applied sciences

MDPI

Article

MALWD&C: A Quick and Accurate Machine Learning-Based Approach for Malware Detection and Categorization

Attaullah Buriro [1], Abdul Baseer Buriro [2], Tahir Ahmad [3,*], Saifullah Buriro [4] and Subhan Ullah [5]

[1] Faculty of Computer Science, Free University Bozen-Bolzano, 39100 Bolzano, Italy
[2] Electrical Engineering Department, Sukkur IBA University, Sukkur 65200, Pakistan
[3] Center for Cybersecurity, Brunno Kessler Foundation, 38123 Trento, Italy
[4] Faculty of Engineering Science & Technology, Deparment of Electrical Engineering, Indus University Karachi, Karachi 75500, Pakistan
[5] Department of Computer Science, National University of Computer and Emerging Sciences (NUCES-FAST), Islamabad 44000, Pakistan
* Correspondence: ahmad@fbk.eu

Abstract: Malware, short for malicious software, is any software program designed to cause harm to a computer or computer network. Malware can take many forms, such as viruses, worms, Trojan horses, and ransomware. Because malware can cause significant damage to a computer or network, it is important to avoid its installation to prevent any potential harm. This paper proposes a machine learning-based malware detection method called MALWD&C to allow the secure installation of Programmable Executable (PE) files. The proposed method uses machine learning classifiers to analyze the PE files and classify them as benign or malware. The proposed MALWD&C scheme was evaluated on a publicly available dataset by applying several machine learning classifiers in two settings: two-class classification (malware detection) and multi-class classification (malware categorization). The results showed that the Random Forest (RF) classifier outperformed all other chosen classifiers, achieving as high as 99.56% and 97.69% accuracies in the two-class and multi-class settings, respectively. We believe that MALWD&C will be widely accepted in academia and industry due to its speed in decision making and higher accuracy.

Keywords: malware detection and categorization; pattern matching; binary and multi-class classification

Citation: Buriro, A.; Buriro, A.B.; Ahmad, T.; Buriro, S.; Ullah, S. MALWD&C: A Quick and Accurate Machine Learning-Based Approach for Malware Detection and Categorization. *Appl. Sci.* **2023**, *13*, 2508. https://doi.org/10.3390/app13042508

Academic Editors: Konstantinos Rantos, Konstantinos Demertzis and George Drosatos

Received: 13 January 2023
Revised: 3 February 2023
Accepted: 13 February 2023
Published: 15 February 2023

1. Introduction

Currently, computers are playing an unavoidable and omnipresent role in our daily lives. They are used for various reasons: storing data, performing research, gaming, social networking, and entertainment, to name a few. However, they are facing continuous, ever-increasing, and constantly evolving cyber security threats. Therefore, it is crucial to implement effective and efficient security measures to protect computers and the sensitive data they contain. One such measure is the use of machine learning-based malware detection methods, such as MALWD&C, to identify and prevent the installation of malicious software on computers. By leveraging the power of machine learning algorithms, these methods can quickly and accurately classify files as benign or malware, allowing users to install only safe and secure software on their computers. In addition to protecting against cyber security threats, machine learning-based malware detection methods can also help organizations comply with regulations and standards, such as HIPAA and GDPR, which mandate the protection of sensitive data.

The injection of malicious code (malware) in benign software programs is the most common and powerful form of cyberattack to penetrate the target computer system and perform malicious activities, e.g., stealing confidential data and asking for ransomware, exploiting it for attacking other computers, etc. While all Operating Systems (OS), such as

Windows, Linux, macOS, etc., are vulnerable to such attacks and could be exploited using various file formats, e.g., Portable Executables (PE), Executable and Linkable Format (ELF), Portable Document Format (PDF), etc., we focus on malware with PE files in Windows OS domain because it holds 76.33% of the global OS market, followed by macOS 14.64% and Linux 2.42% [1] and has shown to be extremely popular among the end users. Consequently, the malware in the PE file format to attack Windows-based computer systems is the most studied threat in the wild [2]. To this end, Kaspersky lab reported the detection of 360,000 malware files in 2020 (including 90% Windows PE malwares) [3]. The same trend was shown to be continued in the year 2021 as well [4].

To combat this threat, researchers and security experts have proposed various approaches, such as signature-based [5–7], behaviour-based [8], and machine learning-based [9,10] detection, to identify and prevent the installation of malware in PE files on Windows-based computer systems. Signature-based detection [5] relies on known malware's unique patterns or characteristics to identify and block them. In contrast, behaviour-based detection monitors the behavior of software programs to detect suspicious activities that may indicate the presence of malware. These schemes depend on comparing the extracted signature from the current file to all the signatures maintained in the database of earlier collected and confirmed malware. Whether the current file is malware or benign depends on the similarity of the signatures. The main flaw of this approach is its dependence on the previously collected and confirmed malware signatures; hence, it can only detect the known malware. However, with the continuous and ever-increasing malware types, these schemes are not considered to be effective any more [11]. Consequently, the focus of the present research has been diverted to machine learning-based malware detection schemes [12,13].

Machine learning, including deep learning, has shown to be extremely successful in various domains such as computer vision, biometrics, speech recognition, etc. To this end, these schemes are also evaluated in the PE malware detection domain. Due to their generalizing capability, these models have been shown to generalize well on unseen testing malware samples. The schemes have shown to be accurate; however, their performance measurement (in terms of training and decision time) and their vulnerability to adversarial attacks are often neglected.

In this paper, we propose MALWD&C—a lightweight, accurate, and quick machine-learning-based approach for PE malware detection and prevention. The proposed MALWD&C approach is a lightweight and efficient solution for detecting and preventing the installation of malware in PE files on Windows-based computer systems. The proposed approach uses machine learning techniques to learn from large datasets of benign and malicious software programs and accurately classify new files as benign or malware. The MALWD&C approach is designed to serve in two scenarios: malware detection and categorization. In the malware detection scenario, the proposed approach processes the download .exe file, extracts the representative features and applies machine learning classifiers to predict the file's state (benign or malware). In the malware categorization scenario, the proposed approach predicts the type of the predicted malware and classifies it into one of the 14 families. The proposed MALWD&C approach achieved high accuracy, with a maximum of 99.56% for the malware detection scenario and 97.69% for the malware categorization scenario on a publicly available (BODMAS) [14] dataset. Overall, MALWD&C is a promising solution for detecting and preventing the installation of malware in PE files on Windows-based computer systems.

