IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2023.0322000*

# PDF Malware Detection: Towards Machine Learning Modeling with Explainability Analysis

**G.M. Sakhawat Hossain[1,2], Kaushik Deb[1], Helge Janicke[3,4] and Iqbal H. Sarker[3,4]**

[1]Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chattogram, 4349, Bangladesh
[2]Department of Computer Science and Engineering, Rangamati Science and Technology University, Chattogram, 4500, Bangladesh
[3]Cyber Security Cooperative Research Centre, Australia
[4]Security Research Institute, School of Science, Edith Cowan University, Perth, WA-6027, Australia

Correspondence: debkaushik99@cuet.ac.bd, m.sarker@ecu.edu.au

**ABSTRACT** The Portable Document Format (PDF) is one of the most widely used file types, thus fraudsters insert harmful code into victims' PDF documents to compromise their equipment. Conventional solutions and identification techniques are often insufficient and may only partially prevent PDF malware because of their versatile character and excessive dependence on a certain typical feature set. The primary goal of this work is to detect PDF malware efficiently in order to alleviate the current difficulties. To accomplish the goal, we first develop a comprehensive dataset of 15958 PDF samples taking into account the non-malevolent, malicious, and evasive behaviors of the PDF samples. Using three well-known PDF analysis tools (PDFiD, PDFINFO, and PDF-PARSER), we extract significant characteristics from the PDF samples of our newly created dataset. In addition, we generate a number of derivations of features that have been experimentally proven to be helpful in classifying PDF malware. We develop a method to build an efficient and explicable feature set through the proper empirical analysis of the extracted and derived features. We explore different baseline machine learning classifiers and demonstrate an accuracy improvement of approx. 2% for the Random Forest classifier utilizing the selected feature set. Furthermore, we demonstrate the model's explainability by creating a decision tree that generates rules for human interpretation. Eventually, we make a comparison with previous studies and point out some important findings.

**INDEX TERMS** Cybersecurity, PDF Malware, Data Analytics, Machine Learning, Decision Rule, explainable AI, Human Interpretation.

## I. INTRODUCTION

In today's digital world, the majority of our tasks are associated with the use of the global web, making it increasingly essential to protect our data, information, and applications, in the face of a variety of cyber criminals who continually attempt to construct brand-new illicit programs and strikes to harm the facilities [1]. Despite ever-increasing security improvements over time, PDF remains a favorite breach vector for adversaries to distribute malware and launch their attack activities [2]. There are several possible damaging acts perpetrated by PDF malware, including the creation of backdoors, password theft, spyware deployment, internet browser compromise, data spilling, social engineering, and scams. Therefore, one of the biggest hurdles in the modern world is the identification of PDF malware because attackers generate many kinds of such malware and additionally its traits are changing swiftly on a daily basis. There are primarily two methods for identifying malware: behavior-based detection and signature-based detection. The features of the underlying object are used to establish a unique signature in the signature-based approach. The method effectively detects the existence of a digital signature by inspecting the object. On the other hand, using machine intelligence and other techniques, the behavior-based strategy can recognize unidentified and sophisticated malware to some extent, although it is a complex process.

The fundamental structure of a PDF that contains header, body, xref (cross reference) table, and trailer is illustrated in Fig. 1 [3]. The PDF's header indicates which version of the parser format will be used. Text blocks, typefaces, file-specific metadata, and images are all included in the PDF's body, which also specifies its content [4]. There are four categories into which the contents of PDF can be placed: numbers, strings, streams, and booleans [5]. Each item in the PDF file has an entry in the cross-reference table that details its byte offset or placement in the file as well as enables speedy random access to particular objects, facilitating effective document exploration and content retrieval. A PDF

reader or parser may traverse and access the different items within the file by using the trailer, which gives them all the necessary information such as PDF size, root object, metadata info, encryption info, and unique identifier of the PDF file. PDF malware can usually be created by injecting malicious content or programs into the elements of the fundamental structure of a PDF.

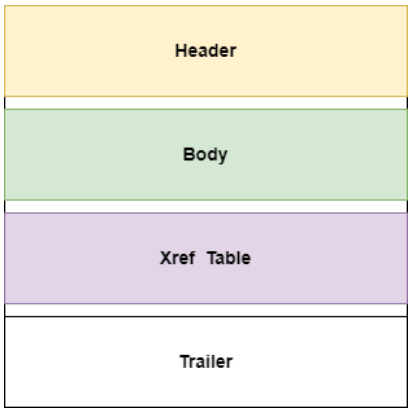


**FIGURE 1. A sample structure of PDF.**

PDF malware can be analyzed using a static, dynamic, or hybrid approach [6]. The static technique inspects malware refraining from executing the program that it embeds, but the dynamic method inspects malware by executing its code [7], [8]. Static analysis becomes susceptible when extensive evasion and fraudulent methods are used to disguise harmful execution behavior. In the present cybersecurity circumstances, depending solely on static inspection is often inadequate since a perpetrator who is dedicated to their attack would disguise and encode their code, making it normally invisible to static inspection [4]. Dynamic techniques, on the contrary, are more resilient to code deception, causing them to be a better defense against advanced viruses [9]. Dynamic analysis is often slow and challenging, but static analysis tends to be fast. Integrating the two approaches results in hybrid analysis, which is more effective in combating advanced malware than either method alone but additionally consumes a longer period and necessitates an additional complex analysis method [10].

Current malware identification methods frequently choose feature sets according to findings from a manual inspection of harmful PDF files and are guided by the expertise of the specialist. The chosen features, nevertheless, are occasionally exclusive for fraudulent files, luring adversaries to possibly gain authority over how and what a malicious file looks like, and evading the current detectors (while preserving their malicious properties). For instance, the Mimicry [11] and Reverse Mimicry [12] incidents have been exacerbated by the observation that the program builders infrequently disclose comprehensive information about the measures adopted to maintain their integrity and resistance to risks. In addition, the data that is accessible to developers, such as clean and harmful samples, datasets, vulnerabilities, payloads, and attack vectors employed within, also constrains their work. Such situations result in the produced solutions becoming outdated considerably earlier than what the diligent developers had planned.

Machine learning applications have advanced to the point where they can now protect systems from threats or aid forensic professionals in their investigations by spotting likely malicious PDF files [13]. However, adversarial techniques have grown capable of compromising threat document analyzers. Numerous machine-learning-based detection tools are at risk because their identification of well-crafted evasive scenarios may be erroneous [14], [15]. Various evaluations or detection methods have been created to detect specific incidents, but the immediate threat posed by evasive attacks has not yet been mitigated.

Developing feature engineering improvements integrating the adversarial behaviors of malicious PDFs for creating harmful PDF classifiers is challenging, yet necessary, and has an opportunity to have a significant impact in the field. We look at ways to improve the identification approach for PDF malware by 1) introducing an inclusive dataset that contains evasive characteristics of suspicious PDFs along with clean and harmful PDF samples, 2) extracting the features of the PDF samples and 3) merging the most significant features to develop an effective feature set that can be fed into a classifier to produce a higher level of accuracy. We provide a comprehensive analysis of the most significant features identified for PDF malware detection and interpret the classifier's performance. In summary, our contributions can be outlined as follows:

- We have developed a comprehensive dataset that consists of a total of 15958 PDF samples including 7500 clean PDFs, 7666 malicious PDFs, and 792 evasive PDFs by considering the non-malicious, malicious, and evasive natures of the PDF samples. For this, we use three popular PDF analysis tools viz. PDFiD [16], PDFINFO [17], and PDF-PARSER [18].
- We develop a method to build an explicable feature set by taking into account the feature's characteristics and importance score.
- We design an architecture for malicious PDF detection and explored different machine learning classifiers to analyze and compare their efficacy in different cases.
- We have demonstrated the model's explainability by creating a decision tree that generates rules for human interpretation.
- We have conducted a wide range of experimental analysis and compare our results to previous studies. We also highlight some key observation of our study.

The rest of the paper is organized as follows: Section II provides an in-depth and organized overview of current research in the same field of study. Section III describes the rec-

ommended approach for PDF malware detection. Section IV presents and evaluates our findings, as well as the experimental outcomes. Section V provides an in-depth discussion while pointing out a few observations. Finally, closing remarks are offered in Section VI.

## II. RELATED WORKS

In recent years, PDFs have been widely used to disseminate malicious documents and malware. To mitigate the subsequent and crucial growth of malicious PDF developments, numerous effective studies on detecting and categorizing technologies for malware and other dangerous files were established [19]. The tools that have been designed throughout the past years range greatly from being general and straightforward to specific and complex. Certain techniques try to find differences by scanning the whole file [20]. An additional kind of technique searches an intended file for resemblance to typical trends found in harmful PDF files [21]–[25]. Another set of tools concentrated on extracting, analyzing, and identifying attack methods, for instance, detecting JavaScript-based attacks [26]–[32]. Most of these approaches are heavily reliant on machine learning methods, including one- and two-class Support Vector Machines, Random Forests, and decision trees.

The study in [33] focused on developing an approach to recognize a group of features derived through currently available tools as well as generated a new group of features aimed at improving PDF maldoc identification and prolonging the useful life of current analysis and detection techniques. The importance of the produced features was assessed using a wrapper function that leveraged three key supervised learning methods as well as a feed-forward deep neural network. Subsequently, a novel classifier that significantly improved classification efficacy with shorter training times was constructed deploying features of the highest significance. With the use of huge datasets from VirusTotal [34] the findings were verified.

From top to bottom, authors in [35] looked into PDF design and JavaScript content contained in PDFs. They developed a wide range of features for design and metadata, including the number of bytes per second, the encoding method, catchphrases, object names, and intelligible strings in JavaScript. Additionally, since subtle changes have a significant impact on AI calculations, it is challenging to develop hostile models when the attributes vary. To reduce the risk of malicious attacks while maintaining structures and data properties, they developed a classification model using discovery-type models. They created an adversarial attack in order to accept the suggested paradigm. An outline of the PDF was provided in [36], and contemporary attacks on PDF malware were carried out using reliable attack models obtained from nature. They gave an example of how to use programming skills to perform a quantitative analysis of a PDF file to look for signs of contained malware. They looked at some of the emerging AI-powered tools for detecting PDF malware which may assist computational scientific analyses and can flag questionable documents before a more thorough, more conclusive statistical analysis is published. They looked at the PDF restrictions alongside various unresolved problems, especially how their flaws might be used to potentially misdirect measured investigations. Finally, they offered advice on how to make those structures more effective in withstanding attacks and sketched a possible assessment.

Obfuscation strategies used by PDF maldoc authors were noted by the study in [37]; these techniques hinder automated evaluation and identification methods and make manual analysis more difficult. This involves exploiting PDF filters, comments, and white space to spread harmful code across numerous objects. Other strategies include gathering around strewn harmful code fragments throughout the page using a "Names" dictionary. Furthermore, hazardous substances can be concealed in odd places like document metadata or the fields (comments) of annotations. Moreover, memory spraying and the use of shellcodes to download malicious files or documents were included in the study of [37] for the classification of PDF-based attacks as JavaScript code exploits.

Because a PDF document acts identically on several devices, the authors in [38] developed a detection method based on behavioral inconsistencies on those platforms using a software engineering idea. On the other hand, a malicious document will behave differently depending on the platform. The study in [39] emphasized malware inserted into PDF files as a representative example of contemporary cyberattacks. They began by classifying the various production processes for PDF malware scientifically. They used a proven adversarial AI framework to counter PDF malware detectors that rely on learning. This strategy, for instance, made it possible to discover existing faults in learning-oriented PDF malware trackers as well as novel threats that may threaten such architectures, as well as the likelihood of protective actions.

In [40], the authors outlined an innovative approach to detect data problems of an ensemble classifier. The ensemble classifier's prediction was shown to be false when enough individual classifier votes clashed during detection. The recommended method, ensemble classifier consensus evaluation, facilitated the findings of various sorts of system evasions without the necessity for additional external ground truth. The authors tested the suggested approach using PDFrate, a PDF malware detector, and revealed that a significant number of assumptions could be derived utilizing improved ensemble classifier concordance using the entire network's data.

The authors of [25] demonstrated how the least optimistic case behavior of a malware detector in terms of specified intensity features could be examined. Additionally, they discovered that creating classifiers with legally verified efficient

features may raise the expense of avoiding unrestrained attackers by simply skipping over simple assault avoidance techniques. They put forth an alternative distance measure that relies on the tree structure of PDF and identified two groups of strong features, such as erasures and subtree inclusions.

In [32], the researchers presented Lux0R, further referred to as "Lux 0n discriminant References," a novel and adaptable approach for detecting malicious code in JavaScript. The recommended strategy hinged on describing code in JavaScript using API references, which contained elements that a JavaScript Application Programming Interface (API) can intuitively comprehend such as objects, constants, functions, attributes, methods, and keywords. To isolate suggestive risky code of a certain subgroup from API references, the proposed approach made use of machine learning which was subsequently used to spot JavaScript malware. The important application domain that the author focused on in this work was the detection of potentially harmful JavaScript code in PDF files. The weaknesses within existent extractors of features for PDFs were uncovered by the authors of [41] by evaluating them alongside analyzing how the framework of the fraudulent documents was set up. The researchers subsequently developed FEPDF (feature extractor-PDF), a sophisticated feature extractor, that was capable of discovering characteristics that traditional extraction methods could lose and recorded accurate data concerning the PDF components. To investigate the most recent antivirus frameworks along with pattern extractors, the authors created numerous fresh harmful PDFs as samples. The results indicate that a number of existing antivirus applications were unable to identify the fresh dangerous PDFs, however, FEPDF was able to retrieve the essential components for improved dangerous PDF classification.

In [42], an integrated detection technique was suggested to track the JavaScript code's runtime behavior along with the recognition of features related to obfuscation, which included concealing certain keywords' presence with ASCII hexadecimal when several compression filters were used and the existence of any void objects. The research in [43] was based on the odd disparities between harmful and clean document construction. The authors adopted the tools that extracted the feature set utilizing the document hierarchy or structure path. A tree was constructed from the document hierarchy and on the basis of the presence of specific paths the harmful and clean files were identified, according to the authors. However, to combat the existing threats caused by PDF malware as well as to mitigate the challenges posed by the evasive behavior of PDFs, we certainly require an effective classifier that works with an explainable feature set covering the wide range of behaviors of PDFs. In this research, we developed a dataset that covers the characteristics of clean and harmful PDFs along with a limited introduction to the evasive behaviors of PDFs. Moreover, we identified an explainable feature set by extracting useful features from the PDFs by adopting three well-known PDF analysis tools. Furthermore, we identified an effective machine learning classifier that leverages the newly developed feature set to detect PDF malware with improved accuracy. Finally, we provided a thorough explanation of the performance of the classifier by describing a decision tree built from one of the estimators of the classifier and extracting a few crucial decision rules for detecting PDF malware effectively. In the subsequent section, the details of the methodology used in this research will be discussed thoroughly.

## III. METHODOLOGY

PDF files are among the most extensively used file types in the world. However, hackers can utilize PDF files, which are usually non-threatening, to introduce security dangers via malicious code, just as they can with PNG files, dot-com files, and Bitcoin [4]. As a result, PDF malware appears, demanding techniques for recognizing malicious from benign files. This section discusses the proposed detection system for analyzing and categorizing PDF files as benign or malicious. Fig.2 represents the inclusive graphical architecture of the proposed approach utilized to conduct this research. In our proposed approach, initially, we accumulate 29901 raw PDF samples from [44] which are originally picked from Contagio Data Dump [45] and VirusTotal [34]. Then, PDF samples are divided into Benign, Malicious, and Evasive categories according to their preassigned label as mentioned in [44], [46]. Then, we choose 15958 PDF samples of Benign, Malicious, and Evasive categories from the 29901 raw samples and develop a comprehensive dataset for our experimental study. We utilize three up-to-date PDF analysis tools viz. PDFiD [16], PDFINFO [17], and PDF-PARSER [18] to extract effective standard feature set $F_1$, $F_2$, and $F_3$ respectively from the raw PDF samples of our experimental dataset. In addition to the standard feature set, seven more features are derived by carefully observing the characteristics of PDF samples from the feature set $F_1$. The standard feature sets $F_1$, $F_2$, and $F_3$ are then implemented in the model selection phase, which includes a set of baseline machine learning classifiers. The model selection phase determines the best model amongst the baseline classifiers employed in this study based on their effectiveness for each feature set.

The extracted derived features are then merged with the standard feature set $F_1$, $F_2$, and $F_3$ respectively, to develop the derived feature set $F_1'$, $F_2'$, and $F_3'$ correspondingly. Later, we aim to generate feature subsets from $F_1'$, $F_2'$, and $F_3'$ based on their importance and employ them in the best-performing model to determine the most effective feature subsets. Finally, we execute a union operation on the three best subsets acquired from $F_1'$, $F_2'$, and $F_3'$ to construct the final feature set used for malicious PDF detection. The performance of our proposed approach is measured using various performance metrics such as precision, recall, f1-measure, and accuracy.

