An Exploratory Analysis of Feature Selection for Malware Detection with Simple Machine Learning Algorithms

Md Ashikur Rahman, Syful Islam, Yusuf Sulistyo Nugroho, *Member, IEEE,* Fatah Yasin Al Irsyadi, and Md Javed Hossain*

Original scientific article

***Abstract*—Computers have become increasingly vulnerable to malicious attacks with an increase in popularity and the proliferation of open system architectures. There are numerous malware detection technologies available to protect the computer operating system from such attacks. This type of malware detector targets programs based on patterns detected in the properties of computer applications. As the amount of analytical data increases, the computer defense system is adversely affected. The performance of the detection mechanism has been hindered due to the presence of numerous irrelevant characteristics. The goal of this study is to provide a feature selection approach that will help malware detection systems be more accurate by detecting pertinent and significant traits. Furthermore, by selecting the most important features, it is possible to maintain an acceptable level of accuracy in the detection of malware while significantly lowering the computational cost. The proposed method displays the most important features (MIFs) obtained from each machine learning method, including data cleaning and feature selection. Furthermore, the method applies six machine learning classification techniques to the selected feature set. Several classifiers were evaluated based on several characteristics for malware detection, including Support Vector Machines (SVM), Logistic Regression (LR), K-nearest neighbor (K-NN), Decision Tree (DT), Naive Bayes (NB), and Random Forest (RF). Our suggested model was tested on two malware datasets to determine its effectiveness. In terms of accuracy, precision, F1 scores, and recall, the experimental findings show that RF and DT classifiers beat other techniques.**

***Index Terms*—Malware Detection, Machine Learning, Feature Selection, Information Gain, Cybersecurity.**

I. Introduction

Malware stands for malicious software and is intended to harm systems and users by stealing information, destroying data, or simply irritating them. Malware is widely distributed and computer security issues are on the rise, according to reports [1]. Malicious programs or attacks, such as malware and ransomware families, continue to pose serious cybersecurity concerns, with potentially devastating effects [2]. Computer systems, data centers, the Web, and mobile devices have been damaged, with applications in a variety of sectors and companies [3], [2], [4]. By encrypting data in an unbreakable format that can only be decoded by the attacker, the majority of malware is designed to restrict victims from accessing computer data [5]. Since removing the infection results in irreversible damage, victims are compelled to comply with the attacker's demands [6]. If anyone does not comply with the attacker's demands, data will be permanently gone. Assailants are using current technology to turn traditional malware into developing malware families, making it more difficult to recover from a malware infection [7], [8].

Malware is a sophisticated and diverse threat that affects people around the world and prohibits them from accessing their system or data until they pay for it [4]. It does this by locking the system's screen or encrypting the files of the users. The two primary varieties of malware, depending on attack strategies, are locker malware, which prevents access to the computer or device, and crypto malware, which prevents access to files or data [9]. It is quite difficult to recover without paying the extortion after these attacks. Due to the above problems, malware detection has become very important for us. Traditional malware detection methods, such as event-based, statistical-based, and data-centric methods, are ineffective [9]. As a result, the research community should focus on achieving the maximum degree of optimal protection and security by using futuristic technologies against such advanced hostile assaults.

Today, many researchers use machine learning in malware detection, it is a novel research field that might be very useful in the creation of creative malware solutions [10]. The use of Machine Learning (ML) techniques allows for the automated identification of malware, including ransomware, based on their dynamic characteristics, which improves security [11], [12]. Support Vector Machines (SVM), Logistic Regression (LR), Decision Tree (DT), Naive Bayes (NB), Random Forest (RF), and Neural Network (NN) based architectures have the ability to identify and classify malware [13].

For malware detection with machine learning in general follow two steps, feature extract from the program file and then classify malware based on these features. Feature extraction from a program file is more costly and time-consuming for

Manuscript received June 20, 2023; revised June 30, 2023. Date of publication September 13, 2023. Date of current version September 13, 2023.

M. A. Rahman and M. J. Hossain are with the Noakhali Science and Technology University, Bangladesh (mdashikur567@gmail.com, javed@nstu.edu.bd).

S. Islam is with the Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Bangladesh (syfulcse@bsmrstu.edu.bd).

Y. S. Nugroho and F. Y. Al Irsyadi are with the Universitas Muhammadiyah Surakarta, Indonesia ({yusuf.nugroho, fatah.yasin}@ums.ac.id).

Digital Object Identifier (DOI): 10.24138/jcomss-2023-0091

*Corresponding author

malware detection. That's why in this study we thoroughly analyze and explore the use of machine learning algorithms for selecting features to detect malware.

Using automated technologies, a malware producer can easily build a significant number and variety of malware. As a result, we're employing machine learning to tackle the following issues: (i) How to get the best feature set for malware detection? (ii) How to classify malware with better accuracy?

In this paper, we present a feature selection methodology for malware detection. Optimal feature selection and malware categorization are the two key aspects of this approach. The following contributions have been made by us: To classify malware, look for the most important features (MIFs). We use a data cleaning strategy (removing zero and near-zero variance predictors) and a feature selection method to get the best results (Information Gain). Then, on the basis of these attributes, we employ machine learning classifier methods to categorize malware. The algorithms used to classify are Support Vector Machines (SVM), Logistic Regression (LR), K-Nearest Neighbors (K-NN), Decision Tree (DT), Naive Bayes (NB), and Random Forest (RF). In this case, RF and DT demonstrate a greater accuracy.

Using our model, we are able to identify the most effective features for a given malware dataset. When these features are extracted from a program file during the feature extraction process, it results in reduced time and cost. Initially, our malware dataset comprised 53 features. However, after selecting the 9 features that yielded the highest accuracy, subsequent extractions only require these 9 chosen features from the program file. This streamlined approach of extracting 9 features significantly minimizes both the time required and associated costs when compared to extracting all 53 features.

The feature selection that we have done here will be more useful for malware analysis and detection. That is because next time we will only extract the selected feature from the program file so that the feature extraction will be very fast. And the machine learning algorithm will be able to respond much faster with fewer features.

We have shown in our research how to select the best features from a dataset out of many features, maintain good accuracy and also reduce time consumption and memory cost.

The remainder of the paper is laid out as follows: In Section II, we go through some of the ML-based malware categorization work that has been done. The strategies we used in this work are explained in Section III. In Section IV, the experimental setup and findings are described. Finally, Sections V and VI bring the paper to a close.

II. Literature Review

Various malware has been classified using traditional detection approaches. A well-defined behavioral structure can be used to evaluate various malware, and most malware families share common behavioral aspects such as payload persistence, stealth tactics, and network activity. The most extensively used conventional anti-malware system is signature-based analysis, and Abiola and Marhusin suggested a signature-based malware detection model by extracting the Brontok worms and using an n-gram approach to break down the signatures [14]. The architecture allows for malware identification and the creation of a trustworthy solution that removes all dangers. The static-based analysis examines the application's code for malicious activities without executing or running the code[15], while dynamic analysis monitors the behavior of malicious intent processes. Processes that exhibit malicious intent behavior will be flagged as suspicious and terminated in order to address this flaw [16]. Both static and dynamic analysis have limitations in terms of detecting undiscovered malware and are ineffective against code obfuscation, high variation output, and targeted assaults.

EldeRan, a machine learning-based technique for dynamically evaluating and categorizing ransomware, analyzes benign software actions based on probable distinctive symptoms of ransomware [17]. EldeRan employs two types of machine learning components: feature selection and classification, both of which are implemented in the Cuckoo Sandbox environment. To obtain and dynamically analyze datasets, Windows API calls, Registry Key Operations, File System Operations, the set of file operations done per File Extension, Directory Operations, Dropped Files, and Strings are utilized. The method was tested with 582 ransomware files from 11 different families, as well as 942 good-ware programs, producing a 0.995 area under the ROC curve, showing accuracy. Another prior work has developed a method for categorizing API calls in binary sequences that use Long-Short-Term Memory (LSTM) networks to classify malware based on its behaviors [18]. In the sandbox environment, a dynamic analysis approach was used to retrieve API calls from the changed log. The suggested LSTM-based system obtained 96.67% accuracy in automatically categorizing ransomware activity from a malware dataset, according to the evaluation. The authors acknowledge the potential for accuracy improvement through larger datasets and LSTM network optimization.

