Journal of Information Systems Engineering and Management
2025, 10(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Multiclass Malware Detection in Operational Technology Systems Using Machine Learning on PE Header Specifications

Ashwini Kumar Verma[1*] and Sanjay Kumar Sharma[2]
[1,2]Department of CSE, University School of ICT, Gautam Buddha University, Greater Noida, India
[1]cyberneticsjustice@gmail.com, https://orcid.org/0000-0001-7580-7392;
[2]sanjay.sharma@gbu.ac.in & https://orcid.org/0000-0001-7918-4938

ARTICLE INFO

Received: 05 Nov 2024
Revised: 27 Dec 2024
Accepted: 21 Jan 2025

ABSTRACT

Malware, short for malicious software, presents a substantial cybersecurity threat within operational technology (OT) systems. It delineates the diverse array of malware threats, encompassing, backdoors, trojans, viruses, worms, and trojan-droppers, highlighting their potential to disrupt industrial operations and compromise sensitive data. In this paper, realm of multi-class classification of malware within OT systems is focused underscoring the pressing need for tailored malware detection techniques in such environments. To effectively counter these threats, Signature-based Detection (SD) method alongside machine learning algorithms are employed on labeled datasets. Multiclass classification of malware is focused in which the intricate process of data pre-processing is elucidated which involves extracting, cleaning, and transforming raw PE file data to facilitate machine learning analysis. Moreover, it elucidates the integration of H2O AutoML for optimizing models and evaluates the performance of various machine learning algorithms using key metrics. The proposed approach provides valuable insights into the sophisticated methodologies employed for multi-class malware classification in OT systems, thereby enhancing cybersecurity measures in critical infrastructure sectors. Results shows that the proposed approach has the 96.8% accuracy better than the state of the art techniques which focus only on two classes such as malicious and benign.

Keywords: Multi-class classification, Malware detection, Operational technology (OT) systems, Signature-based Detection (SD), Machine learning algorithms, H2O AutoML.

1. INTRODUCTION

Computer networks play a critical role in our daily lives, facilitating various activities, including the operation of essential infrastructure such as operational technology (OT) systems [1]. In smart cities, interconnected networks contribute significantly to urban efficiency and sustainability by optimizing resource management and facilitating sustainable development. However, despite the seamless nature of network interaction, there's a constant flow of data packets transmitted between destinations. If these packets are not adequately secured, they can lead to severe issues such as breaches of personal data, unauthorized access, financial fraud, or disruption of essential services. Moreover, the integration of OT systems within these networks introduces additional complexities and security challenges [2]. OT systems are used to monitor and control physical processes in critical infrastructure sectors such as energy, transportation, and manufacturing. These systems often rely on specialized hardware and software tailored to the specific needs of industrial operations. However, the convergence of OT with information technology (IT) systems has expanded the attack surface and increased the risk of cyber threats, including malware infections [3].

Malware, a contraction of "malicious software," poses a significant cybersecurity risk in the realm of OT systems. Malicious actors target OT environments with various types of malware, including backdoors, trojans, viruses, worms, and other forms of malicious software. Ransomware, considered one of the most notorious forms, encrypts user files and demands payment for decryption. Backdoors provide unauthorized remote access to a computer, enabling attackers to control it without detection. Spyware operates covertly, surreptitiously collecting user data such as activity logs and browsing history for remote monitoring. Trojans, disguised as legitimate software, execute malicious actions upon activation, often facilitating the installation of additional harmful software. Worms exploit system vulnerabilities to autonomously spread and replicate across networks without human intervention. Lastly, viruses infect and replicate within host systems, spreading through human actions like file sharing or opening infected files. These malware variants can exploit vulnerabilities in OT software and hardware, disrupt industrial processes, steal sensitive data, or cause physical damage to equipment [4-5].

To combat these threats effectively, organizations employ advanced malware detection techniques tailored to the unique characteristics of OT environments. One common approach involves analysing Portable Executable (PE) headers[6], which provide essential information about executable files commonly found in OT systems. By examining the PE header, security systems can identify potential threats and classify them based on known characteristics of malicious activities. Additionally, organizations leverage sophisticated detection methods such as Signature-based Detection (SD) and Anomaly-based Detection (AD) to identify and mitigate malware threats in OT environments. Signature detection involves analysing network traffic or executable files to identify patterns that match known characteristics of malicious activities, including specific signatures within the PE header. Meanwhile, anomaly-based detection establishes rules for "normal" behaviour and detects unusual activity that deviates from this norm, potentially indicating a malware infection [7].

Machine learning algorithms, such as decision trees, support vector machines (SVMs), neural networks, and ensemble methods, are trained on labelled datasets containing various types of network traffic or PE headers, including normal and malicious instances. These algorithms learn to recognize patterns indicative of different types of malware, enabling them to classify incoming traffic or executable files into multiple categories, such as backdoor, trojan, benign, trojan-downloader, trojan-dropper, virus, and worm. By identifying the specific type of threat, security teams can tailor their response strategies accordingly, deploying appropriate countermeasures to mitigate the threat effectively [2][5]. Once trained, the multi-class classification model continuously monitors incoming traffic or executable files in real-time, analyzing data and assigning each instance to the most appropriate class based on learned patterns. This proactive approach enables organizations to detect, classify, and respond to malware threats effectively, reducing the impact of attacks and safeguarding critical services and assets in OT systems.

In current research, authors address multi-class malware detection using various machine learning models in OT systems. Initially, raw PE file data is parsed to extract important attributes such as file size, header information, section characteristics, and import/export tables. This extracted data is then cleaned and feature engineering techniques are applied to convert numerical data into categorical data suitable for multiclass classification. The data is transformed into seven classes, enhancing its usability for modelling. Subsequently, feature selection is performed which reduces dimensionality and enhances the efficiency and accuracy of the classification model. After feature selection, the approach integrates H2O AutoML, a powerful tool for hyperparameter tuning and model optimization. This integration further improves classification efficiency by automating the selection of the best machine learning algorithms and parameter settings.

Various machine learning algorithms, including decision trees, adaboost, gradient boosting machines, support vector machines, and logistic regression, are then applied to the pre-processed data. These algorithms are trained on the dataset to learn the underlying patterns and relationships between the features and the target variable, enabling effective malware classification. Once the models are trained, model evaluation is performed using appropriate evaluation metrics such as accuracy,

precision, recall, and F1-score. This comprehensive evaluation ensures that the models effectively classify malware and meet the desired performance criteria and achieved an accuracy of 96.8% by applying Random forest with H2O.

Major contributions in this work are:

- This research addresses cybersecurity challenges arising from the constant flow of data packets in computer networks. Inadequate security measures can lead to severe consequences, including data breaches, unauthorized access, financial fraud, or service disruptions. Integration of OT systems in networks further complicates these challenges.
- The study examines the risks posed by malware, including various types such as backdoor, trojan, benign, trojan-downloader, trojan-dropper, virus, and worm. It explores advanced detection techniques like PE header analysis, Signature-based Detection (SD), and Anomaly-based Detection (AD) to mitigate these risks effectively.
- This research highlights the role of machine learning algorithms in detecting and classifying malware in OT systems. Various models, including decision trees, support vector machines (SVMs), neural networks, and ensemble methods, are employed for multi-class malware detection, contributing to enhanced security measures in OT environments.
- Further, H2O AutoML has been integrated to reduce the time complexity and enhances the efficacy of the proposed approach.