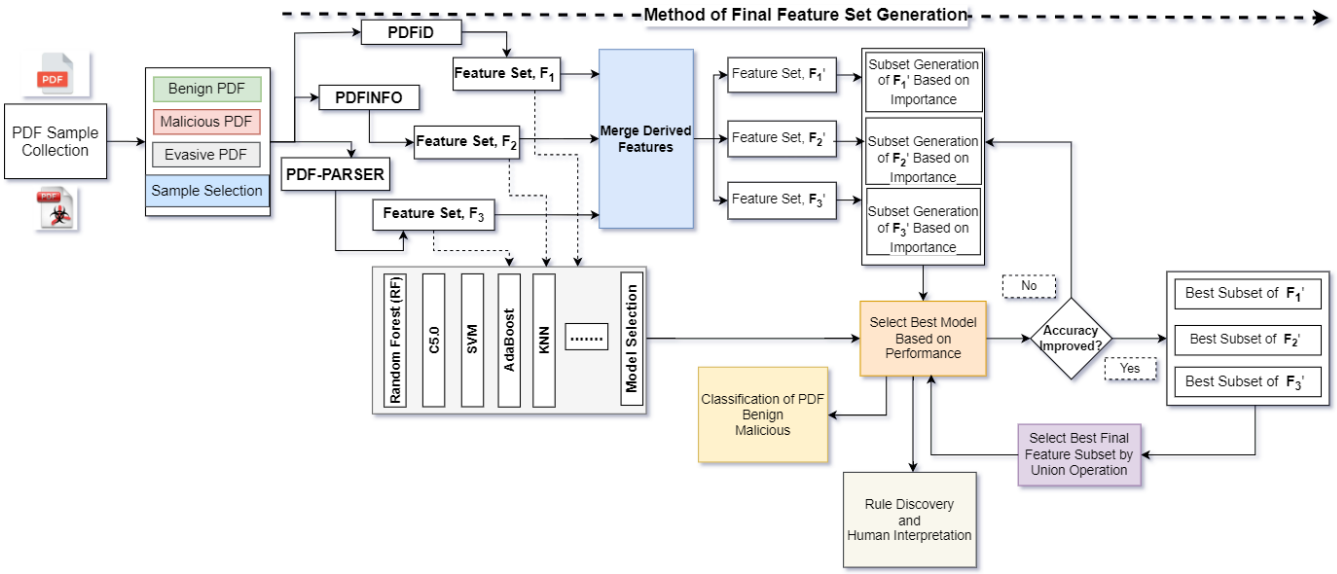


**FIGURE 2.** **Proposed architecture for malicious PDF detection.**

Furthermore, we highlight the impact of the final feature set and how much it contributes to the classification activities. In addition, leveraging the strength of the best-performing model, we offer an explanation to make it more humanly understandable by extracting some important decision rules responsible for the classification activities. The details of our proposed methodology for malicious PDF detection are described below in the following subsections.

### *A. DATASET DEVELOPMENT*

Existing datasets may not represent the entire range of harmful PDFs. Creating a new dataset enables us to include a broader range of samples, capturing adversaries' crafting approaches and strategies. The world of cybersecurity is continuously changing, and attackers are constantly devising new evasion strategies. A fresh dataset enables us to capture novel circumstances that may not have been present in previous datasets. By focusing on the aforementioned directions, we aim to create an all-inclusive dataset that includes not only hazardous and clean PDF samples but also a few elusive PDF samples that displayed the opposite features of their preassigned class which may assist in developing an effective malicious PDF classifier.

#### 1) PDF Sample Collection

To carry out our experiments, we gather a large corpus of raw PDF files from [44] which consists of 29901 PDFs. The original sources of the PDF files are from two well-known sites i.e. Contagio Data Dump [45] and VirusTotal [34]. Among the collected PDF files, we get 9109 Benign PDFs from Contagio Data Dump and 20000 malicious PDFs from VirusTotal. From [44], we also gather 792 evasive PDF files among which 400 are labeled as benign evasive and 392 are labeled as malicious evasive. Table 1 shows the distribution of the collected PDF files with their sources and their preassigned label.

**TABLE 1.** **Collected PDF files for the experimental study.**

| Source | Benign | Malicious |
|---|---|---|
| Contagio Data Dump | 9109 | - |
| VirusTotal | - | 20000 |
| CIC-Evasive-PDFMal2022 [44] | 400 (Evasive) | 392 (Evasive) |
| Total | 9509 | 20392 |

#### 2) PDF Sample Selection

The primary aim of our study is to develop an effective malicious PDF detector by leveraging the explainable and efficient feature set, thoughtfully extracted from the PDF files. As a result, we mainly concentrate on sample selection for building our dataset based on a few constraints. Firstly, the PDF files that preserve the pure malicious activities, for instance, JavaScript are mainly employed by the attackers in creating malicious PDF documents as an attack method [33], we select these types of PDFs exhibiting such malicious activities. Secondly, we consider the PDF files that pose opposite behavior than the malicious ones, for example, legitimate PDF files usually do not contain JavaScript-related features that can possibly damage the user's systems, though it is very much possible that a clean PDF file can be generated using the nonmalicious JavaScript feature. However, developing a machine learning classifier solely based on both of these categories can lead to an overfitted model for malicious PDF identification. Moreover, JavaScript is not only the key characteristic that malicious PDF exhibits rather there are other triggering features. For instance, OpenAction, and Aditional Action (AA) are some of the few features that may indicate potential malicious activity of a PDF [33]. Besides, adding

**TABLE 2. Selected PDF files for our operational dataset.**

| Class | Number of PDFs |
|---|---|
| Benign (Clean) | 7500 |
| Benign Evasive | 400 |
| Total (Benign) | 7900 |
| Malicious | 7666 |
| Malicious Evasive | 392 |
| Total (Malicious) | 8058 |
| Total (Benign + Malicious) | 15958 |

too many diversities for both of these categories can potentially skew the feature weights and importance which may cause an ineffective classification accuracy for our proposed model. Considering the aforementioned reasoning, thirdly, we thoughtfully select a few PDF files that exhibit malicious behaviors but are labeled as benign (benign evasive) and the PDF files that pose the benign behavior but are labeled as malicious (malicious evasive). In this pilot experiment, we concentrate on developing an operational dataset that has a total of 15958 PDF files including 7500 benign (clean), 7666 malicious, 400 benign evasive, and 392 malicious evasive PDF files, by setting a limited scope to ensure that we get reliable findings as quickly as possible. Table 2 represents the dispersion of the selected PDF files for our experimental study.

### B. STANDARD FEATURE SET EXTRACTION

We adopt three tools viz. PDFiD, PDFINFO, and PDFPARSER, which extract features from PDF files. Although the three tools serve the same objective, the results they produce are different and can be leveraged to create three different feature sets. The feature set extracted by the tools can be categorized into the following groups:

- Content-Related Features: Content-related features obtained from the PDF file yield clues regarding the file's textual and visual content. For instance, features like */Image*, */Font*, */ProcSet* etc. are a few examples of the content-related features, we observe in our dataset.
- Structure Related Features: The structural feature refers to the construction elements utilized to create a PDF document. This type of feature provides an internal relationship and exhibits the hierarchy among various elements of PDFs. We have considered various structural features, for instance, *obj*, *endobj*, *%EOF*, *startxref*, *trailer* etc. in our operational dataset.
- Metadata Features: Metadata features of a PDF file provide valuable information about the file itself, including its title, author, creation date, and more. In our operational dataset, we have examined metadata features such as *Filesize_kb*, */ID*, */CreationDate*, */ModDate*, *pages*, etc.
- Triggering Features: Triggering features refer to particular traits or components in a PDF file that can possibly cause various behaviors or actions, including harmful ones. Attackers might deploy these features to distribute malware, execute scripts, or perform other malicious acts. In our dataset, we carefully analyze triggering features such as */JavaScript*, */JS*, */OpenAction*, */AA (Aditional Action*, */Launch*, and so on.

#### 1) PDFiD Features

PDFiD is a Python-based tool [16] for scanning PDF documents in order to discover specific features and traits that may signal possible maliciousness. PDFiD does not run any code inside the PDF; instead, it concentrates on examining the parts and arrangement of the PDF to shed light on its characteristics. We have gone through all of our dataset's PDF files and used the PDFiD tool to extract 22 features, as shown in Fig. 3. These extracted features are considered for the standard feature set, $F_1$. In the following, we describe the features in brief:

- PDF Header: The PDF header is required for applications and software to appropriately identify and comprehend PDF documents. The "%PDF" identification is followed by a version number in the PDF header. For instance, "%PDF-1.3" denotes that the PDF file complies with PDF standard version 1.3. This code notifies applications and PDF viewers that the document is in PDF format.
- obj: PDF documents are made up of objects such as fonts, text, images, forms, etc. The term obj refers to the opening of an object definition. This feature provides the total number of obj keywords that can be identified within the PDF structure.
- endobj: The term endobj specifies the closing of the object definition. In the case of PDFiD, this feature points out how many times the endobj keyword appears inside the PDF structure.
- stream: A stream object is employed in PDF documents to hold binary data, such as fonts, images, or other binary material, within the document. This feature provides the number of stream keywords that exist within the PDF file.
- endstream: This keyword denotes the completion of the stream's binary data portion. In the context of PDFiD, this feature indicates how many endstream keywords can be found within a PDF document.
- xref: The xref (cross reference) table assists in maintaining links between the structured objects that are stored in PDF files. PDFiD provides the number of xref tables that exist inside a PDF document.
- trailer: The trailer is the final component of the PDF file and contains crucial details about the byte offset to the beginning of the cross-reference (XRef) table. In the case of PDFiD, this feature indicates how many trailers can be found within the structure of a PDF file.
- startxref: The startxref keyword designates the location where the Xref table of the PDF is started. This feature yields how many times we can find startxref keyword inside a PDF.
- /Page: This feature indicates the total number of pages of a PDF.

```
┌──(gm㉿kali)-[~/Downloads/puremal]
└─$ pdfid 02e9452cc00abd1e71bc0b0f0b9dc81e54b8c4e9aabd2af72bd0109689783c3e
PDFiD 0.2.8 02e9452cc00abd1e71bc0b0f0b9dc81e54b8c4e9aabd2af72bd0109689783c3e
 PDF Header: %PDF-1.0
 obj                    9
 endobj                 9
 stream                 2
 endstream              2
 xref                   0
 trailer                1
 startxref              0
 /Page                  1
 /Encrypt               0
 /ObjStm                0
 /JS                    2
 /JavaScript            3
 /AA                    0
 /OpenAction            0
 /AcroForm              0
 /JBIG2Decode           0
 /RichMedia             0
 /Launch                0
 /EmbeddedFile          0
 /XFA                   0
 /Colors > 2^24         0
```

**FIGURE 3.** A snapshot of the output of PDFiD scanning a PDF file.

```
┌──(gm㉿kali)-[~/Downloads/puremal]
└─$ pdfinfo 02e9452cc00abd1e71bc0b0f0b9dc81e54b8c4e9aabd2af72bd0109689783c3e
Custom Metadata: no
Metadata Stream: no
Tagged:          no
UserProperties:  no
Suspects:        no
Form:            none
JavaScript:      yes
Pages:           1
Encrypted:       no
Page size:       612 x 792 pts (letter)
Page rot:        0
File size:       1430 bytes
Optimized:       no
PDF version:     1.0
```

**FIGURE 4.** A snapshot of the output of PDFINFO scanning a PDF file.

- /Encrypt: The feature outputs the number of /Encrypt keywords present within the PDF structure.
- ObjStm: The total number of object streams is counted with /ObjStm. The ObjStm possesses the ability to hold other objects, making it useful for hiding things.
- /JS: The number of objects that contain the /JS keyword which reveals the objects having JavaScript code.
- /JavaScript: This feature demonstrates the number of objects containing JavaScript code, a common and prevalent obfuscation technique.
- /AA: This feature denotes the number of /AA (Additional Action) keywords observed inside a PDF document.
- /OpenAction: When a page or document is viewed, an automated action is indicated by the /OpenAction command. This feature demonstrates how many /OpenAction keywords a PDF document has inside its structure.
- /AcroForm: The feature denotes the number of /AcroForm keywords that exist within a PDF file. The Acrobat forms used in PDF files can be exploited by the attackers.
- /JBIG2Decode: This feature reveals the number of /JBIG2Decode keywords that exist within the structure of a PDF file. The feature explains whether the PDF uses the JBIG2 compression or not, although it does not provide any direct indication of maliciousness but requires further analysis.
- /RichMedia: The feature demonstrates the number of /RichMedia keywords that can be found within the PDF structure that provides an indication of flash files.
- /Launch: This outputs the number of /Launch keywords that exist within the PDF.
- /EmbeddedFIle: This indicates the number of /Embed-

dedFIle keywords that can be found inside the structure of a PDF.
- /XFA: Certain PDF files contain XFAs, which are XML Form architectures that offer scripting capabilities that can be abused by attackers. This feature outputs the number of /XFA keywords that can be observed inside a PDF file.
- /Colors: This feature indicates the number of different colors utilized in the PDF structure.

### 2) PDFINFO Features

PDFINFO is a command-line program that is a part of the Poppler utility suite commonly used for extracting metadata from PDF files. We extract 14 features from our operational dataset utilizing the PDFINFO tool as depicted in Fig. 4. These 14 features are considered as the standard feature set $F_2$. In the following, we describe the features of the feature set $F_2$:

- Custom Metadata: This feature indicates the presence of user-defined custom metadata inside a PDF document. The feature provides the value as 'yes' or 'no'.
- Metadata Stream: This feature outputs the presence of metadata stream within a PDF file in the form of 'yes' or 'no'.
- Tagged: This feature demonstrates whether the PDF file is tagged for accessibility or not.
- UserProperties: UserProperties are extra characteristics or data that users can add to a PDF file for a variety of functions, including document administration or private annotation. This feature reveals whether the PDF contains any UserProperties or not.
- Suspects: The feature informs whether any potential flaws or errors have been spotted in the PDF document.
- Form: This feature outputs the Form types utilized in the PDF documents. We have observed the XFA, AcroForm, or none as output from this feature.
- JavaScript: This feature informs whether the PDF file contains any JavaScript or not.
- Pages: We can observe the total number of pages that a PDF contains with the help of this feature.
- Encrypted: The feature demonstrates whether the PDF is encrypted or not.
- Page size: This feature exhibits the page dimensions of the PDF document. We have encountered PDFs with a variety of page dimensions, including A4, Letter, A3, and other uncommon page forms. If the shape of the page is peculiar, we have labeled it as miscellaneous, i.e. *Page size_miscsize*.
- Page rot: The feature provides the rotation information about the pages of the PDF document.
- File size: This feature outputs the size of the PDF file in bytes. However, for the simplicity of the experiment, we have converted the file size to kilobytes. Hence, we have denoted this feature as *Filesize_kb* throughout the study.
- Optimized: This feature informs whether the PDF document is optimized (such as size compression) or not.
- PDF version: The version of the PDF document can be observed using this feature.