Such precision suggests that machine learning might be a realistic and effective method for detecting new ransomware variations and families. Deep Neural Network (DNN) can tackle complicated detection issues and may be used to identify ransomware by building a revolutionary dynamic detection approach. In a prior study, Bayesian Hyperparameter Optimization was applied for deep neural network-based network intrusion detection [19]. Ghanei et al. released a study lately in which they proposed a dynamic malware detection system based on Deep Neural Network (DNN) and Convolutional Neural Network (CNN) [20]. The machine learning model is built using Long Short-Term Memory (LSTM). According to the evaluation report, the combination of DNN with LSTM is successful in the identification of new malware, with an accuracy of 91.63%. Masum et al. suggested a feature selection-based framework for ransomware detection and prevention that uses a variety of machine learning methods, including neural network-based designs [21]. They used a variety of machine learning methods for ransomware classification, including Decision Tree (DT), Random Forest (RF), Nave Bayes (NB), Logistic Regression (LR), and Neural Network (NN) based classifiers. By using a variance threshold and VIF criterion value, they choose 13 features from a data collection. The suggested framework was optimized using a small number of key characteristics and tested with several machine learning classifiers. The experimental findings demonstrate the resilience and efficacy of the suggested framework.

In Android, deep learning has also been used to identify malware. Hossain and Riaz presented a malware detection method for Android that combines static and dynamic analysis for both machine learning and deep learning classifiers [22]. They attempted to analyze a multilayer detection approach that would work on both static and dynamic data from user permissions and network traffic. The model will use user permissions to identify malware before it is installed from the Android Manisfest.xml file, and network traffic data will be used to detect malware at runtime. Bakour & Ünver demonstrated how to use deep learning and image-based features to identify malware on Android devices [23]. By transforming various files from Android application sources into grayscale images, they produced four datasets of grayscale photos. The suggested model was then trained using two categories of image-based characteristics, local features and global features, which were taken from the created image dataset. They achieved extremely effective run time overhead ranging between 0.11 to 2.02 s for each sample, which resulted in classification accuracy of more than 98%. Additionally, they tested the robustness of Android's anti-malware systems against hybrid obfuscation techniques and injection attacks. Two injection attacks (i.e. the benign permissions injection attack and the benign permissions code injection attack) were suggested [24].

III. Methodology

This research is experimental research, in which we take datasets and find the best feature for classification. Even after removing some less important features, our accuracy remained at a good level, while the memory cost and computational cost also decreased. As illustrated in Fig.1, our model consists of modules for *data cleaning*, *feature selection*, *classification* and finaly *evaluation*.

A. Data Cleaning (Remove Zero and Near Zero Variance Predictors)

For data cleaning, we removed zero and near zero variance predictors. Removing zero and near-zero-variance predictors refers to the process of identifying and eliminating variables (predictors) that have very little or no variability in their values. These predictors do not provide useful information for predictive modeling tasks and can potentially introduce noise or cause computational issues[25].

Zero Variance Predictors: Features with zero variance have a fixed and meaningless value across all data points. It is safe to eliminate these features because they do not increase the variability of the data.

Near-Zero Variance Predictors: Near-zero variance features have values that are essentially constant with little variation. Although they may not be completely constant, they offer relatively little information and are not likely to significantly affect the performance of the model.

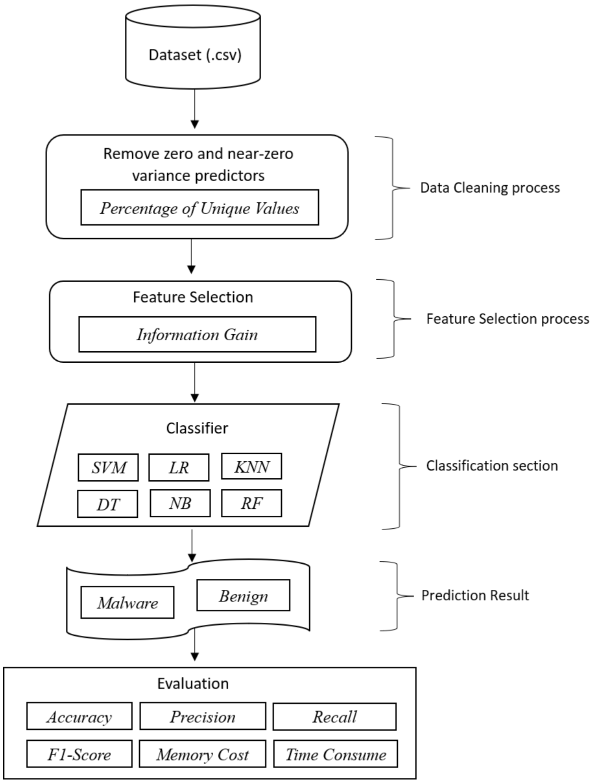


Fig. 1. Malware Classification Model. Our model consists of 5 steps. These are Data Cleaning, Feature Selection, Classification, Prediction and Evaluation.

In comparison to the number of samples, zero and near-zero variance predictors have a small number of unique values or a single dominant value that occurs frequently. Identifying these variables may be done in a number of ways, including checking for unique values, constructing frequency tables, and so on. In this experiment, the percentage of unique values is utilized to discover zero variance features.

1) Percentage of Unique Values: It's calculated by dividing the total number of samples by the number of unique values, which decreases as the data's granularity increases.

$$\text{PUV} = \frac{Number\ of\ Unique\ Value}{Total\ row\ number\ of\ dataset} \times 100 \qquad (1)$$

A variable will be classified as a zero variance predictor if it exhibits a frequency of unique values lower than a specified limit and exceeds the predetermined frequency ratio criterion. We remove various features from our dataset based on the percentage of unique values. It helps us to reduce the number of features in the beginning.

B. Feature Selection (Information Gain (IG))

Through the data cleaning process, some features will be removed. Then we'll apply the information gain methods to find out which features have a good impact on our classification accuracy. Information Gain (IG) is a feature assessment approach based on entropy that is widely used in machine learning. IG can also be used to measure the amount of information provided by a feature with respect to a target variable [26]. The Information Gain (IG) metric evaluates the extent to which a feature contributes valuable information about the target class. IG can detect the features with the highest information based on the target class. High IG characteristics are highly relevant to the target class and are frequently used to improve classification results. As a result, we will have to keep deleting non-essential features. Entropy generates IG, as seen in the (2) to (4) equations [27]. The formula for analyzing more than two classes using entropy is presented below.

$$\mathrm{H(X)}=-\sum_{i=1}^{K} P\left(x_i\right) \log_2 P\left(x_i\right) \tag{2}$$

The number of classes/features of a dataset is K. And the following formula is used to determine the IG of a feature X and the class label Y.

$$\mathrm{IG(X,Y)}=\mathrm{H(X)}-\mathrm{H(X\mid Y)} \tag{3}$$

$$\mathrm{H(X\mid Y)}=-\sum_{j} P\left(y_j\right) \sum_{i} P\left(x_i \mid y_i\right) \log_2\left(P\left(x_i \mid y_i\right)\right) \tag{4}$$

Entropy is a measure of a class's unpredictability based on the likelihood of a particular event or feature occurring. Entropy and IG are inversely proportional. The amount of information accessible before learning the attribute value and the amount of information available after learning the attribute value are the two main determinants of the amount of knowledge gained. For multiclass, the maximum IG value is 1.

Although there are many other methods besides IG for feature selection, such as PCA, autoencoders etc, our primary focus in this study was to evaluate the effectiveness of the chosen approach specifically. In addition, information gain has been widely used in various datasets; it is also useful for a dataset that has a hybrid data type. Agrawal and Kshirsagar proposed an information gain-based feature selection method for Android malware detection [28].Kumar et al. proposes a method for mobile malware detection that uses information gain and support vector machine [29]. Shu et al. proposes an information gain-based semi-supervised feature selection algorithm for hybrid data, considering symbolic, numerical, and missing features, and demonstrates its superiority over other feature selection methods using experiments on ten UCI datasets. The article presents a sentiment analysis system for distance education that integrates information gain (IG) methods for feature selection[30].

C. Classification and Parameter Tuning

The algorithms used to classify are Support Vector Machines (SVM), Logistic Regression (LR), K-Nearest Neighbors (K-NN), Decision Tree (DT), Naive Bayes (NB), and Random Forest (RF).

1) Support Vector Machine (SVM): SVM is a powerful classification algorithm that finds an optimal hyperplane to separate malware classes.

Parameter Tuning: Utilizing the radial basis function (RBF) kernel to enhance performance, with a specific focus on optimizing the values of gamma and C, which are the key parameters associated with the RBF kernel.

2) Logistic Regression (LR): LR estimates the probability of malware or non-malware instances based on their features.

