Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.048183

ARTICLE

Enhancing PDF Malware Detection through Logistic Model Trees

Muhammad Binsawad*

Department of Information Systems, King Abdulaziz University, P.O. Box 80217, Jeddah, 21589, Saudi Arabia

*Corresponding Author: Muhammad Binsawad. Email: mbinsawad@kau.edu.sa

Received: 30 November 2023 Accepted: 22 January 2024 Published: 26 March 2024

ABSTRACT

Malware is an ever-present and dynamic threat to networks and computer systems in cybersecurity, and because of its complexity and evasiveness, it is challenging to identify using traditional signature-based detection approaches. The study article discusses the growing danger to cybersecurity that malware hidden in PDF files poses, highlighting the shortcomings of conventional detection techniques and the difficulties presented by adversarial methodologies. The article presents a new method that improves PDF virus detection by using document analysis and a Logistic Model Tree. Using a dataset from the Canadian Institute for Cybersecurity, a comparative analysis is carried out with well-known machine learning models, such as Credal Decision Tree, Naïve Bayes, Average One Dependency Estimator, Locally Weighted Learning, and Stochastic Gradient Descent. Beyond traditional structural and JavaScript-centric PDF analysis, the research makes a substantial contribution to the area by boosting precision and resilience in malware detection. The use of Logistic Model Tree, a thorough feature selection approach, and increased focus on PDF file attributes all contribute to the efficiency of PDF virus detection. The paper emphasizes Logistic Model Tree's critical role in tackling increasing cybersecurity threats and proposes a viable answer to practical issues in the sector. The results reveal that the Logistic Model Tree is superior, with improved accuracy of 97.46% when compared to benchmark models, demonstrating its usefulness in addressing the ever-changing threat landscape.

KEYWORDS

Malware detection; PDF files; logistic model tree; feature selection; cybersecurity

1 Introduction

Malware is a type of malicious code that can affect networks and computer systems. It has become more common in recent years, making signature-based detection techniques useless [1,2]. Advanced malware has become increasingly complicated and adaptable over the past ten years, making it difficult to categorize and identify since it may intelligently conceal or change its code and behavior, making outmoded detection and classification approaches less effective [3]. As a result, the emphasis now is on using machine learning to identify and classify malware more accurately.

Cybercriminals frequently utilize infected PDF documents as one of their methods [4]. However, separating malicious PDFs from huge PDF files is a challenge that makes forensic analysis of these documents more complex. The complexity of machine learning has allowed for the detection of malicious PDFs, which helps forensic investigators and improves system security [5]. However, advances in adversarial strategies to get around hostile document classifiers have been made. In particular, adversarial examples based on precision manipulation are carefully designed to cause misclassifications, which might pose a danger to many machine learning-based detectors [6]. Although several analysis and detection techniques have been put out to thwart these assaults, the threat posed by adversarial attacks still needs to be properly addressed.

In academic and industrial contexts, machine learning (ML) based solutions are becoming more and more common. One area of specialization is the detection of malware concealed in infection vectors, including malicious PDF files [7]. Research has demonstrated that learning-based systems are capable of detecting obfuscated assaults, which frequently avoid being detected by basic heuristics. It is crucial to remember that even while the number of assaults discovered has significantly increased, there has been doubt about how reliable learning algorithms are against deliberately designed adversarial attacks [8].

Comparable ideas have been used to create adversarial malware samples, as was first shown in [9]. These assaults entail precisely calibrated, tiny alterations to malicious samples that have been accurately recognized, leading to their misidentification as legitimate. As a result, it is now possible to get around machine learning-based detection more covertly without using intrusive changes like code obfuscation.

In this research, we offer a novel detection method that can distinguish between malicious and benign PDF files using document analysis. The Logistic Model Tree (LMT) with ideal hyperparameters is used by the suggested system. The LMT is employed on a PDF malware dataset obtained from the Canadian Institute for Cybersecurity (CIC)-Evasive-PDFMal2022[1]. The projected model is compared with state-of-the-art ML models including Locally Weighted Learning (LWL), Credal Decision Tree (CDT), Naïve Bayes (NB), Average One Dependency Estimator (A1DE), and Stochastic Gradient Descent (SGD). These models are compared based on standard assessment measures which are F1-Score, precision, recall, MCC, and accuracy.

The major contributions of this study in the field of PDF malware detection are multifaceted. To begin, by presenting the LMT, which combines decision trees and logistic regression to successfully model PDF malware detection, this work presents an innovative and robust technique. This novel approach can handle a wide range of feature types, including missing values, numeric, nominal, binary, and multi-class target variables. Furthermore, this study employs four separate approaches for feature selection, boosting the possibility of uncovering key properties for effective identification. The use of a consistent dataset from the CIC improves the study's dependability and repeatability. Finally, this work emphasizes visible resilience qualities in PDFs, which is a novel and insightful divergence from prior literature's main focus on structural aspects or JavaScript code within PDFs. In conclusion, this work significantly improves the accuracy and resilience of PDF malware detection by adding a novel model and a comprehensive feature selection approach, all while increasing awareness of the importance of PDF file attributes in the detection process.

In contrast to previous research, this study adopts a comprehensive approach that not only acknowledges the prevalence of malware in PDF documents but also systematically addresses the challenges posed by adversarial attacks on machine learning-based classifiers. The novel and robust model introduced in this study, the LMT, demonstrates the versatility of LMT in handling various feature types, such as missing values, numeric, nominal, binary, and multi-class variables. It is noteworthy that the emphasis has shifted from structural elements or JavaScript code to attributes of apparent resilience

[1]https://www.unb.ca/cic/datasets/pdfmal-2022.html.

in PDFs, offering a new angle on the detection procedure. To sum up, this work is noteworthy for its novel technique, which highlights the significance of model robustness and provides a thorough and perceptive approach to improving the precision and robustness of PDF malware detection. This study is important in the real world as it addresses the growing cybersecurity issues related to malicious PDF files. With the increasing sophistication of cyber threats, particularly via damaged PDF documents, the suggested LMT model offers a workable solution by utilizing machine learning to increase detection precision. The objectives of cybersecurity stakeholders are aligned with the emphasis on resilience characteristics, forensic analysis improvement, and dataset dependability. All things considered, this work develops machine learning-based methods for detecting PDF malware, tackling important security issues for businesses that depend on safe online communication and document exchange.

The rest of this paper is organized as: Section 2 presents the literature study. Section 3 describes the research design and procedure, Section 4 presents the result analysis and discussion, and finally, Section 5 concludes this study.