The main contributions of this paper are as follows:

- The proposal of an accurate, lightweight, and quick machine-learning-based approach, MALWD&C, for detecting and categorizing PE malware on Windows-based computer systems;
- An experimental validation methodology on a publicly available dataset to evaluate the feasibility and effectiveness of the proposed MALWD&C approach in detecting and categorizing PE malware;

- The identification of important features and their impact on the performance of MALWD&C in detecting and categorizing PE malware;
- An evaluation of multiple classifiers in terms of training and decision-making time, providing insights into the performance and efficiency of MALWD&C. This is the first study to report such results to the authors' knowledge.

Overall, the proposed MALWD&C approach offers a promising solution for detecting and preventing the installation of malware in PE files on Windows-based computer systems.

Paper organization

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 explains the basic idea of our approach. Section 4 presents the methodology we adapted for our solution. Section 5 reports and discusses the evaluation results. Section 6 concludes the paper with a summary of our major findings and the potential future research directions.

2. Related Work

Motivated by the tremendous success of ML techniques in different fields, several ML/DL-based malware detection approaches have been proposed in the recent literature [2,14–20]. In this section, we survey the most recent works that adapted machine learning classifiers for malware detection in PE files. Additionally, our survey will be limited to the papers that applied 1D feature classification.

The study by Schultz et al. [2] presents a data mining approach to detect new (previously unseen) malicious executable files. The proposed approach supports multiple methods of feature extraction and incorporates multiple machine learning classifiers. Their created dataset, consisting of 1000 benign and 3265 malicious samples, achieved 97.76% TPR using the Multi-Naive Bayes algorithm as the binary classifier. Similarly, another study [16] also exploits multiple machine learning classifiers, namely, Naive Bayes, J48, Support Vector Machine, and K-Nearest Neighbors for the same reason. On a dataset comprising 1,971 benign and 1,651 malicious samples, they obtained a TPR of 98% at the cost of 0.05% FPR using J48 as the classifier. Another study by Islam et al. [18] proposed a scalable and automated approach for detecting and classifying malware using several machine algorithms and statistical methods used at various stages of the malware analysis life cycle. Using k-fold cross-validation on the malware, which includes Trojans and viruses, along with 151 clean files, the authors achieve an overall classification accuracy of 98.86% using the top-performing Decision Tree as the classifier. Finally, in a recent study [20], the authors proposed a multi-layer hybrid approach, namely, TROJANDETECTOR, to detect the Trojan's abnormal behaviour in Android applications from three different Android levels based on the selected features and then apply multiple classifiers for the evaluation. The authors evaluated their scheme on three publicly available datasets and reported that the Support Vector Machine outperformed its counterparts and attained the highest accuracy of 96.64%.

Recent studies [14,17] present the two latest larger datasets, i.e., EMBER and BODMAS, respectively. The EMBER [17] dataset consists of multiple clusters of raw features (including raw features, format-agonstic histograms, and string counts) for training machine learning models in order to detect malicious Windows PE files. Similarly, the BODMAS [14] describe and release an open PE malware dataset called BODMAS to facilitate research efforts in machine learning-based malware analysis. A preliminary analysis to illustrate the impact of concept drift and discuss how this dataset can help to facilitate existing and future research efforts. The Ember dataset is a larger dataset comprising 900K training samples and 200K testing samples (100K benign, 100K malicious) and BODMAS data consists of 57,293 malware samples and 77,142 benign samples, with carefully curated family information (581 families). At the best-selected threshold of 0.871, the study [17] achieves a TPR of 92.99% at an FPR of 0.1%, and at 1% FPR, their model achieves 98.2%

TPR. The study [14] investigates the concept drift in malware analysis and applies different approaches; hence, their obtained results are not relevant to this study.

Our proposed approach, namely, MALWD&C, is different from the previously described machine learning-based malware analysis schemes in the following ways: (i) MALWD&C is a scenario-based approach to detect and categorize every PE file before allowing the authors to install. (ii) Performance evaluation of multiple classifiers regarding training and decision-making time. To the best of our knowledge, we are the first to report the results of such an evaluation. Finally, (iii) MALWD&C attains the highest accuracies of 99.56% and 97.69% in malware detection and categorization scenarios, respectively.

3. Malware Detection and Categorization

This section explains the use cases and the approach adopted by our solution.

3.1. Malware Detection

In the home environment shown in Figure 1, multiple Windows machines are connected to the same network and may attempt to install PE files. The proposed MALWD&C system, implemented on a macOS machine, is also part of the network. In this scenario, the task for MALWD&C is to predict if the downloaded executable file is malware or a benign application. To achieve this, MALWD&C processes the downloaded file, extracts the features, creates a feature vector, and tests it with a pre-trained model. The permission to install the application will only be granted if the application is predicted to be benign. In this scenario, the benign application (labelled as 0) is considered the positive class and malware (labelled as 1) are considered the negative class. By implementing the MALWD&C system, users can be confident that they are only installing safe and secure applications on their computers.

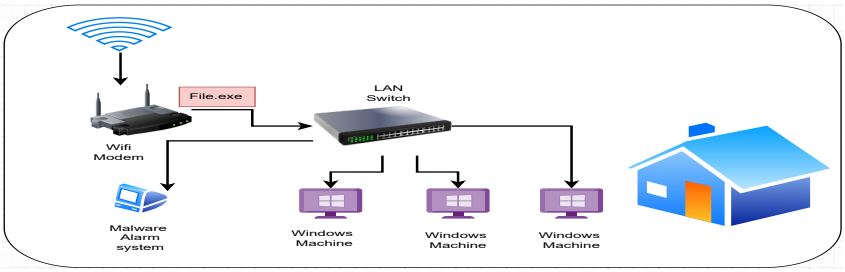


Figure 1. Malware detection scenario visualization.