2. RELATED WORK

In today's interconnected world, computer networks serve as the backbone of essential infrastructure, facilitating various activities and enabling the operation of critical systems such as operational technology (OT) systems. The integration of these networks into smart cities has been instrumental in enhancing urban efficiency and sustainability by optimizing resource management and supporting sustainable development initiatives [5]. However, the seamless flow of data within these networks also poses significant security challenges, as any vulnerabilities could lead to severe consequences such as breaches of personal data, unauthorized access, financial fraud, or disruption of essential services. One of the key challenges in securing computer networks, particularly in the context of OT systems, is the threat posed by malware. A comprehensive comparative analysis of the previous researches is presented in Table 1.

Table 1. Comprehensive Comparative Analysis

Study/Method	Year	Techniques Used	Dataset Size	Best Accuracy
Kolter and Maloof [8]	2004	Decision trees, Naive Bayes, SVM	1971 benign, 1651 malware	99.6%
Karim et al. [9]	2005	N-perms, N-grams	Not specified	Not specified
Henchiri et al. [10]	2006	Iterative Dichotomiser-3, J48, Naive Bayes, SVM	1512 viruses, 1488 benign	92.56%
Blair [11]	2007	Opcodes	67 malware, 20 benign	Not specified
Moskovitch et al. [12]	2008	Text categorization	~30,000 malware, benign	Up to 95%
Moskovitch et al. [13]	2008	N-grams, Boosted DT	Not specified	94.43%
Ye et al. [14]	2008	Windows API, Objective-Oriented Association	636 malicious, 1207 benign	Not specified
Tian et al. [15]	2008	Function length	Not specified	Not specified
Siddiqui et al. [16]	2008	Data mining	2774 (1330 benign, 1444 worms)	95.6%
Tabish et al. [17]	2009	Statistical, Information-theoretic features	37,420 malware, 1800 benign	90%

Mehdi et al. [18]	2009	In-Execution Malware Analysis and Detection (IMAD)	Not specified	90%
Mehdi et al. [19]	2009	Hyper-grams, Variable-length system calls	72 benign, malware files	Not specified
Santos et al. [20]	2011	Single-class learning based on opcode occurrence	1000 benign, 1000 malware executables	~85%
Ravi et al. [21]	2012	Association mining based classification, API call sequence modeled by third-order Markov chain	Not specified	90%
Liangboonprakong et al. [22]	2013	N-grams sequential pattern features, SVM, C4.5 DT, ANN	Not specified	Up to 96.64%
Santos et al. [23]	2013	Opcode sequence occurrence	13,189 malware, 13,000 benign executables	Up to 95.90%
Salehi et al. [24]	2014	Runtime behavior-based feature sets, RF, J48, Rotation RF, FT, NB classifiers	385 benign, 826 malware files	Up to 98.1%
Jikku Kuriakose et al.[25]	2015	Feature ranking methods (TF-IDF, GSS, OR, CMFS, MOR)	Not specified	100%
Mansour Ahmadi et al.[26]	2015	Learning-based system, Portable executables features	Half terabytes of data	99.8%
Ashu et al.[27]	2016	Malware generator kits detection, Random forest, NBT classifier, Optimal k-means clustering	Malicia dataset	Up to 99.11%
Zhixing Xu et al.[28]	2017	Virtual memory access patterns, Logistic regression, Random forest	RIPE benchmark suite	99%
Kotov et al[29]	2018	Static analysis, Hidden Markov model, API calls	Not specified	87.6%
Burnap et al.[30]	2018	Malware Operational Plot Review (MOPR) model, Self-organizing feature maps	Not specified	93.76%
Li et al.[31]	2018	Virtual time control mechanics-based method, Modified Xen hypervisor	Not specified	Not specified
Liu et.al.[32]	2020	Comprehensive review	Not Specified	Not Specified
Kouliaridis et.al. [33]	2021	Comprehensive review	Not specified	Not specified
Tyagi et.al. [34]	2022	Static analysis technique	Not Specified	96.7%
Akhtar et.al.[35]	2023	Dynamic malware detection	301 malicious and 72 benign	100%

2.1 Research Gaps

The following gaps have been identified after conduting the literature review.

- There is a lack of detailed studies focusing specifically on the types and behaviour of malware targeting OT systems compared to IT systems.
- Existing research often focuses on individual detection methods (e.g., signature-based or anomaly-based detection) rather than a comprehensive evaluation and integration of multiple techniques for improved accuracy and robustness.
- Machine learning models in real-time scenarios within OT environments does not focus on latency and detection accuracy.

- Advanced feature engineering methods were not explored that could potentially improve detection accuracy.
- Multiclass malware detection is not performed till date to the best of our knowledge.

3. RESEARCH METHODOLOGY

This section provides an overview of malicious softwares, portable executable files, datasets and feature selection methods used.

3.1 Malicious software

Malicious software, commonly referred to as malware, is designed with the intent to cause harm and disrupt computer systems. Unlike regular programs, malware presents a significant threat due to its destructive capabilities. With numerous releases annually, malware poses a continuous challenge to the security of computer systems and the Internet. Traditional detection methods, like signature-based approaches, rely on identifying specific sequences of bytes associated with known malware. However, this method has limitations as malware creators employ obfuscation techniques to evade detection, making signature-based methods less effective against new and disguised threats.

To overcome these limitations, machine learning-based techniques have emerged as promising solutions. Unlike signature-based methods, machine learning models can learn and generalize patterns from training data, enabling them to detect variations of malware even when their signatures change. Moreover, as malware developers continually modify their code to create new variants, machine learning models can adapt to these changes, offering a more robust approach to malware detection that can keep pace with evolving threats.

In the current research, the authors focus on multi-class malware detection using various machine learning models. While previous studies primarily concentrated on binary classification, distinguishing between benign and malicious files, there is a recognized need to classify malicious files into different categories such as viruses, worms, Trojan horses, spyware, adware, and ransomware. This shift towards multi-class classification allows for a more nuanced analysis of malware behaviour, thereby enhancing the effectiveness of detection strategies. These categories exemplify the diverse nature and potential threat posed by malware, underscoring the importance of robust detection and prevention measures to safeguard computer systems and data. In the context of multi-class classification of malware, machine learning-based techniques offer a promising avenue for enhancing detection capabilities and addressing the evolving landscape of threats.

3.2 PE file format

The Portable Executable (PE) file format, commonly used in Windows operating systems, plays a crucial role in malware detection by offering a structured layout to analyse executable files. By examining PE file structures, researchers can extract key features to differentiate between benign and malicious programs. The PE file consists of a header followed by sections, with each section serving a distinct purpose. The PE header includes the DOS header for file validation, the "PE \ 0 \ 0" signature identifying it as a PE file, and crucial details like the compilation timestamp and the number of sections. The Data_Directory structures within the Optional header point to important tables like the Import, Export, and Resource tables, facilitating access to essential data. The section table provides information about the program sections, and the actual file contains executable code, data, and resources that define its functionality within the Windows environment. As depicted in Figure 1, the PE file consists of a header followed by a sequence of sections, with each section serving a distinct purpose.