2 Literature Study

Numerous research studies have been conducted to investigate the detection of PDF malware using various machine learning (ML) and deep learning (DL) models. Reum et al. [10] provided a detailed examination of JavaScript content and structure within XML-embedded PDFs, elucidating the use of the Portable Document Format. They created a wide range of configuration and metadata features, such as file size, keywords, versions, and content features, as well as encoding schemes, such as keywords, names, and JavaScript-readable strings. The intricacy of these properties, along with the resistance of machine learning algorithms to slight changes, makes creating hostile samples difficult. To reduce the danger of adversarial assaults, they created a recognition model that uses black-box-style models that take into account both structural and content aspects.

A comparative study of machine learning methods for identifying malware in PDF files is carried out by B. Khan and colleagues [6]. K-Nearest Neighbor (KNN) beats other models with an amazing 99.8599% accuracy in 10-fold cross-validation, according to the study, which uses a dataset from the Canadian Institute for Cybersecurity. The research provides important insights for the creation of reliable cybersecurity solutions and highlights the efficacy of ML in precisely detecting and combating PDF malware. An innovative deep learning-based malware detection method that combines the benefits of static and dynamic analysis was presented in a different work by Shaukat et al. [8]. The strategy surpasses state-of-the-art algorithms by 16.56%, achieving 99.06% accuracy on the Malimg dataset. It addresses the problems of unbalanced data and provides the defense sector with a scalable, affordable answer. Using deep learning and feature selection approaches, Alomari et al. [11] provided a high-performance malware detection system that tackles the problems posed by huge and high-dimensional data. The work uses two different malware datasets and correlation-based feature selection to achieve significant feature dimension reductions without sacrificing performance. The suggested technique is effective in preserving detection accuracy, as evidenced by the various reduction ratios of 18.18% to 42.42% for the first dataset and 81.77% to 93.5% for the second dataset. The results show varying performance variations.

ML approaches have been used in several projects to create classifiers for PDF malware. Previous attempts, such as Wepawet [12] and PJScan [13], focused on harmful JavaScript hidden in PDF malware, offering classifiers for recognizing malicious JavaScript and tools for extracting its code. Because not all PDF malware has embedded JavaScript and malware authors use a variety of strategies to conceal JavaScript scripts, more recent PDF malware classifiers have turned their attention to the structural elements of PDF files [14]. The goal of this project is to create sophisticated classifiers using structural characteristics.

Chen et al. [15] demonstrated how to use observable robustness characteristics to build robust PDF malware classifiers. For example, regardless of how many innocuous pages are added to the document, their technique assures that a classifier consistently detects PDF malware as a danger. They provided a thorough evaluation of a malware classifier's worst-case behavior in terms of particular resilience features.

Smutz et al. [16] used information collected from document metadata and file structure to detect counterfeit PDF files using a PDF parser and a random forest classifier. They looked at 202 options, including /Font and /JavaScript. However, according to Liu et al. [17], existing safeguards against malicious PDFs are weak, prone to evasion, and computationally costly for online use. They suggested using both static and runtime characteristics to recognize JavaScript in context. A software engineering method was used to detect fraudulent documents, concentrating on behavioral discrepancies across different systems since a PDF document acts similarly across all platforms, but a malicious one displays platform-specific variances [18].

Previous research has focused on the analysis of JavaScript code in PDFs. Khitan et al. [19] defined attributes of JavaScript scripts based on their functions, constants, objects, methods, keywords, and lexical characteristics. Zhang [20] used PDF structural components, entity characteristics, metadata information, content statistics, and JavaScript features like the number of objects, pages, and stream filtering information. Liu et al. [17] proposed a context-aware method based on the fact that malicious JavaScript functions behave differently from lawful ones. By supplying the original code as input to the "eval" function, this approach analyzes PDF files while detecting suspicious activity.

The limitations of current harmful detection techniques that depend on external extraction tools that follow the Acrobat standard for JavaScript extraction were highlighted by Li et al. [21]. Furthermore, Scofield et al. [22] emphasized the paucity of studies identifying the minimal dataset size necessary for reaching high detection accuracy and contended that growing the training dataset does not always result in better detection. Consequently, a dynamic analysis-based detection approach was presented in their study.

2.1 How This Study Is Different from Other Studies?

This study differentiates significantly from the previous literature on PDF virus detection by taking a holistic approach. It employs four separate feature selection methods: Greedy Stepwise, Best First, Evolutionary, and Particle Swarm Optimization, displaying a rigorous technique to ensure feature identification. Furthermore, the use of a dataset obtained from the Canadian Institute for Cybersecurity strengthens the research's legitimacy and usefulness. This methodology is strengthened by an emphasis on observable robustness features, which provide constant malware detection regardless of harmless material. Specifically, this study proposes an LMT for PDF malware detection, which combines the benefits of decision trees with logistic regression. Finally, the thorough assessment technique used in this study, which includes several measures such as F1-Score, accuracy, precision, MCC, and recall, ensures a full knowledge of the model's performance. This work makes a unique and useful contribution to the field of PDF virus detection by offering this diversified and thorough methodology.

3 Research Design and Procedure

This study aims to propose an LMT-based model for PDF malware detection. Fig. 1 presents the entire study approach. First, a dataset was obtained from the Canadian Institute for Cybersecurity (UNB CIC) at the University of New Brunswick. The dataset may be accessed at https://www.unb.ca/cic/datasets/pdfmal-2022.html. First, missing values are handled, then, features that have a high degree of connection between them and include a significant amount of information are found and kept using feature selection procedures. To guarantee robustness and generalization, several ML models are trained via a 10-fold cross-validation method after feature selection. Next, these models' performance is assessed using recognized evaluation standards, such as F1-Score, accuracy, precision, MCC, and recall. Using a PC with a Core i5 CPU and 12 GB of RAM, the experiments are carried out on a Windows 10 operating system environment, guaranteeing that the computing capabilities are sufficient for the job at hand.

