I. J. Computer Network and Information Security, **2026, 1, 18-32**
Published Online on February 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2026.01.02

Malware Detection and Classification using Shapley Additive Explanations Values in Machine Learning

Balachandra Chikkoppa*
Department of Computer Science, KLE Institute of Technology, Gokul, Hubballi, 580028, Karnataka, India
Email: balachandra.c@kleit.ac.in
ORCID iD: https://orcid.org/0009-0003-9720-7015
*Corresponding author

Hanumanthappa J.
DOS in CS, University of Mysore, Manasgongothri, Mysuru, 10587, Karnataka, India
Email: hanumas@gmail.com
ORCID iD: https://orcid.org/0000-0002-6031-6993

Wai Yie Leong
INTI International University and Colleges, Nilai, Negeri Sembilan, Malaysia
Email: waiyie.leong@newinti.edu.my
ORCID iD: https://orcid.org/0000-0002-5389-1121

Received: 16 July 2025; Revised: 18 September 2025; Accepted: 16 November 2025; Published: 08 February 2026

Abstract: It is essential and unavoidable to detect Malware on the Internet, as a wide range of online IT services are available. Portable Executable files are the most frequently targeted platform by Malware. Malware must be promptly identified and alerted in a real-world environment by establishing a deployable learning system. The researchers applied machine learning to a Malware dataset, observing the model's performance metrics at a high computational cost, but were unable to deploy the model in a real-world environment. A deployable machine learning model using RF, attaining an accuracy of 97.16%, precision of 95.21%, and F1 score of 95.24% is achieved in the proposed research work, which is particularly adept at accurately identifying Malware. We have developed a novel classification model that employs the Support Vector Machine (SVM) to classify preprocessed data, detecting malware and normal instances. Furthermore, the SHAP technique identifies significant features, including SizeOfStackReserve, DllCharacteristics, and MajorImageVersion. The use of SHAP values facilitates an understanding of the characteristics of each feature in the model's prediction. Employing the SHAP algorithm using the trained SVM model to reduce the features, attained an accuracy of 97.16%.

Index Terms: Machine Learning, Malware, Sandbox, PE File, SHAP.

1. Introduction

Hackers use portable executable files (PE files) or unreadable files to develop Malware that contains malicious code. The binary file contains a few traces of readable words, which we call strings, and it is difficult to trace the code, which is why Malware analysis is complicated. Portable binary files are one of the most common file formats for executables on the Windows platform. Because the number of people using the Microsoft operating system is huge, they usually see more Malware written for the Windows platform. The Figure 1 graph reveals that almost 50% of the Malware is written in PDF format, followed by document types like Word, PPT, etc. [1].

Considering all these factors, it is worthwhile to invest our time in Malware analysis using the PE file structure. PE stands for portable executable, and all .exe and .dll files are PE files that contain the information required for the operating system to run the executables. These files provide clues about Malware and its interactions with the operating system during Malware analysis. However, a Malware analyst is not required to know the details of all the parts of the PE file.

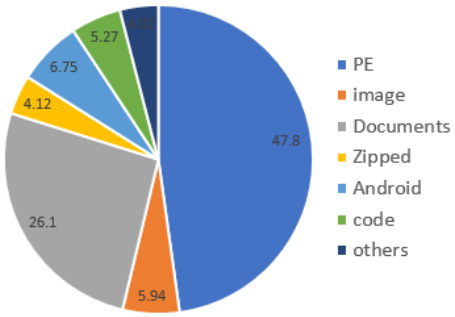

Fig.1. Distribution of Malware files

A typical PE file structure, as shown in Figure 2, comprises various parts [1-3]. In the PE file, the DOS header, or the MZ header, defines the executable binary as the first part of the PE file and contains a file signature with the magic number 5A4D in hexadecimal to represent the type of executable. Also, it includes a value known as the ELF executable link file, instructing the operating system on where to locate the PE header. The DOS stub component in the header is primarily designed for backward compatibility. In the executable, the PDF format signature uses the PE header to represent the PE file PE00 in ASCII and 504500 in hexadecimal. The PE header also holds information about machine types, like 32-bit or 64-bit Intel or AMD chipsets. It provides information about the number of sections and the size of optional headers in the file.