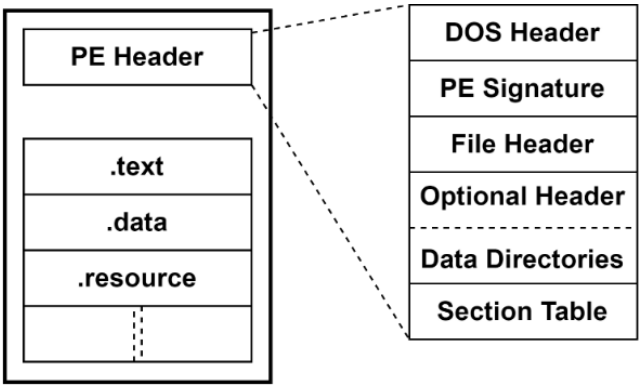


Figure 1. Format of PE Header file

3.3 Dataset used

In this work, datasets used is of Operational Technology, gather from C3i Hub, Indian Institute of Technology, Kanpur. The dataset comprises of 10K PE files. The PE files are further categorized as Benign and Malicious (backdoor, trojan, benign, trojan-downloader, trojan-dropper, virus, and worm). The distribution of the samples files is presented in Figure 2.

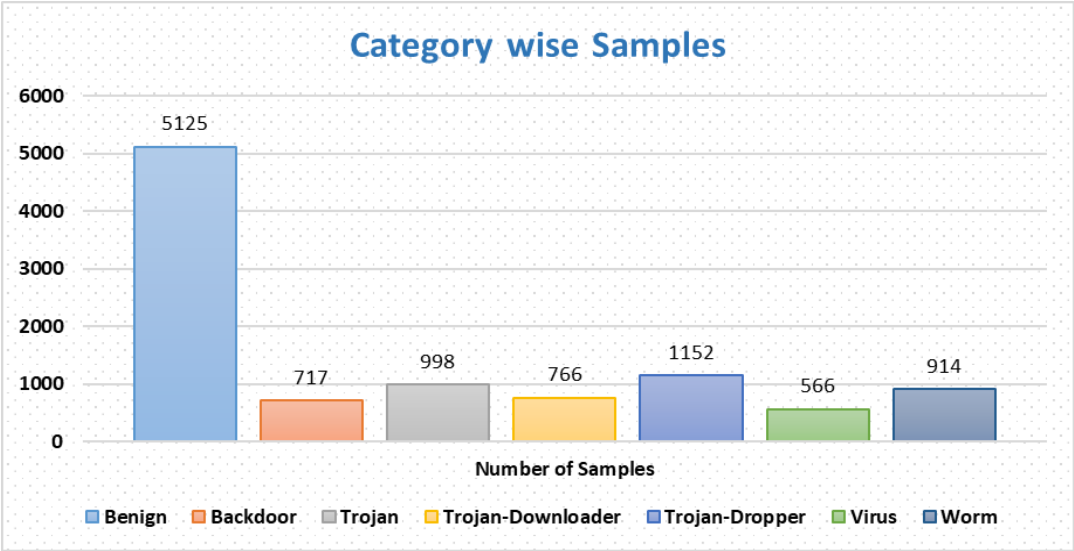


Figure 2. Category wise Samples

3.4 Feature Extraction

In machine learning-based systems designed for malware detection, a critical phase involves the extraction of features. Within this domain, two primary categories of features are commonly employed: Static Features and Dynamic Features. Static feature attributes are derived from a file without the need for executing it, which provide valuable insights into the inherent characteristics and attributes of the file itself. By analyzing such aspects namely; file size, metadata, and code structure, static features enable the identification of potential indicators of malicious behavior. Despite their effectiveness in revealing structural patterns, it suffers in capturing dynamic behaviors exhibited by malware during execution. On the other hand, Dynamic Features are obtained by executing a file, typically within a controlled environment conducive to monitoring and analysis which captures the actual behaviors and actions of the program during runtime. By observing factors such as system calls, memory usage, and network activity, dynamic features offer a more comprehensive understanding of the program's behavior. However, extracting dynamic features often entails greater time and computational resources compared to static analysis and are more susceptible to evasion tactics employed by adversaries. For instance, attackers may introduce time delays or evasion techniques to circumvent dynamic analysis, thereby challenging the efficacy of dynamic feature extraction methods.

In the work, both static and dynamic methods were employed for feature extraction, leveraging the unique strengths of each approach to enhance the effectiveness of malware detection systems. These features are utilized for the detection and classification of different classes of malware presented in Table 2. Table 2 shows features name alongwith the desciptions. 54 features are presented out of 57 as name and labels were not considered as features.

Table 2. PE Header File Features and its description

S. No.	Feature Name	Description
1.	DebugSize	Size of the debug information.
2.	IATRVA	Address of the Import Address Table (IAT).
3.	SizeOfHeapCommit	Size of the heap to commit.
4.	FileSize	Total size of the file.
5.	DebugRVA	Relative Virtual Address (RVA) of the debug section.
6.	ResSize	Size of the resources section.
7.	SizeOfHeaders	Size of the headers.
8.	Machine	Type of machine (e.g., Intel x86).
9.	ImageVersion	Version of the image.
10.	LinkerVersion	Version of the linker.
11.	StackReserveSize	Size of the stack to reserve.
12.	TimeDateStamp	Time and date when the file was created or modified.
13.	OSVersion	Version of the operating system.
14.	VirtualSize2	Virtual size of the section.
15.	SizeOfHeapReserve	Size of the heap to reserve.
16.	Characteristics	Characteristics of the file.
17.	ExportRVA	RVA of the export section.
18.	NumberOfSections	Number of sections in the file.
19.	SizeOfImage	Size of the image.
20.	SizeOfOptionalHeader	Size of the optional header.
21.	ExportSize	Size of the export section.
22.	SizeOfCode	Size of the code section.
23.	SizeOfInitializedData	Size of the initialized data section.
24.	SizeOfStackCommit	Size of the stack to commit.
25.	AddressOfEntryPoint	Address of the entry point.
26.	MajorImageVersion	Major version of the image.
27.	MajorSubsystemVersion	Major version of the subsystem.
28.	SizeOfUninitializedData	Size of the uninitialized data section.
29.	CheckSum	Checksum of the file.
30.	MinorImageVersion	Minor version of the image.
31.	MinorSubsystemVersion	Minor version of the subsystem.
32.	NumberOfRvaAndSizes	Number of RVA and sizes.
33.	SectionAlignment	Alignment of the sections.
34.	MajorLinkerVersion	Major version of the linker.
35.	SectionsLength	Length of the sections.
36.	LoaderFlags	Flags used by the loader.
37.	MajorOperatingSystemVersion	Major version of the operating system.
38.	MinorLinkerVersion	Minor version of the linker.
39.	SectionMinEntropy	Minimum entropy of the sections.
40.	SectionMaxEntropy	Maximum entropy of the sections.
41.	SectionMinVirtualSize	Minimum virtual size of the sections.
42.	SectionMinPhysical	Minimum physical size of the sections.