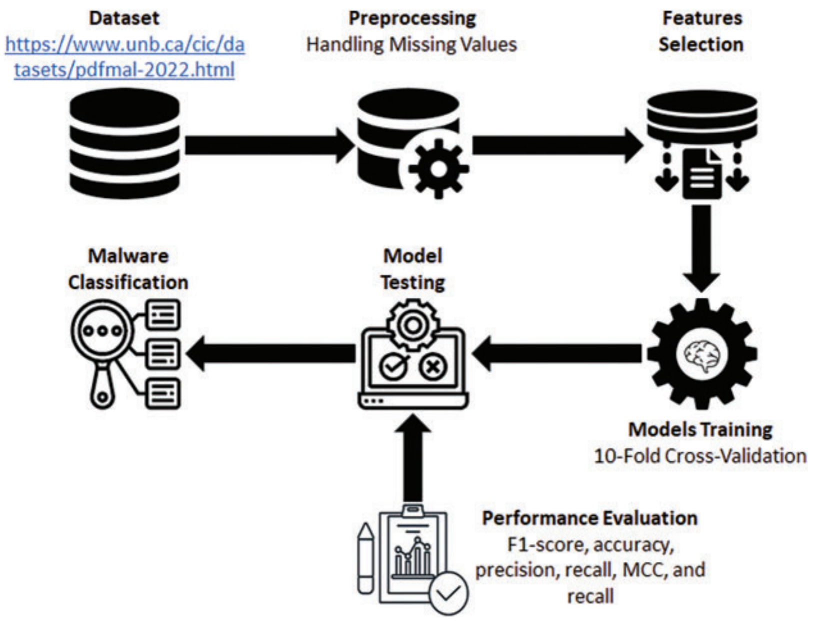


Figure 1: Workflow for PDF malware detection methodology with feature selection and model training

3.1 Dataset Acquisition and Preprocessing

The dataset used in this study has been taken from the Canadian Institute for Cybersecurity (UNB CIC) at the University of New Brunswick available at: https://www.unb.ca/cic/datasets/pdfmal-2022.html. The dataset consists of 10017 instances and initially 31 attributes. Nonetheless, it is important to recognize certain inherent limits in the dataset. There might be issues with the dataset's representativeness, which could affect how well the model applies to a range of real-world situations. The temporal element of the dataset prompts questions about how well it matches current malware patterns, which might restrict the model's capacity to identify new and developing threats. Furthermore, the robustness and generalizability of the machine learning model may be impacted by the dataset's size, makeup, and the imputation of missing information. Even though the suggested LMT-based model shows encouraging results in the dataset, these issues must be resolved if the model is to be used and reliable in actual malware detection situations. Firstly, the preprocessing step is applied to the dataset to handle all the missing values using the mean method. The steps for handling missing values are presented in Algorithm 1.

Algorithm 1: Replacing Missing Values with Mean

```
Function ImputeMissingWithMean(dataset):
    For each feature (column) in dataset:
        If the feature has missing values:
            // Calculate the mean of the non-missing values in the feature
            $mean = \frac{\sum available\ values}{count\ of\ available\ values}$
            For each row in the feature:
                If the value is missing (e.g., marked as NA or null):
                    // Replace the missing value with the calculated mean
                    Set value to mean
    Return the dataset with missing values imputed with means
```

The procedures to impute missing values with the mean for every feature in the dataset are described in this pseudocode. The process iterates over every feature, finds the mean of the non-missing values, and substitutes the mean for the missing values.

3.2 Feature Selection

The feature selection process in the context of PDF malware detection was tackled methodically using four various strategies, including Greedy Stepwise, Best First, Evolutionary, and Particle Swarm Optimization (PSO) searching techniques. The following are the precise values of the control parameters used in the research study for the PSO and Evolutionary searches: PSO and the evolutionary algorithm have 50 and 100 population sizes, respectively. Ten independent runs for PSO and twenty for the Evolutionary algorithm are described. Furthermore, for the PSO and Evolutionary algorithms, the number of iterations each run is fixed at 100 and 200, respectively. Throughout the study's trials, these parameter values were used in the corresponding algorithms. Notably, all four techniques consistently converged on the same set of characteristics, highlighting the resilience and dependability of the found features for PDF virus detection. The selected features are addressed in Table 1.

Table 1: Description of each selected feature

Attributes	Position	Description
Metadata size	3	The metadata area contains information about the PDF file that can be used to incorporate hidden material.
Images	9	A PDF file may include one image or many images.
Text	10	Malicious PDF files may include less text because their goal is not to convey material.
Trailer	17	How many trailers are included in the PDF?
Startxref	18	The number of keywords that include "startxref," which indicates the beginning of the Xref table.
JS	22	The number of objects with Javascript code.

(Continued)

Table 1 (continued)

Attributes	Position	Description
Javascript	23	This indicates how many objects include Javascript code, which is the characteristic that is most frequently exploited, as is clear.
OpenAction	25	Specifies what should happen when the PDF file is opened. Javascript and this functionality together have been found in most common malicious PDF files.

a. Greedy Stepwise Search: The Greedy Stepwise feature selection technique is a heuristic method for choosing a subset of features from a larger feature set [23]. It operates by adding or eliminating features repeatedly depending on a certain assessment criterion. In your case, the purpose is to choose a subset of characteristics from the UNB CIC PDF malware collection. The Greedy Stepwise feature selection method is presented in Algorithm 2.

Algorithm 2: Function Greedy Stepwise Features Selection

```
Initialize an empty set of selected features: S = {}
Initialize the best performance metric value: bestPerformance = 0
While (size(S) < m) AND (bestPerformance improvement is significant OR size(S) < stoppingCriterion):
    For each feature "i" not in S:
        // Evaluate the performance with the feature "i" added to the selected set
        newPerformance = EvaluateModel(dataset, S + i, evaluationMetric)
        // If the new performance is better, update the best feature and performance
        If newPerformance > bestPerformance:
            bestFeature = i
            bestPerformance = newPerformance
       // Add the best feature to the selected set
       S = S + {bestFeature}
    // Optional: Perform feature removal based on post-selection analysis
Return the selected feature set S
```

This pseudocode repeatedly adds the feature to the selected set S that produces the greatest substantial improvement in the evaluation measure. When either the required amount of features (stoppingCriterion) or no meaningful performance gain can be obtained by adding further features, the process terminates.

b. Best First Search: The Best First feature selection technique, also known as Best First Search, is a feature selection heuristic search algorithm [24]. It investigates feature subsets by deciding on the optimal feature to add at each stage based on a predetermined assessment criterion. Algorithm 3 describes the step-by-step process of Best First search for feature selection.