### 3) PDF-PARSER Features

PDF-PARSER is a command-line program and library written in Python that parses and analyzes the internal structure of PDF documents. It is not a PDF creation or editing tool, but rather one for inspecting the internal layout and content of existing PDF files. Though PDF-PARSER does not provide features in a direct manner, we have extracted 27 features from the parsed structure of the PDF as shown in Fig. 5. These features are mainly the keywords frequently observed in the parsed structure of the PDF and are considered as the standard feature set $F_3$. Initially, we have iterated through all the PDFs of our operational dataset to extract the parsed structures. Then, we search for specific keywords i.e. features from these parsed structures, to create the feature set $F_3$. In the following, we introduce these features in brief:

- /JS: Number of /JS keywords that can be found in the parsed structure of a PDF.
- /JavaScript: Number of /JavaScript keywords that can be found in the parsed structure of a PDF.
- /Size: Number of /Size keywords that can be observed in the parsed structure of a PDF. The /Size keyword indicates the total number of objects present in the PDF document.
- startxref: Number of startxref keywords that can be found in the parsed structure of a PDF.
- %EOF: Number of %EOF keywords that can be observed in the parsed structure of a PDF. The keyword is a marker that demonstrates the end of the PDF file.
- /Producer: Number of /Producer keywords that can be spotted in the parsed structure of a PDF. The keyword specifies the tool or software by which the PDF was created.
- /ProcSet: Number of /ProcSet keywords that can be noticed in the parsed structure of a PDF. The set of procedures (or processes) that should be employed while rendering a page or graphic content within a PDF document is specified by /ProcSet. Though this keyword does not directly indicate the maliciousness of a PDF, the keyword has been frequently encountered inside the parsed structures of clean PDF files.
- /ID: Number of /ID keywords that can be discovered in the parsed structure of a PDF. This keyword reveals the document ID that is crucial for the integrity and security of the document which can indicate whether the document was tampered with malicious activity or not.
- /S: Number of /S keywords that can be spotted in the parsed structure of a PDF. The keyword indicates the subtype of various objects or tasks, such as text or link annotations.
- /CreationDate: Number of /CreationDate keywords that can be discovered in the parsed structure of a PDF.
- obj: Number of objects that can be spotted inside the parsed structure of a PDF.

```
┌──(gm㉿kali)-[~/Downloads/puremal]
└─$ pdf-parser 02e9452cc00abd1e71bc0b0f0b9dc81e54b8c4e9aabd2af72bd0109689783c3e
PDF Comment '%PDF-1.0\r\n'

obj 1 0
 Type: /Catalog
 Referencing: 2 0 R, 3 0 R

  <<
    /Type /Catalog
    /Pages 2 0 R
    /Names 3 0 R
  >>


obj 2 0
 Type: /Pages
 Referencing: 4 0 R

  <<
    /Type /Pages
    /Count 1
    /Kids [ 4 0 R ]
  >>


obj 3 0
 Type:
 Referencing: 5 0 R

  <<
    /JavaScript 5 0 R
  >>


obj 4 0
 Type: /Page
 Referencing: 2 0 R, 12 0 R

  <<
    /Type /Page
    /Parent 2 0 R
    /Contents 12 0 R
  >>
```

**FIGURE 5.** A snapshot of the output of PDF-PARSER scanning a PDF file.

- xref: Number of xref that can be observed within the parsed structure of a PDF.
- <<: Number of '<<' keywords that can be noticed in the parsed structure of a PDF. In a PDF file, the '<<' signifies the start of a dictionary object.
- >>: Number of '>>' keywords that can be noticed in the parsed structure of a PDF. In a PDF file, the '>>' signifies the closing of a dictionary object.
- /Font: Number of /Font entries that can be discovered inside the parsed structure of a PDF.
- /XObject: Number of /XObject keywords that can be observed within the parsed structure of a PDF. The /XObject keyword is utilized to indicate and encapsulate external graphical material such as images, forms, and other sophisticated objects.
- /ModDate: Number of /ModDate entries that can be discovered inside the parsed structure of a PDF. The modification date and time of the PDF file are specified using the /ModDate keyword.
- /Info: Number of /Info keywords that can be spotted inside the parsed structure of a PDF. The term /Info describes the document's information dictionary.
- /XML: Number of /XML entries that can be discovered inside the parsed structure of a PDF.
- Comment: Number of comments that are noticed inside the parsed structure of a PDF.
- /Widget: Number of /Widget keywords that are found within the parsed structure of a PDF. The /Widget annotations are interactive components that are employed in PDF files, particularly PDF forms, which enable users to interact with input data.
- Referencing: Number of Referencing keywords that are noticed inside the parsed structure of a PDF.
- /FontDescriptor: Number of /FontDescriptor keywords that are discovered within the parsed structure of a PDF.
- /Image: Number of /Image keywords that can be found within the parsed structure of a PDF.
- /Rect: Number of /Rect keywords that are observed within the parsed structure of a PDF.
- /Length: Number of /Length keywords noticed within a PDF's parsed structure. The /Length keyword specifies the length or size of the content stream related to a PDF object in bytes.
- /Action: Number of /Action keywords noticed within a PDF's parsed structure.

### C. DERIVED FEATURES

We have derived seven more features by carefully observing the characteristics of the PDFs from the standard feature set $F_1$. The derived features are introduced at the following:

- **Headerlength**: The feature considers the length of the filename i.e. length of the title of the PDFs.
- **Headercorrupt**: This is a binary feature that considers the version of the PDFs. If the version of any PDF does not start with %PDF-1.X where X=[0,1,....,7], then the feature value is set to 1 and 0 otherwise.
- **Small content**: This binary feature is derived through the careful observation of the number of objects in a PDF. If a PDF has objects less than or equal to 14 then the feature is set to 1 and 0 otherwise. We have observed

the mean value grouped by class (malicious and benign) for the number of objects present in the PDFs. The mean value for malicious PDFs is 14.1 whereas for the benign ones, the mean is 85.6. Furthermore, we have spotted 6277 malicious PDFs which is 77.90% of the entire malicious PDF sample that fall under or equal to this threshold value of 14. On the other hand, we have discovered 1709 benign PDFs which is 21.6% of the entire benign PDF sample, also fall under the same constraint. Moreover, we have observed 22.10% malicious PDFs that do not meet the threshold requirement. However, through thoughtful inspection, we set the threshold value as 14 for this binary feature.

- **Content corrupt**: This binary feature is set to 1 if the number of objects and endobjects of the PDFs are not the same and 0 otherwise.
- **Stream corrupt**: If the number of streams and endstreams are not the same then this feature is set to 1 and 0 otherwise.
- **Malicecontent**: This binary feature is set to 1 if two features from /JS, /JavaScript, /AA (Aditional Action), /Launch, /OpenAction are found at least for a single instance within a PDF and 0 otherwise. These features pose a risk since they can be exploited to insert and execute malicious code within a PDF document.
- **Hidden File**: The PDFiD tool while scanning a PDF file indicates if there is any hidden file embedded by the adversaries within the document. This binary feature is set to 1 if there is any hidden file found within the document and 0 otherwise.

To find out the effectiveness of the derived features for malicious PDF detection, we have merged these derived features with the standard feature set $F_1$, $F_2$, and $F_3$ respectively to generate derived feature sets $F_1'$, $F_2'$, and $F_3'$ correspondingly.

### D. MACHINE LEARNING MODEL SELECTION

In this pilot study, we intend to develop an effective data-driven approach based on machine learning to detect malicious PDFs. To select an effective machine learning model, at the first step, we initialize the standard feature set $F$ that contains $F_1$, $F_2$, and $F_3$ feature sets. Then, we select $M$ number of baseline machine learning classifiers including Random Forest, C5.0, SVM, J48, AdaBoost, Deep Neural Network (DNN), Gradient Boosting, and KNN for our experimental study. We iterate through each feature set in $F$, for each classifier in $M$, to evaluate each classifier on each feature set based on 10-fold cross-validation and generate the classification report. We compare the classification report yielded for each feature set and choose the best-performing model for PDF malware detection based on the report.

### E. FINAL FEATURE SET

After we generate the derived feature sets $F_1'$, $F_2'$, and $F_3'$ by merging the derived features with the standard feature sets $F_1$, $F_2$, and $F_3$ respectively, we concentrate on developing the final feature set. The intuition behind building the final feature set is to create an effective single feature set by observing all the derived feature sets. Thus, we measure the feature importance, rank the features for each derived feature set, and generate subsets based on the rank of the features. By leveraging the best-performing model, we implement each feature subset utilizing the model to find out the effectiveness of the subset. We iterate through derived feature sets $F_1'$, $F_2'$, and $F_3'$ respectively to create subsets by taking top features from each one of them. After we complete the iteration for subset generation and their evaluation utilizing the best-performing model, we compare the performance of the model among the subsets of each derived feature set to find the best feature subset. We take the best feature subset from each derived feature set $F_1'$, $F_2'$, and $F_3'$ respectively, and perform the union operation among them to develop the final feature set. This newly developed final feature set is then utilized to conduct the final classification activities to detect PDF malware. Algorithm 1 explains the overall approach of the final feature set generation. In the following section, we discuss the experimental results obtained in this research for PDF malware detection.

**Algorithm 1** Steps For Final Feature Set Generation

**Input:** Derived Feature Sets, $F' = \{F1', F2', F3'\}$
**Output:** Final Subset containing the union of best subsets

```
for f in F' do
    Find the importance of features in f
    Sort the features in f based on importance
    Generate subset S = {S1, S2, S3, ...} by taking top features from f
    Initialize best_subset as an empty set
    for s in S do
        Apply s on the Best ML Model Selected for PDF Malware Detection
        Generate Classification Report
        if subset performance is better than best_subset performance then
            Set best_subset to s
        end if
    end for
    Perform a union operation to append best_subset to Final_Subset
end for
```

## IV. EXPERIMENTAL RESULTS

In this pilot experiment, we aim to 1) build an efficient and improved feature set for identifying PDF malware, 2) empirically explore various baseline machine learning classifiers to select an effective machine learning classifier that can leverage the freshly created feature set to identify PDF malware with an improved detection accuracy, 3) explain how much the features of the final feature set contribute to the classifier to detect maliciousness in PDF and 4) extract a few crucial decision rules leveraging the power of the classifier that is easily understood and interpretable by humans to aid

in the detection of potential maliciousness in PDF. To accomplish the objectives, we carry out experiments based on the following cases:

- *Case I*: In this instance, we mainly concentrate on finding the answer to the following questions: 1) How do the standard feature set $F_1$, $F_2$, and $F_3$ assist in identifying PDF malware? and 2) What is the suitable machine learning model for PDF malware detection?
- *Case II*: In this scenario, our key focus is to uncover the findings of the following queries: 1) What is the impact of the derived feature sets in identifying PDF malware? and 2) How do the derived feature sets $F_1'$, $F_2'$, and $F_3'$ contribute to the detection of PDF malware?
- *Case III*: In this case, we investigate the answers to the following questions: 1) What features are selected for the final feature set? and 2) Does the final feature set boost the performance of the classifier?
- *CASE IV*: In this particular circumstance, we look into finding the answers to the following queries: 1) How does the combined feature set (i.e. $F_1$ + $F_2$ + $F_3$ + derived features which also can be represented by $F_1'$ $U$ $F_2'$ $U$ $F_3'$ ) help in the detection of malicious PDF? 2) Will the best feature subset generated from the combined feature set by taking into account the feature importance and approach described in *CASE III* differ from the final feature set acquired in *CASE III*? 3) Will the best feature subset produced in this scenario have a positive or negative impact on classification performance? and lastly 4) What is the classification performance when no derived features are used, such as only with $F_1$ + $F_2$ + $F_3$?
- *Case V*: In this instance, we focus on explaining i.e. How does the freshly developed final feature set aid the classifier in detecting PDF malware? Furthermore, we present an analysis of the distribution of the characteristics of the final feature set in the operational dataset to identify a few prospective directions that may effectively aid in identifying PDF malware.

In addition, we utilize the strength of the classifier to discover a few key decision rules that humans can understand easily and apply to identify potentially dangerous PDF content.

### A. EVALUATION METRICS

We employ the abbreviations for the evaluation metrics listed below to analyze the classification report:

- **Acc:** The term accuracy is abbreviated as Acc, and it can be assessed using the following formula:

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

where TP means True Positive, FP means False Positive, TN means True Negative, and FN stands for False Negative.

- **Pr:** The term precision is abbreviated as Pr can be measured by

$$Precision = \frac{TP}{TP + FP}$$

- **Rec:** The abbreviation Rec is used in place of the term Recall, which can be quantified by

$$Recall = \frac{TP}{TP + FN}$$

- **F1:** The term F1-Score is denoted as F1, which can be calculated by using the

$$F1 = 2 * \frac{Pr * Rec}{Pr + Rec}$$

### B. CASE I

In this case, we look at the impact of standard feature sets $F_1$, $F_2$, and $F_3$, as well as baseline machine learning classifiers, to determine the top-performing model for detecting PDF malware. Table 3 demonstrates the performance of the various baseline machine learning classifiers along with a deep neural network (DNN) for PDF malware detection utilizing the standard feature set $F_1$, $F_2$, and $F_3$ based on 10-fold cross-validation. Conspicuously, we can observe that the Random Forest classifier yields the best accuracy for all the standard feature sets compared to the baseline classifiers while identifying PDF malware. We utilized Scikit-Learn, a well-known open-source machine-learning library for Python to implement the baseline classifiers. We constructed the Random Forest classifier with 100 estimators and with random_state = 42 to handle the randomness. On the other hand, we built the C5.0, SVM, J48, AdaBoost, and KNN classifiers with their default hyperparameters as per the Scikit-Learn library.

Furthermore, we developed the DNN model which is an MLPClassifier, with a hidden layer of 100 units and random_state = 42. We executed the DNN model for 100 epochs. And finally, the Gradient Boosting classifier (GBC) was introduced with the 100 estimators and random_state = 42. While implementing the standard feature set $F_1$, we clearly witness that the Random Forest classifier stands out as the top-performing model for identifying PDF malware by acquiring an accuracy of 96.82% compared to the other models used in this work. Upon examining the other classifiers' performance, we find that the C5.0 classifier achieved the second-best accuracy for malicious PDF identification, with 96.59%. However, we find SVM classifier yielded comparatively less effective performance for identifying the malicious PDFs. On the other hand, the J48, AdaBoost, GBC, KNN, and DNN models provided an accuracy of 96.57%, 95.92%, 96.49%, 95.77%, and 96.20% respectively for PDF malware detection. Similarly, we notice that the Random Forest model outperforms the baseline classifiers by showing an accuracy of 96.53% and 97.19% for detecting PDF malware utilizing the standard feature sets $F_2$ and $F_3$ respectively. Fig. 6, 7, and 8 explicitly exhibit the ROC curve comparison of the classifiers implemented in this study on the standard feature

**TABLE 3.** An investigation of the Accuracy, Precision, Recall, and F1 - Score of various machine learning methods for standard feature sets utilizing tools (PDFiD, PDFINFO, and PDF-PARSER) based on 10-fold cross-validation.

| Standard Feature Set | Model | Acc | Pr | Rec | F1 |
|---|---|---|---|---|---|
| $F_1$ (PDFiD) | Random Forest | **0.9682** | **0.9690** | **0.9682** | **0.9682** |
| | C5.0 | 0.9659 | 0.9668 | 0.9659 | 0.9658 |
| | SVM | 0.8450 | 0.8551 | 0.8450 | 0.8437 |
| | J48 | 0.9657 | 0.9667 | 0.9657 | 0.9657 |
| | AdaBoost | 0.9592 | 0.9594 | 0.9592 | 0.9592 |
| | DNN | 0.9620 | 0.9624 | 0.9620 | 0.9620 |
| | GBC | 0.9649 | 0.9653 | 0.9649 | 0.9648 |
| | KNN | 0.9577 | 0.9580 | 0.9577 | 0.9577 |
| $F_2$ (PDFINFO) | Random Forest | **0.9653** | **0.9656** | **0.9653** | **0.9653** |
| | C5.0 | 0.9598 | 0.9599 | 0.9598 | 0.9598 |
| | SVM | 0.8441 | 0.8486 | 0.8441 | 0.8435 |
| | J48 | 0.9602 | 0.9604 | 0.9602 | 0.9602 |
| | AdaBoost | 0.9437 | 0.9439 | 0.9437 | 0.9437 |
| | DNN | 0.9546 | 0.9556 | 0.9546 | 0.9546 |
| | GBC | 0.9602 | 0.9611 | 0.9602 | 0.9601 |
| | KNN | 0.9547 | 0.9549 | 0.9547 | 0.9547 |
| $F_3$ (PDF-PARSER) | Random Forest | **0.9719** | **0.9727** | **0.9719** | **0.9719** |
| | C5.0 | 0.9699 | 0.9708 | 0.9699 | 0.9699 |
| | SVM | 0.8293 | 0.8487 | 0.8293 | 0.8266 |
| | J48 | 0.9698 | 0.9706 | 0.9698 | 0.9698 |
| | AdaBoost | 0.9613 | 0.9616 | 0.9613 | 0.9613 |
| | DNN | 0.9608 | 0.9618 | 0.9608 | 0.9608 |
| | GBC | 0.9699 | 0.9708 | 0.9699 | 0.9699 |
| | KNN | 0.9622 | 0.9629 | 0.9622 | 0.9622 |