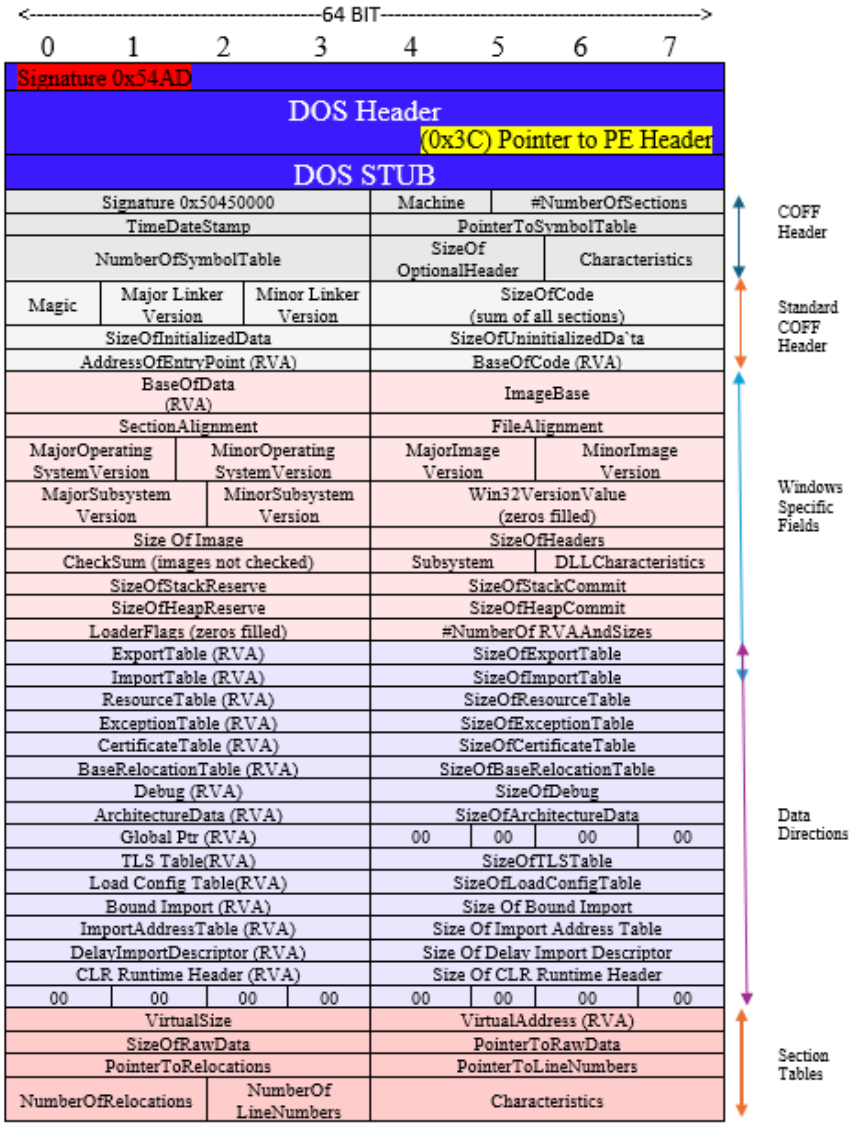

Fig.2. PE file format and its attributes

When it is loaded into memory, the section table contains information such as the virtual size, which is the total size of the section, and also the size of the raw data, which is the size of the initialized data on the disc, and characteristics, which are flags that describe the characteristics of a section, such as whether it is a readable or writable executable, among others. Lastly, the PE file format presents the section portion. This part comprises multiple sections, each tailored to the specific goals of the application. For example, the text section contains the executable code for the application, while the block starting symbol (bss) section includes the application's uninitialized data. During Malware analysis, the portable executable, which employs the simplified B structure of standard PE information, provides insights into Malware's operation and interactions with the operating system. However, a Malware analyst does not need to be intimately familiar with all of the pieces of the PE file; simply selecting specific components and a few attributes for each section allows us to detect Malware.

The detection of Malware in PE files was carried out using different machine learning (ML) models, including Logistic Regression (LR), k-nearest-neighbour (KNN), Decision Trees (DT), Support Vector Machine (SVM), Random Forest (RF), and Naive Bayes (NB). These models are selected based on their ability to handle features derived from PE files' imports, sections, and strings effectively. Several of these models encounter challenges, including higher computational costs, sensitivity to non-essential features, unrealistic assumptions about feature independence, reduced accuracy, complications with broad datasets, and limitations in classifying nonlinear features. Random Forest (RF) demonstrates effectiveness in pattern detection, although it suffers significant computational costs. Additionally, the characteristics of Shapley values offer descriptions of the contributing features for prediction, a capability frequently lacking in traditional machine learning models. The eight selected features suggest potential malicious entry points, outdated versions, bypassing of critical flags, and excessive memory usage, all of which may complicate the Malware detection process. Applying the Shapley values enables the quantification and analysis of each feature's contribution, thereby validating the rules that identify and monitor the model's behavior, ultimately enhancing the stability of its predictions. This technique is essential for Malware detection, as it enables security teams to prioritise the indicator of compromise (IoC) by reducing false positives or negatives. SOC analysts and Malware researchers can use Shapley values to triage alerts faster by refining detection models.

2. Dataset Generation

The file, when executed, is loaded into memory in some random location using unused space, as shown in Fig. 3. When the executable is loaded into memory, it will take a random location in the RAM, and the OS knows all the headers, sections, and options of the file to run. Capturing this information from the PE files helps to generate the Malware dataset. The size of a specific section tells the computer where to locate the next part of the file. We downloaded Malware and benign files from VirusTotal and Malware Bazaar for the Malware dataset. We conducted tests in a controlled environment by configuring a customised hypervisor for the sandbox and utilising various open-source tools, including Hashcalc, Hxd, Exeinfo PE, UPX, Strings/BinText, PE Studio, Process Monitor, RegShot, AutoRuns, ProcDot, and FakeNet. This sandboxing assisted us in observing the Malware's static and dynamic behaviours, which enabled us to classify it as either Malware or benign. Further, by creating two folders of Malware and benign files, the features are extracted using a PE parser from headers, sections, options, and subfields within the PE file. The PE parser collects attributes from the PE files through various headers, sections, and options that form the dataset of 56 features. Further applying feature scaling through normalization and standardization to multiple columns, the preprocessing is complete, and the dataset is ready for machine learning models that can be used for training and testing.

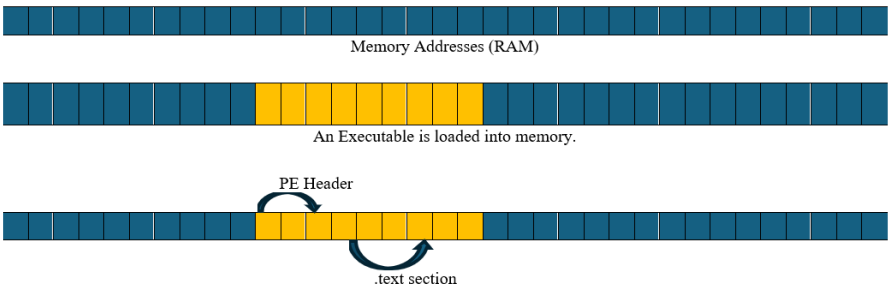

Fig.3. File execution and memory access

- Create two folders containing Malware and regular files collected from Virus Total and Malware Bazaar.
- We use the PE parser library to identify the required attribute or feature for data generation from various files.
- Apply feature scaling using normalization N_{new} and standardization S_{new}.

$$N_{new} = \frac{(\mathrm{N_{instance}} - \mathrm{N_{minimum}})}{(\mathrm{N_{maximum}} - \mathrm{N_{minimum}})} \quad (1)$$

$$S_{new} = \frac{(S_i - \text{Mean})}{\text{Standard Deviation}} \quad (2)$$

- Observe the primary contributing feature and its variations for Malware and normal files.
- Use the generated dataset for further training and testing the various models.

Sandbox using Hypervisor	Data Preparation	Dataset 56 feature
	Analyse binary PE file by parsing, using the various opensource tools and Machine Learning to extract the feature.	md5, Machine, SizeOfOptionalHeader, Characteristics, MajorLinkerVersion, MinorLinkerVersion, SizeOfCode, SizeOfInitializedData, SizeOfUninitializedData, AddressOfEntryPoint, BaseOfCode, BaseOfData, ImageBase, SectionAlignment, FileAlignment, MajorOperatingSystemVersion, MinorOperatingSystemVersion, MajorImageVersion, MinorImageVersion, MajorSubsystemVersion, MinorSubsystemVersion, SizeOfImage, SizeOfHeaders, CheckSum, Subsystem, DllCharacteristics, SizeOfStackReserve, SizeOfStackCommit, SizeOfHeapReserve, SizeOfHeapCommit, LoaderFlags, NumberOfRvaAndSizes, SectionsNb, SectionsMeanEntropy, SectionsMinEntropy, SectionsMaxEntropy, SectionsMeanRawsize, SectionsMinRawsize, SectionMaxRawsize, SectionsMeanVirtualsize, SectionsMinVirtualsize, SectionMaxVirtualsize, ImportsNbDLL, ImportsNb, ImportsNbOrdinal, ExportNb, ResourcesNb, ResourcesMeanEntropy, ResourcesMinEntropy, ResourcesMaxEntropy, ResourcesMeanSize, ResourcesMinSize, ResourcesMaxSize, LoadConfigurationSize, VersionInformationSize