3.2. Malware Categorization

Malware categorization is a critical problem in the field of cybersecurity. In the malware categorization scenario (as seen in Figure 2), MALWD&C not only detects if a downloaded executable file is benign or malware but also determines the specific family to which the malware belongs. This information is important for understanding the characteristics and behaviour of different malware families and developing effective countermeasures and prevention strategies.

However, publicly available datasets for this problem are limited [14,17], as cybersecurity companies and individuals often treat their collected labelled PE malware samples as private and do not share them with the public. This makes it challenging for researchers to develop and evaluate machine learning-based approaches for malware categorization. The proposed MALWD&C approach addresses this challenge by providing a lightweight and accurate solution for detecting and categorizing PE malware on Windows-based computer systems. In this scenario, anytime any executable file is downloaded, MALWD&C checks if it is a benign or malware application. If the file is predicted to be a malware application, it checks to which family this malware belongs. Then, it labels that malware file according to its family and saves it as a record for future analysis.

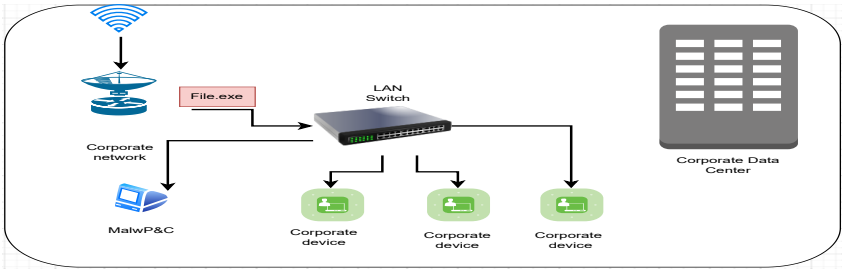


Figure 2. Malware categorization scenario visualization.

In this scenario, the classifier outputs the label of the testing sample for all the available classes. As only 14 classes had more than 1000 samples, we chose them for our analysis. Figure 3 illustrates the distribution of the available samples for each of the 14 classes.

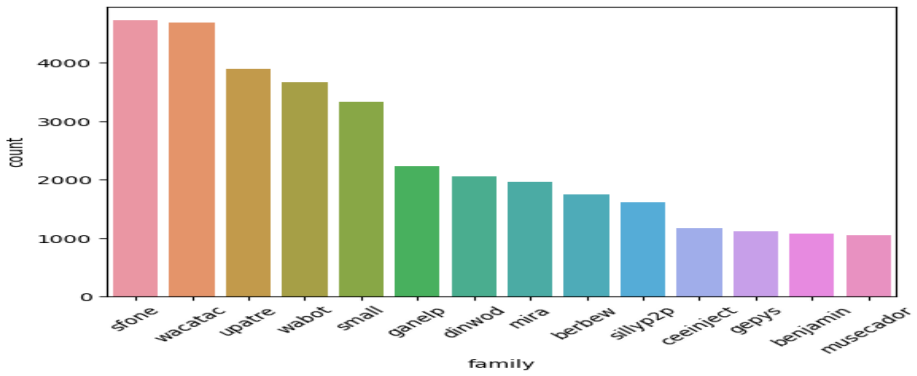


Figure 3. Distribution of the available samples per class.

4. Experimental Validation

4.1. Dataset

We evaluated our approach on the recently released publicly available dataset [14]. The shared **B**lue Hexagon **O**pen **D**ataset for **M**alware **A**nalysiS (BODMAS) dataset contains 57,293 and 77,142 malware and benign samples, respectively. These malware samples were collected during one year (from August 2019 to September 2020) and contain samples from 581 families.

4.2. Features

The BODMAS dataset contains two variants: raw binary files (malware only) and the extracted feature sets from each binary file. The authors extracted 2381 features long vectors from each binary file along with its label (0 for benign and 1 for malware) [14]. We have exploited these extracted feature datasets in this analysis.

In our dataset analysis as shown in Figure 4, we observed a high variance in the features. To address this issue, we applied feature scaling using the *StandardScalar* (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html accessed on 15 February 2023) approach. This method transforms the data so that its mean becomes 0 and its standard deviation is 1. We chose not to use the *MinMaxScalar* (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html accessed on 15 February 2023) approach because it is more suited for image data, which has fixed pixel values between 0 and 255. Since our dataset has a large variance, we considered *StandardScalar* a more appropriate choice for our analysis. By applying feature scaling, we improved the performance and accuracy of our machine-learning models.