Parameter Tuning: Tune Solver and Maximum Iterations to balance fitting and generalization.

3) k-Nearest Neighbors (k-NN): k-NN classifies instances by majority vote from their k nearest neighbors using distance metrics.

Parameter Tuning: Experiment with different k values, metric, and P values for an optimal trade-off between overfitting and smoothing.

4) Decision Tree (DT): DT recursively splits data based on feature thresholds to create a tree-like classifier.

Parameter Tuning: Tune criterion, max depth, min samples to split a node, and min samples for leaf nodes to control complexity.

5) Naive Bayes (NB): is a probabilistic classifier based on Bayes' theorem, employing the 'naive' assumption of independence.

6) Random Forest (RF): Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions for classification. It is robust, handles high-dimensional data well, and helps reduce overfitting.

Parameter Tuning: Important hyperparameters include the number of trees (n_estimators), the maximum depth of each tree (max_depth), and the number of features considered for each split (max_features).

D. Evaluation Metrics

In this section, we explain how we evaluate the performance of our machine learning models. We use a variety of metrics to get a comprehensive understanding of how well our models are working. These metrics measure different aspects of model performance, such as accuracy, precision, recall, and resource utilization. By understanding how our models perform, we can identify areas where they can be improved.

1) Classification Evaluation Metrics: Our evaluation process begins with traditional classification metrics that focus on the models' predictive capabilities:

- Accuracy: Measures overall correctness (correct predictions/total predictions), but may be inadequate for imbalanced datasets.
- Precision: Proportion of true positives over total predicted positives, valuable when false positives are costly.
- Recall (Sensitivity): Proportion of true positives over total actual positives, crucial for identifying all positive instances.
- F1-Score: Harmonic mean of precision and recall, balances evaluation for imbalanced classes.

2) Performance and Resource Metrics: In addition to classification metrics, we consider custom metrics that shed light on the efficiency and resource utilization of the models:

- Build Time: This metric records the time (in seconds) needed to train each model on the training dataset. It reflects the efficiency of the algorithm during the training phase.
- Prediction Time: Prediction time (in seconds) indicates the time taken by each model to make predictions on the

test dataset, showcasing their real-time decision-making speed.

- Build Peak Memory Usage: The peak memory consumption (in bytes) during model training highlights the maximum memory requirement during the training process.
- Prediction Peak Memory Usage: This metric reveals the highest memory usage (in bytes) during prediction, providing insights into memory demands during operational use.

The variety of evaluation metrics chosen guarantees a thorough examination of our machine learning models. We acquire a full grasp of each model's capabilities across various circumstances by taking into account predictive performance and resource utilization.

Here the methodology steps described in our paper considered common practices in the field of machine learning. However, we believe it is essential to highlight these steps as they serve as a foundation for our research and ensure the reliability and validity of our results. Data pre-processing, including data cleaning, is indeed a mandatory step, especially when dealing with datasets containing a large number of features. By explicitly mentioning these steps, we aim to provide clarity and transparency in our methodology, making our work more reproducible and enabling other researchers to build upon our findings. Moreover, the data set we have used, as far as we know, feature selection has not been done using information gain on these data sets, and we have tried to validate the results with some simple algorithms.

IV. Experiments and Results

A. Dataset Details

In this study, we utilized two distinct datasets: the first dataset was extracted from GitHub [31], containing 138,047 samples with 53 attributes. Among these samples, 70% represent malware instances, while the remaining 30% constitute legitimate observations. The attributes encompass a variety of features, including the file's hash, usage frequency, priority, and memory usage. All feature values are derived numerically from the PE (Portable Executable) headers of executable files.

The second dataset, obtained from Kaggle [32], consists of 100,000 samples with 32 attributes. Here, 50% of the samples pertain to malware, and the other 50% correspond to benign observations. These attributes capture behavioral features extracted from software. The dataset's attributes include file sizes, file hashes, and memory usage, with all feature values presented in numerical format.

B. Data Cleaning

For data cleaning, we have 2 parts, *remove duplicate data* and *remove zero or near zero variance predictor*. After deleting duplicates from dataset 1, our dataset contains 104,589 observations, 60% of them being malware and 40% being legal observations. We use the metric *Percentage of Unique Values* to remove zero or near zero variance predictors. If there are less than 10% unique values in the samples, the predictor is said to have near zero variance. In some instances, the threshold is set very close to zero to select the predictor of near-zero variance [25]. For that, in this case, we removed extremely near-zero variance predictors by using a near-zero criterion. For datasets 1 and 2 we established a threshold of 0.05% for features with a very low unique value. Columns with a proportion of less than 0.05% are deleted. After deleting 21 features, there are 32 features left to examine. In the same way, after deleting duplicates from dataset 2, our dataset contains 34,382 observations, with 40% of them being malware and the remaining 60% benign observations and removing zero or near-zero variance predictors. Columns with a proportion of less than 0.05% are deleted. After deleting 18 characteristics, there are 14 features left to examine.

C. Feature Selection

For feature selection, we use Information Gain (IG). In the context of malware classification, the focal point is the class label (benign or malicious). By using IG for feature selection, we can identify the most informative features that are relevant to the classification task and discard the less informative ones, which can improve the accuracy and efficiency of the classification algorithm. The IG score is calculated as the entropy reduction that results from the splitting of the data based on a given feature. Entropy measures the randomness or uncertainty of the data, and reducing the entropy means increasing the homogeneity or purity of the data. Therefore, the features that result in a high reduction in entropy are considered more informative for the classification task.

1) Apply Information Gain on Dataset 1: Upon completing data cleaning for dataset 1, we obtained a total of 32 features for feature selection. We utilized Information Gain (IG) to determine the importance of each feature for classification. The computed IG values ranged from 0.0357 to 0.5681, and we selected a threshold value of 0.5 for feature selection, as it is frequently used in the literature [33]. A higher IG value indicates that the feature is less entropic and more informative [34], [35]. To identify the most significant features, we experimented with different thresholds ranging from 0.1 to 0.5 and found that the accuracy of the model is consistent with all the characteristics of the fitted model up to a threshold value of 0.5. However, features with IG values less than 0.05 or greater than 0.5 resulted in a dramatic alteration in accuracy and were therefore excluded from our final selection of features, resulting in a final set of 9 features. These features were identified as the most important features (MIFs) for our analysis. Table I presents the final selected features along with their corresponding IG values for dataset 1.

2) Apply Information Gain on Dataset 2: After data cleaning from dataset 2, we have 14 features for feature selection. We use Information Gain for knowing a feature's importance for classification. We get IG values for all features from 0.0663 to 0.6623. For this dataset, the model accuracy is closer to that of an all-feature-fitted model up to a threshold of 0.1. From these IG values, we took the threshold value of 0.1. Features with a value greater than 0.1 have been taken. And we found

TABLE I
THE FINAL SELECTED FEATURES FROM DATASET 1 AFTER APPLYING THE INFORMATION GAIN METHOD, WITH CORRESPONDING INFORMATION GAIN (IG) VALUES.

Feature Name	IG Value
ResourcesMaxEntropy	0.5681
ResourcesMinEntropy	0.5513
ResourcesMeanSize	0.5218
ResourcesMaxSize	0.5210
SectionsMaxEntropy	0.5162
Characteristics	0.5089
SectionsMinEntropy	0.5038
SectionMaxVirtualsize	0.5020
AddressOfEntryPoint	0.5015

our final best feature set where 13 features exist. Table II demonstrates the final selected features with the Information Gain value of dataset 2.

TABLE II
THE FINAL SELECTED FEATURES FROM DATASET 2 AFTER APPLYING THE INFORMATION GAIN METHOD, WITH CORRESPONDING INFORMATION GAIN (IG) VALUES.

Feature Name	IG Value
static_prio	0.6623
nvcsw	0.6168
prio	0.6069
utime	0.5807
vm_truncate_count	0.5582
nivcsw	0.2569
map_count	0.2305
free_area_cache	0.2073
mm_users	0.2067
reserved_vm	0.1797
last_interval	0.1451
exec_vm	0.1274
total_vm	0.1250

D. Experimental Setting

We conducted our experiments to assess the performance of our model in comparison with traditional machine learning techniques, including SVM, LR, K-NN, DT, NB, and RF classifiers. This section outlines the hardware and software configurations used for the experiments, as well as the specific methodology employed.