Fig.4. PE files and possible malware dataset

3. Related Works

Machine learning techniques are utilized to complement the detection of Malware in a timely and accurate manner, as standard antivirus and heuristics databases [1] cannot identify the contemporary hostile advancements that attackers utilize through sophisticated tools. The system must possess learning capabilities to detect newly emerging Malware-induced dangers. Various published works propose machine learning-based approaches to combat dynamic and evolving Malware in a new threat landscape. The author of [2] presents the behaviour of Malware by employing dynamic analysis with samples of Windows PE files. Additionally, the author generates datasets that utilize Gini's feature scoring to identify contributing features based on API activities. Regarding ransomware and Trojans, the results were excellent, and they were successfully classified. The graph model framework built in [3] transforms the application software based on feature meta-paths for quick Android Malware applications. The model was able to correlate different application software from the neighbours in the Malware network applications. A recognition rate of 93% was achieved by the model when it came to identifying the unknown application as being associated with suspicious behaviours. The author used the program semantics described in [4] and developed a prototype known as Malsensor, which can do static analysis based on the signatures of portable Windows files. Additionally, the author utilized FCG-based code and string analysis to carry out an accurate function call graph. With the help of Malsensor, it was possible to detect Malware quickly with an accuracy rate of 98% by utilizing the image dataset collection. To identify and classify malicious software, the author of [5] utilized the metric learning technique on Windows PE files and analyzed Malware. Generic metric embeddings can perform essential Malware analysis tasks well, such as locating and classifying them into malicious file families and forecasting their types in two distinct methods, provided subject matter experts provide correct information about the files in question. Examples of these features include the distribution of errors and strings and the location of the API import in PE files. To find a solution to the issue of unbalanced classes, the author [6] implemented the OCC approach. The One-Class Support Vector Machine was implemented by the LibSVM package, which utilized pipelining characteristics extracted from PE file headers. Using the proposed approach, it was possible to differentiate between Malware files that underwent dynamic changes and those classified as having a class imbalance problem. The author uses visualization techniques [7] to transform executable files into colourful images. Following this, CNN is utilized to recognize deep features, ultimately reducing training costs. By applying the feature selection method to a one-class classifier, they could select the most essential features for detecting Malware. They identified 89% of malicious files by utilizing the most influential attributes, which resulted in an accuracy of 92%. In [8], the author uses several machine learning algorithms to analyze the signature-based dataset and identify malicious software. Deep Learning, AdaBoost, Random Forest, NB, and gradient boosting are some approaches the author utilises to achieve accuracy and precision for the dataset. Additionally, binary executable files are utilized. The random forest method is the most efficient one, with a success rate of 99% in terms of accuracy. The author of [9] scrutinizes the increasing trend of attackers employing obfuscation techniques to alter the Malware code embedded in PE files. They have generated DNA databases for static and dynamic Malware and achieved a high level of accuracy in identifying and classifying the Malware during this process. The author of the [10] article utilized the techniques of volatile memory forensics visualization to dump the photos by using the characteristics of PE files to list the Malware files. Through the utilization of optimized ensemble-based machine learning techniques known as hard voting, an accuracy of 95% and an accuracy of 96% was achieved. To classify the various forms of Malware, the author of [11] gathers multiple information from downloaded files and employs several different machine learning models. We determined the accuracy of these models to be 99% by comparing them based on factors like accuracy, efficiency, and F-1 scores. This strategy, which is detailed in [12], converts binary files into images and assigns a label to each image based

on the Malware state of the image. This strategy also allows attackers to develop Malware variants hidden in approaches dependent on visualization. Within the black box detector framework, the author created the GAN model, trained the machine learning model, and labelled the image through a substitute detector. PE files contain a variety of strings in their headers and sections, which provide information regarding Malware investigation. Natural language processing can recognize these strings. By employing this technique, the author of [13] could use the FFRI dataset and achieve an F1 score of 0.981, successfully identifying the newly discovered Malware. To extract sentences and compose the dataset based on GTP-2, the author of [14] utilized the text sections of Windows portable files and used low-level assembly language. The author achieved an F1 score of 98.3%. Integrating bidirectional LSTM methodologies from natural language processing analysis and word embedding has significantly improved model performance.
The literature highlights some of the challenges encountered in all prior research projects.

- In the literature, researchers directly applied machine learning models to a customized dataset, observing high-performance metrics. However, real-world deployments revealed false positives and high computational costs in Malware detection. A two-step solution is proposed in our work; the first step involves training and testing SVM, KNN, LR, NB, DT, and RF models that enable us to detect Malware in a real environment.
- We applied the SVM model to classify the data in our model and used the SHAP algorithm for feature importance analysis once we had trained the model. SHAP values help understand each feature's contribution to the model's prediction. The SHAP analysis identifies only a few features as the most important for classification.
- We utilized the most recent Malware files from the total virus database to analyze their advanced behaviour, which helped develop the detector and classifier into a more robust and efficient system.
- In the previous works, the Malware samples were limited in type; here, we considered samples of various Malware, such as adware, Trojans, backdoors, unknown, multidrop, bots, spam, and ransomware, to generate the dataset.

4. Methodology

Static analysis, which does not involve executing Malware, is a method for detecting it. This initial analysis consists of extracting useful information from the suspect binary file to make an informed decision and analyze and classify the Malware. We then focus our subsequent analysis efforts on observing various attributes with outliers or specific values, aiding the process. We do this to determine the intent of Malware. The dataset's various features enable the detection of most commodity Malware through their signatures.

Dynamic analysis involves running the file in a controlled environment, a sandbox, to observe its flow and comprehend the Malware's intentions.

- Is the device communicating with anyone on the Internet?
- Does it gather information from the system or network?
- Are the files in the systems being encrypted?
- Is there an attempt to halt any services?
- Is the system creating or deleting any files?
- Is it looking for anything specific, like a database or email server?

We observed eight of the above-selected 56 features playing prominent roles in observing the above-listed behaviours for dynamic analysis. After selecting these eight features, we applied various machine-learning models to detect and classify the Malware.