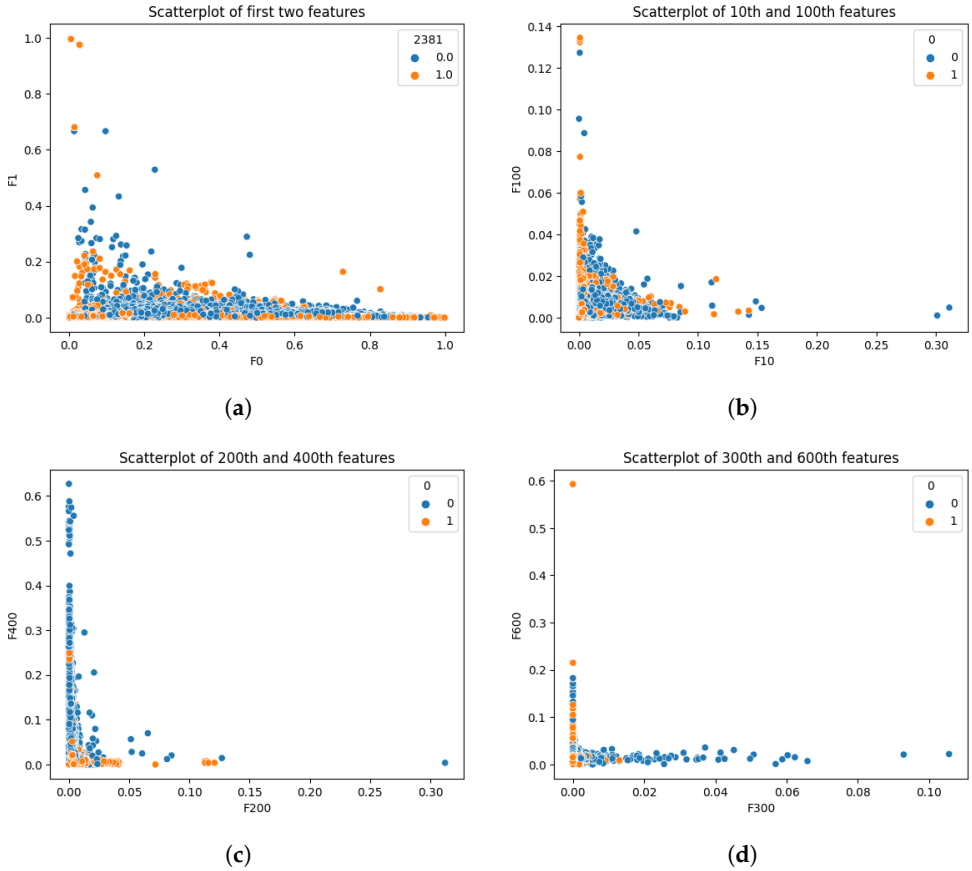


Figure 4. Scatter plot of some random features. (**a**) Scatterplot of first two features; (**b**) Scatterplot of 10th and 100th features; (**c**) Scatterplot of 200th and 400th features; (**d**) Scatterplot of 300th and 600th features.

4.3. Classifier Selection

The selection of classifiers for our analysis was based on several factors, including the size of the dataset, the time taken for model training, and the shape of the data. After initial analysis, we observed that the features were overlapping, which required a careful selection of classifiers. We aimed to use simple, lightweight, effective, quick-to-converge, and accurate classifiers for our analysis. Therefore, we chose several state-of-the-art machine learning classifiers for our analysis, including Gaussian Naive Bayes (GN), K-Nearest Neighbor (KNN), Gradient Boosting (GB), Multilayer Perceptron (MLP), Local Discriminant Analysis (LD), Logistic Regression (LR) and Random Forest (RF). These classifiers were implemented in python using the scikit (https://scikit-learn.org/stable/ accessed on 15 February 2023) learn library. We applied these chosen classifiers in their default settings in both the malware detection and categorization scenarios.

4.4. Feature Selection

Feature, attribute, or variable subset selection involves the computation of the most productive feature subset from the data.

Forward Feature Selection (FFS) [21] is a popular method for feature selection in machine learning. It is a greedy search algorithm that starts with an empty set of features and then iteratively adds the most relevant feature to the subset at each step. The relevant feature is determined by evaluating the model's performance on a validation set using a metric such as accuracy or F1 score. The algorithm stops when the desired number of features is reached or when no further improvement can be made by adding more features.

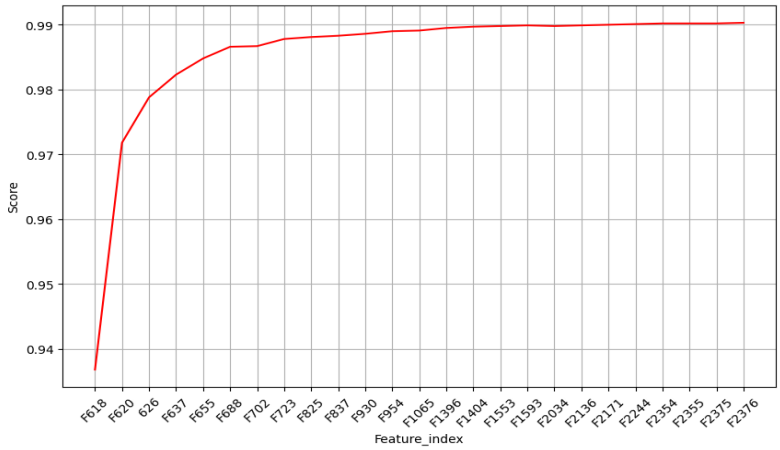
Sequential Feature Selection (SFS), a derivative of FFS, starts with all the features and iteratively keeps removing the least important features until a specific number of features is reached (25 in this paper) or any other stopping criteria are met. The importance of

each of the features can be determined using different methods, e.g., mutual information, chi-squared test or by using any classifier and evaluating it on a validation set. We used classifier-based feature selection and employed a Decision Tree for this task.

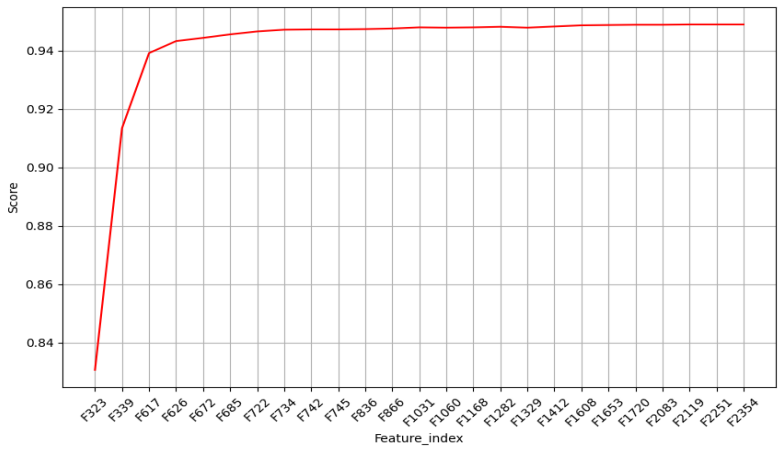
Decision Tree classifiers are commonly used for feature selection because they are simple to understand and interpret and provide a clear measure of feature importance. The feature importances calculated by a DT classifier can be used to identify the most relevant features for a given problem. This can help improve the model's performance by reducing overfitting and increasing generalization.

In summary, the FFS algorithm combined with a DT classifier can be an effective approach for feature selection in machine learning. It allows us to identify the most relevant features in the data, which can improve the model's performance. The approach starts with the computation of the most informative individual feature and stops when the desired number of selected features (25 in our case) is reached.

