concerning the Android malware dataset, the average accuracy of FRAMC outperforms all other methods. However, for both datasets, a feature subset found by the proposed method has a different cardinality. Figures 6-3 illustrate this variance in the cardinalities of the ideal feature subsets. Similarly, figure 6-4 shows the optimal range of features by different methods including the proposed one. In an ensemble process, all the base algorithms operate independently, allowing us to leverage parallel processing for faster results. In our scenario, we establish a pool of worker processes to execute base feature selection algorithms concurrently. This approach enhances the efficiency of the ensemble process by taking full advantage of available computational resources and significantly reducing execution time.

Table 6.4: The classification accuracy of the Windows dataset using the complete feature space

	Classifiers					
	Decision Tree	Random Forest	AdaBoost	Gradient Boost	Naive Bayes	SVM
Full Feature Space (57)	99.02%	99.39%	98.85%	98.95%	69.99%	67.89%

Table 6.5: The Precision, Recall, and F1 score of the Windows dataset using the complete feature space

Classifier	**Precision**	**Recall**	**F1 Score**
Decision Tree	98.66%	98.58%	98.62%
Random Forest	98.9%	99.24%	99.11%
AdaBoost	98.14%	98.21%	98.18%
Gradient Boost	97.9%	97.87%	98.11%
Naive Bayes	68.9%	69.24%	69.37%
SVM	66.2%	65.24%	66.44%

Table 6.6: The performance comparison on the Windows dataset

Methods	Features	Classifiers					
		Decision Tree	Random Forest	AdaBoost	Gradient Boost	Naive Bayes	SVM
Gain Ratio	15	98.98%	99.12%	98.38%	98.71%	70.42%	69.96%
Info Gain	16	98.91%	98.78%	98.43%	97.71%	69.86%	71.03%
SU	14	98.56%	98.67%	99.40%	96.98%	69.97%	68.78%
ReliefF	17	99.10%	95.45%	98.75%	98.85%	68.74%	67.77%
MIFS	14	97.71%	98.91%	98.43%	99.03%	69.99%	69.99%
MRMR	17	98.6%	98.23%	97%	99.03%	69.6%	69.44%
CMIM	15	98.9%	98.6%	97.36%	98.4%	69.68%	69.99%
ChiSqaure	14	97.8%	98%	97.11%	98.41%	68.6%	70.41%
FRAMC	16	99.19%	98.35%	98.68%	98.95%	69.97%	69.98%

Table 6.7: Precision, Recall and F1 -Score on the Windows Dataset

Method	Precision						Recall						F1					
	Decision Tree	Random Forest	Ada Boost	Gradient Boost	Na´ve Bayes	SVM	Decision Tree	Random Forest	Ada Boost	Gradient Boost	Na´ve Bayes	SVM	Decision Tree	Random Forest	Ada Boost	Gradient Boost	Na´ve Bayes	SVM
Gain Ratio	97.6	98.37	96.8	97.67	68.89	67.88	96.67	97.9	96.32	97.7	68.33	66.77	98.1	98.9	97.1	98	69.5	68.3
InfoGain	97.99	97.77	97.57	96.62	68.87	69.98	97.78	97.54	96.99	96.11	68.23	69.23	98.34	98.21	97.9	97	69.2	70.89
SU	97.56	97.66	98.87	95.33	69.12	67.54	97.18	97.24	97.99	95.82	68.67	66.56	98.1	98	99	96.13	69.88	68.54
ReliffF	98.9	94.72	98.11	98.12	68.1	66.99	98.66	93.89	97.74	97.82	67.89	66.23	99	95.1	98.6	98.5	68.3	67.35
MIFS	97.1	98	97.77	98.88	68.89	69	96.88	98	97.49	98.1	68.43	68.56	97.67	98.11	98.18	98.88	69.17	69
MRMR	97.65	97.9	96	98.22	68.3	68.77	97	97.35	95.6	97.89	67.78	68	98.11	98	96.64	98.7	68.8	69
CMIM	98.2	97.4	96.5	98	68.8	69	97.6	96.8	96	97.5	68	68.7	98.34	98	97	98	69.18	69.22
ChiSquare	96.8	97	96.21	97.7	67.11	68.43	96	96.7	95.59	97	66.88	68	97	97.2	96.5	98	67.5	69
FRAMC	98.87	97.77	98.1	98.43	69	68.98	98.63	97	97.84	98	68.65	68.4	99	98.4	98.29	98.7	69	69.59