43.	SectionMaxChar	Maximum characteristics of the sections.
44.	SectionMinRawSize	Minimum raw size of the sections.
45.	SectionMaxVirtualSize	Maximum virtual size of the sections.
46.	SectionMaxPointerData	Maximum pointer data of the sections.
47.	Reserved1	Reserved field.
48.	SectionMaxRawSize	Maximum raw size of the sections.
49.	SectionMaxPhysical	Maximum physical size of the sections.
50.	SectionMinPointerData	Minimum pointer data of the sections.
51.	Dll	Indicates if the file is a Dynamic Link Library (DLL).
52.	ImportFunctionCount	Count of imported functions.
53.	ImportFunctionMethodCount	Count of methods used for importing functions.
54.	MD5Hash	MD5 hash of the file.

4. PROPOSED METHODOLOGY

In this work, a framework for multiclass classification of malwares using machine learning models is proposed. The proposed approach involves several steps namely, data pre-processing, feature selection, integration of h2o automl interfaces and training the machine learning models. The framework of the proposed approach is depicted in Figure 3.

4.1 Data Pre-Processing

Data pre-processing on PE files is a critical step in preparing them for machine learning-based analysis for multiclass classification of malware detection. This process involves several key steps aimed at cleaning, transforming, and structuring the raw PE file data to make it suitable for modeling. Initially, the PE files are parsed to extract essential attributes such as file size, header information, section characteristics, import/export tables, and other relevant features. Next, data cleaning techniques have been applied to handle missing values, outliers, or inconsistencies in the dataset. Additionally, feature engineering techniques have been applied which converts the numerical data to categorical data for multiclass classification. The data have been transformed into seven classes as presented in figure 2. Finally, the pre-processed data is normalized to ensure that all features are on a similar scale, preventing any single feature from dominating the modelling process. Overall, data pre-processing plays a crucial role in optimizing the quality and usability of PE file data for subsequent machine learning analysis.

4.2 Feature Selection

After data pre-processing, feature selection is performed based on correlation. It involves identifying and selecting the most relevant features that exhibit a strong correlation with the target variable, i.e. class of malware. By analyzing the correlation between each feature and the target variable, redundant or irrelevant features has been eliminated, reducing dimensionality and improving the efficiency and accuracy of the classification model. Features with high correlation values are retained, as they provide valuable insights into the characteristics of malware and contribute significantly to the predictive power of the model. This process ensures that only the most informative features are included in the final dataset, enhancing the performance of the machine learning algorithm in accurately detecting and classifying different types of malware.

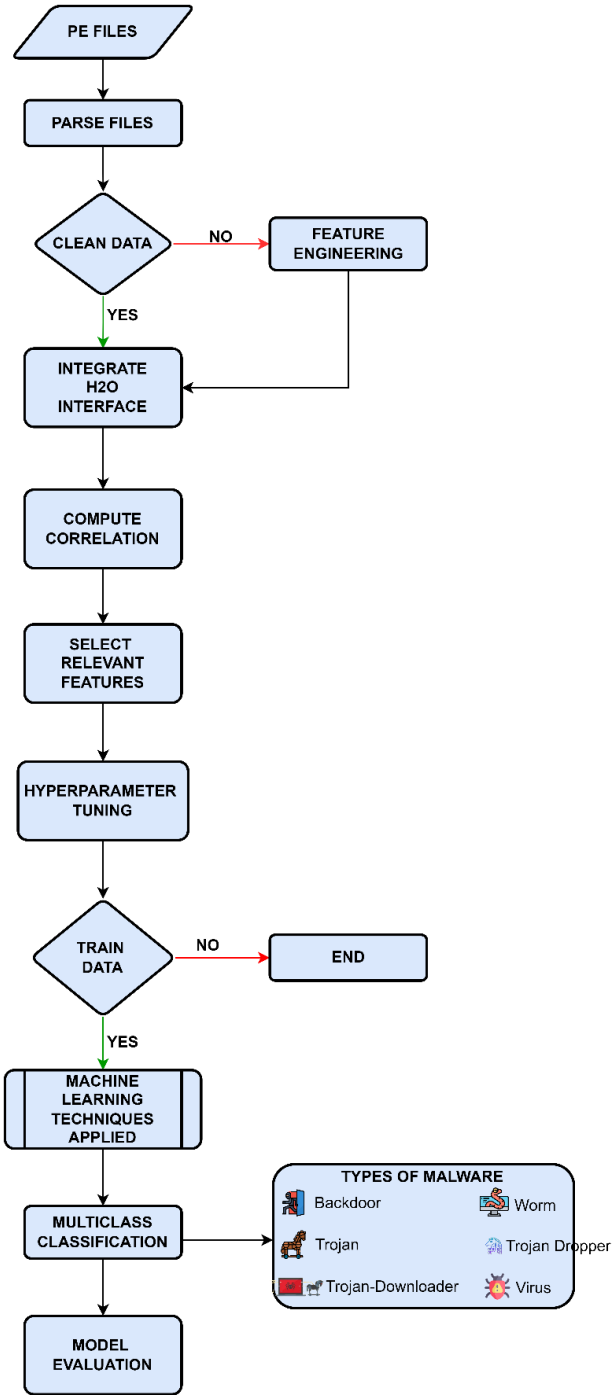


Figure 3. Proposed approach

4.3 Integrating H2O autoML interface

After feature selection, the approach integrates H2O AutoML which further enhances the efficiency and effectiveness of multiclass malware classification. H2O AutoML performs hyperparameter tuning on multi-dimensional dataset which enhances the speed and accuracy of various machine learning algorithms without human intervention.

4.4 Machine learning algorithms applied

Once the pre-processed data is ready and H2O is integrated, then these dataframe various machine learning techniques are applied to train and evaluate the models. These techniques include decision trees, adaboost, gradient boosting machines, support vector machines, and logistic regression. Each

algorithm is trained on the pre-processed dataset to learn the underlying patterns and relationships between the features and the target variable.

4.5 Model Evaluation

The performance of each model is then assessed using appropriate evaluation metrics, such as accuracy, precision, recall, and F1-score, to determine its effectiveness in classifying malware into different categories.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

$$Accuracy = \frac{Number\ of\ Correct\ predication}{Total\ Number\ of\ predication} \quad (3)$$

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

4.6 Pesudocode of the proposed approach

Algorithm: **Multiclass Classification of Malwares Using Machine Learning Models**

```
parsed_data = ParsePEFiles(PE_files)
cleaned_data = DataCleaning(parsed_data)
preprocessed_data = FeatureEngineering(cleaned_data)
transformed_data = TransformData(preprocessed_data)
normalized_data = NormalizeData(transformed_data)
  return normalized_data
// Analyze correlation between features and target variable
correlation_matrix = CalculateCorrelation(data)
 selected_features = SelectFeatures(correlation_matrix)
  return selected_features
// Integrate H2O AutoML for hyperparameter tuning
model = H2OAutoML(data)
  return model
// Train various machine learning techniques on pre-processed data
trained_models = TrainModels(data, model)
// Evaluate performance of each model
evaluation_results = EvaluateModels(trained_models)
   return evaluation_results
```

5. IMPLEMENTATION

Python language was used within a Google Colab environment for training our system across multiple machine learning models. Data reading, conversion to data frames, model splitting, and training were facilitated by key libraries like pandas, scikit-learn, seaborn, and plotly. Additionally, the H2O library was utilized for fast computation and feature reduction, resulting in enhanced dimensionality reduction and model efficacy. Computations were accelerated, and preprocessing time was minimized

by leveraging its scalable architecture. Model building was automated by its feature selection techniques, ensuring faster insights.