**TABLE 4.** Feature importance of standard feature sets $F_1$, $F_2$, and $F_3$.

| Feature Set, $F_1$ (PDFiD) | Importance | Feature Set, $F_2$ (PDFINFO) | Importance | Feature Set, $F_3$ (PDFPARSER) | Importance |
|---|---|---|---|---|---|
| /JS | 0.189384 | Filesize_kb | 0.385827 | /JS | 0.17806 |
| startxref | 0.133327 | Metadata Stream | 0.14866 | /JavaScript | 0.174966 |
| /JavaScript | 0.128817 | Optimized | 0.132621 | /Producer | 0.098267 |
| obj | 0.092926 | Pages | 0.112362 | /Size | 0.079436 |
| xref | 0.080607 | JavaScript | 0.071808 | startxref | 0.053516 |
| endobj | 0.075699 | Page size:_miscsize | 0.054809 | /ProcSet | 0.046 |
| stream | 0.052136 | Tagged | 0.028214 | %EOF | 0.043771 |
| trailer | 0.047182 | Page size:_A4 | 0.0246 | xref | 0.03762 |
| /OpenAction | 0.046888 | Page size:_letter | 0.015741 | /Font | 0.028149 |
| endstream | 0.037798 | Form:_none | 0.009629 | Referencing | 0.024605 |
| /XFA | 0.037407 | Form:_XFA | 0.007045 | /ID | 0.024562 |
| /Page | 0.023114 | Custom Metadata | 0.00654 | /S | 0.02106 |
| /AcroForm | 0.019413 | Page rot | 0.001469 | /Rect | 0.02038 |
| /EmbeddedFile | 0.016478 | Encrypted | 0.000674 | /Widget | 0.020207 |
| /ObjStm | 0.011918 | UserProperties | 0 | << | 0.016886 |
| /AA | 0.002771 | Suspects | 0 | /XObject | 0.016549 |
| /Launch | 0.001869 | | | /CreationDate | 0.016449 |
| /RichMedia | 0.001168 | | | obj | 0.016161 |
| /Colors | 0.000437 | | | >> | 0.015845 |
| /JBIG2Decode | 0.000386 | | | /Image | 0.015456 |
| /Encrypt | 0.000274 | | | Comment | 0.010286 |
| | | | | /ModDate | 0.009622 |
| | | | | /FontDescriptor | 0.008315 |
| | | | | /Length | 0.008113 |
| | | | | /Action | 0.006537 |
| | | | | /Info | 0.005279 |
| | | | | /XML | 0.003905 |

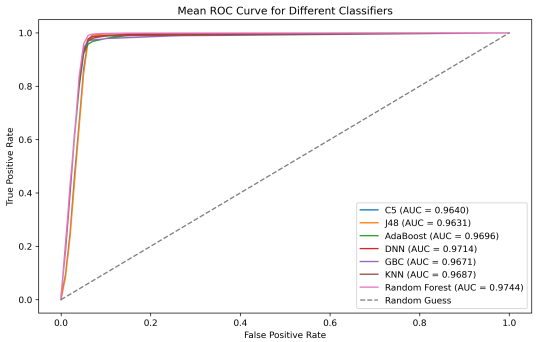


**FIGURE 6. ROC curve comparison of Random Forest model with various classifiers adopted in this study on the standard feature set $F_1$.**

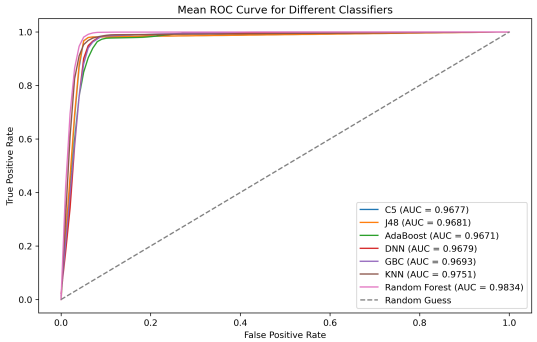


**FIGURE 7. ROC curve comparison of Random Forest model with various classifiers adopted in this study on the standard feature set $F_2$.**

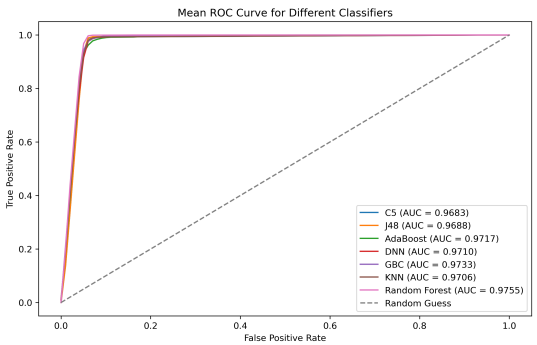


**FIGURE 8. ROC curve comparison of Random Forest model with various classifiers adopted in this study on the standard feature set $F_3$.**

sets $F_1$, $F_2$, and $F_3$ respectively. We can identify that the Random Forest classifier provided the best area under the curve (AUC) score compared to the others for PDF malware detection in each of these standard feature sets. Among the three standard feature sets, we encounter that the feature set $F_3$ turns out to be the best for obtaining better performance of the model. To verify the intuition of the Random Forest model's performance, we investigated the feature significance of the standard feature sets using the model directly.

Table 4 lists the feature importance of the standard feature set $F_1$, $F_2$, and $F_3$ respectively while implementing the Random Forest model for PDF malware detection. From Table 4, we identify the key features of the Random Forest model which aid in achieving a better performance of the model compared to the other baseline classifiers. The features /JS, startxref, and JavaScript are the most important features of $F_1$ whereas the features Filesize_kb, Metadata Stream, and Optimized are the most important features from $F_2$. Besides, we identify that the /JS, /JavaScript, and /Producer are the top three important features from the feature set $F_3$. Leveraging the effectiveness of the Random Forest classifier in high-dimensional feature space as well as the decision-making capability based on the ensemble learning method and certainly observing the aforementioned empirical analysis utilizing the standard feature sets, we consider the Random Forest model as the best-performing model to detect PDF malware. Thus, for further case studies, we take only the Random Forest model to execute our desired experiments.

### C. CASE II

In this case, we discover the impact of the derived feature sets on the classifier's performance as well as how much the derived features incorporate for identifying PDF malware. Table 5 highlights the performance of the Random Forest classifier utilizing both the standard and derived feature sets based on 10-fold cross-validation. The findings explicitly demonstrate that the derived feature set noticeably improved the effectiveness of the classifier compared to the standard feature sets. We observe a nearly 2% increase in accuracy for the classifier when utilizing the derived feature set $F_1'$ instead of the standard feature set $F_1$. Similarly, the other derived feature sets $F_2'$, and $F_3'$ maintain the same consistency for the accuracy improvement of the classifier as like as the $F_1'$. However, for the derived feature set $F_3'$, we obtain the best classification report by attaining an accuracy of 98.90% of the classifier for the detection of PDF malware.

To further assess the significance of the derived features, we estimate the feature importance of the derived feature sets $F_1'$, $F_2'$, and $F_3'$ leveraging the power of the classifier. Table 6 shows the feature importance of the derived feature sets and explicitly exhibits the significance of the derived features and how much they contribute during the classification. The results reveal that the derived feature *Headerlength* contributes most for all the derived feature sets i.e. the feature *Headerlength* contributes 28.19%, 34.15%, and 30.85% when $F_1'$, $F_2'$, and $F_3'$ are utilized respectively for the classification activities, Likewise, another important derived feature *Malicecontent* turns out to be in top three in terms of its significance among all the features within the derived feature sets $F_1'$ and $F_2'$ by exhibiting 12.6%, and 17.10% contributions respectively for the classification purpose. However, we observe the *Malicecontent* feature as the second most important feature yielding 9.7% significance when the classifier utilizes the $F_3'$ for the identification of PDF malware. Apart from the *Headerlength* and *Malicecontent* derived feature, we encounter seldom contributions from other derived features, though we find the *small content* feature assisting a little to

**TABLE 5. Improvement of Random Forest classifier's performance using derived feature sets compared to the standard feature sets for detecting PDF malware based on 10-fold cross-validation.**

| Feature Set | Acc | Pr | Rec | F1 |
|---|---|---|---|---|
| $F_1(Standard)$ | 0.9682 | 0.9690 | 0.9682 | 0.9682 |
| $F_1'(Derived)$ | **0.9868** | **0.9868** | **0.9868** | **0.9868** |
| $F_2(Standard)$ | 0.9653 | 0.9656 | 0.9653 | 0.9653 |
| $F_2'(Derived)$ | **0.9824** | **0.9825** | **0.9824** | **0.9824** |
| $F_3(Standard)$ | 0.9719 | 0.9727 | 0.9719 | 0.9719 |
| $F_3'(Derived)$ | **0.9890** | **0.9891** | **0.9890** | **0.9890** |

the classifier among the rest of the derived features.

### D. CASE III

In this instance, we identify the features from the derived feature sets $F_1'$, $F_2'$, and $F_3'$ that are selected for the final feature set through careful observations. Besides, we investigate the effect of introducing the final feature set on the classifier's performance. Furthermore, we estimate the significance of the final feature set in the identification of PDF malware.

To identify the best feature subsets from the derived feature sets we follow the steps mentioned in the Algorithm 1. We use the findings highlighted in Table 6 where the feature importance of each derived feature set is estimated and then sorted according to their importance. We consider the features with at least 1% of the feature importance score to generate subsets from the derived feature sets $F_1'$, $F_2'$, and $F_3'$. Thus, we find the top 15 features from $F_1'$. top 10 features from $F_2'$, and top 18 features from $F_3'$ that satisfy the aforementioned condition, and these features are selected for the generation of feature subsets initially. We generate the subsets starting from the first feature and gradually increase the features sequentially up to the last feature considered for feature subset generation from each derived feature set.

Therefore, we build 15 subsets from $F_1'$, 10 subsets from $F_2'$, and 18 subsets from $F_3'$ respectively. Further, we follow the steps as mentioned in Algorithm 1 to estimate the effectiveness of each feature subset to select the best feature subset from each derived feature set for final feature set generation. We highlight the mean accuracy obtained utilizing the aforementioned approach for each feature subset of $F_1'$. $F_2'$, and $F_3'$ in Fig.9, Fig. 10, and Fig. 11 respectively. According to the depiction in Fig. 9, the classifier has a maximum mean accuracy of 98.69% when implementing the subset consisting of 11 top features from $F_1'$, and there is a small variation in the mean accuracy for other subsets of $F_1'$. Likewise in Fig. 10, the classifier yields a maximum mean accuracy of 97.91% when applying the subset consisting of 8 top features from $F_2'$, and there is a considerable variation in the mean accuracy for other subsets of $F_2'$. However, we notice in Fig. 11 that the classifier achieves the highest mean accuracy when utilizing all of the top 18 features considered for subset generation from $F_3'$. Thus, the subset comprising the top 11 features of $F_1'$ is identified as the best feature subset from $F_1'$, the top 8 features of $F_2'$ is identified as the best feature subset from $F_2'$, and the top 18 features of $F_3'$ is identified as the best feature subset from $F_3'$. To accommodate the commonness as well as the uncommonness among the newly identified best feature subsets, we perform a union operation to generate the final feature set. Table 7 represents the list of the identified features that are finally considered for the final feature set. We discover three derived features such as *Headerlength*, *Malicecontent*, and *small content* into the final feature set. Since the *JavaScript* feature is found in all three derived feature sets $F_1'$, $F_2'$, and $F_3'$, we only consider this feature once (from $F_1'$) in the list. The features */JS*, *startxref*, *xref* are noticed both in $F_1'$, and $F_3'$, so we take them only once (from

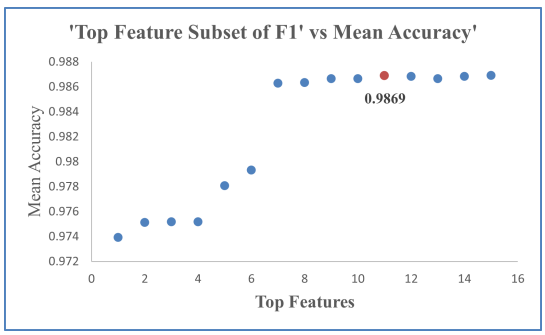


**FIGURE 9. Mean accuracy of Random Forest classifier vs top feature subset of the derived feature set $F_1'$.**

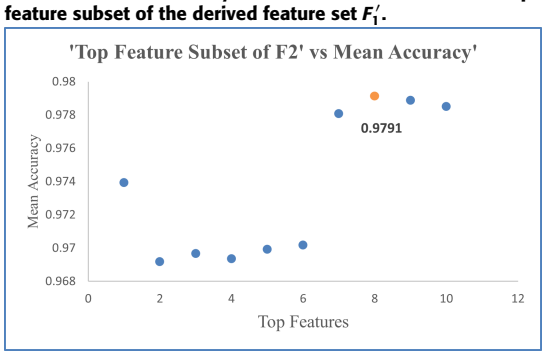


**FIGURE 10. Mean accuracy of Random Forest classifier vs top feature subset of the derived feature set $F_2'$.**

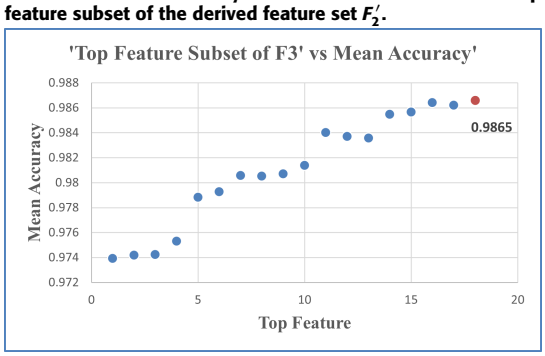


**FIGURE 11. Mean accuracy of Random Forest classifier vs top feature subset of the derived feature set $F_3'$.**

$F_1'$) in the list of the final feature set. We observe the features *obj*, *endobj*, *stream*, */OpenAction*, and */XFA* only from $F_1'$ in the final feature set. Besides the features *Filesize_kb*, *MetadataStream*, *Optimized*, and *Pages* are encountered only

**TABLE 6.** Feature importance of derived feature sets $F_1'$, $F_2'$, and $F_3'$.

| Feature Set, $F_1'$ (PDFiD) | Importance | Feature Set, $F_2'$ (PDFINFO) | Importance | Feature Set, $F_3'$ (PDFPARSER) | Importance |
|---|---|---|---|---|---|
| **Headerlength** | 0.281992628 | **Headerlength** | 0.341565169 | **Headerlength** | 0.308559 |
| /JavaScript | 0.155315511 | Filesize_kb | 0.192103387 | **Malicecontent** | 0.097292 |
| **Malicecontent** | 0.126014543 | **Malicecontent** | 0.171037759 | /JS | 0.093437 |
| /JS | 0.113431885 | Metadata Stream | 0.083803166 | /JavaScript | 0.082273 |
| xref | 0.052227619 | Optimized | 0.057181587 | /Producer | 0.055951 |
| startxref | 0.051961463 | Pages | 0.0405821 | /Size | 0.040859 |
| obj | 0.034711542 | **small content** | 0.035097756 | /CreationDate | 0.03431 |
| endobj | 0.029616909 | JavaScript | 0.018807386 | %EOF | 0.032303 |
| stream | 0.026597245 | Page size:_miscsize | 0.016363782 | startxref | 0.029804 |
| /OpenAction | 0.025362229 | Tagged | 0.014240093 | /ProcSet | 0.028089 |
| /XFA | 0.022755269 | Page size:_A4 | 0.007776304 | /ID | 0.025948 |
| trailer | 0.019853837 | Page size:_letter | 0.006177369 | xref | 0.014015 |
| endstream | 0.016040389 | Custom Metadata | 0.003728307 | /Info | 0.013927 |
| /EmbeddedFile | 0.014248542 | contentcorrupt | 0.002811481 | /Font | 0.013196 |
| /AcroForm | 0.010383106 | Form:_none | 0.002611641 | /S | 0.012968 |
| /Page | 0.006201786 | Form:_XFA | 0.002316382 | Referencing | 0.01296 |
| **small content** | 0.004261149 | headercorrupt | 0.001100524 | /XML | 0.010976 |
| /ObjStm | 0.003643089 | HiddenFile | 0.000976744 | /Rect | 0.010309 |
| /AA | 0.001284778 | streamcorrupt | 0.000798319 | /ModDate | 0.009883 |
| contentcorrupt | 0.001125052 | Page rot | 0.000714061 | obj | 0.009521 |
| streamcorrupt | 0.000936493 | Encrypted | 0.000206682 | /XObject | 0.009186 |
| /Launch | 0.000570549 | UserProperties | 0 | >> | 0.008798 |
| HiddenFile | 0.000351583 | Suspects | 0 | << | 0.007742 |
| /RichMedia | 0.000341436 | | | /Widget | 0.006901 |
| /Encrypt | 0.000328662 | | | **small content** | 0.006212 |
| headercorrupt | 0.000241423 | | | /Image | 0.006063 |
| /Colors | 0.000112839 | | | Comment | 0.004843 |
| /JBIG2Decode | 8.8444E-05 | | | /Length | 0.004401 |
| | | | | /FontDescriptor | 0.003952 |
| | | | | /Action | 0.002423 |
| | | | | contentcorrupt | 0.001105 |
| | | | | HiddenFile | 0.00084 |
| | | | | headercorrupt | 0.000505 |
| | | | | streamcorrupt | 0.000452 |

**TABLE 7. List of identified features of the final feature set.**

| Identified Features for the Final Feature Set | Source |
|---|---|
| Headerlength<br>Malicecontent<br>small content | Derived Features |
| /JavaScript | Common in $F_1'$, $F_2'$, $F_3'$ |
| /JS<br>startxref<br>xref | Common in $F_1'$, $F_3'$ |
| obj<br>endobj<br>stream<br>/OpenAction<br>/XFA | $F_1'$ Only |
| Filesize_kb<br>MetadataStream<br>Optimized<br>Pages | $F_2'$ Only |
| /Size<br>%EOF<br>/Producer<br>/ProcSet<br>/ID<br>/S<br>/CreationDate<br>/Info<br>/Font<br>Referencing<br>/XML<br>/Rect | $F_3'$ Only |

from $F_2'$, and the rest of the features listed in Table 7 are observed only from $F_3'$. We investigate the impact of the final feature set on the Random Forest classifier based on 10-fold cross-validation to detect PDF malware. Table 8 shows the findings of the classifier for several types of feature sets used in this research. We notice an impressive increase in the accuracy of the classifier due to the utilization of the final feature set compared to the standard and derived feature sets. The maximum accuracy improvement for the classifier when employing the final feature set is 2.71% compared to the standard feature set $F_2$. On the contrary, we find that the minimum accuracy improvement of the classifier when executing the final feature set is 0.34% compared to the derived feature set $F_3'$. However, the classifier provides a noticeable performance boost in the case of PDF malware detection, due to the introduction of the freshly developed final feature set.