1) Hardware Configuration: The experiments were conducted on a system with the following specifications:

- Processor: Intel Core i5-6200U (2.30 GHz, 2 physical cores and 4 logical processors)
- Memory: 8 GB DDR4 RAM
- Graphics Card: Intel HD Graphics 620
- Storage: 500 GB HDD

2) Software Configuration: The following software tools and libraries were employed for the experiments:

- Operating System: Windows 10 Home (64-bit)
- Python Version: 3.8.0
- Scikit-Learn Version: 0.23.2

3) Experiment Methodology: Both datasets were randomly divided into training and test sets while maintaining the class distribution between benign and malicious samples. Each machine learning model was trained on the training data and evaluated on the test data. To ensure the reliability of our results, we employed 10-fold cross-validation for each model.

Python's scikit-learn tools were utilized to develop the algorithms, and the hyperparameter settings were carefully tuned for each model.

This comprehensive experimental setup provides insight into the hardware and software context of the experiments, as well as the specific performance metrics used to evaluate the models. These details contribute to the transparency and reproducibility of the research findings.

E. Results

To distinguish between malware and legitimate samples, we used SVM, LR, K-NN, DT, NB, and RF classifiers. The outcomes of the models in terms of accuracy, precision, recall, and F1 score are calculated from confusion matrices. Here, we provide the accuracy, precision, recall, F1 score, memory use, and processing time for each dataset with and without feature selection. To demonstrate the required time and memory, we choose the model build and prediction time. We also choose the highest value of memory during model training and testing. In terms of accuracy, precision, recall, F1 score, required time and memory, Table III shows the outcomes of the models on dataset 1 without feature selection and Table IV shows the outcomes of the models on dataset 1 after feature selection.

For dataset 1, the accuracy before and during feature selection is extremely close in this case. We observed that the accuracy obtained using 53 features is roughly equivalent to the accuracy achieved using 9 features. Here, the RF classifier surpasses other models in terms of precision, accuracy, and F1 score. Although the NB classifier has the best recall, it performs poorly on other performance criteria for before and after feature selection. When compared to RF, DT and K-NN classifiers perform admirably. In comparison to RF, DT, and K-NN, SVM is less accurate. However, when compared to other approaches, LR fails to reach a satisfying F1 score and recall score, even though the accuracy score is fail when compared to DT, RF, and K-NN classifiers.

Tables III and IV further demonstrate that training and testing without feature selection consumes more time and memory compared to the setup involving feature selection. The absence of feature selection leads to increased time and memory requirements. We also observe that NB requires the least amount of training time both before and after feature selection. In terms of prediction, both RF prior to feature selection and LR subsequent to feature selection require less time. In addition, LR requires the least amount of memory during training before feature selection, whereas KNN requires the least memory following feature selection. Additionally, LR consumes less memory for prediction both before and after feature selection. The cost of a model increases with the requirement of more memory and time. Therefore, it can be argued that the cost of the model decreases following feature selection.

Once more, we have demonstrated model performance and results for dataset 2 based on time and memory both before and after feature selection. Table V shows the outcomes, required

TABLE III

FOR DATASET 1, THE EXPERIMENTAL RESULTS OF SEVERAL CLASSIFIERS WERE ANALYZED WITHOUT FEATURE SELECTION.

Classifier	Accuracy	Precision	Recall	F1 Score	Build Time	Prediction Time	Build Peak Memory	Prediction Peak Memory
SVM	0.98 ± 0.01	0.98 ± 0.03	0.98 ± 0.01	0.98 ± 0.01	57.750 ± 1.50	9.140 ± 0.50	$33891300 \pm 1 - 2\%$	$631800 \pm 1 - 2\%$
LR	0.97 ± 0.01	0.97 ± 0.02	0.96 ± 0.01	0.96 ± 0.01	3.150 ± 0.80	0.170 ± 0.01	$\mathbf{3890600 \pm 1\text{-}2}\%$	$\mathbf{591800 \pm 1\text{-}2}\%$
KNN	0.98 ± 0.01	0.97 ± 0.01	0.98 ± 0.02	0.98 ± 0.02	33.130 ± 1.00	70.350 ± 2.50	$4562000 \pm 1 - 2\%$	$4190400 \pm 1 - 2\%$
DT	0.98 ± 0.02	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	1.900 ± 0.40	0.070 ± 0.02	$20546000 \pm 1 - 2\%$	$6381900 \pm 1 - 2\%$
NB	0.45 ± 0.02	0.42 ± 0.03	$\mathbf{0.99 \pm 0.01}$	0.59 ± 0.04	$\mathbf{0.145 \pm 0.05}$	0.078 ± 0.05	$40346000 \pm 1 - 2\%$	$22450100 \pm 1 - 2\%$
RF	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	2.330 ± 0.80	$\mathbf{0.032 \pm 0.02}$	$22769000 \pm 1 - 2\%$	$6664000 \pm 1 - 2\%$

TABLE IV

FOR DATASET 1, THE EXPERIMENTAL RESULTS OF SEVERAL CLASSIFIERS WERE ANALYZED AFTER FEATURE SELECTION.

Classifier	Accuracy	Precision	Recall	F1 Score	Build Time	Prediction Time	Build Peak Memory	Prediction Peak Memory
SVM	0.96 ± 0.01	0.95 ± 0.03	0.96 ± 0.01	0.95 ± 0.01	36.790 ± 1.20	4.880 ± 0.300	$6278900 \pm 1 - 2\%$	$629400 \pm 1 - 2\%$
LR	0.89 ± 0.02	0.88 ± 0.02	0.85 ± 0.03	0.86 ± 0.05	0.203 ± 0.10	$\mathbf{0.002 \pm 0.001}$	$3871600 \pm 1 - 2\%$	$\mathbf{592000 \pm 1\text{-}2}\%$
KNN	0.97 ± 0.01	0.96 ± 0.01	0.97 ± 0.04	0.97 ± 0.02	1.010 ± 0.50	5.305 ± 0.800	$\mathbf{3285700 \pm 1\text{-}2}\%$	$4189700 \pm 1 - 2\%$
DT	$\mathbf{0.99 \pm 0.01}$	0.97 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	0.630 ± 0.10	0.006 ± 0.001	$6740800 \pm 1 - 2\%$	$1779400 \pm 1 - 2\%$
NB	0.44 ± 0.03	0.41 ± 0.05	$\mathbf{0.99 \pm 0.01}$	0.58 ± 0.06	$\mathbf{0.047 \pm 0.01}$	0.005 ± 0.001	$6908700 \pm 1 - 2\%$	$4041900 \pm 1 - 2\%$
RF	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	0.98 ± 0.02	$\mathbf{0.98 \pm 0.01}$	1.670 ± 0.50	0.031 ± 0.010	$8962900 \pm 1 - 2\%$	$2064500 \pm 1 - 2\%$

TABLE V

FOR DATASET 2, THE EXPERIMENTAL RESULTS OF SEVERAL CLASSIFIERS WERE ANALYZED WITHOUT FEATURE SELECTION.

Classifier	Accuracy	Precision	Recall	F1 Score	Build Time	Prediction Time	Build Peak Memory	Prediction Peak Memory
SVM	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	4.598 ± 1.00	1.2800 ± 0.500	$8485600 \pm 1 - 2\%$	$2271900 \pm 1 - 2\%$
LR	0.94 ± 0.02	0.95 ± 0.02	0.95 ± 0.01	0.94 ± 0.01	0.312 ± 0.05	0.1560 ± 0.005	$1299700 \pm 1 - 2\%$	$\mathbf{240900 \pm 1\text{-}2}\%$
KNN	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	1.280 ± 0.40	3.5600 ± 0.700	$\mathbf{1127900 \pm 1\text{-}2}\%$	$1383300 \pm 1 - 2\%$
DT	0.98 ± 0.01	0.97 ± 0.01	$\mathbf{0.99 \pm 0.01}$	0.98 ± 0.02	0.515 ± 0.10	$\mathbf{0.0005 \pm 0.0001}$	$4984900 \pm 1 - 2\%$	$1376800 \pm 1 - 2\%$
NB	0.69 ± 0.03	0.90 ± 0.02	0.56 ± 0.04	0.65 ± 0.03	$\mathbf{0.156 \pm 0.05}$	0.0780 ± 0.050	$8123100 \pm 1 - 2\%$	$4540500 \pm 1 - 2\%$
RF	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	0.651 ± 0.10	0.0312 ± 0.010	$6210200 \pm 1 - 2\%$	$1524400 \pm 1 - 2\%$

TABLE VI

FOR DATASET 2, THE EXPERIMENTAL RESULTS OF SEVERAL CLASSIFIERS WERE ANALYZED AFTER FEATURE SELECTION.