The flowchart in Figure 4 represents a pipeline for detecting Malware using ML techniques. Below is a breakdown of each step:

- Malware Dataset: The Malware features are taken from the RVA of the EP, which we get from the text section. These include the starting address of execution, the version attributes to see what kind of Windows machine it is, DLL characteristics, the reserve size of the stack, section details, and resource size. These will help us find the behaviours that aren't supposed to happen.
- Preprocessing: This step is crucial in preparing the data for machine learning, ensuring that no single feature overpowers the others. We used min-max normalization to scale all the values in the range [0, 1].

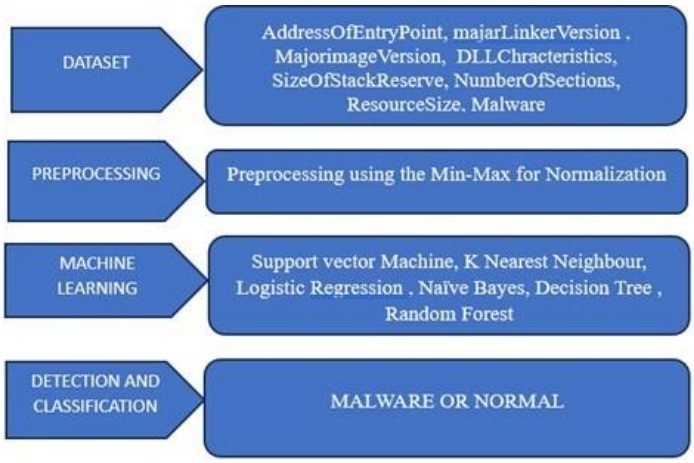

Fig.5. Machine learning models for malware

- Classification Algorithms: Various machine learning classifiers, such as SVM, KNN, LR, NB, DT, and RF, are then used to select the features for Malware detection and classification, determining whether the data represents Malware or normal behavior.

    a. The support vector machine defines the function of the hyperplane, which divides data points to maximize the margin between the hyperplane and the data points of various classes.

$$f(x) = Transpose(weight_{vector})\ x\ Input_{vector} + bias \tag{3}$$

    b. Naive Bayes (NB): Based on Bayes' Theorem, NB is a probabilistic classifier that presumes feature independence of the class label.

The Bayes' Theorem for classification:

$$P(C_k|x) = \frac{P(x|C_k)\,P(C_k)}{P(x)} \tag{4}$$

where
$P(C_k|x)$ is the posterior probability of class C_k given the data x,
$P(x|C_k)$ is the likelihood of data xxx given class C_k,
$P(C_k)$ is the prior probability of class $C_k$and
$P(x)$ is the evidence or total probability of the data x.

    c. Logistic Regression (LR): It's a supervised model that helps transform the linear combination of features into a probability using the sigmoid function.

The logistic (sigmoid) function is defined as:

$$P(\,y = 1 \mid X\,) = \frac{1}{1+e^{-(w^T x+b)}} \tag{5}$$

where:
w is the weight vector,
x is the feature vector,
b is the bias term,

    d. Decision Tree (DT): A non-parametric supervised learning approach for regression and classification is called a DT. It creates a tree structure by dividing the data into subsets according to the input feature values.

$$\text{Gini} = 1 - \sum_{i=1}^{c} p_i^{\,2} \tag{6}$$

where:
pi represents the probability of the class i,
c is the number of classes.

e. Random Forest (RF): It builds several decision trees and outputs the class that represents the mean forecast of each tree or the class that is the mode of the classes. Overfitting is avoided in part by the randomness of tree generation.

$$\hat{y} = \text{mode}\big(T_1(x), T_2(x), \dots, T_n(x)\big) \tag{7}$$

f. K-Nearest Neighbors (KNN): A data point is assigned a class according to the majority class of its k nearest neighbors. Euclidean distance is commonly used to measure the distance between data points.

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} \big(x_{ik} - x_{jk}\big)^2} \tag{8}$$

Performance Evaluation: We looked at six different classifiers (SVM, KNN, LR, NB, DT, and RF) and their performance in terms of F1 score, mean square error, sensitivity, specificity, accuracy, and recall. A summary of the results is in Table 2. Figures 7 to 13 indicate that RF and DT emerge as the top performers in this comparison, excelling in all key metrics, including accuracy, precision, recall, F1 score, and specificity. LR and NB lag in terms of recall and F1 score, indicating they may not be ideal for cases requiring a strong balance between precision and recall.

Table 1. Selected attributes malware dataset sample

Sl. No.	AddressOf EntryPoint	MajorLink erVersion	Majorim ageVersi on	MajorOperating SystemVersion	DllChara cteristics	SizeOfStack Reserve	NumberOfSe ctions	ResourceSi ze	legiti mate
0	10407	9	6	6	33088	262144	4	952	1
1	5354	9	6	6	33088	262144	4	952	1
2	58807	9	6	6	33088	262144	4	135490	1
3	25166	9	6	6	33088	262144	4	1940	1
4	70387	9	6	6	33088	262144	4	83098	1
-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-
137439	123291	11	0	5	33088	1048576	5	81654	0
137440	40000	2	6	1	32768	1048576	8	67624	0
137441	59610	10	0	5	33088	1048576	5	22648	0
137442	51216	2	0	1	0	1048576	8	2216	0
137443	22731	11	0	5	33088	1048576	5	318464	0
137444 rows x 9 columns									