Algorithm 3: Best First Features Selection

```
Initialize an empty set of selected features: S = {}
Initialize the best performance metric value: bestPerformance = 0
While (size(S) < m) AND (bestPerformance improvement is significant):
    For each feature "i" not in S:
        // Evaluate the performance with the feature "i" added to the selected set
        newPerformance = EvaluateModel(dataset, S + i, evaluationMetric)
        // If the new performance is better, update the best feature and performance
        If newPerformance > bestPerformance:
            bestFeature = i
            bestPerformance = newPerformance
    // Add the best feature to the selected set
    S = S + {bestFeature}
  // Optional: Perform feature removal based on post-selection analysis
Return the selected feature set S
```

The method in this pseudocode iteratively picks the feature that delivers the greatest substantial improvement in the evaluation measure and adds it to the chosen collection S. When either the required amount of features (stoppingCriterion) or no meaningful performance gain can be obtained by adding further features, the algorithm terminates.

c. Evolutionary Search: Evolutionary search is a heuristic optimization strategy that looks for the best solutions within a wide solution space by simulating the process of natural evolution [25]. When utilized in malware detection feature selection. The step-by-step process of evolutionary search is described in Algorithm 4.

Algorithm 4: Evolutionary Search for Feature Selection

```
Initialize population:
For each individual in the population:
    Create a random binary string where each bit represents a feature (0: exclude, 1: include)
Define the fitness function:
For each individual in the population:
    Train a classifier using the selected features
    Evaluate the classifier's performance on a validation dataset
    Set the individual's fitness score based on the classifier's performance
Main loop:
Repeat for a specified number of generations or until convergence criteria are met:
    Select parents for the next generation:
    For each pair of parents:
        Use tournament selection, roulette wheel selection, or other selection methods to choose two individuals based on their fitness scores
    Perform crossover:
    For each pair of parents:
        Perform one-point or two-point crossover to create two children by swapping bits between the parents
```

(Continued)

Algorithm 4 (continued)

```
    Perform mutation:
    For each child:
        With a certain probability, flip some randomly selected bits in the child's feature string
    Evaluate the fitness of the new individuals:
    For each child:
        Train a classifier using the selected features
        Evaluate the classifier's performance on a validation dataset
        Set the child's fitness score based on the classifier's performance
    Select survivors:
    Combine the current population and the new children
    Select the top individuals based on their fitness scores to form the next generation's population
Return the best individual in the final population based on their fitness score
```

A basic evolutionary method for feature selection in a dataset is described in this algorithm. The fitness function, crossover, and mutation techniques would need to be modified depending on your unique dataset and machine learning model, and the hyperparameters would need to be adjusted.

d. Particle Swarm Optimization: PSO is an algorithm for population-based optimization that imitates fish or bird social behavior. Regarding the feature selection process [26,27]. The overall process is discussed through Algorithm 5.

Algorithm 5: Particle Swarm Optimization for Features Selection

```
Initialize PSO parameters:
- population_size
- max_iterations
- inertia_weight
- cognitive_weight
- social_weight
Initialize particle swarm:
For each particle in the population:
    Initialize random binary feature subset
    Initialize random velocity for each feature
    Set the particle's best-known position to the initial random subset
    Evaluate the fitness of the particle's subset and set it as the best-known fitness
    Update the global best position if this particle's subset is the best so far
Main PSO loop:
For iteration = 1 to max_iterations:
    For each particle in the population:
        For each feature in the particle's subset:
            Update the velocity of the feature:
            velocity = inertia_weight * current_velocity
                + cognitive_weight * random_value() * (best_known_position_for_particle - current_position) + social_weight * random_value() * (global_best_position - current_position)
            Update the position of the feature:
            current_position = current_position + velocity
```

(Continued)

Algorithm 5 (continued)

```
        Evaluate the fitness of the particle's new subset:
        Set the particle's best-known position to the new subset if it has better fitness
        Update the global best position if this particle's subset is the best so far
Return the global best position as the selected feature subset
```

For PDF malware detection, this technique performs feature selection using PSO. PSO parameters, such as population size, the maximum number of iterations, and weights for inertia, cognitive, and social components, are first initialized. Next, a population of particles, each representing a binary feature subset, is created using the method. Based on the global best-known position and the particles' past performance, iteratively updates the positions and velocities of the particles. A classifier is used to assess each particle's feature subset's fitness. After a predetermined number of rounds, the particles adjust their subgroups as the algorithm progresses. Ultimately, the outcome is returned as the global best position, which represents the feature subset that was chosen.

3.3 Model Training and Performance Evaluation

Data splitting for model training is an important part of any ML-based research study. To this end, this study uses the 10-fold cross-validation to split the data for training and testing purposes. By testing the model on several data subsets, it reduces the possibility of overfitting [6].

Another important aspect of any research study is to test the performance of employed models. To this end, this study focuses on standard assessment measures including F1-Score, accuracy, precision, MCC, and recall [28–30]. These measures can be calculated as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{F1} - \text{Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

$$\text{MCC} = \frac{(\text{TN} * \text{TP}) - (\text{FN} * \text{FP})}{\sqrt{(\text{FP} + \text{TP})(\text{FN} + \text{TP})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \tag{4}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5}$$

where the rate of true negative values (TN), true positive values (TP), false positive values (FP), and false negative values (FN) are expressed by the equations.

3.4 Logistic Model Tree

LMT is an ML technique that combines decision trees and logistic regression to generate prediction models that are particularly successful for binary outcomes such as PDF virus detection. It builds "logistic model trees," with the leaves representing logistic regression functions. LMT can handle missing values, numeric and nominal features, as well as binary or multi-class target variables, making it useful for a wide range of jobs [29,31]. The LMT was chosen over other viable models in our research due to its numerous features, which collectively position it as the best option for

PDF virus detection. LMT's unique combination of decision trees and logistic regression gives it the capacity to capture complicated data connections inside the PDF malware dataset. This property improves the model's ability to recognize subtle linkages and patterns, which contributes to its improved accuracy in classification tasks. Furthermore, the adaptability of LMT is demonstrated by its strong handling of the many feature types included in our dataset, which includes missing values, and nominal, numeric, and binary variables. The model's resilience to missing data is very important for real-world applications in situations where datasets may contain partial information, ensuring that LMT retains its performance integrity. Importantly, the logistic regression functions at the tree's leaves highlight LMT's applicability for binary outcome tasks such as malware detection, allowing probabilistic predictions and improving result interpretability. The experimental findings confirm LMT's constant outperformance over benchmark models in accuracy, recall, F1-Score, and Matthews Correlation Coefficient, highlighting its ability to properly detect both harmful and benign occurrences while minimizing false positives and false negatives. Furthermore, the use of four separate feature selection methodologies in tandem with LMT demonstrates a painstaking consideration for effective identification, cementing its position as an innovative and resilient solution for accurate PDF malware detection. The step-by-step process of LMT for PDF malware detection is presented through Algorithm 6.