Classifier	Accuracy	Precision	Recall	F1 Score	Build Time	Prediction Time	Build Peak Memory	Prediction Peak Memory
SVM	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	2.466 ± 0.50	0.390 ± 0.10	$2891700 \pm 1 - 2\%$	$\mathbf{208400 \pm 1\text{-}2}\%$
LR	0.91 ± 0.02	0.92 ± 0.02	0.94 ± 0.01	0.93 ± 0.01	0.187 ± 0.05	0.005 ± 0.001	$1292900 \pm 1 - 2\%$	$240500 \pm 1 - 2\%$
KNN	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	0.156 ± 0.05	0.890 ± 0.10	$\mathbf{1127000 \pm 1\text{-}2}\%$	$1382400 \pm 1 - 2\%$
DT	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.078 \pm 0.01}$	$\mathbf{0.0003 \pm 0.0001}$	$2678200 \pm 1 - 2\%$	$724200 \pm 1 - 2\%$
NB	0.56 ± 0.03	0.76 ± 0.02	0.42 ± 0.04	0.55 ± 0.03	0.150 ± 0.05	0.0155 ± 0.05	$3341900 \pm 1 - 2\%$	$1924600 \pm 1 - 2\%$
RF	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$	0.281 ± 0.05	0.0156 ± 0.01	$3425700 \pm 1 - 2\%$	$867900 \pm 1 - 2\%$

time and memory of the models on dataset 2 without feature selection.

Then, Table VI shows the outcomes, required time and memory of the models on dataset 2 after feature selection.

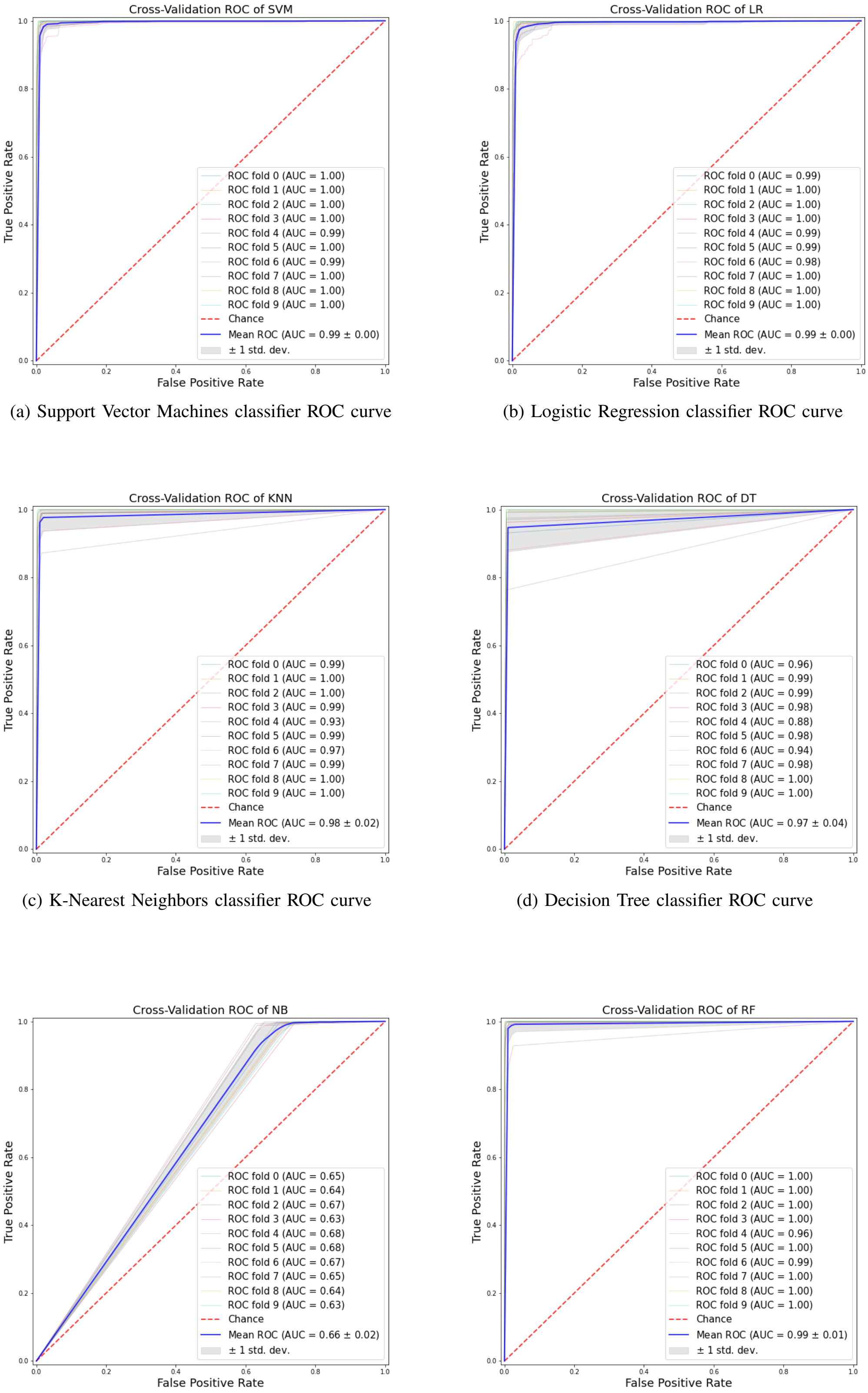
The performance of various machine learning models was evaluated on dataset 2 based on precision, accuracy, and F1 score, both before and after feature selection. The SVM, K-NN, DT, and RF classifiers were found to outperform other models. In contrast, the NB classifier failed to meet the performance requirements, and the LR classifier achieved a reasonable accuracy score but failed to attain satisfactory precision, F1 score, and recall score, when compared to the aforementioned classifiers. Additionally, the required time and memory for the models were compared before and after feature selection. It was observed that NB and DT required the least training time before and after feature selection, respectively. The DT classifier was found to be the fastest for prediction both before and after feature selection, while K-NN consumed the least amount of memory during training before and after feature selection. SVM was found to use less memory after feature selection, while LR used less memory for prediction before feature selection. The cost of the model significantly decreased after feature selection.

At last, we produced ROC curves as part of our cross validation using 10-fold cross validation before and after feature selection.

2a- 2f shows the ROC curve for dataset 1 for each of the classifiers before feature selection, which includes 10-fold curves and a mean curve. The RF, SVM, and LR all had the highest mean Area Under Curve (AUC) scores of 0.99, K-NN and DT also provide a good response, while the NB had the lowest (mean AUC: 0.66)

On the other hand, 3a- 3f shows the ROC curve for dataset 1 for each of the classifiers after feature selection, which includes 10-fold curves and a mean curve also. The RF, SVM, and K-NN all had the highest mean Area Under Curve (AUC) scores of 0.99, while the NB had the lowest (mean AUC: 0.85) From the ROC curves before and after feature selection of dataset 1, we can see that the classifiers are giving almost close results. That means even after we reduced the features, our model still responds like all the previous features are present. The ROC curve before features selection of dataset 2 for each of the classifiers is shown in 4a- 4f , which contains 10-fold curves and a mean curve. The RF had the greatest mean Area Under Curve (AUC) ratings of 0.95, followed by the SVM with 0.93 and the LR with 0.91, while the NB had the lowest (mean AUC: 0.84)