In Figure 5a,b, we illustrate the selected features and their score for malware detection and malware categorization scenarios, respectively. The X-tick on the X-axis shows the index of the features; e.g., F618 means Feature number 618, and on the Y-axis, we show its obtained score. For example, F618 and F323, when evaluated alone, yielded validation scores of 0.937 and 0.831 for malware detection and categorization scenarios, respectively. It can be seen that the maximum accuracy has already been reached using these 25 features, so we stopped SFS.



(**a**)



(**b**)

Figure 5. List of selected features and their scores. (**a**) SFS features for malware detection; (**b**) SFS features for malware categorization.

4.5. Experimental Settings

We divided the dataset into three parts: we used 60% of the samples for training, 15% for validation and 25% for testing to obtain fair and generalized testing results in both scenarios. Dividing the dataset into training, validation, and testing sets is a common practice in machine learning to ensure that the model is not overfitted to the data. We can avoid overfitting and obtain a more generalized model by using a separate validation set for evaluating the model during the training process. In preliminary analysis steps, i.e., feature selection, we used the validation set to test the Decision Tree classifier and obtain the validation accuracy. We intentionally kept the test set unseen during the initial analysis and used it only to obtain the test accuracy to compute the final classifier performance. Further, it is worth repeating that we applied to scale individually on all splits, i.e., training, validation, and testing sets, to avoid any data leakage.

5. Experimental Results

5.1. Performance Evaluation

We report our results in terms of the True Positive Rate (TPR), False Negative Rate (FNR), False Positive Rate (FPR), True Negative Rate (TNR), and Accuracy. We briefly define these performance indicators as such:

- True Positive Rate (TPR): The rate of correct classification of benign samples.
- False Positive Rate (FPR): The rate of incorrect classification of malware samples as benign.
- False Negative Rate (FNR): The rate of incorrect classification of benign samples as malware.
- True Negative Rate (TNR): The rate of correct classification of malware samples.
- Accuracy: The ratio of correct classifications to all the classification attempts. Mathematically,

$$\text{Accuracy} = \frac{\text{TPR} + \text{TRR}}{\text{TPR} + \text{FNR} + \text{FPR} + \text{TNR}} \quad (1)$$

- Receiver Operating Characteristics (ROC): It is a graphical plot between False Positive Rate (on the x-axis) and True Positive Rate (on the y-axis) used to depict the classification ability of a classifier over the different thresholds. The curve starts from coordinates (0,0) and ends at (1,1). The curve closer to coordinates (0,1) shows higher quality.
- Training time: The time taken by the classifier (in s) for training on the training set.
- Decision-making time: The average testing time (in s) taken by the classifier on the test set.

5.2. Results

In binary classification, where the goal is to predict whether a sample belongs to one of two classes, the classifier's predictions can be summarized using several evaluation metrics. True Positive Rate (TPR), also known as sensitivity or recall, is the number of samples correctly classified as positive divided by the total number of positive samples. False Positive Rate (FPR), also known as the fall-out, is the number of samples incorrectly classified as positive divided by the total number of negative samples. Accuracy is the number of correct predictions divided by the total number of samples.

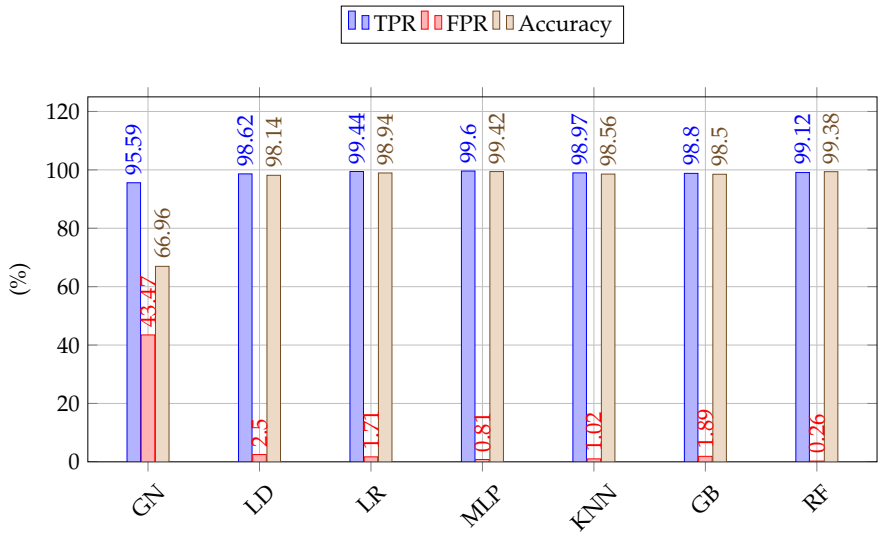
In the malware detection scenario, TPR corresponds to the probability that a benign sample is correctly classified as benign. In contrast, FPR corresponds to the probability that a malware sample is incorrectly classified as benign. In this case, a high TPR and low FPR are desirable, as it means that the classifier can accurately detect benign .exe files while avoiding false alarms.

In the malware categorization scenario, where the goal is to predict the family type of a malware sample, TPR corresponds to the probability that a sample is correctly classified as belonging to its true family type. FPR corresponds to the probability that a sample is incorrectly classified as belonging to a different family type. In this case, a high TPR and

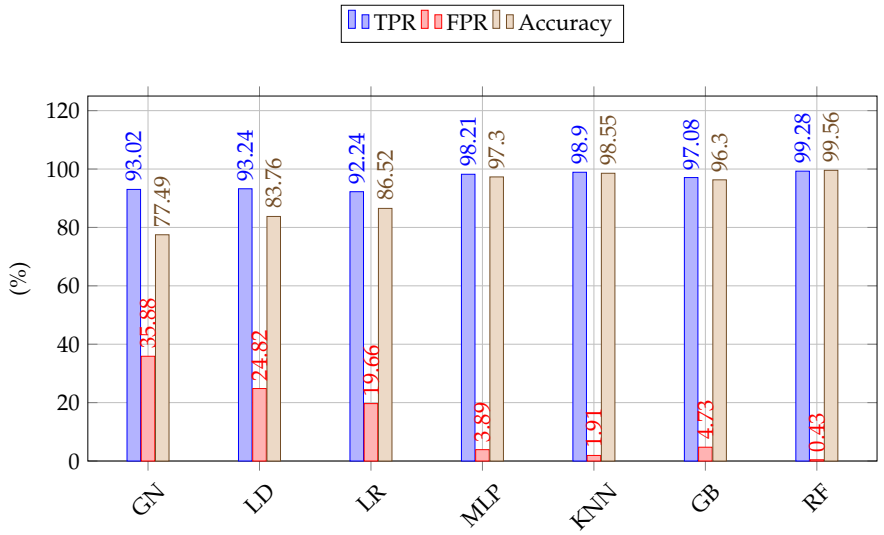
low FPR are desirable, as the classifier can accurately categorize malware samples into their correct family types.