Fig. 12 illustrates the accuracy curve of the Random Forest classifier implemented using the final feature set based on 10-fold cross-validation. We observe that the model attains 99.56% accuracy during the sixth fold whereas the model yields the minimum accuracy of 99.05% during the ninth fold of the 10-fold cross-validation. The log loss curve of the Random Forest model during the various folds of the 10-fold cross-validation is depicted in Fig. 13. We observe the maximum loss during the third fold whereas the minimum loss during the first fold from the entire 10-fold cross-validation. Fig.14 represents the Reciever Operating Characteristics (ROC) curve of the Random Forest model for the various folds of the 10-fold cross-validation on the final

**TABLE 8. Impact of final feature set on the Random Forest classifier's performance based on 10-fold cross-validation for PDF malware detection.**

| Feature Set | Acc | Pr | Rec | F1 |
|---|---|---|---|---|
| **Final Feature Set** | **0.9924** | **0.9924** | **0.9924** | **0.9924** |
| F1 (Standard) | 0.9682 | 0.9690 | 0.9682 | 0.9682 |
| F1'(Derived) | 0.9868 | 0.9868 | 0.9868 | 0.9868 |
| F2(Standard) | 0.9653 | 0.9656 | 0.9653 | 0.9653 |
| F2'(Derived) | 0.9824 | 0.9825 | 0.9824 | 0.9824 |
| F3(Standard) | 0.9719 | 0.9727 | 0.9719 | 0.9719 |
| F3'(Derived) | 0.9890 | 0.9891 | 0.9890 | 0.9890 |

feature set. We notice an area under the curve of 1.00 for the Random Forest model throughout the entire 10-fold cross-validation.

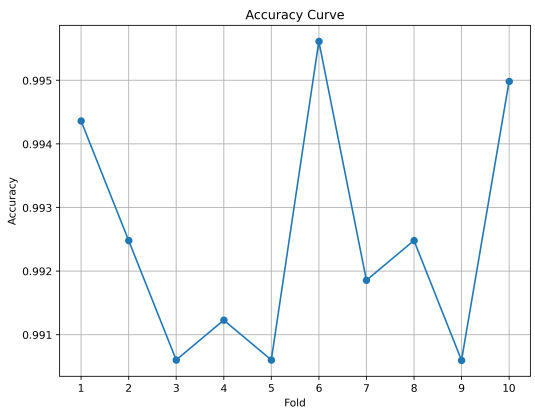


**FIGURE 12. Accuracy curve of Random Forest model on final feature set based on 10-fold cross-validation.**

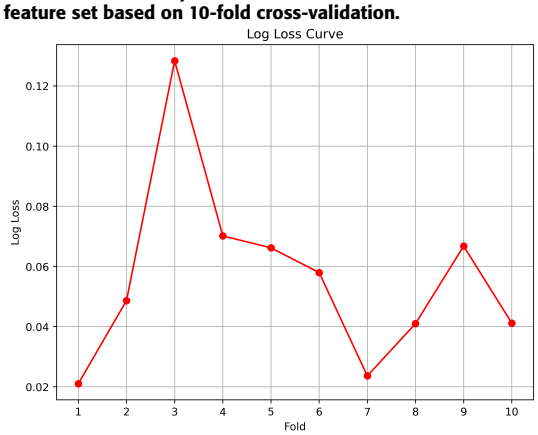


**FIGURE 13. Loss curve of Random Forest model on final feature set based on 10-fold cross-validation.**

### *E. CASE IV*

In this circumstance, we present the findings of implementing the combined feature set (i.e. $F_1 + F_2 + F_3$ + derived features) for detecting PDF malware. We identify the best feature subset from the combined feature set by considering the approach as stated in *CASE III* and discover the difference with the final feature set. Moreover, we find the impact of the best subset obtained in this case on the classification performance.

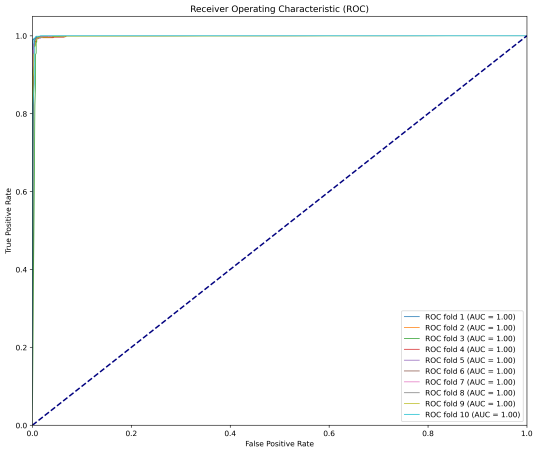


**FIGURE 14. ROC curve of Random Forest model on final feature set based on 10-fold cross-validation.**

**TABLE 9. Impact of combined feature set as well as the feature set with no derived features (i.e. F1+ F2+ F3) on the Random Forest classifier's performance based on 10-fold cross-validation for PDF malware detection.**

| Feature Set | Acc | Pr | Rec | F1 |
|---|---|---|---|---|
| F1 (Standard) | 0.9682 | 0.9690 | 0.9682 | 0.9682 |
| F1'(Derived) | 0.9868 | 0.9868 | 0.9868 | 0.9868 |
| F2(Standard) | 0.9653 | 0.9656 | 0.9653 | 0.9653 |
| F2'(Derived) | 0.9824 | 0.9825 | 0.9824 | 0.9824 |
| F3(Standard) | 0.9719 | 0.9727 | 0.9719 | 0.9719 |
| F3'(Derived) | 0.9890 | 0.9891 | 0.9890 | 0.9890 |
| **F1+F2+F3+Derived (Combined)** | **0.9919** | **0.9919** | **0.9919** | **0.9919** |
| **F1+F2+F3** | **0.9728** | **0.9729** | **0.9728** | **0.9728** |

Also, we discuss the potency of the classifier when no derived features are used for malicious PDF detection.

Table 9 explicitly depicts the performance of the Random Forest classifier when utilizing the combined feature set with derived features as well as the merged feature set with no derived features and presents a comparison of the classifier's efficacy with the final feature set. We observe that the classifier acquired 99.19% accuracy when utilizing the combined feature set with derived features (i.e. $F_1$ + $F_2$ + $F_3$+ derived features) for identifying malicious PDFs. On the other hand, the classifier yielded an accuracy of 97.28% for the merged featured set (i.e. $F_1$ + $F_2$ + $F_3$) with no derived features. Thus, we can notice the impact of the derived features in enhancing the performance of the classifier for at least 1.91% for detecting PDF malware. However, we discovered comparatively better efficacy of the classifier for the utilization of the final feature set to identify malicious PDFs. The reason behind this is that the combined feature set consists of comparatively a larger number of features than the final feature set. Because of the higher number of features, in the high-dimensional spaces, the classifier sometimes may find patterns in noise rather than genuine relationships and provides somewhat less effective performance. Furthermore, the elimination of unnecessary features can improve the model's ability to generalize and classify malicious PDFs effectively. Thus, the final feature set with less but by taking into account the important features based on feature importance produced better accuracy than the combined feature set.

Table 10 represents the feature importance of the top features of the combined feature set obtained utilizing the strength of the classifier. To generate the subset, from the table, initially, we consider the features having at least 1% feature importance score (as mentioned in *CASE III*). Thus to construct the best subset of the combined feature set, we evaluate only the top 10 features from Table 10. Then, we develop 10 subsets from these features by following the approach as described in *CASE III*. The mean accuracy obtained from these subsets adopting the classifier is illustrated in Fig. 15. We notice that the subset containing all the top 10 features produced the best mean accuracy of 98.53%. Therefore, we identify the subset having the features ( Malicecontent, /JavaScript, Filesize_kb, /Producer, Headerlength, /S, /ProcSet, /ID, startxref, /Info ) as the best subset of the combined feature set. Conspicuously, we can strongly differentiate between the final feature set and the best subset obtained from the combined feature set. However, we note that the best subset achieved in this case does not improve the classifier's accuracy when compared to the classifier's performance for the final feature set.

Furthermore, to validate the effectiveness of the final feature set we utilize the Correlation-based Feature Selection (CFS) technique to generate the best feature subset from the combined feature set. We implement the *CfsSubsetEval* feature selection method (with Best First Search approach) using a popular machine learning software *Weka 3* [47]. We find the following features in the best subset as output from the method: Headerlength, contentcorrupt, /Encrypt, Malicecontent, /Colors, Metadata Stream, Optimized, Page size:_A4, Page size:_miscsize, /Size, and /Action. Similarly, we also implement the *ReliefFAttributeEval* feature selection method (with Ranker approach) to produce the best subset from the combined feature set using *Weka 3*. We consider the features having at least a 1% merit score and then adopt the approach as mentioned in *CASE III* to develop and evaluate the feature subset. Finally, we identify the best subset from this approach having the following features: Headerlength, Optimized, Malicecontent, Metadata Stream, Tagged, /EmbeddedFile, Custom Metadata, Form:_none, /FontDescriptor, /XFA, /Font, small content, /Producer, /AcroForm, /ModDate, %EOF, Form:_XFA, /XML, /Action, /CreationDate, and xref. To further evaluate the potency of the final feature set, we employ these feature sets using the classifier's strength and compare the results. We find that for the subset of *CfsSubsetEval*, the classifier yields an accuracy of 97.59% whereas for the subset of *ReliefFAttributeEval*, the classifier provides 98.75% accuracy. Notably, we identify that the classifier produces the highest accuracy when using the final feature set to detect malicious PDFs.

Overall, we find that from *CASE I*, the classifier delivers the highest accuracy of 97.19% on the standard feature set $F_3$, from *CASE II*, the classifier yields the highest accuracy

**TABLE 10. Feature importance of top features of combined feature set.**

| Combined Feature Set (F1+F2+F3+Derived) | Importance |
|---|---|
| Malicecontent | 0.159703144 |
| /JavaScript | 0.149509462 |
| Filesize_kb | 0.138701347 |
| /Producer | 0.122005002 |
| Headerlength | 0.109485574 |
| /S | 0.093529154 |
| /ProcSet | 0.026969307 |
| /ID | 0.020228097 |
| startxref | 0.018314721 |
| /Info | 0.017243893 |
| obj | 0.009457851 |
| endobj | 0.009146312 |
| xref | 0.008258587 |
| Optimized: | 0.007503771 |
| Metadata Stream: | 0.006817359 |
| stream | 0.006458339 |
| << | 0.005902468 |
| endstream | 0.005836158 |
| /OpenAction | 0.0058075 |
| /JS | 0.005585594 |
| >> | 0.005467207 |
| /XFA | 0.005415037 |
| /Font | 0.004927339 |
| Referencing | 0.004888336 |
| trailer | 0.004624181 |
| /AcroForm | 0.003871471 |
| %EOF | 0.003766663 |
| /Page | 0.00370188 |
| /Length | 0.003536077 |

of 98.90% on the derived feature set $F_3'$, from *CASE III* the classifier produces the best accuracy of 99.24% utilizing the final feature set, and from *CASE IV* the classifier outputs the highest accuracy of 99.19% on the combined feature set with derived features. Thus, conspicuously we can identify that the final feature set assists the classifier to deliver the highest efficacy for detecting PDF malware among all the feature sets.

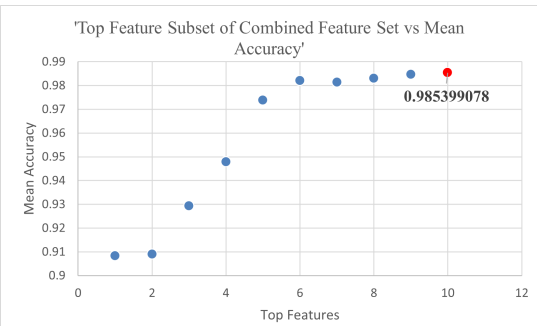


**FIGURE 15. Mean accuracy of Random Forest classifier vs top feature subset of the combined feature set.**

### F. CASE V

In this case, we provide an explanation of how the freshly created final feature set contributes to the classifier for identifying maliciousness in PDF. To analyze the significance of the final feature set, we estimate the importance of the top features from this feature set utilizing the Random Forest classifier which is illustrated in Fig. 16. This illustration uncovers the important features and how much they contribute to the classification activities. The illustration reveals that the derived features *Headerlength*, and *Malicecontent* contribute largely to the identification of PDF malware. On the other hand, */JS* and */Javascript* features are also proven to be very crucial for malicious PDF detection.

We observe all the features from the newly developed final feature set to explore the traits of both categories of PDFs toward these features in our operational dataset. The average value of the derived feature *Headerlength* is 16.50 with a standard deviation of 12.45 for the benign PDFs whereas for the malicious ones, the average value is 42.48 with a standard deviation of 7.28 as depicted in Fig.17. The illustration also reveals that the title length of 75.83% of the benign PDFs is under or equal to the mean value of benign ones while on the other hand 89.16%, malicious PDFs satisfy the condition of their title length less or equal to their mean value. However, this explains the fact that in our operational dataset, the average title length of the benign PDFs is much smaller than the malicious ones. This finding provides a potential indication of identifying PDF malware by just looking at the length of the title of the PDF, though this feature alone does not necessarily point to the maliciousness within a PDF. Because, in a real-world scenario, a clean PDF often may have a large title length.

The distribution of another derived feature *Malicecontent* that is constructed through inspecting the triggering features across the malicious and benign PDFs is illustrated in Fig. 18. During our pilot study, we noticed 7246 malicious PDFs identified by this feature whereas 651 benign PDFs also fell under the same condition. This demonstrates that malicious PDFs mostly contain triggering features compared to the clean ones which also point out a potential direction of identifying PDF malware in real-world cases. The characteristics of the */JavaScript* feature, a popular choice by cyber attackers to build a PDF maldoc plotted in Fig. 19 through the proper inspection of our operational dataset. The finding explains that the clean PDFs close to 92%, hardly have JavaScript features while nearly one-third of malicious PDFs from our dataset contain this feature. The presence of this feature within a PDF exhibits the strong possibility of malicious activity. Similar to the */JavaScript* feature, we discover very little presence of */JS* feature among the clean files but in the case of malicious ones, we observe frequent presence of this keyword.