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 137444 entries, 0 to 137443
Data columns (total 9 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   AddressOfEntryPoint           137444 non-null  int64
 1   MajorLinkerVersion            137444 non-null  int64
 2   MajorImageVersion             137444 non-null  int64
 3   MajorOperatingSystemVersion   137444 non-null  int64
 4   DllCharacteristics            137444 non-null  int64
 5   SizeOfStackReserve            137444 non-null  int64
 6   NumberOfSections              137444 non-null  int64
 7   ResourceSize                  137444 non-null  int64
 8   legitimate                    137444 non-null  int64
dtypes: int64(9)
memory usage: 9.4 MB
```

Fig.6. Details of malware dataset

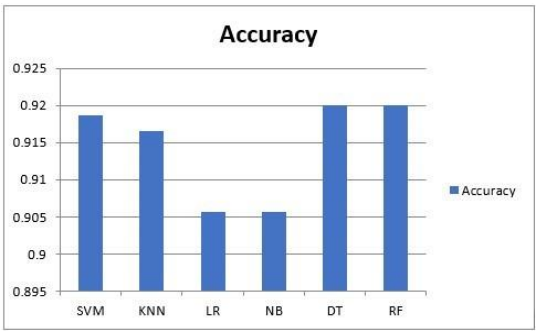

Fig.7. 8-attribute: accuracy

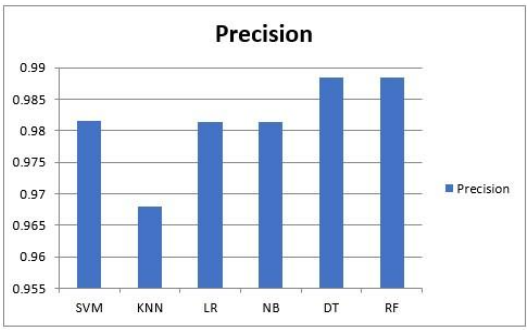

Fig.8. 8-attribute: precision

Figure 7 shows that DT and RF achieve the highest accuracy of 92%, meaning they perform correct classification compared to other classifiers. Similarly, LR and NB exhibit poor accuracy, leading to a higher likelihood of false classification by the models.

Figure 8 demonstrates that DT and RF achieve a height precision of 98.8%, meaning they can correctly identify true positives with fewer false positives.

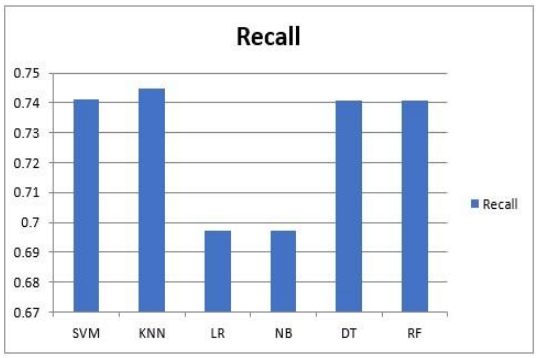

Fig.9. 8-attribute dataset: Recall

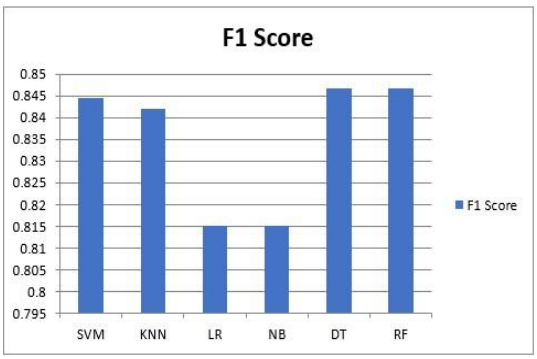

Fig.10. 8-attribute: F1 score

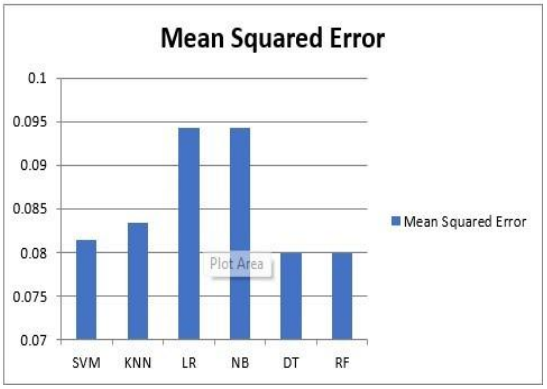

Fig.11. 8-attribute: Mean squared error

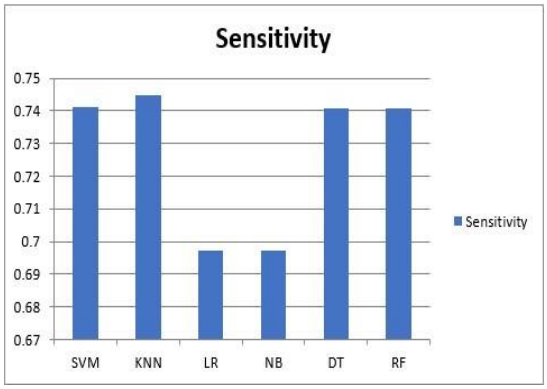

Fig.12. 8-attribute: Mean squared error

Figure 9 clearly shows that KNN achieves the highest recall at 74.4%. SVM, DT, and RF have similar recall values, approximately 74.1%. This model demonstrates its ability to identify many true positives accurately. On the other hand, LR and NB show the lowest recall value.

Figure 10 shows a good balance between precision and recall for DT and RF, achieving the highest F1 score of 84.6%. LR and NB have the lowest F1 score, indicating a trade-off between lower recall and higher precision value.

With a mean squared error of 0.0799, DT and RF have the lowest error rate, indicating fewer incorrect classifications made by these two classifiers. LR and NB have the highest mean squared value and a high error rate.

Figure 12 demonstrates that KNN achieves the highest sensitivity of 74.4%. SVM, DT, and RF have similar sensitivity values, approximately 74.1%. This model demonstrates its ability to accurately identify many true positives. On the other hand, LR and NB show the lowest value for sensitivity.

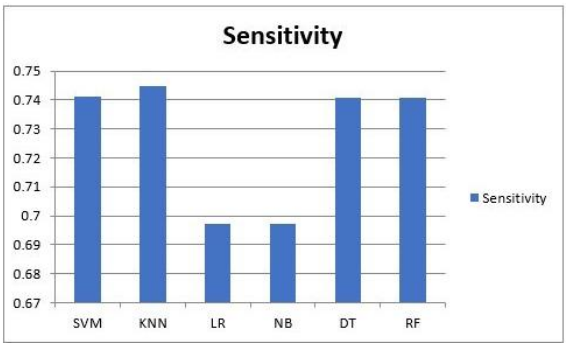

Fig.13. 8-attribute: Specificity

DT and RF have obtained a 99.6% specificity value, indicating that they can easily identify negative cases. KNN has the lowest specificity of 98.9%, indicating a slightly higher false positive rate than other models.

Table 2. Evaluation of the classifier using an 8-feature dataset

Classifier	SVM	KNN	LR	NB	DT	RF
Accuracy	0.9186	0.9166	0.9057	0.9057	0.9201	0.9201
Precision	0.9815	0.9679	0.9814	0.9814	0.9884	0.9884
Recall	0.741	0.7449	0.6971	0.6971	0.7408	0.7408
F1 Score	0.8445	0.8419	0.8151	0.8151	0.8468	0.8468
Mean Squared Error	0.0814	0.0834	0.0943	0.0943	0.0799	0.0799
Sensitivity	0.741	0.7449	0.6971	0.6971	0.7408	0.7408
Specificity	0.9941	0.9895	0.9944	0.9944	0.9963	0.9963

The model operates with high-performance metrics, identifying all Malware in files and classifying them as Normal or Malware. However, due to its extensive feature set, its deployment in the real environment can be challenging, necessitating further enhancements to improve performance. We propose a new classification model, Figure 5, that utilizes the support vector machine (SVM) model to classify data following preprocessing. SVM is a supervised learning technique that identifies the optimal hyperplane to separate data into distinct classes and performs well in classification problems. After training the model, we use the SHAP technique to assess the importance of each feature. The use of SHAP values simplifies the understanding of each feature's role in the model's prediction.

Proposed Methodology

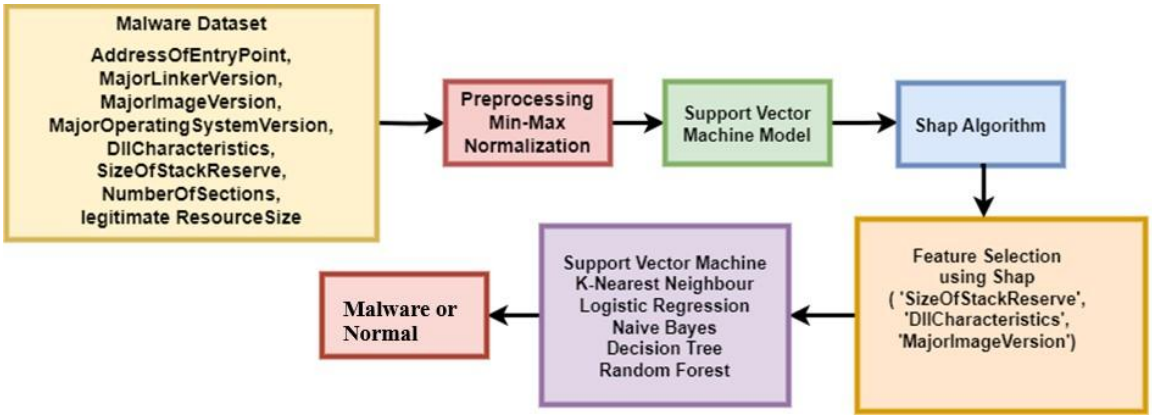