Overall, TPR, FPR, and accuracy are useful metrics for evaluating the performance of a classifier in both malware detection and malware categorization scenarios. These metrics can help us understand how well the classifier can identify malware samples and classify them into their correct family types. We summarize our results in terms of TPR, FPR, and Accuracy. We do not report FRR and TRR because they can be computed easily by FNR = 1 − TPR and TRR = 1 − FPR, to avoid redundancy.

It is clear from the results reported in Figure 6a,b that the classifiers achieved high performance in the malware detection task. In particular, the MLP classifier performed well on the full set of features, achieving a TPR of 99.7% and an FPR of 0.81%. The RF classifier also performed well on the full set of features, achieving a slightly lower TPR of 99.12% but a lower FPR of 0.2%.



(**a**)



(**b**)

Figure 6. Results of different classifiers for malware detection problem. (**a**) Full Features; (**b**) Selected Features.

On the selected feature subset, the RF classifier was the top performer, achieving a TPR of 99.28%, an FPR of 0.43%, and an accuracy of 99.56%. This suggests that the RF classifier

can effectively identify malware samples using a smaller number of features, which can be useful in situations where the number of features is limited.

These results indicate that the classifiers could effectively identify malware samples with high accuracy and low false positive rates. The RF classifier showed strong performance on the selected feature subset, which can be useful in situations where the number of available features is limited.

The results reported in Table 1 show the training and testing times for the different classifiers. As expected, the training time decreases when the number of features is reduced since the classifiers have to process fewer features to learn the hidden patterns in the data. For example, the training time for the RF classifier is 117.09 s on the full set of features but only 4.2691 s on the selected feature subset.

It is also worth noting that the testing time, which is the time taken by the classifier to make predictions on a new sample, also decreases when the number of features is reduced. This can be beneficial in real-world scenarios, where making predictions quickly and efficiently is important.

Overall, the results in Table 1 show that reducing the number of features can lead to faster training and testing times for the classifiers without sacrificing performance. This can be useful in situations where the processing time is a critical factor.

Table 1. Training and average testing time of all chosen classifiers for malware detection scenario.

	Full Features		**Selected Features**	
Classifiers	**Training Time (s)**	**Testing Time (s)**	**Training Time (s)**	**Testing Time (s)**
GN	0.76	0.0000622	0.0294	0.0000458
MLP	4736.90	0.013	63.375	0.0000308
KNN	0.10	0.37	0.00976	0.00476
GB	1399.86	0.0000715	9.02	0.0000707
LR	10.97	0.0000220	0.1791	0.0000275
LD	49.58	0.0000232	0.1624	0.0000236
RF	117.69	0.00294	4.2691	0.00270

The results reported in Figure 7a,b show that the classifiers achieved high performance in the malware categorization task. In particular, the RF classifier performed well on the full features and the selected feature subset, achieving TPRs of 95.6% and 95.8%, respectively. This indicates that the RF classifier can effectively categorize malware samples into their correct family types with high accuracy.

It is also worth noting that the performance of the RF classifier improved slightly on the selected feature subset, achieving a higher TPR and accuracy. This suggests that the RF classifier can achieve better performance using a smaller number of features, which can be useful in situations where the number of features is limited.

These results indicate that the classifiers could effectively categorize malware samples into their correct family types with high accuracy. The RF classifier showed strong performance on both the full set of features and the selected feature subset, demonstrating its ability to achieve high performance using a smaller number of features.

In Table 2, we report the training time taken by the classifiers to learn the hidden patterns in the data and perform multi-class classification. It is important to note that the specific training times reported in Table 2 are specific to the dataset and classifiers used in that particular scenario and may not apply to other datasets or classifiers. Training time can vary widely depending on several factors, such as the size and complexity of the dataset, the type of classifier being used, the number of features in the dataset, and the computing power available.