Algorithm 6: LMT_PDF_Malware_Detection

```
Input: Labeled dataset with features (PDF_files, Labels)
Output: LMT model for PDF malware detection
Function BuildLMT(dataset):
    if StoppingCriteria(dataset) then
        return CreateLeafNode(dataset)
    else
        feature, threshold = ChooseFeatureAndThreshold(dataset)
        left_data, right_data = SplitDataset(dataset, feature, threshold)
        left_subtree = BuildLMT(left_data)
        right_subtree = BuildLMT(right_data)
        return CreateInternalNode(feature, threshold, left_subtree, right_subtree)
Function CreateLeafNode(dataset):
   # Logistic Regression Model
   model = TrainLogisticRegression(dataset)
   return model
Function StoppingCriteria(dataset):
   # Check if stopping criteria are met (e.g., max depth or min samples)
   return ...
Function ChooseFeatureAndThreshold(dataset):
   # Select feature and threshold for data splitting
   return feature, threshold
Function SplitDataset(dataset, feature, threshold):
   # Split dataset into left and right branches
   left_data = [data for data in dataset if data[feature] <= threshold]
   right_data = [data for data in dataset if data[feature] > threshold]
   return left_data, right_data
```

(Continued)

Algorithm 6 (continued)

```
    Function TrainLogisticRegression(dataset):
      # Train a logistic regression model using dataset
      model = LogisticRegressionTrain(dataset)
      return model
    LMT_model = BuildLMT(PDF_files)
    return LMT_model
# Logistic Regression Function
Function LogisticRegressionTrain(dataset):
    Initialize coefficients (β0, β1, β2, ..., βn) with zeros
    Initialize learning rate and maximum iterations
    repeat for each iteration until convergence:
        for each data point in the dataset:
          Calculate logistic function using coefficients and features
          Compute the gradient of the log-likelihood
          Update coefficients using gradient and learning rate
    return coefficients
```

The algorithm describes the technical process of creating an LMT for detecting PDF malware. It starts with a labeled dataset of PDF files and their labels. The build LMT function repeatedly builds the tree, making decisions based on feature selection and data splitting and led by parameters such as halting conditions and feature threshold selection. Using a basic logistic regression training function, leaf nodes train logistic regression models to predict the likelihood of a PDF file being harmful. The Logistic Model Tree combines the advantages of decision trees with logistic regression to produce a model capable of categorizing PDF files as benign or malicious, taking into account extracted attributes and maximizing the chance of accurate predictions.

3.5 Other Employed Models

The main aim of this study is to propose LMT for PDF malware detection. However, the proposed model is compared with some standard ML models that are referenced in Table 2.

Table 2: List of ML models employed in this study as benchmark compared with LMT

S. no.	Model	References
1	Locally weighted learning (LWL)	[32]
2	Credal decision tree (CDT)	[28,33]
3	Naïve Bayes (NB)	[34,35]
4	Average one dependency estimator (A1DE)	[28,30]
5	Stochastic gradient descent (SGD)	[36,37]

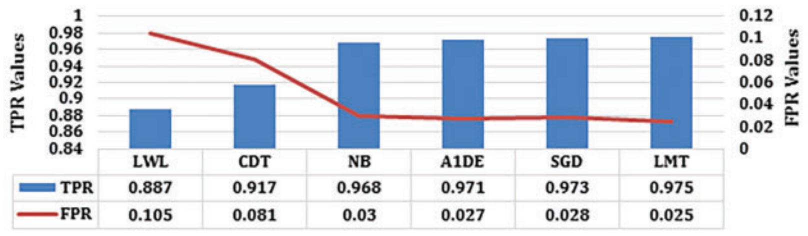
4 Results Analysis and Discussion

This section discusses the overall outcomes of the study. A new LMT-based model is presented for PDF malware detection. LMT and other benchmarked models are evaluated using standard assessment measures which are accuracy, recall, precision, F1-Score, and MCC. The outcome for each measure is obtained using the confusion matrix, which is a table used in machine learning to evaluate the effectiveness of a classification model. It summarizes the numbers of TP, TN, FP, and FN. Table 3 summarizes the classification performance of the used ML models in the context of PDF malware detection, with "Malicious" and "Benign" as the two classes. The number of cases identified as "Malicious" or "Benign" by the corresponding model is represented by each cell in the table. Notably, the table shows that, while the LWL and CDT models perform similarly in correctly classifying both "Malicious" and "Benign" instances, the NB and A1DE models perform better in correctly classifying "Malicious" instances with fewer false negatives, but with a slightly higher number of false positives for "Benign" instances. The SGD and LMT models, on the other hand, emphasize accuracy, as indicated by low false positive counts for "Benign" occurrences and a somewhat greater number of false negatives for "Malicious" examples. These contrasts emphasize the trade-offs between accuracy and recall, emphasizing the importance of model selection in PDF virus detection settings.

Table 3: Confusion matrix values achieved through each employed model

Model	Class	Malicious	Benign
LWL	Malicious	4747	804
	Benign	329	4137
CDT	Malicious	5049	502
	Benign	328	4138
NB	Malicious	5322	229
	Benign	95	4371
A1DE	Malicious	5332	219
	Benign	74	4392
SGD	Malicious	5410	141
	Benign	133	4333
LMT	Malicious	5408	143
	Benign	111	4355

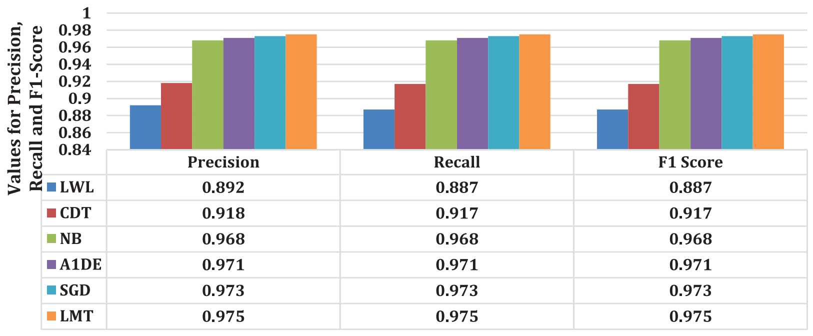
Fig. 2 depicts the TPR and FPR values for several ML models used to identify PDF malware. TPR is the percentage of real "Malicious" occurrences accurately categorized as such, reflecting the model's ability to detect true positive cases. Notably, all models have high TPR values, with the LMT model having the highest (0.975), demonstrating their efficiency in detecting fraudulent PDF files. FPR, on the other hand, represents the proportion of genuine "Benign" cases that were wrongly labeled as "Malicious," emphasizing the incidence of false alarms. The models, particularly the A1DE, SGD, and LMT, have low FPR values (0.027, 0.028, and 0.025, respectively), demonstrating their ability to reduce false positives, which is important in the context of PDF malware detection to minimize excessive security warnings. The results show that these models have a fair mix of sensitivity and specificity, making them useful tools for robust virus detection in PDF files.