Fig.14. Proposed model for Malware detection and classification

Algorithm for Malware Detection:

Step 1: Data Collection and Preprocessing

- Load the Malware dataset with the provided features.
- Apply Min-Max normalization to the dataset to scale the features between 0 and 1.

Step 2: Train the model on SVM.

- Initialize and train an SVM model on the preprocessed data to classify Malware versus normal data.

Step 3: Feature Importance Analysis using SHAP

- Apply the SHAP algorithm to the trained SVM model to determine the feature importance scores. Identify the key features based on SHAP values.

Step 4: Feature Selection

- Use the important features identified by SHAP for further classification.

Step 5: Train Various Classifiers

- Using the selected features, train different classifiers (SVM, KNN, LR, NB, DT, RF).

Step 6: Classification

- Use the trained models to determine whether the data represents Malware or normal behaviour.

Step 7: Result Evaluation

- We evaluated the classifiers' performance using metrics like accuracy, recall, F1 score, precision, simplicity, and sensitivity. We compared the results of the 8-attribute and reduced features using the SHAP algorithm.

5. Result and Discussion

The plot in Figure. 15 helps to explain the contribution of different features to the model's output for individual predictions. The model orders the listed features SizeOfStackReserve, DllCharacteristics, MajorImageVersion, NumberOfSections, and MajorOperatingSystemVersion based on their overall impact on the model's output. The SHAP value serves as an indicator of how a feature influences the model's prediction. A positive SHAP value means the feature pushes the prediction towards a higher output (e.g., more likely to be Malware).

In comparison, a negative SHAP value pushes it towards a lower output (e.g., less likely to be Malware), and the dot on the plot represents a single instance (e.g., a single file is classified). The dot's position on the horizontal axis shows the SHAP value for that feature in that specific instance. The colour of the dots represents the feature value for that instance,

and the colour bar on the right indicates that blue dots correspond to low features and red dots correspond to high features. The overall trend of the dots for each feature provides insight into how the feature affects the prediction. For example, suppose red dots tend to cluster on the right side (positive SHAP values) for a particular feature. In that case, higher values of that feature generally increase the likelihood of the instance being classified as Malware. In general, this plot helps visualize the most important features.

Features higher on the vertical axis have a larger overall impact. The impact of each feature and its value on individual predictions is explained as follows: The position of each dot shows the specific effect of that feature for a given instance, and the colour of the dots helps understand whether the high or low values of a feature contribute more to the prediction. Thus, the study of SHAP identified SizeOfStackReserve, DllCharacteristics, and MajorImageVersion as the most important features for classification. Based on SHAP values, we were able to determine the following essential features: We further applied the classification algorithm to these crucial features, including SizeOfStackReserve, DllCharacteristics, and MajorImageVersion, and observed the performance behaviour for various parameters. Fig. 17 explains that DT and RF achieve the highest accuracy at 97.16%, indicating a high proportion of correct predictions. KNN follows closely with an accuracy of 97.1%, reflecting its robustness. SVM shows moderate accuracy at 90.68%, while NB and LR trail with 87.96% and 85.36%, respectively.

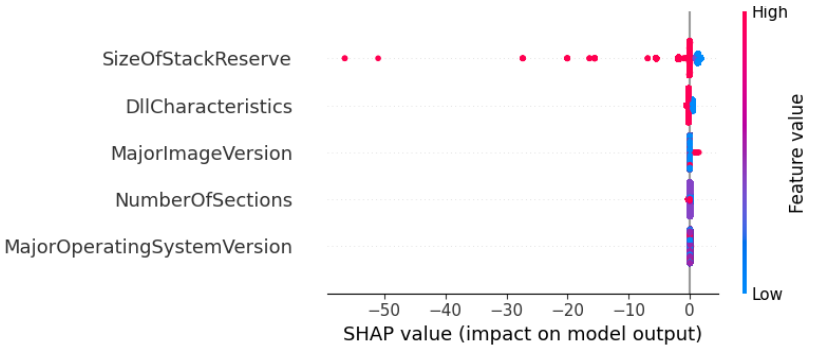

Fig.15. SHAP Algorithm on input attributes

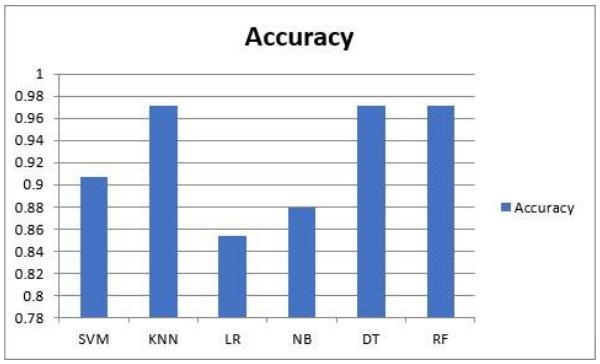

Fig.16. SHAP attributes: Accuracy

Figure 17 shows NB achieves the highest precision at 98.44%, which reflects its strong ability to minimize false positives. SVM comes next with 98.13%, while KNN, DT, and RF have similar precision scores around 95.18%. LR performs the weakest in precision, with a score of 73.21%, indicating a higher rate of false positives.