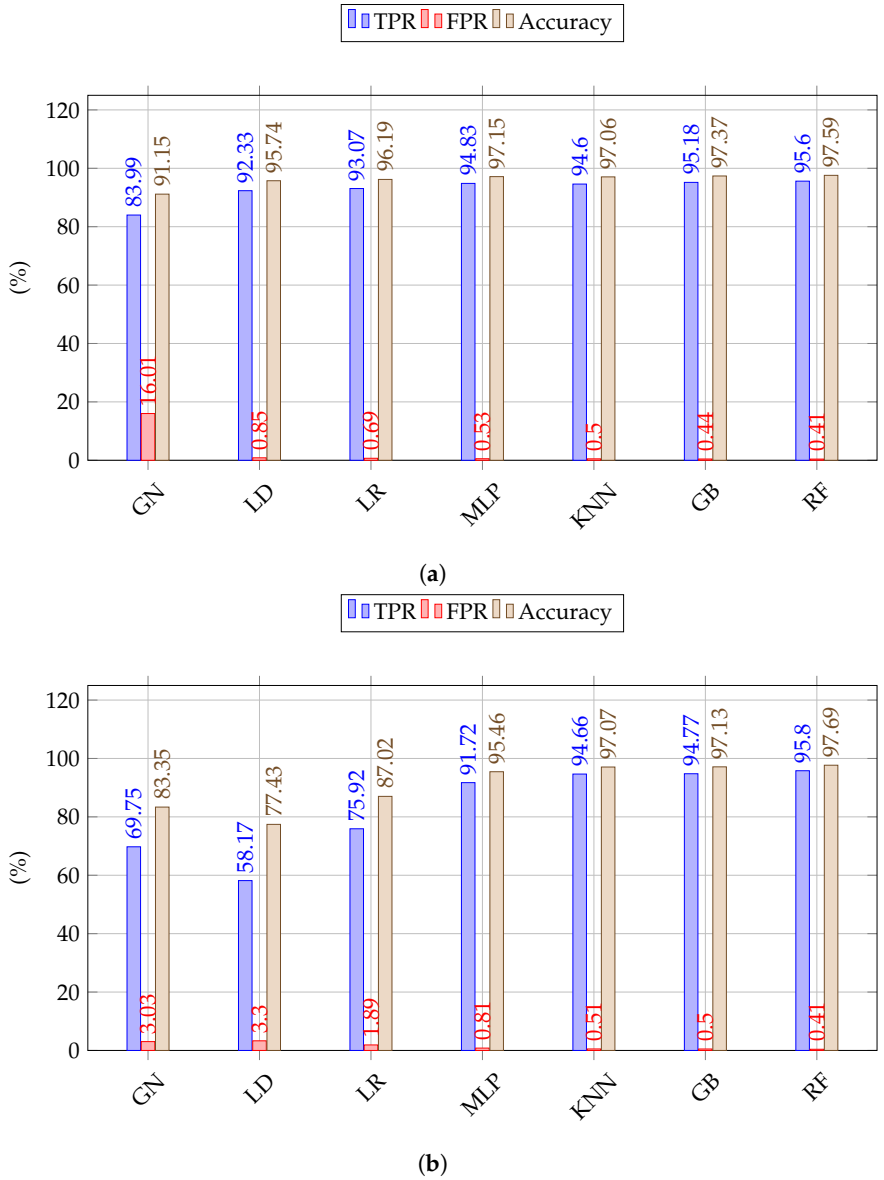


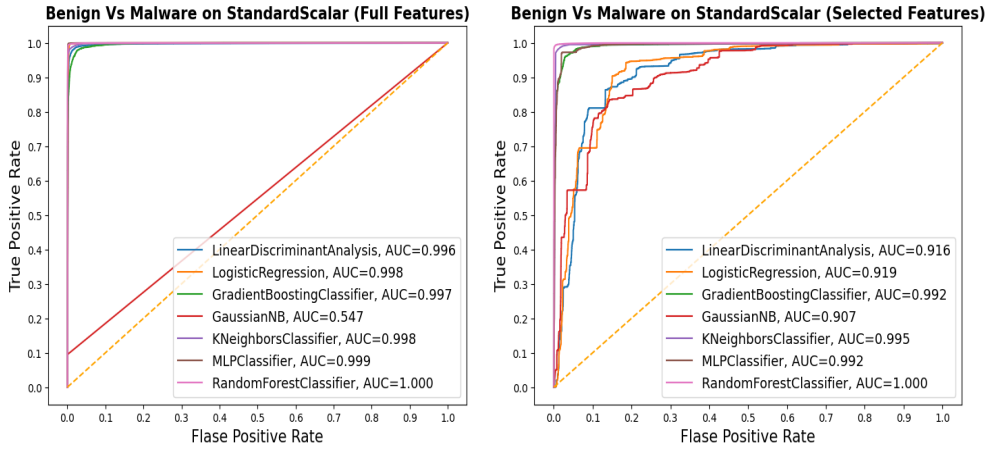
Figure 7. Results of different classifiers for malware categorization (**a**,**b**) problem. (**a**) Full features; (**b**) selected features.

Table 2. Training and average testing time of all chosen classifiers for malware categorization scenario.

	Full Features		**Selected Features**	
Classifiers	**Training Time (s)**	**Testing Time (s)**	**Training Time (s)**	**Testing Time (s)**
GN	0.457	0.000303	0.0258	0.00015
MLP	123.689	0.000185	43.586	0.0000352
KNN	0.194	0.02194	0.0289	0.00078
GB	3352.254	0.00025	24.910	0.000205
LR	14.31	0.0000426	3.469	0.0000213
LD	23.379	0.0000620	0.0954	0.0000379
RF	14.855	0.00288	1.251	0.00280

In the scenario described, the training time for the classifiers decreased when using selected features rather than the original ones. This may be because fewer features can make the training process more efficient, as the classifier has less information to process. However, it is also possible that the classifier's performance may be affected by using only a subset of the original features, so it is important to carefully evaluate the trade-off between training time and performance when deciding which features to use. Similar to the malware detection scenario, the training time for the chosen classifiers in multi-class settings also decreases on selected features compared to the original ones. In this scenario, the best-performing RF classifier took 14.855 s and 1.251 s, respectively, on the full and selected features without affecting the performance.

We also summarize the performance of the chosen classifiers as ROC curves in Figure 8a (on full features) and Figure 8b (on selected features). We illustrate the ROC curves for malware detection problems because, in this scenario, they make more sense. ROC curves show that the RF classifier was accurate and consistent in both settings: on the full dataset Figure 8a and the selected subset of features Figure 8b.



(**a**) (**b**)

Figure 8. ROC curves for difference classifiers for malware detection problem. (**a**) Full Features; (**b**) Selected Features.

6. Conclusions and Future Work

In this paper, we have proposed a lightweight, quick, and accurate machine-learning-based approach for malware detection and categorization, namely MALWD&C. We tested our proposed scheme in two scenarios: (i) malware detection (benign vs. malware) and (ii) malware categorization (multi-class classification).

The proposed MALWD&C approach uses machine learning to detect and classify malware. The approach was tested on the BODMAS [14] dataset, and the random forest classifier was found to be the most effective. In the malware detection scenario, the approach achieved an accuracy of 99.38% on full features and 99.56% on a selected subset of 25 features, with a reduced training time of 4.269 s. In the malware categorization scenario, the approach achieved an accuracy of 97.59% on full features and 97.69% on selected features, with a reduced training time of 1.25 s. Overall, the experiments show the effectiveness of the proposed approach in detecting and classifying malware with high accuracy and relatively fast training and testing times.