Step 1: Data Pre-processing:

The samples file is shown in Figure 4, after being read through the Colab environment comprising of 10238 PE files with total 57 attributes presented in columns. Assigning the class labels as per the different category, shown in Table 3.

Table 3. Category wise class labels and number of samples

Category	Class	Number of Samples
Benign	0	5125
Virus	1	566
Backdooor	2	717
Trojon	3	998
Trojan-dropper	4	1152
Worm	5	914
Trojan-downloader	6	766

filesize	Machine	TimeDateStamp	Characteristics	SizeOfOptionalHeader	DebugSize	DebugRVA	ImageVersion	OSVersion	...	section_min_physical
15360	0x14c	998081615	271	224	28	4272	5	5	...	68
16384	0x14c	998081342	271	224	28	4384	5	5	...	3080
77824	0x14c	997261829	271	224	28	29264	4	4	...	7711
118834	0x14c	993603227	271	224	28	66240	0	4	...	1240
13312	0x14c	998081627	271	224	28	4416	5	5	...	92

Figure 4. Reading of Sample files using python language

Step 2: Feature Selection:

In the initial step, certain features were removed from consideration due to their lack of significant variation across different samples. These features include "Name," “label", filesize," "Machine," "TimeDateStamp," "Reserved1," and "SizeOfOptionalHeader." As a result, the system was trained using 50 remaining features. Figure 5 illustrates the selected features.

Step 3: Integration of H2O interface

Following the removal of these features, we proceeded to employ the H2O Python library for additional feature reduction and expedited computation.

import h2o (5)

h2oinit() (6)

Some of the features of h2o is shown in Figure 6 as;

```
Index (['Characteristics', 'DebugSize', 'DebugRVA', 'ImageVersion', 'OSVersion',
'ExportRVA', 'ExportSize', 'IATRVA', 'ResSize', 'LinkerVersion','VirtualSize2',
'NumberOfSections', 'SizeOfCode', 'SizeOfHeapCommit',  'SizeOfHeaders',
'StackReserveSize', 'SizeOfHeapReserve', 'SizeOfImage',SizeOfInitializedData',
'SizeOfStackCommit', 'SizeOfUninitializedData',     'NumberOfRvaAndSizes',
'LoaderFlags', 'AddressOfEntryPoint', 'CheckSum', 'SectionAlignment',
'MajorOperatingSystemVersion',
'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion',
'MajorLinkerVersion', 'MinorLinkerVersion', 'MajorSubsystemVersion',
'MinorSubsystemVersion', 'sections_length', 'section_min_entropy',
```

```
'section_max_entropy', 'section_min_rawsize', 'section_max_rawsize',
'section_min_virtualsize', 'section_max_virtualsize',
'section_max_physical', 'section_min_physical',
'section_max_pointer_data', 'section_min_pointer_data',
'section_max_char', 'Dll', 'ImportFunctionCount',
'ImportFunctionMethodCount', 'md5hash', 'label'],
dtype='object')
```

Figure 5. Selected Features after the dimension Reduction

H2O_cluster_uptime:	07 secs
H2O_cluster_timezone:	Etc/UTC
H2O_data_parsing_timezone:	UTC
H2O_cluster_version:	3.46.0.1
H2O_cluster_version_age:	26 days
H2O_cluster_name:	H2O_from_python_unknownUser_pxibl9
H2O_cluster_total_nodes:	1
H2O_cluster_free_memory:	3.170 Gb

Figure 6. H2O Features Set

Step 4: Machine learning models applied

The system was trained using the H2O-based data frame generation. Various machine learning algorithms were utilized for this purpose, including logistic regression, decision trees, random forests, gradient boosting, and Adaboost. The dataset was divided into training and testing samples at a ratio of 70:30, respectively. Subsequently, the system underwent training with the machine learning model, followed by testing to evaluate its performance.

6. RESULTS AND DISCUSSION

The authors works on multi-class malware detection through the utilization of diverse machine learning models. Unlike previous studies that predominantly targeted binary classification, distinguishing solely between benign and malicious files, there exists a recognized necessity to categorize malicious files into various classes such as viruses, worms, Trojan horses, spyware, adware, and ransomware. This transition towards multi-class classification facilitates a more intricate examination of malware behaviour, consequently bolstering the efficacy of detection approaches. As discussed above authors has classified the data into 7 different labels and the different machine learning models are trained on 70:30 ratio. The models are trained on 7167 samples of different classes and tested on 3071 samples. The models are evaluated on the basic of evaluation metrics such as precision, recall accuracy and F1 score. Here show the results before applying h2o, table 4 shows the evaluation metrics of different machine learning models.

Table 4. Evaluation metrics over the various ML models without h2o

Model Applied	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.78	0.77	0.77	0.783
Decision Tree	0.9456	0.945	0.9465	0.9456
Random Forest	0.9528	0.9524	0.9533	0.9528
Gradient Boost	0.9504	0.9504	0.9517	0.9504
Ada boost	0.3603	0.2364	0.1817	0.3603

However, after applying the H2O the overall accuracy is increased. Table 5 shows the different evaluation parameters after applying h2o.

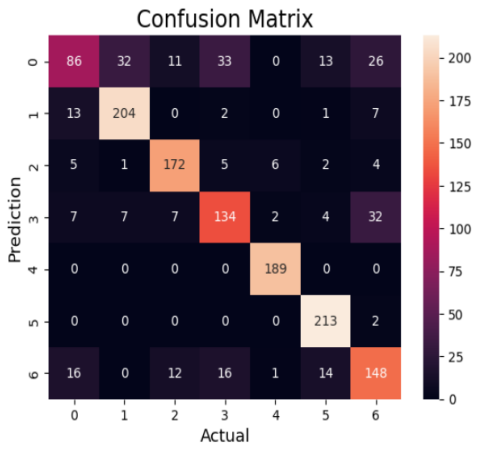
Table 5. Evaluation metrics over the various ML models with h2o

Model Applied	Accuracy	F1 Score	Precision	Recall
Logistic Regression	0.8031	0.7938	0.7945	0.8031
Decision Tree	0.949	0.9386	0.9386	0.939
Random Forest	0.9682	0.969	0.9691	0.96495
Gradient Boost	0.9586	0.9576	0.9566	0.9556
Ada boost	0.3104	0.187	0.1408	0.3104

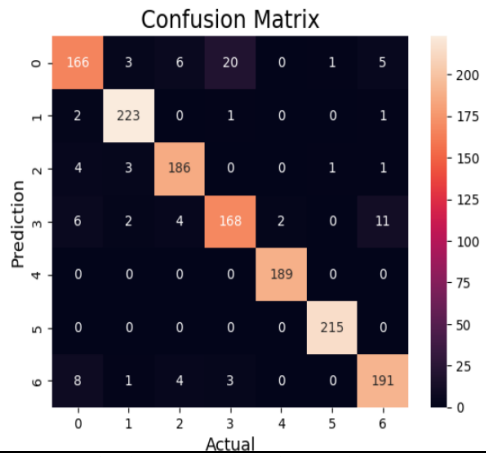
Figure 7 shows the confusion matrix of the LR, DTC, RF, GB and Ada Boost. Random forest has the maximum accuracy of 96.8% however Ada boost has the lowest as 31%.