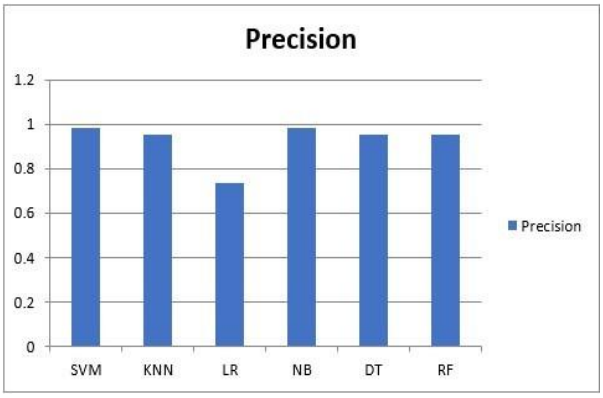

Fig.17. SHAP attributes: Precision

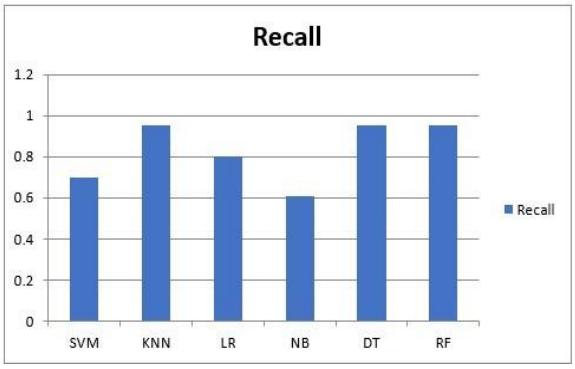

Fig.18. SHAP attributes: Recall

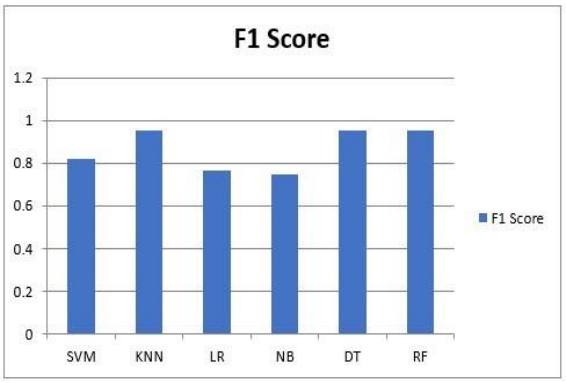

Fig.19. SHAP attributes: F1 score

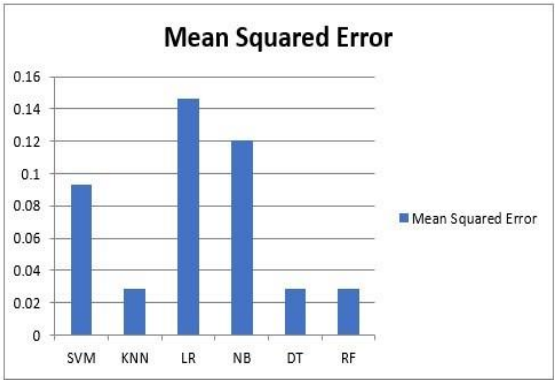

Fig.20. SHAP attributes: Mean squared error

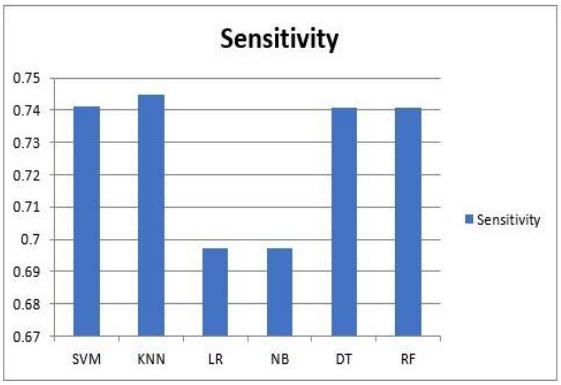

Fig.21. SHAP attributes: Sensitivity

Figure 19 displays DT and RF recall leads at 95.24% and 95.27%, showing their excellent capability to identify true positives. KNN also performs well with 95.1%, while SVM recalls are at 70.06%. NB falls behind at 60.6%, showing difficulty capturing true positives, and LR stands at 80.27. Figure 21 shows that DT and RF have the lowest MSE at

0.0284, indicating the fewest incorrect classifications. KNN also has a low MSE of 0.029, while SVM and NB have higher MSE values of 0.0932 and 0.1204, respectively. LR has the highest MSE at 0.1464, showing more misclassifications than other models.

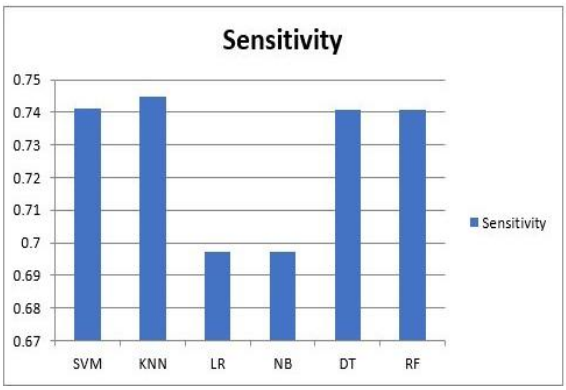

Fig.22. SHAP attributes: Specificity

Figure 21 shows that DT and RF lead in sensitivity with 95.24% and 95.27%, respectively, indicating a strong ability to detect true positives. KNN also performs well with 95.1%, while SVM shows moderate sensitivity at 70.06%. NB has the lowest sensitivity at 60.6%, and LR stands at 80.27%.

From Figure 16 to Figure 22, we know that DT and RF are the best-performing models across all metrics, excelling in accuracy, precision, recall, F1 score, and specificity. KNN also performs exceptionally well, with high accuracy and balanced precision-recall performance. SVM shows moderate results, with substantial precision but a lower recall and F1 score than the top performers. LR and NB struggle in comparison, with lower precision, recall, and F1 scores, making them less suitable for tasks where both positive and negative classifications are critical.

Table 3 indicates that NB achieves the highest specificity at 99.59%, correctly identifying true negatives. SVM also performs well with a specificity of 99.43%, while KNN, DT, and RF have similar scores of around 97.95%. LR has the lowest specificity at 87.52%, indicating more false positives.

Table 3. Applied the SHAP Algorithm on the original data

Classifier	SVM	KNN	LR	NB	DT	RF
Accuracy	0.9068	0.971	0.8536	0.8796	0.9716	0.9716
Precision	0.9813	0.9518	0.7321	0.9844	0.9523	0.9521
Recall	0.7006	0.951	0.8027	0.606	0.9524	0.9527
F1 Score	0.8175	0.9514	0.7658	0.7502	0.9524	0.9524
Mean Squared Error	0.0932	0.029	0.1464	0.1204	0.0284	0.0284
Sensitivity	0.7006	0.951	0.8027	0.606	0.9524	0.9527
Specificity	0.9943	0.9795	0.8752	0.9959	0.9798	0.9796

6. Conclusions

Applying the SHAP set of rules to the dataset significantly improved the overall performance of the classifiers, particularly KNN, Decision Tree, and Random Forest. Key measurements such as accuracy, F1 score, and Mean Squared Error (MSE) demonstrated outstanding gains for those classifiers, demonstrating that the SHAP-applied records most likely comprised more significant function insights, enhancing version efficacy. KNN attained an impressive 97.1% accuracy, a 74% enhancement in forgetting, and the minimal MSE among all models utilizing SHAP, rendering it especially advantageous. The rise in recall and F1 scores for KNN, Decision Tree, and Random Forest signifies that these classifiers are more adept at identifying significant events, demonstrating enhanced sensitivity and predictive capability.

Although specific measures like precision and specificity exhibited slight declines for Decision Tree and Random Forest with the use of SHAP, this trade-off yielded enhanced overall performance and reliability for real-world applications where recall and accuracy are paramount, which illustrates that SHAP-based comprehensive functional insights facilitated enhanced stability in identifying genuinely remarkable events while sustaining a lower error rate across all classifiers. SHAP enhanced the interpretability of models and optimized their performance on key evaluation measures, making it a valuable tool for refining machine learning models, particularly in high-stakes applications that require reliable quality detection.