In Fig. 20, we portray the traits of the *FIlesize_kb* feature that explicitly reveals that the average size of the clean PDFs is much larger than the harmful ones. We discover a high standard deviation for the file size of the benign PDFs, as well as the fact that only 68.77% of clean PDFs have a file size less than or equal to their mean size. We also detect a substantial standard deviation for the malicious PDFs, with nearly 10% of the malicious PDFs file size falling outside of their mean bounds. The mean value of the */Size* keyword is 2.27 for benign PDFs and 1.00 for malicious PDFs, emphasizing that attackers try to embed their intended payload inside PDFs rather than focusing on the content of the PDFs. The

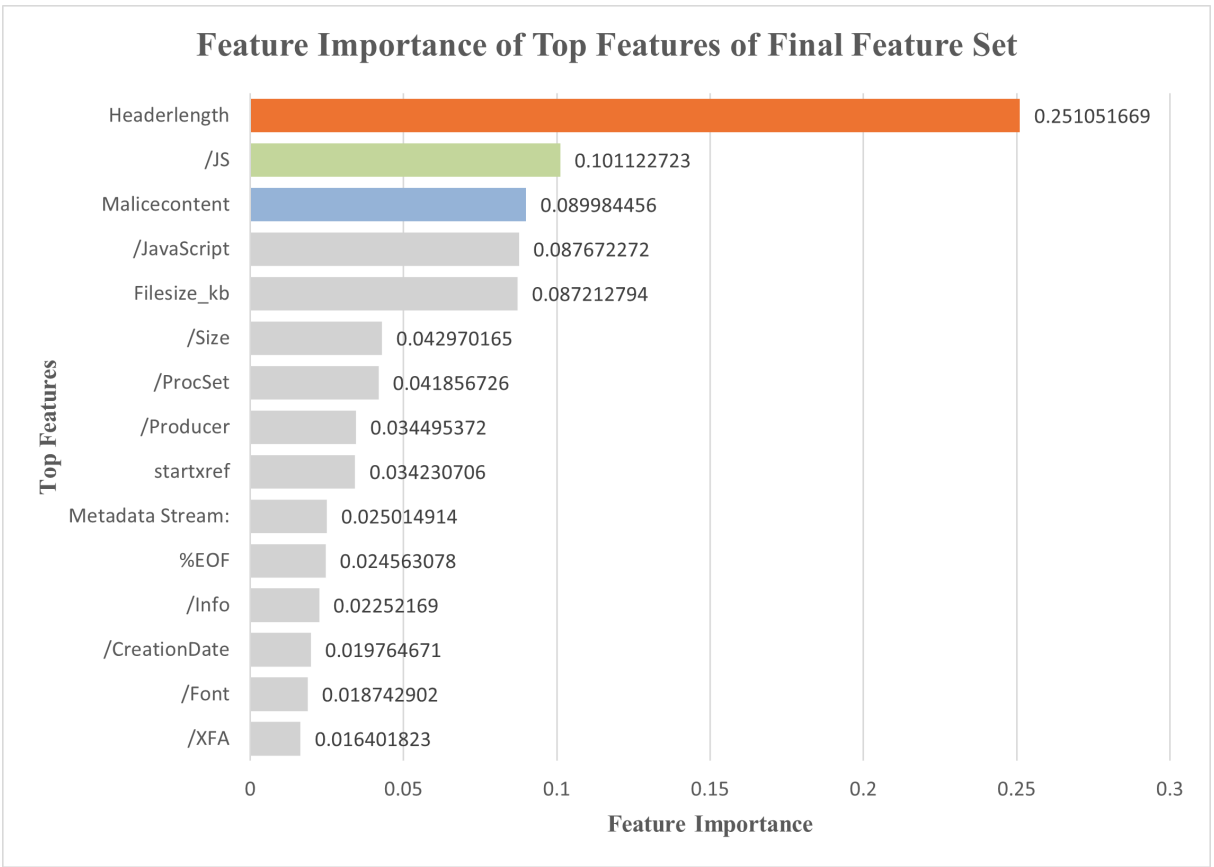


**FIGURE 16.** **A bar plot to display the importance of the top features of the final feature set.**

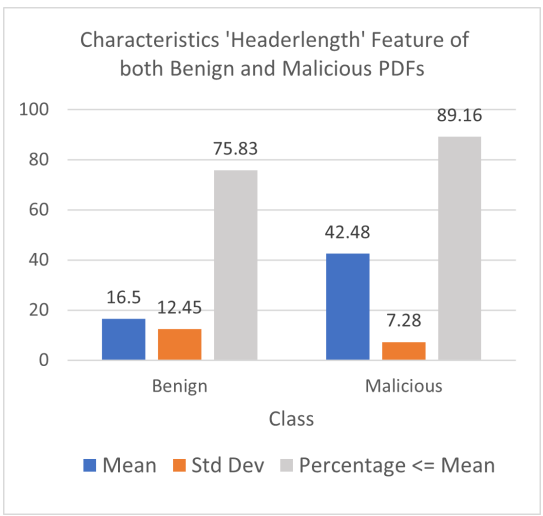


**FIGURE 17.** **Characteristics of Headerlength feature for both benign and malicious PDFs in the operational dataset.**

metadata features */Producer*, */ID*, */CreationDate*, and */Info* are commonly observed in clean PDFs but are rarely observed in the hazardous ones in our operational dataset. Likewise, the feature *MetadataStream* is spotted more often in the clean ones compared to the malicious ones.

This finding leads to the possibility that hazardous PDFs include fewer metadata features than clean PDFs. Fig.21 depicts the distribution of the obj characteristic across both PDF categories. The average number of objects for the clean PDFs is 85.61, with a significantly high standard deviation of 169.81, whereas the hazardous PDFs have a comparatively small mean of 14.11, with a standard deviation of 21.89. This underscores the fact that harmful files typically contain fewer objects since the attacker's objective is to create malicious material with as few objects as feasible in order to execute their attack as quickly as possible. We notice a similar pattern with the endobj feature, as every object declaration should be followed by an endobj, ideally. However, we inspect a small variation in the mean size for the endobj feature compared to the obj feature for malicious PDFs, with a mean value of 16.50. For the *stream* feature, we observe that malicious files occupy a limited number of streams compared to the clean files. Attackers exploit *startxref* and *xref* as well to avoid detection. For a PDF document, a reader program will render and show it as follows: The EOF (End Of File) mark at the bottom of the document serves as the starting point for reading. It is going to be the *startxref* preceded by the offset of the cross-reference table immediately on above. The offset of the root dictionary, which serves as the starting point of the hierarchical structure (PDF file), by which all objects can be

retrieved, is contained in the cross-reference table. If either the *xref* or the *startxref* are missing, stringent readers and parsers will reject the file as malformed. A versatile reader, on the other hand, will be capable to navigate the root dictionary and render the file, much like contemporary readers. The features *%EOF* can also be manipulated by the attackers to perform malicious activities.

The features */XFA* and */OpenAction* are observed to be more prominent in malicious PDFs than in clean files in our experimental dataset. Because the clean PDFs contain more objects and are larger in size than the hazardous ones, we discover that the features */Font*, *Referencing*, *XML*, */Rect*, */S* (subtype of objects or tasks), */ProcSet* (set of procedures) are more prevalent in the clean files than in the malformed ones. Furthermore, the average number of pages in clean files is 5.79, whereas it is 1.44 in malicious files. This demonstrates that harmful files in our operational dataset are rather short, generally consisting of one or two pages with limited information (often a blank page). This discovery points to a possible path for spotting questionable PDF files. We inspect a little significance of the feature *Optimized* and the derived feature *small content* throughout our entire operational dataset for detecting malicious PDFs.

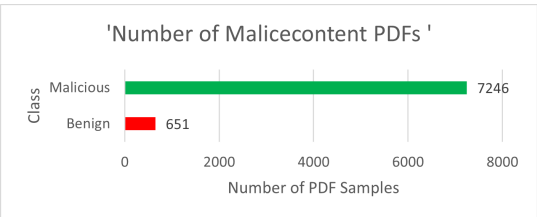


**FIGURE 18.** Distribution of Malicecontent feature for both benign and malicious PDFs in the operational dataset.

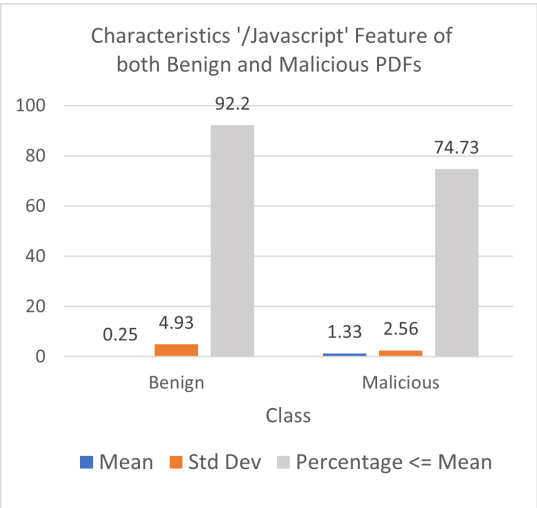


**FIGURE 19.** Characteristics of /JavaScript feature for both benign and malicious PDFs in the operational dataset.

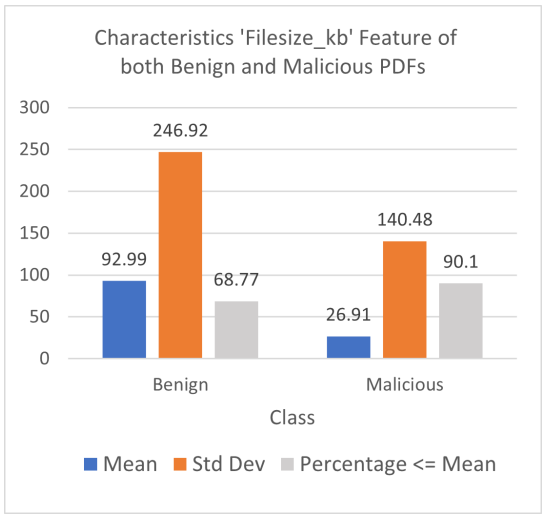


**FIGURE 20.** Characteristics of Filesize_kb feature for both benign and malicious PDFs in the operational dataset.

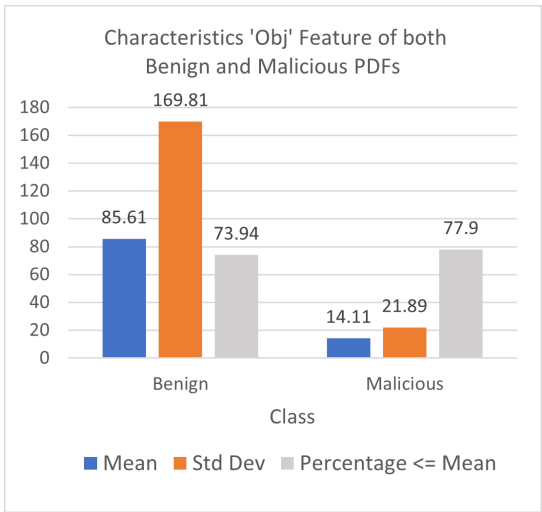


**FIGURE 21.** Distribution of Obj feature for both benign and malicious PDFs in the operational dataset.

### G. RULE DISCOVERY AND HUMAN INTERPRETATION

We explore the interpretation of the Random Forest classifier i.e. how the classifier predicts maliciousness in PDF with the help of the final feature set by constructing a decision tree from one of the estimators of the classifier as well as extracting a few important decision rules from the implementation of the classifier to detect PDF malware. Moreover, we provide interpretations of the decision rules so that they can be easily comprehended. To decode the classifier's performance, in Fig. 22, we illustrate one of the decision trees from the 100 estimators of our Random Forest classifier. The generated decision tree provides an explanation of the performance of the classifier by representing the conditions of the various features from the final feature set in its nodes for detecting malicious or benign PDFs. As we observe the tree, we notice that each node of the tree specifies a feature with a certain

threshold condition which is used to split the samples and also mentions the percentage of samples reached to the node. Moreover, each node also provides the proportionate class distribution of the samples that reached the node and indicates the final class label that has the majority vote. The feature in the root node indicates the most important feature of the tree. If the condition in the root node is true then the control transfers to the left child of the root node or to the right child otherwise. This process continues for each node until we reach the leaf node which indicates a decision rule specifying a certain class label.

In Fig. 22, we find that the *Malicecontent* feature is in the root node of the tree specifying a threshold condition of *Malicecontent* $<= 0.5$ which is used to split the PDF samples. The condition implies that the initial decision point in the tree is based on the *Malicecontent* feature and evaluates whether or not its value is less than or equal to 0.5. We observe that the root node deals with all the samples considered for the tree. The values in square brackets show the proportionate distribution of classes at this node. The first value indicates 49% of benign occurrences, whereas the second value specifies 51% presence of malicious class among the samples that reached the root node. We find the majority vote for the malicious class at this node. The colors show each node's majority class (box, with rusty representing the majority benign and blue representing the majority malicious). The colors become darker as the node gets closer to becoming completely benign or malicious. Similarly, the colors become lighter if the node contains closer distribution of the samples among themselves.

As we can identify one of the decision rules in Fig. 22 that indicates *If* $(Malicecontent <= 0.5$ *and* $/ID > 0.5$ *and Headerlength* $<= 39.5)$ *then Class* : *Benign*. This reveals that if any PDF sample from our operational dataset does not have *Malicecontent* feature but contains metadata feature */ID* and its title length is less than or equal to 39.5, then the sample belongs to the benign class. Fig. 23 illustrates a decision plot of the Random Forest classifier for one of the instances that satisfies the above decision rule and belongs to the benign class. We adopted the SHAP library to construct the decision plot which renders the decision-making process easier to understand by highlighting how each feature affects the output of the classifier. The plot highlights the features that push the model toward classifying benign or malicious classes for a particular instance. The blue region (left side of the vertical line) of the plot represents the features that push the classifier's prediction toward the benign class whereas the red region (right side of the vertical line) highlights the features that push the classifier's prediction toward the malicious class. From Fig.23, notably we can observe that the *Malicecontent*, *Headerlength*, */ID* etc. features push the classifier towards the benign class prediction.

Similarly, we discover another decision rule that specifies *If* $(Malicecontent > 0.5$ *and stream* $<= 24.5$ *and* $/XFA > 0.5$ *and Headerlength* $<= 52)$ *then Class* : *Malicious*. This means that if a PDF sample from our dataset contains the *Malicecontent* feature with a number of streams less than or equal to 24.5, as well as the XFA form and its title length less than or equal to 52, it belongs to the malicious class. Fig. 24 visualizes the decision plot for one of the instances of the above decision rule where we can notice that *Headerlength*, */XFA*, *Malicecontent*, *stream* etc. features push the classifier towards the malicious class prediction.

We discovered a total of 230 decision rules from the illustrated decision tree of the Random Forest classifier that explains the predictions for detecting PDF malware. Since our Random Forest classifier has 100 estimators, to further investigate the decision rules, we generate all the decision trees of the classifier and derive all the potential decision rules from the trees. Fig, 25 depicts the number of decision rules discovered from each of the decision trees originating from the Random Forest classifier. We identify a total of 23183 decision rules from the 100 estimators of the classifier. The findings also demonstrate that the maximum number of decision rules (i.e. 333) is obtained from the tree_index = 53 whereas the minimum number of decision rules (i.e. 162) is discovered from the tree_index = 07. We observe a mean of 231.83 with a standard deviation of 34.11 for the number of decision rules extracted from the decision trees of the classifier. This implies the average number of decision rules used by the decision trees is 231, which aids in the Random Forest classifier's predictions.

Table 11 and Table 12 present a few important decision rules for detecting clean and malicious PDF files respectively, including the conditions of the rule, the total number of samples that come within the rules, as well as the right predictions and the rule's confidence, which indicates how frequently the rules are found to be true. Both of these tables provide a comprehensive explanation that can easily be interpreted by humans and aid them in identifying clean and harmful PDFs. As we look into Table 11, we find that the first decision rule signifies that if the PDF sample does not contain any */Javascript* and its title length is less or equal to 39.5 then the PDF sample falls into the benign category. This rule accurately identifies 7154 PDFs from our operational dataset, suggesting that the rule has 100% confidence. Similar to the first decision rule, all the other rules of the table can be explained and interpreted by humans for clearly detecting benign PDFs. Moreover, we find several strong rules (such as Rule ID 2 to 8) that yield 99% to 100% confidence as well as cover a wide range of samples for identifying benign PDFs. On the contrary, we find comparatively less effective rules at the bottom of Table 11 (such as Rule ID 9 to 15) where the rules yield a confidence level of around 90% to 98% covering a small number of samples to clearly identify them as clean files.