Figure 8 shows the Overall comparison of accuracy, precision, recall and F1 score with h2o and without h2o, results depict in all the model's performance is increased after applying h2o and also reduces the model building time. The results indicate that the proposed approach, utilizing H2O with a random forest model, achieved the highest accuracy of 96.8% compared to other machine learning models. Notably, this is the first application of this approach to multiclass classification; previous research primarily focused on binary classification. In two-class classification, the proposed approach attained 100% accuracy with selected features. Hence, it is better than the existing state-of-the-art techniques.

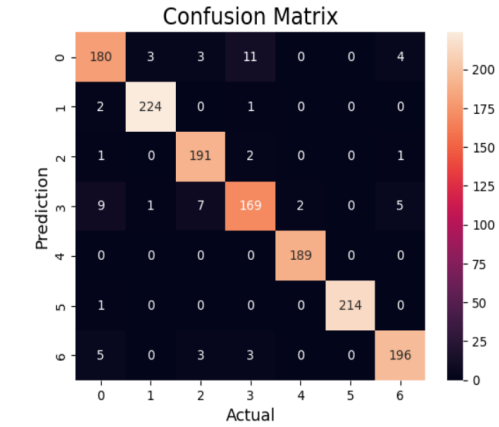
Confusion Matrix of Logistic Regression (LR)



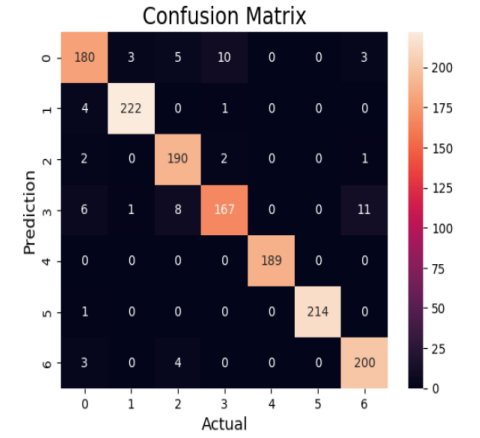
Confusion Matrix of Decision Tree (DTC)



Confusion Matrix of Random Forest (RF)



Confusion Matrix of Gradient Boost (GB)