The ROC curve after features selection for dataset 2 for



(a) Support Vector Machines classifier ROC curve

(b) Logistic Regression classifier ROC curve

(c) K-Nearest Neighbors classifier ROC curve

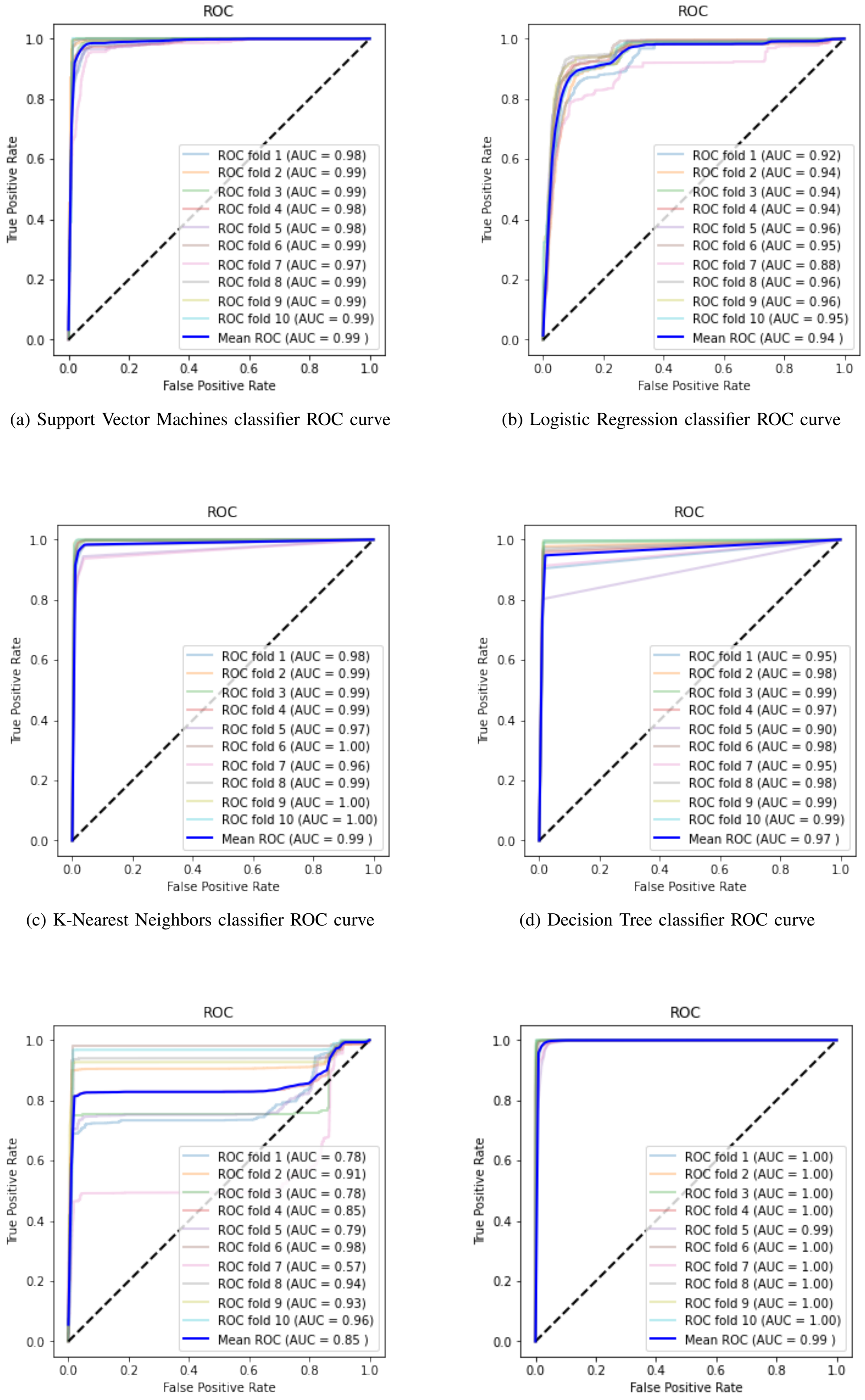
(d) Decision Tree classifier ROC curve

(e) Nave Bayes classifier ROC curve

(f) Random Forest classifier ROC curve

Fig. 2. Receiver Operating Characteristic (ROC) Curves for Various Classifiers (SVM, LR, K-NN, DT, NB, RF) on Dataset 1 Before Feature Selection. Here RF, SVM, and LR had the highest mean Area Under Curve (AUC) scores of 0.99, K-NN and DT also provided a good response, while the NB had the lowest (mean AUC: 0.66).

these classifiers, RF, SVM, and K-NN achieved the highest mean AUC scores, registering an impressive 0.99, while NB exhibited the lowest mean AUC of 0.85.



(a) Support Vector Machines classifier ROC curve

(b) Logistic Regression classifier ROC curve

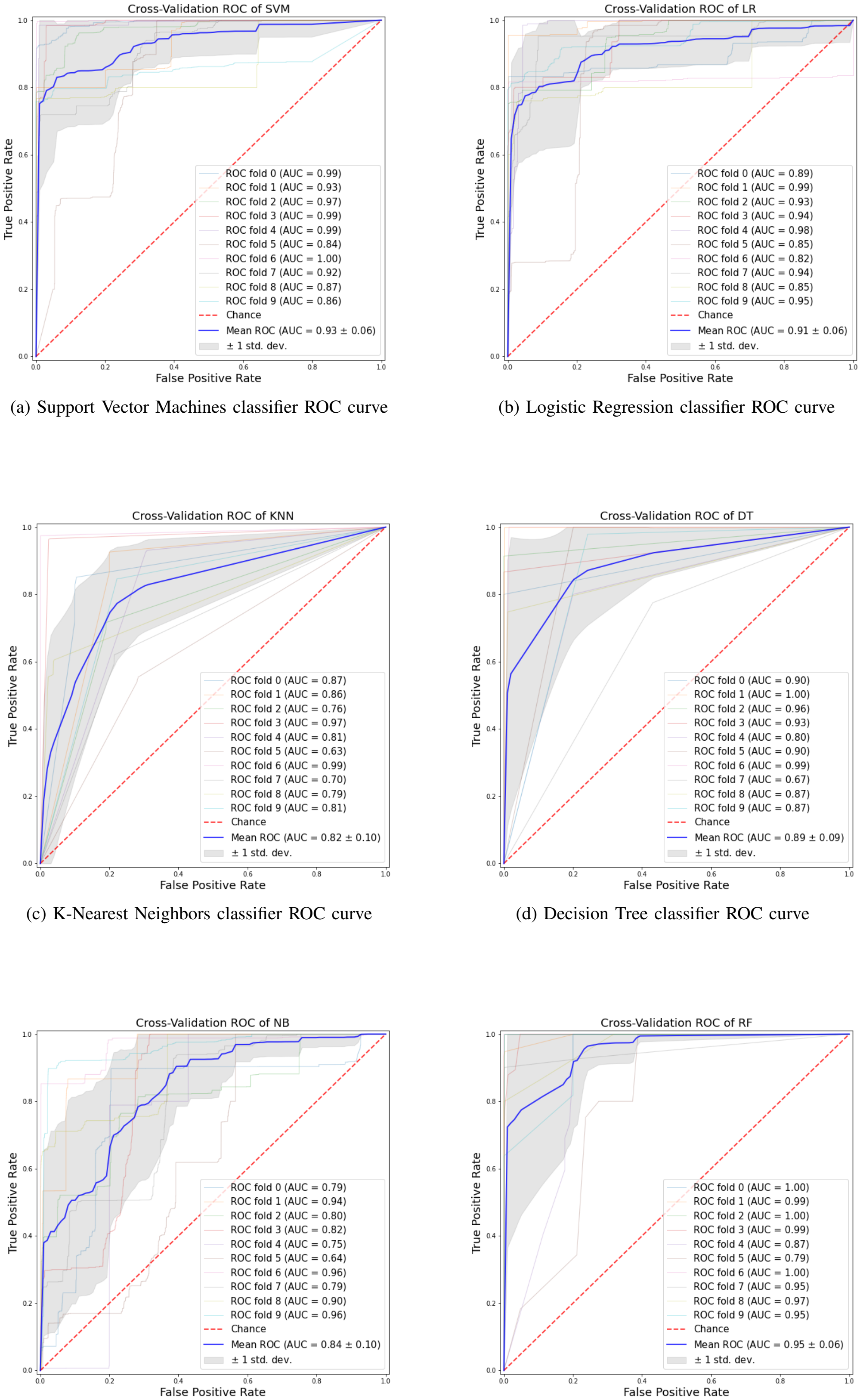
(c) K-Nearest Neighbors classifier ROC curve

(d) Decision Tree classifier ROC curve

(e) Nave Bayes classifier ROC curve

(f) Random Forest classifier ROC curve

Fig. 3. Receiver Operating Characteristic (ROC) Curves for Various Classifiers (SVM, LR, K-NN, DT, NB, RF) on Dataset 1 After Feature Selection. Among these classifiers, RF, SVM, and K-NN achieved the highest mean AUC scores, registering an impressive 0.99, while NB exhibited the lowest mean AUC of 0.85.