References

[1] Gibert, D., "Machine Learning for Windows Malware Detection and Classification: Methods, challenges and ongoing research", arXiv (Cornell University), 2024. DOI: https://doi.org/10.48550/arxiv.2404.18541.

[2] Syeda, D. Z., & Asghar, M. N., "Dynamic Malware classification and API categorization of Windows portable executable files using machine learning", Applied Sciences, 14(3), 1015, 2024.DOI: https://doi.org/10.3390/app14031015

[3] Li, T., Shou, P., Wan, X., Li, Q., Wang, R., Jia, C., & Xiao, Y. "A fast Malware detection model based on heterogeneous graph similarity search. Computer Networks", 110799, 2024. DOI: https://doi.org/10.1016/j.comnet.2024.110799

[4] Zhao, H., Wu, Y., Zou, D., Liu, Y., & Jin, H. "MalSensor: Fast and Robust Windows Malware Classification." ACM Transactions on Software Engineering and Methodology" (2024). DOI: https://doi.org/10.1145/3688833

[5] Rudd, E. M., Krisiloff, D., Coull, S., Olszewski, D., Raff, E., & Holt, J. "Efficient Malware Analysis Using Metric Embeddings. Digital Threats Research and Practice", 5(1), 1–20, 2024. https://doi.org/10.1145/3615669

[6] Al-Khshali, H. H., Ilyas, M., Sohrab, F., & Gabbouj, M. "Malware Detection with Subspace Learning-based One-Class Classification." IEEE Access, 12, 81017–81029, 2024. https://doi.org/10.1109/access.2024.3409937

[7] Shaukat, K., Luo, S., & Varadharajan, V. "A novel machine learning approach for detecting first-time-appeared Malware. Engineering Applications of Artificial Intelligence", 131, 107801 2024. https://doi.org/10.1016/j.engappai.2023.107801

[8] Tyagi, S., Baghela, A., Dar, K. M., Patel, A., Kothari, S., & Bhosale, S., "Malware Detection in PE files using Machine Learning." OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON), 2023. https://doi.org/10.1109/otcon56053.2023.10113998

[9] Ngwobia, S. C., Ralescu, A., Kapp, D., & Kebede, T. "Detection of malicious PE files using synthesized DNA artifacts. Computers & Security", 134, 103457 2023. https://doi.org/10.1016/j.cose.2023.103457

[10] Vashishtha, L. K., Chatterjee, K., & Rout, S. S. "An Ensemble Approach for Advanced Malware Memory Analysis Using Image Classification Techniques. Journal of Information Security and Applications", 77, 103561, 2023. https://doi.org/10.1016/j.jisa.2023.103561

[11] Kamboj, A., Kumar, P., Bairwa, A. K., & Joshi, S. "Detection of Malware in Downloaded Files Using Various Machine Learning Models." Egyptian Informatics Journal, 24(1), 81–94, 2022. https://doi.org/10.1016/j.eij.2022.12.002

[12] Fascí, L. S., Fisichella, M., Lax, G., & Qian, C. "Disarming visualization-based approaches in Malware detection systems", Computers & Security, 126, 103062, 2022. https://doi.org/10.1016/j.cose.2022.103062

[13] Mimura, M. "Evaluation of printable character-based malicious PE file detection method. Internet of Things", 19, 100521, 2022. https://doi.org/10.1016/j.iot.2022.100521

[14] Demirci, D., Sahin, N., Sirlancis, M., & Acarturk, C. Static Malware detection using stacked BILSTM and GPT-2. IEEE Access, 10, 58488–58502, 2022. https://doi.org/10.1109/access.2022.3179384

Authors' Profiles

Balachandra Chikkoppa: Assistant Professor of the Department of Computer Science, KLE Institute of Technology, Gokul, Hubballi, 580028, Karnataka, India. Areas of scientific interests: network security, Malware analysis, computer networking, and AIML.

Hanumanthappa J.: Doctor of Sciences (Engineering), Full Professor, Department of Science in Computer Science, University of Mysore, Manasgangothri, Mysuru, 10587, Karnataka, India. Areas of scientific interests: IoT security, Cybersecurity, computer networking, and cloud technologies.

Wai Yie Leong: Pro Vice Chancellor, INTI International University and Colleges, Nilai, Negeri Sembilan, Malaysia. Areas of scientific interests: IoT, Cybersecurity, computer networking, and cloud technologies.

How to cite this paper: Balachandra Chikkoppa, Hanumanthappa J, Wai Yie Leong, "Malware Detection and Classification using Shapley Additive Explanations Values in Machine Learning", International Journal of Computer Network and Information Security(IJCNIS), Vol.18, No.1, pp.18-32, 2026. DOI:10.5815/ijcnis.2026.01.02