Table 6.8: The classification accuracy of the Android dataset using the complete feature space

	Classifiers					
	Decision Tree	Random Forest	AdaBoost	Gradient Boost	Naive Bayes	SVM
Full Feature Space (347)	91.65%	90.20%	89.41%	87.68%	76.48%	91.90%

Table 6.9: The Precision, Recall and F1 score of the Android dataset using the complete feature space

Classifier	**Precision**	**Recall**	**F1 Score**
Decision Tree	91.66%	91.58%	91.62%
Random Forest	90.9%	90.24%	90.11%
AdaBoost	88.14%	88.21%	88.18%
Gradient Boost	87.9%	87.87%	88.11%
Naive Bayes	75.9%	77.24%	79.37%
SVM	90.2%	90.24%	91.44%

Table 6.10: The performance comparison on the Android dataset

Methods	Features	Classifiers					
		Decision Tree	Random Forest	AdaBoost	Gradient Boost	Naive Bayes	SVM
Gain Ratio	12	92.45%	92.60%	87.89%	91.98%	85.68%	92.67%
Info Gain	10	92.31%	90.41%	86.45%	92.30%	86.41%	92.45%
SU	11	91.68%	92.05%	89.91%	91.88%	83.77%	91.67%
ReliefF	13	90.33%	91.67%	88.70%	90.97%	84.56%	90.58%
MIFS	10	91.44%	92.37%	86.91%	91.67%	85.01%	91.85%
MRMR	11	90%	91.5%	87.33%	91.8%	83.6%	91.77%
CMIM	13	91.2%	91.34%	88.6%	92%	86%	92.65%
ChiSquare	14	87.8%	89%	86.32%	89.6%	83.21%	87.88%
FRAMC	11	92.60%	92.42%	88.73%	92.40%	85.58%	92.82%

Table 6.11: Precision, Recall and F1 -Score on the Android Dataset

Method	Precision						Recall						F1					
	Decision Tree	Random Forest	Ada Boost	Gradient Boost	Na´ve Bayes	SVM	Decision Tree	Random Forest	Ada Boost	Gradient Boost	Na´ve Bayes	SVM	Decision Tree	Random Forest	Ada Boost	Gradient Boost	Na´ve Bayes	SVM
Gain Ratio	91.8	92	87.1	91.2	84.78	91.9	91.65	91.89	86.8	90.79	84.21	91.65	92.1	92.43	87.3	91.22	85	92.3
InfoGain	92	88.89	86	91.8	85.9	92	91.7	88.59	85.88	91.32	85.64	91.8	92	89.1	86.1	92	86.22	92
SU	90.88	91.67	89.3	91	83.1	90.9	90.2	91	88.7	90.98	83	90.81	91.23	91.78	89.33	91.2	83.19	91.1
ReliefF	89.8	91	88.12	89.9	84	89.99	89.28	90.88	87.99	89.78	84	89.86	90	91.11	88.3	90.13	84.19	90
MIFS	89.98	91.8	86	90.88	84.77	91	89.81	91	85.87	90.45	84.19	90.79	90.11	92.1	86.11	91	84.98	91.31
MRMR	88.66	90.23	86	89.12	82	89.71	88.21	89.5	85.43	88.78	81.49	88.61	89	90.78	86.24	89.9	82.6	91
CMIM	90	89.78	87	90.78	84.89	91.67	88.64	89	86.65	89.71	84	90.7	90.3	90	87.9	91.22	85.1	92
ChiSquare	86.44	87.9	84.33	88	82.56	86.2	86	87.1	83.77	87.8	81.6	85.5	87	88.3	85.1	88.5	82	87.1
FRAMC	92	91.8	88	91.98	84.9	92.3	91.8	91.6	87.69	91.64	84.67	92	92.29	92	88.22	92	85.19	92.46