Confusion Matrix of Ada Boost

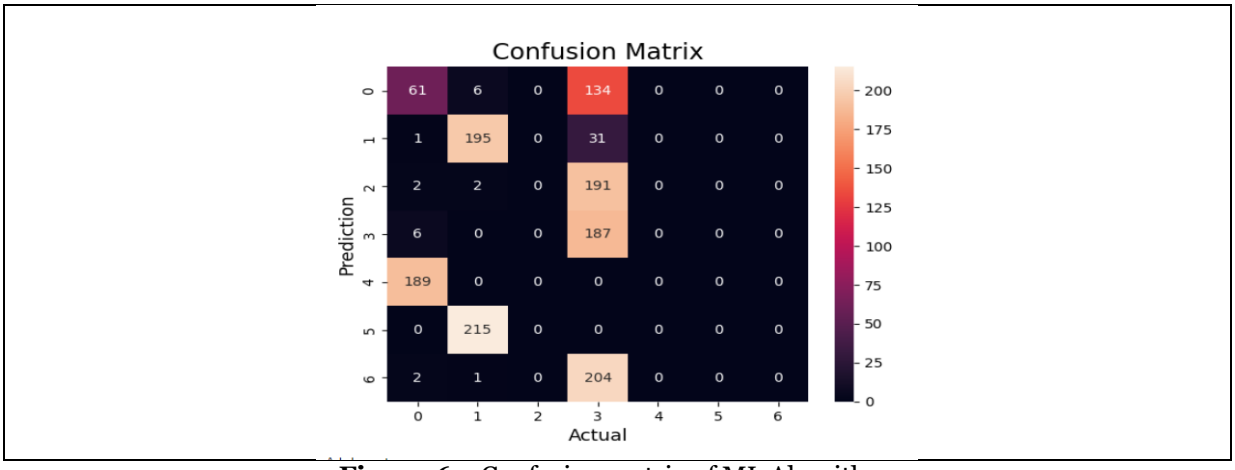
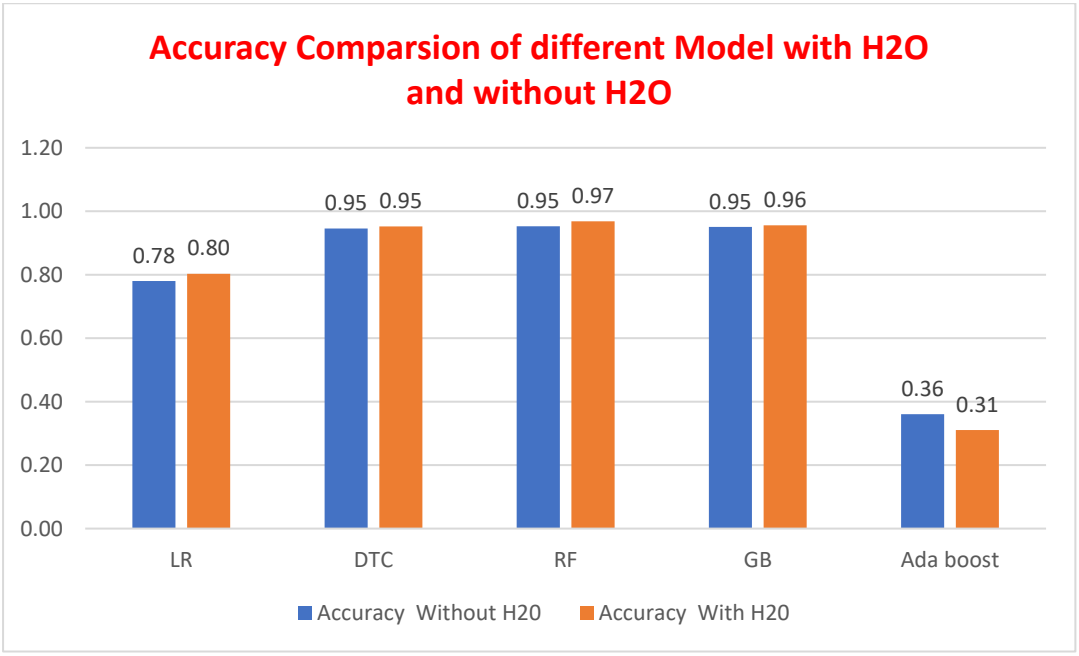
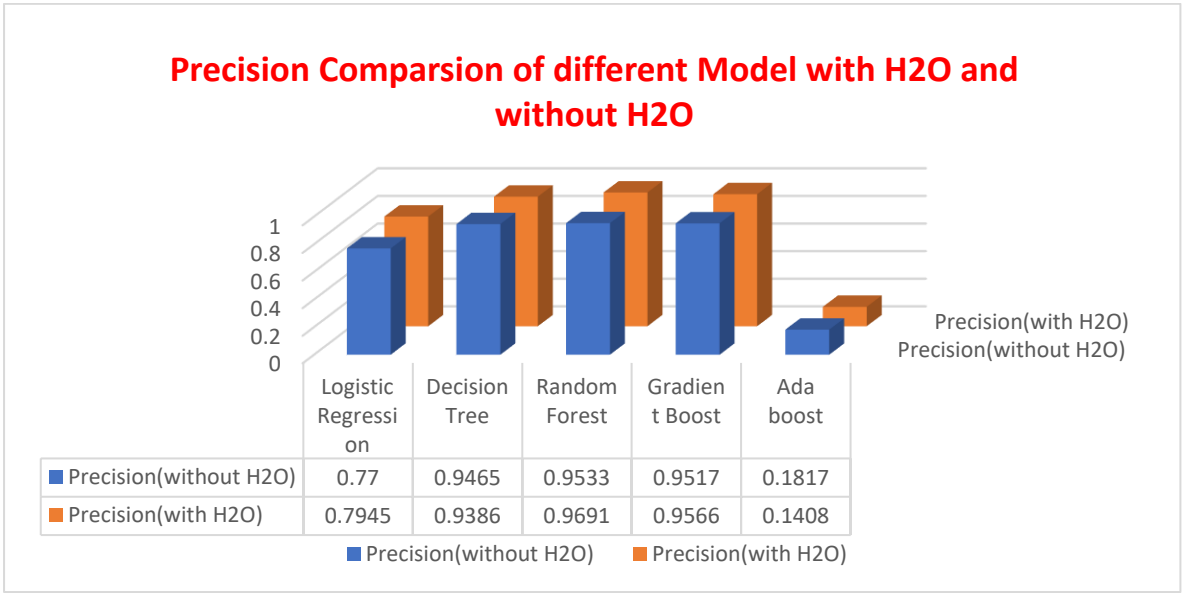
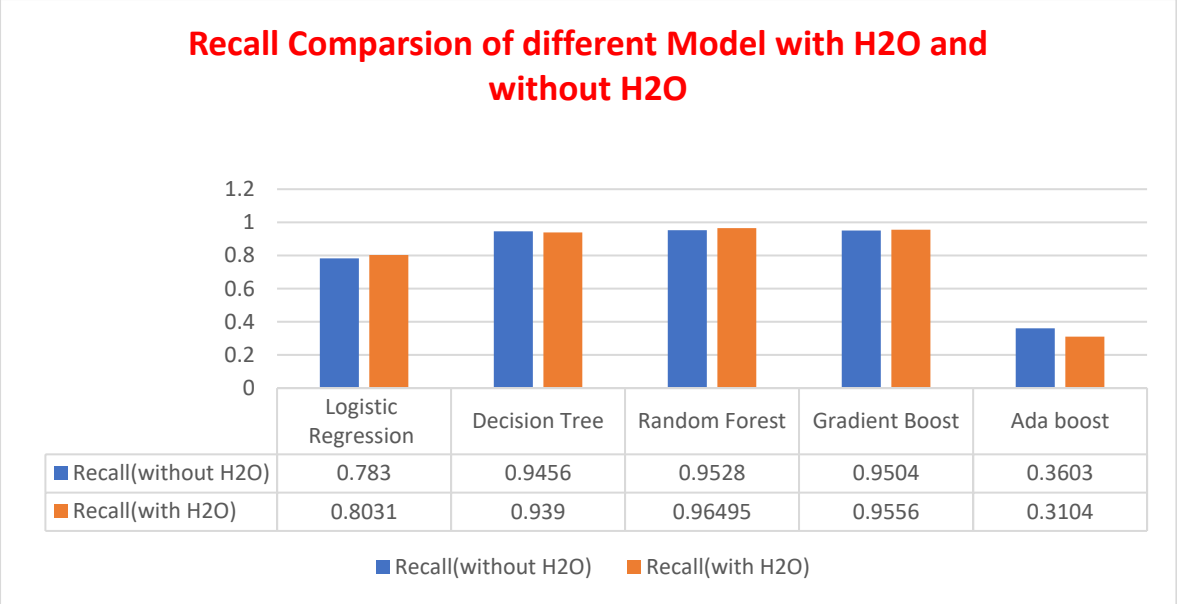


Figure 6. Confusion matrix of ML Algorithms







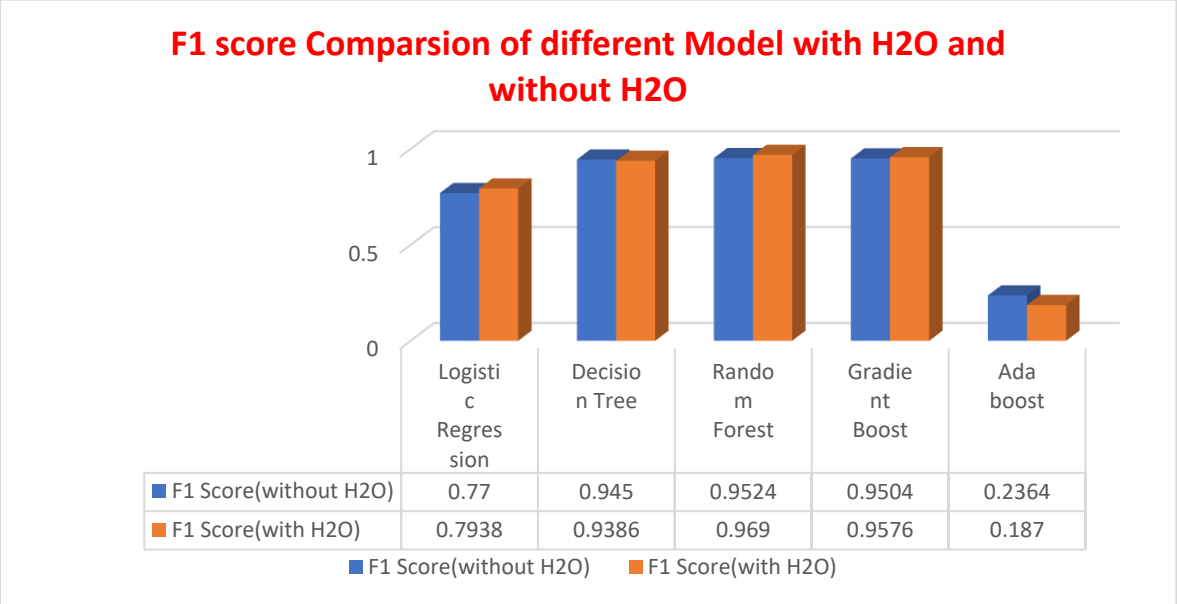


Figure 8. Comparison of Different ML models without H2O and with H2O

7. CONCLUSION

In the proposed research a comprehensive approach to multi-class malware detection in OT systems, employing various machine learning techniques is presented. Through the parsing and extraction of crucial attributes from raw PE file data, followed by feature engineering and selection processes, the dataset is optimized for multiclass classification. Integration of H2O AutoML further enhances model efficiency by hyperparameter tuning. By applying a range of machine learning algorithms and evaluating their performance using appropriate metrics, this study demonstrates the effectiveness of the proposed approach in accurately classifying malware. Random forest models explicit the maximum accuracy of 96.8 % as compared to other applied methods. Earlier research has maximum accuracy of 99 % having only two class labels however the proposed approach has 100% accuracy in two class labelled dataset, which is better than the state-of the art methods. These findings contributes to the advancement of cybersecurity measures in OT environments, ensuring enhanced protection against malicious threats.