	LWL	CDT	NB	A1DE	SGD	LMT
TPR	0.887	0.917	0.968	0.971	0.973	0.975
FPR	0.105	0.081	0.03	0.027	0.028	0.025

Figure 2: True positive rates (TPR) and false positive rates (FPR) of PDF malware detection models

Fig. 3 shows precision, recall, and F1-Score values for various ML models used to identify PDF malware. Notably, the LMT performs well, with a precision of 0.975, recall of 0.975, and F1-Score of 0.975. This outstanding performance can be ascribed to the distinct qualities of LMT. LMT combines decision trees and logistic regression to efficiently capture extensive data interactions. Furthermore, the flexibility of LMT to handle missing values, nominal and numeric properties, and binary or multi-class target variables makes it an excellent candidate for detecting PDF malware. Because of these characteristics, LMT outperforms other models in terms of precision and recall, making it the best choice for effectively recognizing fraudulent PDFs.

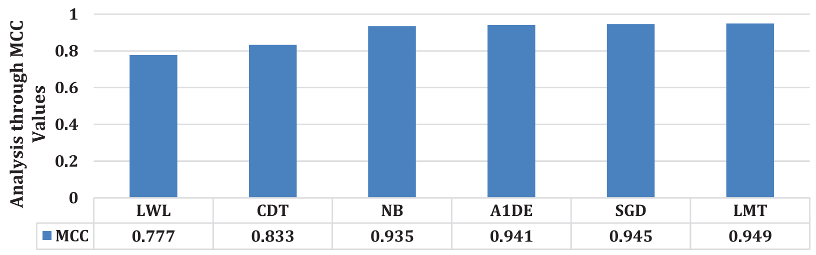


	Precision	Recall	F1 Score
LWL	0.892	0.887	0.887
CDT	0.918	0.917	0.917
NB	0.968	0.968	0.968
A1DE	0.971	0.971	0.971
SGD	0.973	0.973	0.973
LMT	0.975	0.975	0.975

Figure 3: Precision, recall, and F1-Score of PDF malware detection models

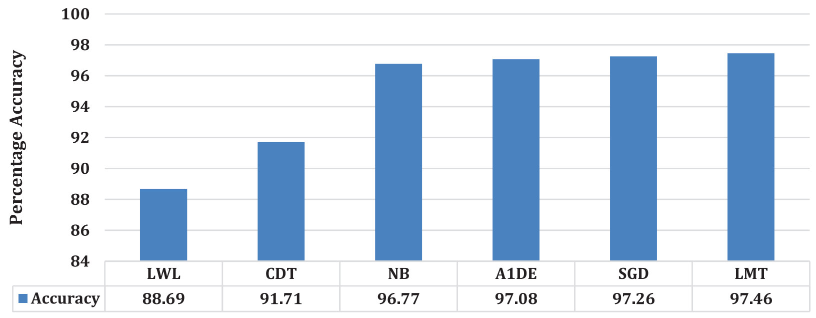
A detailed summary of the MCC values for the employed ML models used in PDF malware detection is shown in Fig. 4. With the highest MCC score of 0.949 among these models, the LMT stands out as having an excellent capacity to distinguish between malicious and benign PDF files. Since the MCC metric takes into account both true and false positives as well as negatives, it is a very useful measure of a model's prediction ability when working with unbalanced datasets. The better MCC score of LMT can be attributed to its improved handling of missing data, intricate feature interactions, and the combination of logistic regression and decision tree approaches. These findings highlight the applicability of LMT in big data analysis for sound decision-making in complicated commercial marketplaces as well as its adaptability for precise risk prediction in the context of software needs.

Fig. 5 depicts the accuracy of the models used in the detection of PDF malware. Among the models, the LMT performs best, with an accuracy of 97.46%, suggesting its ability to reliably categorize PDF files as harmful or benign. LWL, on the other hand, has the lowest accuracy of 88.69%, indicating relatively poor performance. The accuracy disparities can be ascribed to the LMT's unique technique of merging decision trees with logistic regression, which allows it to capture complicated relationships within the dataset, whereas LWL appears to struggle in successfully distinguishing between malicious and benign samples. Furthermore, the performance of LMT is enhanced by its capacity to handle missing values, different feature types, and binary or multi-class target variables, making it a more robust alternative for PDF malware detection than other models. These findings emphasize the significance of model selection and point to LMT as a possible method for improving the accuracy of PDF virus detection.



	LWL	CDT	NB	A1DE	SGD	LMT
■ MCC	0.777	0.833	0.935	0.941	0.945	0.949

Figure 4: Matthews correlation coefficient (MCC) scores for machine learning models in PDF malware detection



	LWL	CDT	NB	A1DE	SGD	LMT
■ Accuracy	88.69	91.71	96.77	97.08	97.26	97.46

Figure 5: Accuracy comparison of machine learning models for PDF malware detection

The study presents a novel LMT-based model for PDF malware detection and performs a thorough comparison analysis against well-known benchmark models using industry-standard assessment criteria including as accuracy, precision, recall, F1-Score, and MCC. The findings clarify the models' ability to classify across "Malicious" and "Benign" classes, and they are displayed in a comprehensive confusion matrix (Table 3). LMT demonstrates a high level of competence in distinguishing between the two groups. Analyzing TPR and FPR values (Fig. 2) reveals that all models have persistently high TPR values, while LMT is the best at minimizing FPR. The remarkable performance of LMT is highlighted by its recall, precision, and F1-Score values (Fig. 3), which may be attributed to its skill in managing a variety of data sources and identifying complex connections. The accuracy of LMT in differentiating between malicious and benign PDF files is confirmed by the MCC scores (Fig. 4). Notably, the accuracy visualization (Fig. 5) emphasizes the resilience of LMT in PDF malware detection by showcasing its pinnacle accuracy of 97.46%. Because of its unique features, especially the way it incorporates logistic regression and decision trees, and because it can work with a variety of data types, LMT is a viable option for improving the detection accuracy of PDF viruses. These results highlight how important careful model selection is, thus proving that LMT is a powerful method for accurate PDF malware detection in the field of large-scale data analysis, which is essential for well-informed decision-making in complex business environments.