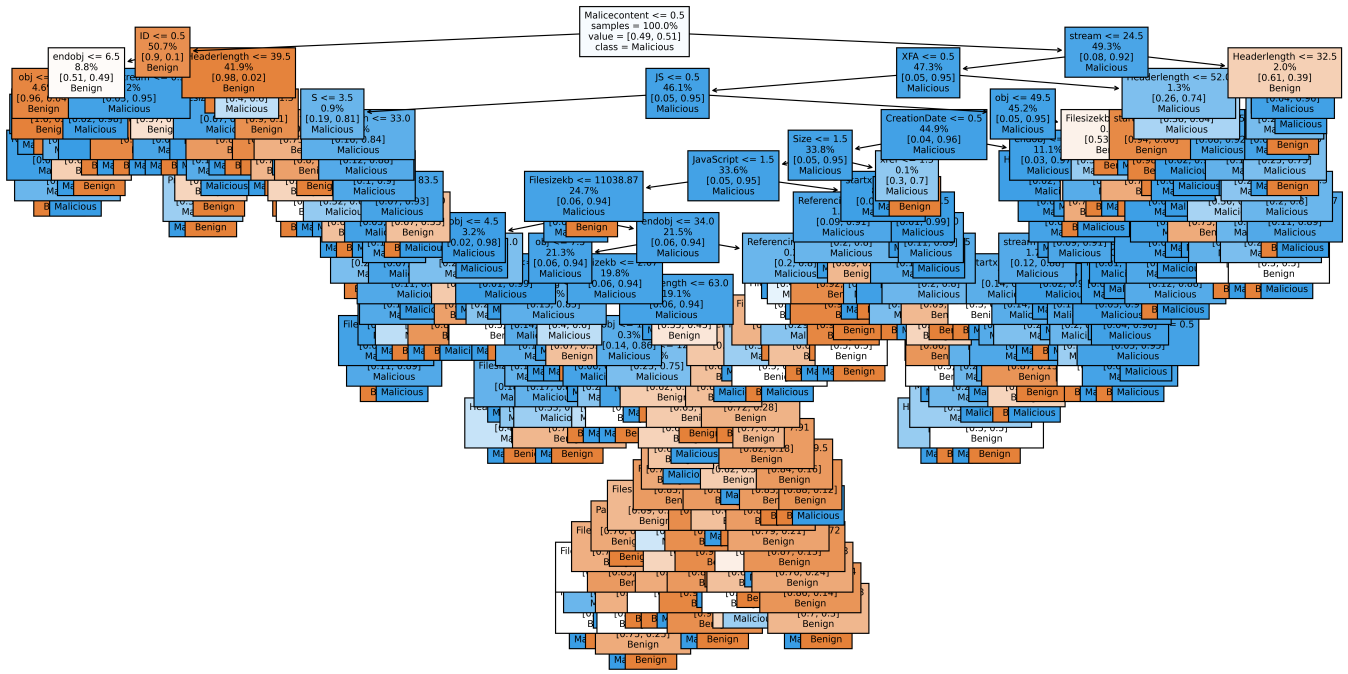


**FIGURE 22.** A decision tree from one of the estimators of the Random Forest classifier used to detect PDF malware utilizing the final feature set.

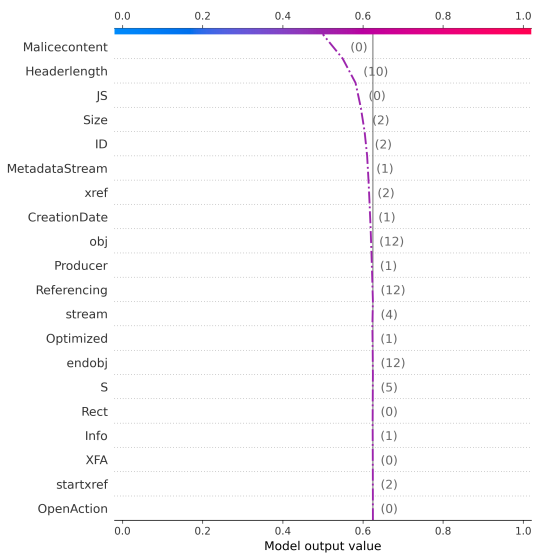


**FIGURE 23.** A sample decision plot of an instance of Benign Class using SHAP values.

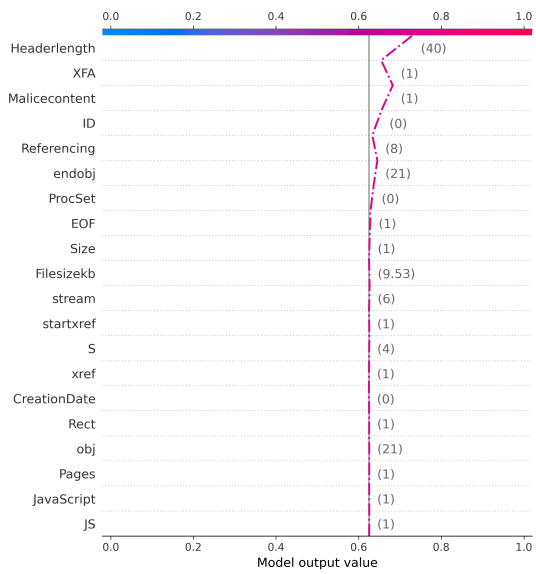


**FIGURE 24.** A sample decision plot of an instance of Malicious Class using SHAP values.

Looking at Table 12, we see that in the case of harmful PDF detection, a far larger number of requirements must be verified than in the case of benign PDF detection. Despite the large constraints, we find a few strong rules (such as Rule ID 1 to 5) that offer nearly 100% confidence in recognizing thousands of samples from our operational dataset as malicious. On the contrary, we identify somewhat less effective decision rules that only apply to a very tiny number of samples from our dataset as we carefully examine the rules from 6 to 10 in the table. Similar to the explanation indicated in Table 11, the decision rules of Table 12 can be extensively described and interpreted. For instance, the first rule of Table12 states that a PDF sample is considered malicious if it lacks the */XML* feature, is not optimized, has a file size greater than 1.41 kilobytes and less than or equal to 46.52 kilobytes, a title length greater than 39 and less than or equal to 63, contains a cross-reference table less than or equal to 1.5 times, and does not have the */CreationDate* feature. The rule correctly identifies 5079 malicious PDF samples from the dataset.

**TABLE 11.** Top decision rules extracted from Random Forest classifier to detect clean PDF.

| Rule ID | Conditions of the Rule | Total Sample | Correct Predictions | Confidence (%) |
|---|---|---|---|---|
| 1 | If (/JavaScript <= 0.5 and Headerlength <= 39.5) then | 7154 | 7154 | 100 |
| 2 | If (/ProcSet <= 1.5 and Malicecontent <= 0.5 and Headerlength <= 38.5) then | 1723 | 1723 | 100 |
| 3 | If (/Producer > 0.5 and Malicecontent <= 0.5 and %EOF > 1.5 and /ID > 0.5 and MetadataStream > 0.5) then | 4530 | 4530 | 100 |
| 4 | If (Malicecontent <= 0.5 and /XFA <= 0.5 and Filesize_kb > 10.25 and Headerlength <= 39.5) then | 7133 | 7133 | 100 |
| 5 | If (/XML > 0.5 and MetadataStream > 0.5 and obj > 6.5 and %EOF <= 9.0 and /ID > 0.5 and stream > 2.5 and /Font > 0.5 and Headerlength <= 62.0) then | 4434 | 4433 | 99.97 |
| 6 | If (%EOF <= 1.5 and Malicecontent <= 0.5 and /Font > 0.5 and MetadataStream > 0.5 and /ProcSet <= 1.5) then | 411 | 409 | 99.51 |
| 7 | If (/Producer <= 0.5 and Referencing > 36.0 and Malicecontent <= 0.5) then | 85 | 84 | 98.82 |
| 8 | If (/Producer > 0.5 and Malicecontent <= 0.5 and %EOF <= 1.5 and endobj <= 18.5 and xref > 0.5 and Filesize_kb > 12.82) then | 819 | 817 | 99.75 |
| 9 | If (/Producer > 0.5 and Malicecontent <= 0.5 and %EOF > 1.5 and /ID > 0.5 and MetadataStream <= 0.5 and /Font <= 140.5 and /ProcSet <= 0.5 and /XFA <= 0.5) then | 13 | 12 | 92.30 |
| 10 | If (/Producer > 0.5 and Malicecontent > 0.5 and endobj > 76.0 and /S <= 163.5 and /Font > 45.5 and stream <= 100.5 and /ProcSet > 0.5) then | 31 | 30 | 96.77 |
| 11 | If (/Producer > 0.5 and Malicecontent > 0.5 and endobj > 76.0 and /S > 163.5 and /Rect > 5.0 and /XML <= 14.0 and /OpenAction <= 0.5) then | 129 | 127 | 98.44 |
| 12 | If (Malicecontent <= 0.5 and /XFA <= 0.5 and Filesize_kb <= 10.25 and /Size <= 1.5 and Filesize_kb <= 1.85 and /Rect <= 1.0) then | 15 | 14 | 93.33 |
| 13 | If (Malicecontent > 0.5 and stream <= 24.5 and endobj > 62.5 and Filesize_kb > 20.8 and /ID > 1.5) then | 16 | 15 | 93.75 |
| 14 | If (/Size <= 1.5 and Malicecontent > 0.5 and obj <= 95.5 and obj > 16.5 and Headerlength > 31.5 and /JS <= 0.5 and /Font <= 4.0 and stream > 7.5 and Referencing <= 20.0) then | 35 | 34 | 97.14 |
| 15 | If (/Size <= 1.5 and Malicecontent > 0.5 and obj <= 95.5 and obj <= 16.5 and /XFA <= 0.5 and CreationDate > 0.5 and Headerlength > 57.0 and smallcontent <= 0.5 and /S <= 62.0 ) then | 19 | 17 | 89.47 |

**TABLE 12.** Top decision rules extracted from Random Forest classifier to detect malicious PDF.

| Rule ID | Conditions of the Rule | Total Sample | Correct Predictions | Confidence (%) |
|---|---|---|---|---|
| 1 | If (/XML <= 0.5 and Optimized <= 0.5 and Filesize_kb <= 46.52 and Headerlength > 39.0 and xref <= 1.5 and Filesize_kb > 1.41 and /CreationDate <= 0.5 and Headerlength <= 63.0) then | 5079 | 5079 | 100 |
| 2 | If ( /Size <= 1.5 and Malicecontent > 0.5 and obj <= 95.5 and obj <= 16.5 and /XFA <= 0.5 and /CreationDate > 0.5 and Headerlength <= 57.0 and /Rect <= 4.5 and Pages <= 1.5) then | 1267 | 1267 | 100 |
| 3 | If (Malicecontent > 0.5 and stream <= 24.5 and endobj <= 62.5 and /OpenAction > 0.5 and Filesize_kb <= 11038.86 and /JS > 0.5 and /Info <= 0.5 and Headerlength <= 63.0) then | 3769 | 3769 | 100 |
| 4 | If (/Size <= 1.5 and Malicecontent <= 0.5 and /CreationDate <= 0.5 and obj <= 76.0 and Pages <= 33.5 and endobj > 11.5) then | 528 | 528 | 99.62 |
| 5 | If (Filesizekb <= 18.27 and xref <= 1.5 and /JS > 0.5 and /ProcSet <= 105.0 and /S <= 372.5 and Headerlength <= 57.0 and Referencing <= 126.0) then | 5719 | 5717 | 99.96 |
| 6 | If (Malicecontent <= 0.5 and /XFA <= 0.5 and Filesize_kb > 10.25 and Headerlength > 39.5 and /Font <= 6.0 and Optimized <= 0.5) then | 57 | 56 | 98.24 |
| 7 | If (Malicecontent > 0.5 and stream <= 24.5 and endobj <= 62.5 and /OpenAction <= 0.5 and Headerlength > 57.0 and obj > 20.0 and endobj > 25.5 and /Font > 7.0) then | 82 | 79 | 96.34 |
| 8 | If (/XML <= 0.5 and Optimized > 0.5 and Headerlength > 39.5 and /ProcSet <= 1.5 and ID <= 1.5) then | 17 | 16 | 94.11 |
| 9 | If (Malicecontent > 0.5 and stream <= 24.5 and endobj > 62.5 and Filesize_kb <= 20.88 and /Font > 1.5 ) then | 24 | 22 | 91.67 |
| 10 | If (%EOF <= 1.5 and Malicecontent <= 0.5 and /Font > 0.5 and MetadataStream <= 0.5 and stream <= 8.5 and /XFA <= 0.5 and /Producer <= 0.5) then | 11 | 10 | 90.90 |







IEEE Access

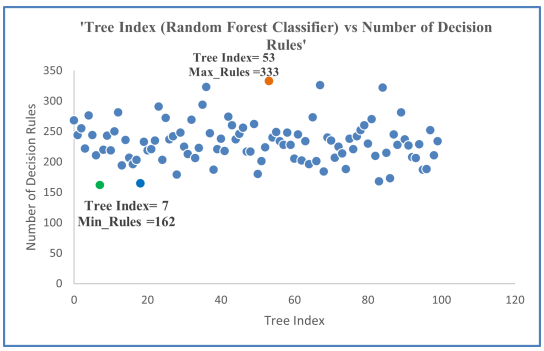


**FIGURE 25. Number of decision rules derived from decision trees of the Random Forest classifier.**

Similarly, the rest of the decision rules of Table 12 can be explained and explained and interpreted for recognizing malicious PDFs. Nevertheless, these crucial decision rules mentioned in Table 11 and Table 12 can significantly contribute to a clear understanding of humans for categorizing benign and malicious PDFs.

Finally, to assess this study of PDF malware detection, we perform a comparison with various existing works of the same study discipline. Table 13 summarizes this comparative study, in which our work is evaluated from a variety of perspectives, including the PDF sample source, the number of samples considered for the study, PDF labels, PDF analysis tools, the total number of PDF features considered for the study, the number of derived features developed for the analysis, the machine learning model used in the study, the accuracy observed during the study, and whether the study provides decision rules and human interpretation. The authors of [1] described a method that used machine learning classifiers to evaluate a given PDF both statistically and interactively to identify the hazardous nature of the document. They ran their trials on 1200 PDF samples with PhoneyPDF (a PDF analysis tool) and discovered that the Random Forest classifier was the best fit to detect malicious PDFs, with an accuracy of 98.6%. Similarly, the authors in [33] implemented the Random Forest classifier to identify malicious PDFs. They only used 1000 PDF samples in their pilot investigation and employed the PDFiD and PeePDF PDF analysis tools to extract the potential features that were critical for the classification task. However, their suggested strategy provided a maximum accuracy of 97.4% in completing the task. Besides, the authors in [4] proposed an approach called O-DT (Optimizable Decision Tree) for PDF malware detection. The authors utilized a benchmark dataset to perform their intended experiments and got an accuracy of 98.84% for the suggested approach. The study in [46] presented a dataset consisting of 10,025 PDF samples based on evasive characteristics of PDF files. Moreover, the authors suggested an ensemble classifier based on stacking learning which provided 98.69% accuracy to detect PDF malware. However, we notice that our work outperforms the existing works presented in Table 13, covering a large number of PDF samples with the use of three advanced tools for feature extraction, deriving important features, and utilizing the power of the Random Forest classifier to achieve a much better accuracy of 99.24% for detecting PDF malware. Furthermore, to the best of our knowledge, none of the research presented in Table 13 gives a thorough human interpretation of the classifier's performance by illustrating a decision tree and identifying decision rules for PDF malware detection.

## V. DISCUSSION

We created a dataset considering the malicious, clean, and evasive PDFs to detect PDF malware. However, we take only 792 evasive PDFs which is approximately 5% of the entire dataset. The intuition of introducing the evasive PDFs is to reduce the bias of the classifier. Moreover, the evasive characteristics of the PDFs make the classifier more robust in the detection of PDF malware. We maintained an approximately balanced distribution between the benign and malicious PDFs to overcome the problem of skewness of the classifier to a particular class. To analyze the PDFs and extract the useful features, we used three well-known and highly accurate tools PDFiD, PDFINFO, and PDF-PARSER. The idea of using these tools is to develop an effective feature set for PDF malware detection by exploring multiple efficient tools that ensure the acceptability of the extracted standard features of the PDFs. Additionally, we derived a few important features and merged them with the standard features to generate a merged feature set. We built the final feature set by generating subsets from the merged feature set and assessed them utilizing the strength of the classifier.

We performed an in-depth experimental analysis of the final feature set to explain the traits of the feature set. We found the title length of the PDF files is crucial, as malicious files tend to have an unusual length of the title compared to clean files. Furthermore, metadata and structural features such as */Producer*, */ProcSet*, */ID*, */CreationDate* etc. are frequently observed inside the clean PDFs and seldomly found within the harmful ones. Attackers usually keep fraudulent files as small as possible by restricting the contents, pages, fonts, and size with the intuition of carrying out their attacks as swiftly as possible. We discovered that cyber criminal's primary intention is to insert malice-related contents such as inserting JavaScript code, OpenAction files, etc. within the structure of the PDFs to harm the victim's systems.