In summary, our proposed MALWD&C approach demonstrates promising malware detection and categorization results. Using a RF classifier and feature selection enables an accurate and efficient classification of malware samples. Our approach can potentially be used in real-world applications to quickly and accurately identify and classify malware samples. Future work includes exploring other advance classifiers, such as Convolutional Neural Networks (CNNs), on the best feature subset. Since the RF classifier achieves a higher accuracy and has comparatively better training and testing times, we pick this

classifier for our final proof-of-the-concept application. We leave the implementation of the proof-of-the-concept application and its evaluation for future work. Additionally, another avenue for future work is to expand the dataset and include more diverse malware samples. This will increase the robustness of the proposed approach and ensure that it can handle a wide range of malware variants. Furthermore, incorporating other feature extraction techniques, such as graph-based representation or embeddings, could also be interesting. Finally, investigating the potential to use MALWD&C in a real-time setting, and evaluating its performance on large-scale datasets can be a promising direction. In conclusion, the results of our proposed MALWD&C approach are promising and offer new opportunities for further research in the field of malware detection and categorization.

Author Contributions: A.B.—Conceptualization, Data curation, Software, Writing—original draft, Investigation, Validation and Visualization; A.B.B.—Methodology, Formal Analysis, Resources, Writing—original draft, Validation, and Investigation; T.A.—Writing—original draft, Writing—review and editing, Funding acquisition, Validation, and Visualization; S.B.—Methodology, Supervision, Writing—original draft, Project Administration, and Visualization; S.U.—Methodology, Software, Resources, Writing—original draft, and Visualization. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Available upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. How Many People Use Windows in 2022? (New Statistics). Available online: https://earthweb.com/windows-users/ (accessed on 20 November 2022).
2. Schultz, M.G.; Eskin, E.; Zadok, F.; Stolfo, S.J. Data mining methods for detection of new malicious executables. In Proceedings of the 2001 IEEE Symposium on Security and Privacy, S&P 2001, Oakland, CA, USA, 13–16 May 2001; pp. 38–49.
3. The Number of New Malicious Files Detected Every Day Increases by 5.2% to 360,000 in 2020. Available online: https://www.kaspersky.com/about/press-releases/2020_the-number-of-new-malicious-files-detected-every-day-increases-by-52-to-360000-in-2020 (accessed on 20 November 2022).
4. New Malicious Files Discovered Daily Grow by 5.7% to 380,000 in 2021. Available online: https://www.kaspersky.com/about/press-releases/2021_new-malicious-files-discovered-daily-grow-by-57-to-380000-in-2021 (accessed on 20 November 2022).
5. Savenko, O.; Nicheporuk, A.; Hurman, I.; Lysenko, S. Dynamic Signature-based Malware Detection Technique Based on API Call Tracing. *ICTERI Work.* **2019**, *2393*, 633–643.
6. Sahoo, A.K.; Sahoo, K.S.; Tiwary, M. Ignature based malware detection for unstructured data in Hadoop. In Proceedings of the IEEE International Conference on Advances in Electronics Computers and Communications, Bangalore, India, 10–11 October 2014; pp. 1–6.
7. Panigrahi, C.R.; Tiwari, M.; Pati, B.; Prasath, R. Malware detection in big data using fast pattern matching: A hadoop based comparison on GPU. In Proceedings of the Second International Conference on Mining Intelligence and Knowledge Exploration, MIKE 2014, Cork, Ireland, 10–12 December 2014; pp. 407–416.
8. Urmila, T.S. Machine learning-based malware detection on Android devices using behavioral features. *Mater. Today Proc.* **2022**, *62*, 4659–4664. [CrossRef]
9. Allix, K.; Bissyande, T.F.D.A.; Klein, J.; Le Traon, Y. *Machine Learning-Based Malware Detection for Android Applications: History Matters!*; Tech Report; University of Luxembourg, SnT: Esch-sur-Alzette, Luxembourg, 2014.
10. Chumachenko, K. *Machine Learning Methods for Malware Detection and Classification*; Kaakkois-Suomen Ammattikorkeakoulu: Kouvola, Finland, 2017.
11. Malware Detection Methods Struggle to Keep up with Evolving Threats. Available online: https://www.techtarget.com/searchsecurity/news/252472645/Malware-detection-methods-struggle-to-keep-up-with-evolving-threats (accessed on 20 November 2022).
12. Ucci, D.; Aniello, L.; Baldoni, R. Survey of machine learning techniques for malware analysis. *Comput. Secur.* **2019**, *8*, 123–147. [CrossRef]
13. Ceschin, F.; Gomes, H.M.; Botacin, M.; Bifet, A.; Pfahringer, B.; Oliveira, L.S.; Grégio, A. Machine learning (in) security: A stream of problems. *arXiv* **2020**, arXiv:2010.16045.

14. Yang, L.; Ciptadi, A.; Laziuk, I.; Ahmadzadeh, A.; Wang, G. BODMAS: An open dataset for learning based temporal analysis of PE malware. In Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27–27 May 2021; pp. 78–84.
15. Singh, J.; Singh, J. A survey on machine learning-based malware detection in executable files. *J. Syst. Archit.* **2021**, *112*, 101861. [CrossRef]
16. Kolter, J.Z.; Maloof, M.A. Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.* **2006**, *7*, 2721–2744.
17. Anderson, H.S.; Roth, P. Ember: An open dataset for training static pe malware machine learning models. *arXiv* **2018**, arXiv:1804.04637.
18. Islam, R.; Tian, R.; Batten, L.; Versteeg, S. Classification of malware based on string and function feature selection. In Proceedings of the 2010 Second Cybercrime and Trustworthy Computing Workshop, Ballarat, VIC, Australia, 19–20 July 2010; pp. 9–17.
19. Rieck, K.; Holz, T.; Willems, C.; Düssel, P.; Laskov, P. Learning and classification of malware behavior. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Paris, France, 10–11 July 2008; pp. 108–125.
20. Ullah, S.; Ahmad, T.; Buriro, A.; Zara, N.; Saha, S. TrojanDetector: A Multi-Layer Hybrid Approach for Trojan Detection in Android Applications. *Appl. Sci.* **2022**, *12*, 10755. [CrossRef]
21. Forward Feature Selection and Its Implementation. Available online: https://www.analyticsvidhya.com/blog/2021/04/forward-feature-selection-and-its-implementation/ (accessed on 23 April 2022).