However, there are certain drawbacks to this study report. To begin, the suggested LMT is largely examined in the context of PDF malware detection, which may limit its application to more generic malware detection tasks. Furthermore, the benchmark models in the study may not cover the complete range of existing ML models, potentially leaving out more sophisticated or specialized methods. While the research focuses on hostile assaults, it does not cover the entire spectrum of potential antagonistic techniques, potentially leaving vulnerabilities unmet. Finally, while the study focuses on feature selection approaches and metrics, it may ignore other crucial issues such as computing resource needs and model training timeframes, which might be critical in real-time detection settings.

4.1 Ablation Study for PDF Malware Detection

In this technical ablation study for PDF malware detection, we carefully evaluate the different components and methodologies that support the research. We investigate the impact of several feature selection algorithms on overall model performance, including Greedy Stepwise Search, Best First Search, Evolutionary Search, and PSO. We identify and quantify the unique impact of each benchmark model, such as LWL, CDT, NB, A1DE, and SGD. We investigate the importance of dataset provenance and size, looking at how these affect model efficacies. Furthermore, we examine the model's resilience features and its capacity to survive adversarial assaults, as well as its interpretability and robustness in the face of class imbalance. Finally, we investigate the model's scalability in real-world scenarios. The thorough examination of these components and approaches elucidates their unique contributions to PDF malware detection as well as their overall influence on the research's conclusion.

5 Conclusion

In the ever-changing world of cybersecurity, combating malware has become increasingly difficult, needing creative defense tactics. Malware has gotten increasingly complex over time, with the capacity to adapt, disguise, or alter its code and behavior, leaving traditional signature-based detection approaches ineffective. This work focused on the specific problem of malware hidden within PDF documents, a popular channel for thieves, and addressed adversarial tactics that aim to defeat ML-based detectors. We wanted to improve the accuracy and robustness of malicious PDF file identification by developing a unique technique based on the LMT and applying document analysis. This study thoroughly compared the performance of LMT to that of existing machine learning models such as LWL, CDT, NB, A1DE, and SGD, and assessed them using standard metrics. This study produced significant advances, such as an original model, extensive feature selection methods, the utilization of consistent datasets, and an emphasis on PDF file properties. Finally, our findings show that the LMT model is capable of differentiating between malicious and benign PDF files. LMT outperforms other models in precision, recall, F1-Score, MCC, and accuracy, achieving 97.5% precision, recall, F1-Score, MCC, and correctness. These findings highlight the importance of new models like LMT in efficiently combatting malware threats, improving the accuracy and resilience of PDF malware detection, and bolstering the cybersecurity community's skills in the face of emerging threats.

5.1 Future Work

Future objectives for this research include investigating advanced ML and DL algorithms to improve the accuracy of PDF virus detection. Combining static analysis with behavioral and anomaly detection technologies may boost system efficacy even more. Countering adversarial attacks on ML models customized for PDF virus detection is critical. Collaboration with industry professionals and threat intelligence groups will improve system capabilities by providing real-time threat data and insights. Furthermore, continual research into novel evasion tactics and attack channels is required to keep up with increasing cybersecurity threats.

Acknowledgement: This research work was funded by Institutional Fund Projects. The authors gratefully acknowledge technical and financial support provided by the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

Funding Statement: This research work was funded by Institutional Fund Projects under Grant No. (IFPIP: 211-611-1443). The authors gratefully acknowledge technical and financial support provided by the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

Author Contributions: The author confirms contribution to the paper as follows: study conception and design, data collection, analysis and interpretation of results, and draft manuscript preparation were performed by Muhammad Binsawad. The author reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data used in this study is publically available at: https://www.unb.ca/cic/datasets/pdfmal-2022.html.

Conflicts of Interest: The author declares that he has no conflict of interest to report regarding the present study.

References

[1] B. Ndibanje, K. Kim, Y. Kang, H. Kim, T. Kim and H. Lee, "Cross-method-based analysis and classification of malicious behavior by API calls extraction," *Appl. Sci*, vol. 9, no. 2, pp. 239, 2019. doi: 10.3390/app9020239.

[2] Q. A. Al-Haija, A. A. Badawi, and G. R. Bojja, "Boost-defence for resilient IoT networks: A head-to-toe approach," *Expert Syst.*, vol. 39, no. 10, pp. e12934, 2022. doi: 10.1111/exsy.12934.

[3] M. J. H. Faruk *et al.*, "Malware detection and prevention using artificial intelligence techniques," in *2021 IEEE Int. Conf. Big Data (Big Data)*, Orlando, FL, USA, 2021, pp. 5369–5377.

[4] H. Ghanei, F. Manavi, and A. Hamzeh, "A novel method for malware detection based on hardware events using deep neural networks," *J. Comput. Virol. Hacking Tech.*, vol. 17, no. 4, pp. 319–331, 2021. doi: 10.1007/s11416-021-00386-y.

[5] A. Gaurav, B. B. Gupta, and P. K. Panigrahi, "A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system," *Enterp. Inf. Syst.*, vol. 17, no. 3, pp. 2023764, 2023. doi: 10.1080/17517575.2021.2023764.

[6] B. Khan, M. Arshad, and S. S. Khan, "Comparative analysis of machine learning models for PDF malware detection: Evaluating different training and testing criteria," *J. Cybersecur.*, vol. 5, pp. 1–11, 2023. doi: 10.32604/jcs.2023.042501.

[7] X. Ling *et al.*, "Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art," *Comput. Secur.*, vol. 128, no. 3, pp. 103134, 2023. doi: 10.1016/j.cose.2023.103134.

[8] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Eng. Appl. Artif. Intel.*, vol. 122, no. 4, pp. 106030, 2023. doi: 10.1016/j.engappai.2023.106030.

[9] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Mach. Learn. Know. Discov. Databases: Eur. Conf., ECML PKDD 2013*, Prague, Czech Republic, 2013, pp. 387–402.