(a) Support Vector Machines classifier ROC curve

(b) Logistic Regression classifier ROC curve

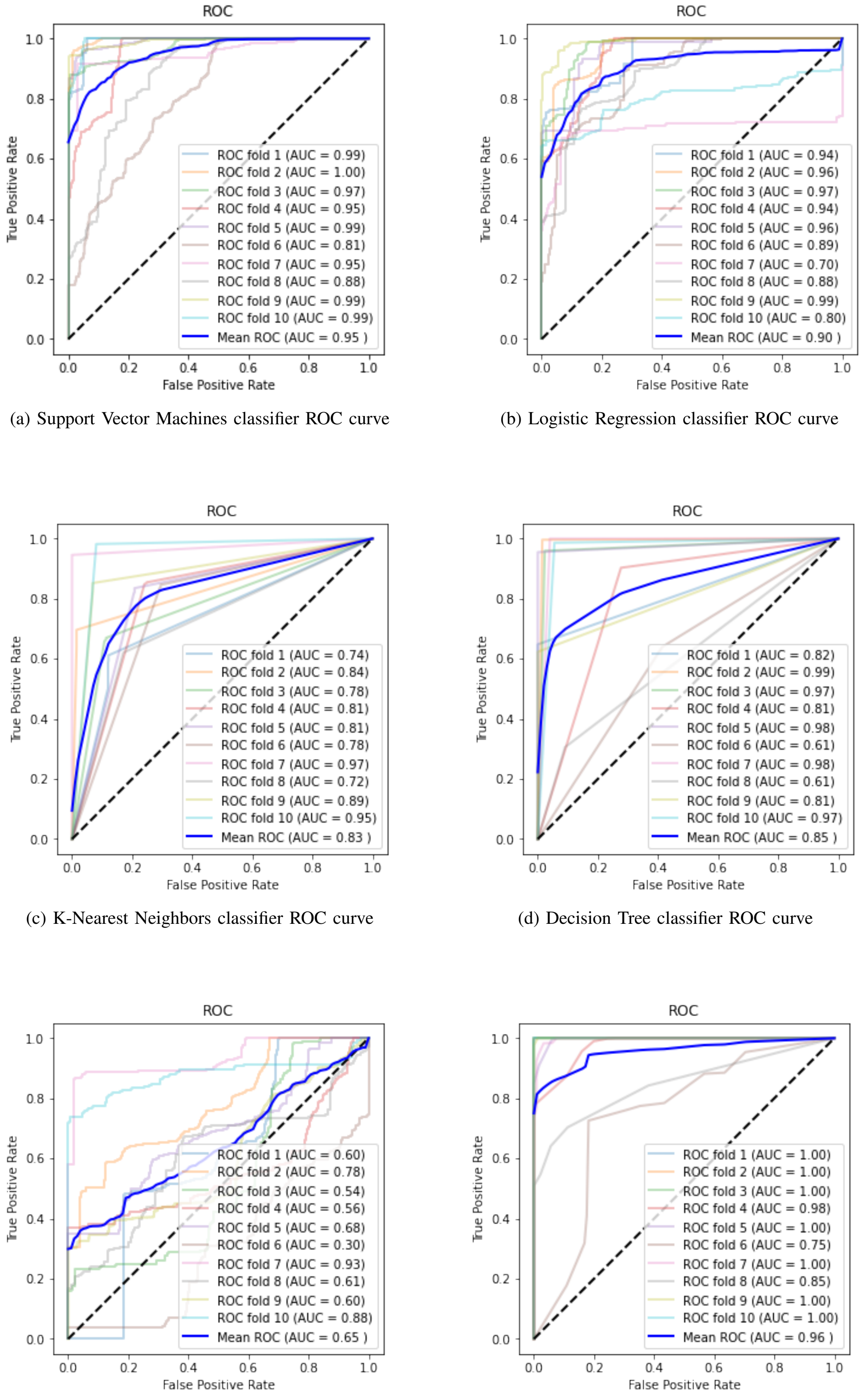
(c) K-Nearest Neighbors classifier ROC curve

(d) Decision Tree classifier ROC curve

(e) Nave Bayes classifier ROC curve

(f) Random Forest classifier ROC curve

Fig. 4. Receiver Operating Characteristic (ROC) Curves for Various Classifiers (SVM, LR, K-NN, DT, NB, RF) on Dataset 2 Before Feature Selection. RF secured the highest mean AUC score at 0.95, closely followed by SVM with 0.93 and LR with 0.91, whereas NB exhibited the lowest mean AUC of 0.84.



(a) Support Vector Machines classifier ROC curve

(b) Logistic Regression classifier ROC curve

(c) K-Nearest Neighbors classifier ROC curve

(d) Decision Tree classifier ROC curve

(e) Nave Bayes classifier ROC curve

(f) Random Forest classifier ROC curve

Fig. 5. Receiver Operating Characteristic (ROC) Curves for Various Classifiers (SVM, LR, K-NN, DT, NB, RF) on Dataset 2 After Feature Selection. RF achieved the highest mean Area Under Curve (AUC) score at 0.96, closely followed by SVM with 0.95 and LR with 0.90, whereas NB recorded the lowest mean AUC of 0.65.

each of the classifiers is shown in 5a- 5f , which contains 10-fold curves and a mean curve. The RF had the greatest mean Area Under Curve (AUC) ratings of 0.96, followed by the SVM with 0.95 and the LR with 0.90, while the NB had the lowest (mean AUC: 0.65)

In a manner akin to dataset 1, for dataset 2, ROC curves are presented both prior to and following the implementation of feature selection. Notably, the classifiers produce highly comparable outcomes. This observation implies that, despite the reduction in features, our model's performance maintains consistency with that achieved using the full feature set.

V. Discussion

When we analyze the findings presented above, we can observe that the use of feature selection techniques can greatly reduce the computational cost of our model without compromising its accuracy in detecting malware. This is a significant advantage as it allows for faster and more efficient processing of large datasets. However, it is important to note that our work has some limitations. Using existing datasets has its benefits as it provides a thoroughly tested approach. However having access, to malware files for feature extraction and dataset creation is crucial. This autonomy allows us to tailor the datasets to perfectly align with our research needs potentially leading to accuracy, in our models. Moreover, while removing features, we had to use different threshold values across various datasets since we were unable to apply them uniformly. This can introduce some inconsistencies in the feature selection process and affect the performance of the model. Another important factor to consider is that the dataset already showed a level of accuracy, in all aspects, which limits how much we can improve it further. However, by maintaining the accuracy and lowering the computational cost, we have still achieved a practical improvement in the overall efficiency of the model. In terms of the implications of the study, our findings have practical applications in the field of cybersecurity. The ability to detect malware quickly and accurately is crucial in protecting computer systems and networks from potential threats. By reducing the computational cost of malware detection, we can improve the efficiency and effectiveness of cybersecurity measures. Our study also highlights the importance of feature selection in machine learning models and the potential benefits it can provide. In summary, our research imparts valuable perspectives on the significance of feature selection in the realm of malware detection. Moreover, it illuminates possible paths for future investigations aimed at refining the precision and effectiveness of these models.

VI. Conclusions and Future Work

Financial institutions, corporations, and individuals are all becoming increasingly vulnerable to malware. It's critical to have an automated system that can successfully identify and detect malware while also reducing the danger of unwanted activity. We proposed a feature selection-based model for successful malware detection in this research and we used several machine learning classifier techniques. There is a variation in the best feature set for each classification method. The results show that the malware detection tools cannot focus on any of the single aspects independently. We ran all of the trials on two malware datasets and used a rigorous comparative analysis to compare the performance of SVM, LR, K-NN, DT, NB, and RF classifiers. The RF and DT classifiers outperformed other classifiers in the experiments, attaining the best accuracy, F1, and precision scores with good consistency. Additionally, feature selection lowers the cost of the computation, which saves us a ton of time and memory.

In future work, it is possible to experiment with more features and find the best features for classifications. Also, we can extract these selected features from program file directly. And we can implement a whole pipeline for malware classification. Additionally, algorithmic improvements can be achieved, by suggesting innovations in this field that combine deep learning with ensemble learning. Different types malware analysis has different types of datasets, we can apply our method on those datasets.

Acknowledgement(s)

The Department of Computer Science and Telecommunication Engineering at Noakhali Science and Technology University, Bangladesh has helped us with our project by allowing us to use the laboratory.

References

[1] P. Jain, I. Rajvaidya, K. K. Sah, and J. Kannan, "Machine learning techniques for malware detection-a research review," in *2022 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE, 2022, pp. 1–6.

[2] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222 310–222 354, 2020.

[3] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.

[4] A. Ocaka, D. Ó. Briain, S. Davy, and K. Barrett, "Cybersecurity threats, vulnerabilities, mitigation measures in industrial control and automation systems: A technical review," in *2022 Cyber Research Conference-Ireland (Cyber-RCI)*. IEEE, 2022, pp. 1–8.

[5] F. Noorbehbahani, F. Rasouli, and M. Saberi, "Analysis of machine learning techniques for ransomware detection," in *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*. IEEE, 2019, pp. 128–133.