REFRENCES

[1] Stouffer, K., Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., ... & Thompson, M. (2023). Guide to operational technology (ot) security (p. 9). US Department of Commerce, National Institute of Standards and Technology.

[2] Murray, G., Peacock, M., Rabadia, P., & Kerai, P. (2018). Detection techniques in operational technology infrastructure.

[3] Khadpe, M., Binnar, P., & Kazi, F. (2020, July). Malware injection in operational technology networks. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

[4] Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., & Stoddart, K. (2016). A review of cyber security risk assessment methods for SCADA systems. Computers & security, 56, 1-27.

[5] Mansfield-Devine, S. (2019). The state of operational technology security. Network security, 2019(10), 9-13.

[6] Schultz, M. G., Eskin, E., Zadok, F., & Stolfo, S. J. (2000, May). Data mining methods for detection of new malicious executables. In Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001 (pp. 38-49). IEEE.

[7] Fürnkranz, J., Gamberger, D., Lavrač, N., Fürnkranz, J., Gamberger, D., & Lavrač, N. (2012). Rule learning in a nutshell. Foundations of Rule Learning, 19-55.

[8] Kolter, J. Z., & Maloof, M. A. (2004, August). Learning to detect malicious executables in the wild. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 470-478).

[9] Karim, M. E., Walenstein, A., Lakhotia, A., & Parida, L. (2005). Malware phylogeny generation using permutations of code. Journal in Computer Virology, 1(1-2), 13-23.

[10] Henchiri, O., & Japkowicz, N. (2006, December). A feature selection and evaluation scheme for computer virus detection. In Sixth International Conference on Data Mining (ICDM'06) (pp. 891-895). IEEE.

[11] Bilar, D. (2007). Opcodes as predictor for malware. International journal of electronic security and digital forensics, 1(2), 156-168.

[12] Moskovitch, R., Elovici, Y., & Rokach, L. (2008). Detection of unknown computer worms based on behavioral classification of the host. Computational Statistics & Data Analysis, 52(9), 4544-4566.

[13] Moskovitch, R., Feher, C., Tzachar, N., Berger, E., Gitelman, M., Dolev, S., & Elovici, Y. (2008). Unknown malcode detection using opcode representation. In Intelligence and Security Informatics: First European Conference, EuroISI 2008, Esbjerg, Denmark, December 3-5, 2008. Proceedings (pp. 204-215). Springer Berlin Heidelberg.

[14] Ye, Y., Wang, D., Li, T., Ye, D., & Jiang, Q. (2008). An intelligent PE-malware detection system based on association mining. Journal in computer virology, 4, 323-334.

[15] Tian, R., Batten, L. M., & Versteeg, S. C. (2008, October). Function length as a tool for malware classification. In 2008 3rd international conference on malicious and unwanted software (MALWARE) (pp. 69-76). IEEE.

[16] Siddiqui, M., Wang, M. C., & Lee, J. (2009). Detecting internet worms using data mining techniques. Journal of Systemics, Cybernetics and Informatics, 6(6), 48-53.

[17] Tabish, S. M., Shafiq, M. Z., & Farooq, M. (2009, June). Malware detection using statistical analysis of byte-level file content. In Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics (pp. 23-31).

[18] Mehdi, S. B., Tanwani, A. K., & Farooq, M. (2009, July). Imad: in-execution malware analysis and detection. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (pp. 1553-1560).

[19] Mehdi, B., Ahmed, F., Khayyam, S. A., & Farooq, M. (2010, May). Towards a theory of generalizing system call representation for in-execution malware detection. In 2010 IEEE international conference on communications (pp. 1-5). IEEE.

[20] Santos, I., Nieves, J., & Bringas, P. G. (2011). Semi-supervised learning for unknown malware detection. In International Symposium on Distributed Computing and Artificial Intelligence (pp. 415-422). Springer Berlin Heidelberg.

[21] webmaster@vxheaven.org. Viruses don't harm, ignorance does. http://vx.netlux.org (2017)

[22] Liangboonprakong, C., & Sornil, O. (2013, June). Classification of malware families based on n-grams sequential pattern features. In 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA) (pp. 777-782). IEEE.

[23] Santos, I., Brezo, F., Ugarte-Pedrero, X., & Bringas, P. G. (2013). Opcode sequences as representation of executables for data-mining-based unknown malware detection. information Sciences, 231, 64-82.

[24] Salehi, Z., Sami, A., & Ghiasi, M. (2014). Using feature generation from API calls for malware detection. Computer Fraud & Security, 2014(9), 9-18.

[25] Kuriakose, J., & Vinod, P. (2015). Unknown metamorphic malware detection: Modelling with fewer relevant features and robust feature selection techniques. IAENG International Journal of Computer Science, 42(2), 139-151.

[26] Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M., & Giacinto, G. (2016, March). Novel feature extraction, selection and fusion for effective malware family classification. In Proceedings of the sixth ACM conference on data and application security and privacy (pp. 183-194).

[27] Sharma, A., & Sahay, S. K. (2018). An investigation of the classifiers to detect android malicious apps. In Information and Communication Technology: Proceedings of ICICT 2016 (pp. 207-217). Springer Singapore.

[28] Xu, Z., Ray, S., Subramanyan, P., & Malik, S. (2017, March). Malware detection using machine learning based analysis of virtual memory access patterns. In Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017 (pp. 169-174). IEEE.

[29] Kotov, V., & Wojnowicz, M. (2018). Towards generic deobfuscation of windows API calls. arXiv preprint arXiv:1802.04466.

[30] Burnap, P., French, R., Turner, F., & Jones, K. (2018). Malware classification using self organising feature maps and machine activity data. computers & security, 73, 399-410.

[31] Li, Z. Q., Qiao, Y. C., Hasan, T., & Jiang, Q. S. (2018, January). A similar module extraction approach for android malware. In Proceedings of the 2018 International Conference on Modeling, Simulation and Optimization (MSO 2018), Shenzhen, China (pp. 21-22).

[32] Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A review of android malware detection approaches based on machine learning. IEEE access, 8, 124579-124607.

[33] Kouliaridis, V., & Kambourakis, G. (2021). A comprehensive survey on machine learning techniques for android malware detection. Information, 12(5), 185.

[34] Tyagi, S., Baghela, A., Dar, K. M., Patel, A., Kothari, S., & Bhosale, S. (2023, February). Malware Detection in PE files using Machine Learning. In 2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON) (pp. 1-6). IEEE.

[35] Akhtar, M. S., & Feng, T. (2023). Evaluation of machine learning algorithms for malware detection. Sensors, 23(2), 946.