[10] A. R. Kang, Y. S. Jeong, S. L. Kim, and J. Woo, "Malicious PDF detection model against adversarial attack built from benign PDF containing javascript," *Appl. Sci.*, vol. 9, no. 22, pp. 4764, 2019. doi: 10.3390/app9224764.

[11] E. S. Alomari *et al.*, "Malware detection using deep learning and correlation-based feature selection," *Symmetry*, vol. 15, no. 1, pp. 123, 2023. doi: 10.3390/sym15010123.

[12] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," in *Proc. 19th Int. Conf. World Wide Web*, Raleigh, North Carolina, USA, 2010, pp. 281–290.

[13] P. Laskov and N. Šrndić, "Static detection of malicious JavaScript-bearing PDF documents," in *Proc. 27th Annu. Comput. Secur. Appl. Conf.*, Orlando, Florida, USA, 2011, pp. 373–382.

[14] B. W. Langdon and M. Harman, "Optimizing existing software with genetic programming," *IEEE. Trans. Evolut. Comput.*, vol. 19, no. 1, pp. 118–135, 2014. doi: 10.1109/TEVC.2013.2281544.

[15] Y. Chen, S. Wang, D. She, and S. Jana, "On training robust {PDF} malware classifiers," in *29th USENIX Secur. Symp. (USENIX Security 20)*, Boston, MA, USA, 2020, pp. 2343–2360.

[16] C. Smutz and A. Stavrou, "Malicious PDF detection using metadata and structural features," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, Orlando, Florida, USA, 2012, pp. 239–248.

[17] D. Liu, H. Wang, and A. Stavrou, "Detecting malicious javascript in pdf through document instrumentation," in *2014 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Atlanta, GA, USA, 2014, pp. 100–111.

[18] M. Xu and T. Kim, "{PlatPal}: Detecting malicious documents with platform diversity," in *26th USENIX Secur. Symp. (USENIX Security 17)*, Vancouver, BC, Canada, 2017, pp. 271–287.

[19] S. J. Khitan, A. Hadi, and J. Atoum, "PDF forensic analysis system using YARA," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 5, pp. 77–85, 2017.

[20] J. Zhang, "MLPdf: An effective machine learning based approach for PDF malware detection," arXiv preprint arXiv:1808.06991, 2018.

[21] M. Li, Y. Liu, M. Yu, G. Li, Y. Wang and C. Liu, "FEPDF: A robust feature extractor for malicious PDF detection," in *2017 IEEE Trustcom/BigDataSE/ICESS*, Sydney, NSW, Australia, 2017, pp. 218–224.

[22] D. Scofield, C. Miles, and S. Kuhn, "Fast model learning for the detection of malicious digital documents," in *Proc. 7th Softw. Secur., Prot., Rev. Eng./Softw. Secur. Prot. Workshop*, Orlando, FL, USA, 2017, pp. 1–8.

[23] F. Asdaghi and A. Soleimani, "An effective feature selection method for web spam detection," *Knowl.-Based Syst.*, vol. 166, pp. 198–206, 2019. doi: 10.1016/j.knosys.2018.12.026.

[24] A. O. Balogun, A. O. Bajeh, V. A. Orie, and A. W. Yusuf-asaju, "Software defect prediction using ensemble learning: An ANP based evaluation method," *J. Eng. Technol.*, vol. 3, no. 2, pp. 50–55, 2018. doi: 10.46792/fuoyejet.v3i2.200.

[25] M. Gambhir and V. Gupta, "Recent automatic text summarization techniques: A survey," *Artif. Intell. Rev.*, vol. 47, no. 1, pp. 1–66, 2017. doi: 10.1007/s10462-016-9475-9.

[26] Q. Bai, "Analysis of particle swarm optimization algorithm," *Computer and Information Science*, vol. 3, no. 1, pp. 180–184, 2010. doi: 10.5539/cis.v3n1p180.

[27] H. Jiang, Z. He, G. Ye, and H. Zhang, "Network intrusion detection based on PSO-Xgboost model," *IEEE Access*, vol. 8, pp. 58392–58401, 2020. doi: 10.1109/ACCESS.2020.2982418.

[28] B. Khan *et al.*, "Software defect prediction for healthcare big data: An empirical evaluation of machine learning techniques," *J. Healthc. Eng.*, vol. 2021, no. 2, pp. 1–16, 2021. doi: 10.1155/2021/8899263.

[29] R. Naseem *et al.*, "Investigating tree family machine learning techniques for a predictive system to unveil software defects," *Complex.*, vol. 2020, no. 6, pp. 1–21, 2020. doi: 10.1155/2020/6688075.

[30] R. Naseem *et al.*, "Performance assessment of classification algorithms on early detection of liver syndrome," *J. Healthc. Eng.*, vol. 2020, no. 4, pp. 1–13, 2020. doi: 10.1155/2020/6680002.

[31] S. Sahu, "Network intrusion detection system using J48 decision tree," in *2015 Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Kochi, India, 2015, pp. 2023–2026.

[32] L. Jiang, Z. Cai, H. Zhang, and D. Wang, "Naive bayes text classifiers: A locally weighted learning approach," *J. Exp. Theor. Artif. In.*, vol. 25, no. 2, pp. 273–286, 2013. doi: 10.1080/0952813X.2012.721010.

[33] Q. He *et al.*, "Novel entropy and rotation forest-based credal decision tree classifier for landslide susceptibility modeling," *Entropy.*, vol. 21, no. 2, pp. 106, 2019. doi: 10.3390/e21020106.

[34] M. M. Saritas, “Performance analysis of ANN and naive bayes classification algorithm for data classification,” *Int. J. Intell. Syst. Appl. Eng.*, vol. 7, no. 2, pp. 88–91, 201. doi: 10.18201/ijisae.2019252786.

[35] M. Panda, “Combining multi-objective evolutionary algorithm with averaged one-dependence estimators for big data analytics,” *Int. J. Comput. Intell. Stud.*, vol. 7, no. 1, pp. 1–18, 2018. doi: 10.1504/IJCISTUD-IES.2018.090160.

[36] S. Wojtowytsch, “Stochastic gradient descent with noise of machine learning type Part I: Discrete time analysis,” *J. Nonlinear. Sci.*, vol. 33, no. 3, pp. 45–52, 2023. doi: 10.1007/s00332-023-09903-3.

[37] A. A. Syed, F. L. Gaol, and T. Matsuo, “A survey of the state-of-the-art models in neural abstractive text summarization,” *IEEE Access*, vol. 9, pp. 13248–13265, 2021. doi: 10.1109/ACCESS.2021.3052783.