We provided an explicit interpretation and explanation of the classifier's performance by generating a decision tree from one of the classifier's estimators, as well as highlighting a few critical decision rules for recognizing malicious and clean PDFs. We discovered some strong decision rules for recognizing both types of PDFs that provide up to 100% confidence and can identify a large number of samples; nevertheless, we noticed a number of rules that require a significant number of constraints to be verified, making them somewhat less effective. In addition to that, these weak rules can accurately identify a small number of samples yet yield

**TABLE 13.** Comparision of our work with various existing studies for PDF malware detection.

| Reference | PDF Sample Source | No. of Samples | PDF Labels | PDF AnalysisTools | No. of Features | No. of Derived Features | Model | Accuracy (%) | Rule Discovery and Human Interpretation |
|---|---|---|---|---|---|---|---|---|---|
| [1]/2022 | Internet | 1200 | Safe- 571<br>Malicious-629 | PhoneyPDF | - | - | Random Forest | 98.6 | No |
| [4]/2022 | Evasive-PDFMal2022 | 10,025 | Benign Evasive- 4468<br>Malicious Evasive- 5557 | - | 37 | - | O-DT | 98.84 | No |
| [33]/2021 | VirusTotal,<br>Contagio,<br>Personal Files | 1000 | Clean-470,<br>Malicious-530 | PDFiD, PeePDF | 14 | 05 | Random Forest | 97.4 | No |
| [46]/2022 | Evasive-PDFMal2022 | 10,025 | Benign Evasive- 4468<br>Malicious Evasive- 5557 | - | 37 | - | Stacking | 98.69 | No |
| **This Paper** | VirusTotal,<br>Contagio,<br>Evasive-PDFMal2022 | 15,958 | Benign- 7500<br>Malicious- 7666<br>Benign Evasive-400<br>Malicious Evasive-392 | PDFiD,<br>PDFINFO,<br>PDF-PARSER | 28 | 07 | Random Forest | 99.24 | **Yes** |

high confidence. However, the decision rules offer a clear understanding and interpretation of how the features can be utilized to detect PDF malware.

In this study, we added evasive behaviors to our experimental dataset to make our classifier more resilient. Nevertheless, the strength of the classifier can be enhanced to combat modern advanced attacks more precisely if we can include more evasive PDF properties through careful inspection. We extracted characteristics in this experiment using three tools: PDFiD, PDFINFO, and PDF-PARSER. These tools while very popular, are known to have vulnerabilities (for instance, PDFiD tool) to some attacks. One such attack is known as the parser confusion attack where the fraudulent material is disguised and concealed using a variety of approaches to avoid detection while retaining the ability to execute and exploit. Also, run-time and other dynamic characteristics may be leveraged to further investigate questionable documents. We intend to address each of these constraints in our future work. Additional analysis can be performed by combining aspects from various parsers and analysis techniques to investigate complicated content such as JavaScript code.

Furthermore, the present feature set is derived from three extraction methods, and the features employed by the three programs depend on heuristics and insights made by their developers. A greater comprehensive feature set can be added by incorporating new sources, such as malicious document generation tools or in-depth study of malicious PDF documents. One such analysis can be to consider the internal text of malicious PDFs where the attackers can hide their harmful code segment behind the text content. Also, we want to assess the generalizability of our suggested method against multiple types of PDF malware by investigating how the model performs against different types of PDF malware, including newer or more advanced variations. Plus, we want to implement the proposed method in real-world scenarios or simulations to justify its practical effectiveness in identifying various types of PDF malware. Besides, we intend to investigate certificateless signcryption [48] and proxy signcryption [49] as advanced strategies for safeguarding PDFs which can add additional layers of security for PDFs that could potentially mitigate the risks posed by PDF malware and leading the way for future research that integrates cryptographic techniques with malware detection. In addition to that, adversarial PDF malware still poses a great threat to a secure cyberspace. In the future, to combat such threats we want to develop a data-driven intelligent approach that can tackle adversarial PDF malware effectively. Besides, we want to publish an additional dataset comprised solely of evasive PDF samples covering a wide range of approaches to cyber attacks.

## VI. CONCLUSION

In this study, we performed an extensive analysis for PDF malware detection. For this, we first developed a comprehensive dataset of 15958 PDF samples by taking into account the non-malicious, malicious, and evasive natures of the PDF samples. We also developed a method to generate an effective and explainable feature set by extracting important traits from our freshly constructed dataset's PDF samples using multiple PDF analysis tools. Further, we also derived features that are empirically demonstrated to be useful for classifying PDF malware. We investigated different machine learning classifiers and highlighted the effectiveness of the Random Forest model not only for performance comparison but also for the explainability analysis with generating decision rules. Moreover, we clarified the behaviors of the characteristics in charge of detecting PDF malware and pointed out a few relevant observations that may aid in the detection of hazardous PDF files. Finally, we compared our findings to several state-of-the-art research and highlighted some key observations of our study.

## REFERENCES

[1] S. S. Alshamrani, "Design and analysis of machine learning based technique for malware identification and classification of portable document format files." *Security & Communication Networks*, vol. 2022, 2022.

[2] P. Singh, S. Tapaswi, and S. Gupta, "Malware detection in pdf and office documents: A survey," *Information Security Journal: A Global Perspective*, vol. 29, no. 3, pp. 134–153, 2020.

[3] N. Livathinos, C. Berrospi, M. Lysak, V. Kuropiatnyk, A. Nassar, A. Carvalho, M. Dolfi, C. Auer, K. Dinkla, and P. Staar, "Robust pdf document conversion using recurrent neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 17, 2021, pp. 15 137–15 145.

[4] Q. Abu Al-Haija, A. Odeh, and H. Qattous, "Pdf malware detection based on optimizable decision trees," *Electronics*, vol. 11, no. 19, p. 3142, 2022.

[5] Y. Wiseman, "Efficient embedded images in portable document format," *International Journal*, vol. 124, pp. 129–38, 2019.

[6] M. Ijaz, M. H. Durad, and M. Ismail, "Static and dynamic malware analysis using machine learning," in *2019 16th International bhurban conference on applied sciences and technology (IBCAST)*. IEEE, 2019, pp. 687–691.

[7] Y. Alosefer, "Analysing web-based malware behaviour through client honeypots," Ph.D. dissertation, Cardiff University, 2012.

[8] N. Idika and A. P. Mathur, "A survey of malware detection techniques," *Purdue University*, vol. 48, no. 2, pp. 32–46, 2007.

[9] M. Abdelsalam, M. Gupta, and S. Mittal, "Artificial intelligence assisted malware analysis," in *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, 2021, pp. 75–77.

[10] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, "Botmark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors," *Information Sciences*, vol. 511, pp. 284–296, 2020.

[11] N. Šrndić and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 197–211.

[12] D. Maiorca, I. Corona, and G. Giacinto, "Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 119–130.

[13] S. Atkinson, G. Carr, C. Shaw, and S. Zargari, "Drone forensics: The impact and challenges," *Digital Forensic Investigation of Internet of Things (IoT) Devices*, pp. 65–124, 2021.

[14] C. Liu, C. Lou, M. Yu, S. Yiu, K. Chow, G. Li, J. Jiang, and W. Huang, "A novel adversarial example detection method for malicious pdfs using multiple mutated classifiers," *Forensic Science International: Digital Investigation*, vol. 38, p. 301124, 2021.

[15] Q. A. Al-Haijaa and A. Ishtaiwia, "Machine learning based model to identify firewall decisions to improve cyber-defense," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 11, no. 4, pp. 1688–1695, 2021.

[16] D. Stevens, "pdfid (Version 0.2.8)," 2023, [Software]. [Online]. Available: https://blog.didierstevens.com/programs/pdf-tools

[17] pdf-info, "pdf-info (Version 2.1.0)," 2021, [Software]. [Online]. Available: https://pypi.org/project/pdf-info/

[18] D. Stevens, "pdf-parser (Version 0.7.8)," 2023, [Software]. [Online]. Available: https://blog.didierstevens.com/programs/pdf-tools

[19] M. Yu, J. Jiang, G. Li, C. Lou, Y. Liu, C. Liu, and W. Huang, "Malicious documents detection for business process management based on multi-layer abstract model," *Future Generation Computer Systems*, vol. 99, pp. 517–526, 2019.

[20] H. Pareek, P. Eswari, N. S. C. Babu, and C. Bangalore, "Entropy and n-gram analysis of malicious pdf documents," *International Journal of Engineering*, vol. 2, no. 2, 2013.

[21] C. Smutz and A. Stavrou, "Malicious pdf detection using metadata and structural features," in *Proceedings of the 28th annual computer security applications conference*, 2012, pp. 239–248.

[22] D. Maiorca, G. Giacinto, and I. Corona, "A pattern recognition system for malicious pdf files detection," in *International workshop on machine learning and data mining in pattern recognition*. Springer, 2012, pp. 510–524.

[23] H. Pareek, P. Eswari, and N. S. C. Babu, "Malicious pdf document detection based on feature extraction and entropy," *International Journal of Security, Privacy and Trust Management*, vol. 2, no. 5, pp. 31–35, 2013.

[24] D. Maiorca, D. Ariu, I. Corona, and G. Giacinto, "A structural and content-based approach for a precise and robust detection of malicious pdf files," in *2015 international conference on information systems security and privacy (icissp)*. IEEE, 2015, pp. 27–36.

[25] N. Šrndić and P. Laskov, "Hidost: a static machine-learning-based detector of malicious files," *EURASIP Journal on Information Security*, vol. 2016, pp. 1–20, 2016.

[26] P. Laskov and N. Šrndić, "Static detection of malicious javascript-bearing pdf documents," in *Proceedings of the 27th annual computer security applications conference*, 2011, pp. 373–382.

[27] Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. P. Markatos, "Combining static and dynamic analysis for the detection of malicious documents," in *Proceedings of the Fourth European Workshop on System Security*, 2011, pp. 1–6.

[28] C. Vatamanu, D. Gavriluţ, and R. Benchea, "A practical approach on clustering malicious pdf documents," *Journal in Computer Virology*, vol. 8, no. 4, pp. 151–163, 2012.

[29] F. Schmitt, J. Gassen, and E. Gerhards-Padilla, "Pdf scrutinizer: Detecting javascript-based attacks in pdf documents," in *2012 tenth annual international conference on privacy, security and trust*. IEEE, 2012, pp. 104–111.

[30] S. Karademir, T. Dean, and S. Leblanc, "Using clone detection to find malware in acrobat files," in *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, 2013, pp. 70–80.

[31] X. Lu, J. Zhuge, R. Wang, Y. Cao, and Y. Chen, "De-obfuscation and detection of malicious pdf files with high accuracy," in *2013 46th Hawaii International Conference on System Sciences*. IEEE, 2013, pp. 4890–4899.

[32] I. Corona, D. Maiorca, D. Ariu, and G. Giacinto, "Lux0r: Detection of malicious pdf-embedded javascript code through discriminant analysis of api references," in *Proceedings of the 2014 workshop on artificial intelligent and security workshop*, 2014, pp. 47–57.

[33] A. Falah, L. Pan, S. Huda, S. R. Pokhrel, and A. Anwar, "Improving malicious pdf classifier with feature engineering: A data-driven approach," *Future Generation Computer Systems*, vol. 115, pp. 314–326, 2021.

[34] Virustotal. Available online. [Online]. Available: https://www.virustotal.com/gui/home/upload

[35] 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," *Applied Sciences*, vol. 9, no. 22, p. 4764, 2019.

[36] D. Maiorca and B. Biggio, "Digital investigation of pdf files: Unveiling traces of embedded malware," *IEEE Security & Privacy*, vol. 17, no. 1, pp. 63–71, 2019.

[37] N. Nissim, A. Cohen, C. Glezer, and Y. Elovici, "Detection of malicious pdf files and directions for enhancements: A state-of-the art survey," *Computers & Security*, vol. 48, pp. 246–266, 2015.

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

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

[40] C. Smutz and A. Stavrou, "When a tree falls: Using diversity in ensemble classifiers to identify evasion in malware detectors." in *NDSS*, 2016.

[41] 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*. IEEE, 2017, pp. 218–224.

[42] D. Liu, H. Wang, and A. Stavrou, "Detecting malicious javascript in pdf through document instrumentation," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 100–111.

[43] N. Šrndic and P. Laskov, "Detection of malicious pdf files based on hierarchical document structure," in *Proceedings of the 20th annual network & distributed system security symposium*. Citeseer, 2013, pp. 1–16.

[44] Canadian Institute for Cybersecurity (CIC). (2022) Pdf dataset: Cic-evasive-pdfmal2022. Available online. [Online]. Available: https://www.unb.ca/cic/datasets/pdfmal-2022.html

[45] (2013) Contaigo, 16,800 clean and 11,960 malicious files for signature testing and research. Available online. [Online]. Available: http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html

[46] M. Issakhani, P. Victor, A. Tekeoglu, and A. H. Lashkari, "Pdf malware detection based on stacking learning." in *ICISSP*, 2022, pp. 562–570.

[47] E. Frank, M. A. Hall, and I. H. Witten, *The WEKA Workbench*, 4th ed. Morgan Kaufmann, 2016, online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques". [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/book.html

[48] I. Ullah, N. Ul Amin, M. Zareei, A. Zeb, H. Khattak, A. Khan, and S. Goudarzi, "A lightweight and provable secured certificateless signcryption approach for crowdsourced iiot applications," *Symmetry*, vol. 11, no. 11, 2019. [Online]. Available: https://www.mdpi.com/2073-8994/11/11/1386

[49] A. Waheed, A. I. Umar, M. Zareei, N. Din, N. U. Amin, J. Iqbal, Y. Saeed, and E. M. Mohamed, "Cryptanalysis and improvement of a proxy signcryption scheme in the standard computational model," *IEEE Access*, vol. 8, pp. 131 188–131 201, 2020.

**G.M. SAKHAWAT HOSSAIN** received Bachelor of Science in Computer Science and Engineering at Rajshahi University of Engineering and Technology. He is currently pursuing his Master of Science in Computer Science and Engineering at Chittagong University of Engineering and Technology. He is also serving as a Lecturer at Rangamati Science and Technology University, Chattogram. His research interest includes Malware Analysis, Natural Language Processing, Computer Vision, and Machine Learning.

**KAUSHIK DEB** received the B.Tech. and M.Tech. degrees from the Department of Computer Science and Engineering, Tula State University, Tula, Russia, in 1999 and 2000, respectively, and the Ph.D. degree in Electrical Engineering and Information Systems from the University of Ulsan, Ulsan, South Korea, in 2011. Since 2001, he has been a Faculty Member of the Department of Computer Science and Engineering (CSE), Chittagong University of Engineering and Technology (CUET), Chattogram, Bangladesh. He is currently a Professor with the Department of CSE, CUET. Moreover, he was in various administrative positions with CUET, such as the Dean of the Faculty of Electrical and Computer Engineering (ECE), from 2017 to 2019, the Director of the Institute of Information and Communication Technology (IICT), from 2015 to 2017, and the Head of the CSE Department, from 2012 to 2015. He made a variety of contributions to managing and organizing conferences, workshops, and other academic gatherings. He has published more than 110 technical papers with peer reviews. His research interests include computer vision, deep learning, pattern recognition, intelligent transportation systems (ITSs), and human–computer interaction. He was a Steering Member. He acted as the Chair or a Secretary in a variety of international and national conferences: the International Conference on Electrical, Computer, and Communication Engineering (ECCE), the International Forum on Strategic Technology (IFOST), the International Workshops on Human System Interactions (HSI), and the National Conference on Intelligent Computing and Information Technology (NCICIT).

**HELGE JANICKE** is a Professor of Cybersecurity at Edith Cowan University (ECU), Australia. He is the Director of ECU's Security Research Institute and the Research Director for Australia's Cyber Security Cooperative Research Centre. He optained his PhD in 2007 from De Montfort University, UK, where he established DMU's Cyber Technology Institute and its Airbus Centre of Excellence for SCADA cybersecurity and digital forensics research, as well as heading up DMU's School of Computer Science. His research interests are Cybersecurity in Critical Infrastructure, Human Factors of Cybersecurity, Cybersecurity of Emerging Technologies, Digital Twins and Industrial IoT.

**IQBAL H. SARKER** is a Research Fellow of the Cyber Security Cooperative Research Centre (CSCRC) in association with the ECU Security Research Institute, Edith Cowan University (ECU), Australia. He received his Ph.D. in Computer Science from Swinburne University of Technology, Melbourne, Australia in 2018. His research interests include Cybersecurity, AI/XAI and Machine Learning, Data Science and Behavioral Analytics, Digital Twin, Smart City Applications and Critical Infrastructure Security. He has published 100+ journal and conference papers in various reputed venues published by Elsevier, Springer Nature, IEEE, ACM, Oxford University Press, etc. Moreover, he is a lead author of a research monograph BOOK titled "Context-Aware Machine Learning and Mobile Data Analytics", published by Springer Nature, Switzerland, 2021. He has also been listed in the world's top 2% of most-cited scientists, published by Elsevier & Stanford University, USA. In addition to research work and publications, Dr. Sarker is also involved in a number of research engagement and leadership roles such as Journal editorial, international conference program committee (PC), student supervision, visiting scholar and national/international collaboration. He is a member of IEEE, ACM and Australian Information Security Association.