[6] C. Kateryna, "Machine learning methods for malware detection and classification," *University of Applied Sciences*, pp. 1–93, 2017.

[7] A.-A. M. Majid, A. J. Alshaibi, E. Kostyuchenko, and A. Shelupanov, "A review of artificial intelligence based malware detection using deep learning," *Materials Today: Proceedings*, vol. 80, pp. 2678–2683, 2023.

[8] L. Liu, B.-s. Wang, B. Yu, and Q.-x. Zhong, "Automatic malware classification and new malware detection using machine learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 9, pp. 1336–1347, 2017.

[9] S. K. Sahay, A. Sharma, and H. Rathore, "Evolution of malware and its detection techniques," in *Information and Communication Technology for Sustainable Development: Proceedings of ICT4SD 2018*. Springer, 2020, pp. 139–150.

[10] A. Mishra and A. Almomani, "Malware detection techniques: A comprehensive study," *Insights: An International Interdisciplinary Journal*, vol. 1, no. 1, pp. 1–5, 2023.

[11] D. W. Fernando, N. Komninos, and T. Chen, "A study on the evolution of ransomware detection using machine learning and deep learning techniques," *IoT*, vol. 1, no. 2, pp. 551–604, 2020.

[12] M. Memon, A. A. Unar, S. S. Ahmed, G. H. Daudpoto, and R. Jaffari, "Feature-based semi-supervised learning approach to android malware detection," *Engineering Proceedings*, vol. 32, no. 1, p. 6, 2023.

[13] M. E. Z. N. Kambar, A. Esmaeilzadeh, Y. Kim, and K. Taghva, "A survey on mobile malware detection methods using machine learning," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022, pp. 0215–0221.

[14] A. M. Abiola and M. F. Marhusin, "Signature-based malware detection using sequences of n-grams," *International Journal of Engineering and Technology (UAE)*, 2018.

[15] Y. Pan, X. Ge, C. Fang, and Y. Fan, "A systematic literature review of android malware detection using static analysis," *IEEE Access*, vol. 8, pp. 116 363–116 379, 2020.

[16] G. Kirubavathi, S. Sreevarsan, and P. VARADHAN, "Behavioural based detection of android ransomware using machine learning techniques," 2023.

[17] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," *arXiv preprint arXiv:1609.03020*, 2016.

[18] S. Maniath, A. Ashok, P. Poornachandran, V. Sujadevi, P. S. AU, and S. Jan, "Deep learning lstm based ransomware detection," in *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)*. IEEE, 2017, pp. 442–446.

[19] M. Masum, H. Shahriar, H. Haddad, M. J. H. Faruk, M. Valero, M. A. Khan, M. A. Rahman, M. I. Adnan, A. Cuzzocrea, and F. Wu, "Bayesian hyperparameter optimization for deep neural network-based network intrusion detection," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 5413–5419.

[20] H. Ghanei, F. Manavi, and A. Hamzeh, "A novel method for malware detection based on hardware events using deep neural networks," *Journal of Computer Virology and Hacking Techniques*, vol. 17, no. 4, pp. 319–331, 2021.

[21] M. Masum, M. J. H. Faruk, H. Shahriar, K. Qian, D. Lo, and M. I. Adnan, "Ransomware classification and detection with machine learning algorithms," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022, pp. 0316–0322.

[22] M. S. Hossain and M. H. Riaz, "Android malware detection system: a machine learning and deep learning based multilayered approach," in *Intelligent Computing & Optimization: Proceedings of the 4th International Conference on Intelligent Computing and Optimization 2021 (ICO2021) 3*. Springer, 2022, pp. 277–287.

[23] K. Bakour and H. M. Ünver, "Deepvisdroid: android malware detection by hybridizing image-based features with deep learning techniques," *Neural Computing and Applications*, vol. 33, pp. 11 499–11 516, 2021.

[24] K. Bakour, H. M. Ünver, and R. Ghanem, "A deep camouflage: evaluating android's anti-malware systems robustness against hybridization of obfuscation techniques with injection attacks," *Arabian Journal for Science and Engineering*, vol. 44, pp. 9333–9347, 2019.

[25] M. Kuhn, K. Johnson *et al.*, *Applied predictive modeling*. Springer, 2013, vol. 26.

[26] X. Zhu, Z. Feng, J. Wu, and W. Deng, "A novel feature selection based on vmd and information gain for pipe blockages," *Applied Sciences*, vol. 11, no. 22, p. 10824, 2021.

[27] A. Bar-Or, A. Schuster, R. Wolff, and D. Keren, "Decision tree induction in high dimensional, hierarchically distributed databases," in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 466–470.

[28] P. V. Agrawal and D. D. Kshirsagar, "Information gain-based feature selection method in malware detection for maldroid2020," in *2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN)*. IEEE, 2022, pp. 1–5.

[29] W. Shu, Z. Yan, J. Yu, and W. Qian, "Information gain-based semi-supervised feature selection for hybrid data," *Applied Intelligence*, vol. 53, no. 6, pp. 7310–7325, 2023.

[30] S. Rijal, P. A. Cakranegara, E. M. S. Ciptaningsih, P. H. Pebriana, A. Andiyan, and R. Rahim, "Integrating information gain methods for feature selection in distance education sentiment analysis during covid-19." *TEM Journal*, vol. 12, no. 1, 2023.

[31] M. Mathur, "Ransomware detection," https://github.com/muditmathur2020/RansomwareDetection, 2021.

[32] N. Saravana, "Malware detection," https://www.kaggle.com/datasets/nsaravana/malware-detection, 2018.

[33] M. I. Prasetiyowati, N. U. Maulidevi, and K. Surendro, "Determining threshold value on information gain feature selection to increase speed and prediction accuracy of random forest," *Journal of Big Data*, vol. 8, no. 1, p. 84, 2021.

[34] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine learning research*, vol. 7, pp. 1–30, 2006.

[35] Z. Yang, Q. Ye, Q. Chen, X. Ma, L. Fu, G. Yang, H. Yan, and F. Liu, "Robust discriminant feature selection via joint l2, 1-norm distance minimization and maximization," *Knowledge-Based Systems*, vol. 207, p. 106090, 2020.

Md Ashikur Rahman received his B.Sc. in Engineering degree from the Department of Computer Science and Telecommunication Engineering at Noakhali Science and Technology University, Bangladesh. Currently, he is enrolled in the M.Sc. Engineering program within the same department. His academic pursuits center around machine learning and deep learning, representing his keen research interests. Contact him at mdashikur567@gmail.com.

Syful Islam is working as a Lecturer in the department of Computer Science and Engineering at Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Gopalganj. He also worked as an assistant professor in the department of Computer Science and Telecommunication Engineering at Noakhali Science and Technology University, Bangladesh. He received the M.E. and Ph.D. degree from Nara Institute of Science and Technology, Japan. His research interests include software ecosystem, mining Stack Overflow, mining software repositories etc. Contact him at syfulcse@bsmrstu.edu.bd, syfulcste@gmail.com

Yusuf Sulistyo Nugroho is a lecturer at the Department of Informatics, Universitas Muhammadiyah Surakarta, Indonesia. He received his Ph.D degree from Nara Institute of Science and Technology, Japan in 2020. His research interests include Empirical Software Engineering, Software Documentation, and Mining Software Repositories. Contact him at yusuf.nugroho@ums.ac.id.

Fatah Yasin Al Irsyadi is a senior lecturer at the Department of Informatics, Universitas Muhammadiyah Surakarta, Indonesia. He received his master degree from Universitas Gadjah Mada, Indonesia in 2005. His research interests include Game Development and Network Security. Contact him at fatah.yasin@ums.ac.id.

Md Javed Hossain holds B.Sc. (Honors) and M.Sc. degrees from Dhaka University in Applied Physics, Electronics, and Communication Engineering. He also earned a Diploma in 'Semiconductor Technology' from Osaka University, Japan. He further pursued an MS degree from Wonkwang University, South Korea, specializing in Electrical, Electronic, and Information Engineering. With a teaching career spanning 6 years at Atish Dipankar University of Science and Technology and National University, he has rich experience in Computer Science Courses. His research areas encompass digital signal processing, fuzzy logics, wireless communication systems, and machine learning. Joining Noakhali Science and Technology University in 2006, he progressed from Lecturer to Associate Professor upto February 2023. He is now a Professor in the Department of Computer Science and Telecommunication Engineering. He has an impressive publication record, featuring numerous peer-reviewed articles and conference papers. Contact him at javed@nstu.edu.bd