IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 May 2026, accepted 22 June 2026, date of publication 25 June 2026, date of current version 1 July 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3707213

RESEARCH ARTICLE

RIT-PDFMal-2026: A Comprehensive Benchmark Dataset for PDF Malware Detection

MOHAMMED M. ALANI[1], (Senior Member, IEEE), AND ERNESTO DAMIANI[2], (Senior Member, IEEE)
[1]Rochester Institute of Technology of Dubai, Dubai, United Arab Emirates
[2]Università degli Studi di Milano, 20122 Milan, Italy

Corresponding author: Mohammed M. Alani (m@alani.me)

This work was supported by Rochester Institute of Technology of Dubai under Grant RITD-ARC-2025-001.

ABSTRACT Portable Document Format (PDF) has gained rapid popularity in the past two decades due to its platform-independence, and portability. A single PDF file can contain all the elements needed to convey a document in a fixed format, such as text, images, and fonts. This popularity made PDF documents an important attack vector for malicious activity. In this paper, we present a comprehensive, carefully curated benchmark dataset for PDF malware detection. The dataset captures a large number of real-life malicious and benign samples to build a 24,337-samples dataset suitable for training and testing of machine learning-based malicious PDF detection systems. The work also presents an extended set of static features extracted from PDF files to support the detection process. The presented dataset was used in the training and testing of a pipeline of different machine learning classifiers. The testing results showed that the extreme gradient-boost classifier was capable of detecting malicious PDF with F_1 score exceeding 0.99. The model also performed similarly well in 10-fold cross validation. Additionally, the model was explained using Shapley additive explanation to ensure the model's transparency, increase confidence, and ensure the alignment of the classifier decision with human experience.

INDEX TERMS Detection, machine learning, malware, pdf.

I. INTRODUCTION

According to a report published in 2026 [1], approximately 402 million terabytes of data is generated every day. A total of 181 zettabytes of data were generated in 2025, while the projected data to be generated in 2026 is 221 zettabytes. These huge numbers indicate a steady, if not rapid, growth in reliance on internet-connected devices and internet-based data exchange.

Another report published in 2025 by CloudFiles says that on a daily basis, about 100 million invoices, 50 million payslips, and 200-300 million reports, contracts, and certificates are generated, in Portable Document Format (PDF) [2]. The report claims that the total number of PDF files around the world is estimated to be 2.5 trillion files.

PDF was first introduced by Adobe in 1992 as a file format that is capable of containing text, images, and formatting information combined into one file [3]. The goal was to avoid the need to have separate types of content in separate files and to maintain formatting information such as font types and sizes. This proved particularly important when smart phones and tablet devices became more popular and documents needed to be accessible not only across different computer platforms, but also on mobile devices.

PDF has witnessed steady growth in adoption over the past three decades. This growth stems from the platform-independence, interactivity, and portability of this information exchange format.

The ability to embed objects, such as images, fonts, and files, within a PDF file for portability reasons has also enabled malicious actors to embed malicious content. In some cases, malicious content is in the form of Javascript, to exploit vulnerability in an internet browser, such as

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

CVE-2025-1918 in Google Chrome. This vulnerability, with a Common Vulnerability Scoring System CVSS-score of 8.8, enables the attacker to perform buffer overflow attack on Google Chrome PDFium component used to render PDF documents inside the browser. By exploiting this vulnerability, a remote attacker can perform out-of-bounds memory access, which could lead to information disclosure or arbitrary code execution [4].

In other cases, attackers embed code to exploit vulnerabilities in PDF readers, such as CVE-2018-4990. With a CVSS-score of 8.8 as well, this vulnerability in Adobe Reader software also enables the attacker to execute arbitrary code in the context of the reader application. Even in the case of failed exploitation, the attack will result in denial-of-service [5].

In addition to such vulnerabilities, there are inherent risks in the design of PDF files. The file architecture, discussed in Section II, enables embedding different types of objects inside the file to ensure portability. This same feature can be easily misused by malicious actors to embed malicious content. The attacker can rely on simple spoofing techniques or just built-in commands in the PDF files to trigger the execution of malicious code simply by opening the file in a reader or in a browser. Recent detection research attempted to treat PDFs as binary files to facilitate better detection [6].

PDF malware detection, like other malware types, relies on static analysis, dynamic analysis, or a hybrid of both. Static analysis examines the raw content of the file without running the file or activating it. The goal is to inspect the file for malicious indicators without triggering the malware. Dynamic analysis involves running the file and activating the malware while monitoring its behavior in a safe environment. Although behavioral indicators can help overcome sophisticated obfuscation, static analysis remains a faster, safer, and less expensive choice.

A. PROBLEM STATEMENT

PDF file format has become a worldwide standard for document exchange due to its portability and platform independence, while maintaining presentation and formatting. It is now the standard form of information produced by billing systems, books, catalogs, and virtually any form of document exchange over the internet. With that expansion in use, its misuse by threat actors has become more prevalent.

In 2019, Nissim et al. conducted a thorough study that inspected about 2 million PDF files published on a famous scientific papers search engine named CiteSeerX [7]. The study found that about 2% of these PDF files were malicious or contained malicious content. This indicates that the use of malicious PDF files is widespread and makes the detection of malicious PDF files an urgent research problem that requires addressing. Such research has witnessed noticeable hurdles due to the lack of a benchmark dataset that can help researchers study the structure and features of malicious PDF files, to build efficient, effective, and robust detection systems.

Existing datasets, such as [8], [9], and [10], suffer from multiple issues such as using significantly outdated samples, utilizing small number of samples, or extracting a limited number of features that do not necessarily capture the essence of malicious content within PDF files. Other issues were also noticed in older datasets, such as bias toward a specific type of malicious malware.

B. RESEARCH CONTRIBUTION

In this paper, we present the following contributions:

1) Building a new realistic dataset of malicious PDF files with more than 24,000 samples.
2) Building a reliable and extensible feature extraction tool that is capable of extracting features accurately from PDF files.
3) Training and testing a PDF malware detection system based on machine learning to test a newly built dataset.
4) Utilizing SHAP explanations to analyze the behavior of the malware detection system and find the most important features contributing to the detection process.

C. PAPER LAYOUT

The next section explains preliminary information about PDF file architecture and format. Related works are reviewed and summarized in Section III. The details of the dataset collection, extraction and curation are presented in Section IV. Section V presents the use of the dataset in building a pipeline of machine learning classifiers along with testing and validation results. The detection model's explainability using SHAP is introduced in Section VI. The discussion of the results along with a comparative analysis can be found in Section VII. The study conclusions, and potential future research directions are finally presented in Section VIII.

II. PRELIMINARIES

A. HISTORY OF THE PDF STANDARD

PDF standard specifications were published in eight version from 1993 to 2007 the PDF, numbered 1.0 to 1.7 by Adobe Systems as a freely available, and royalty-free standard to encourage its use. In 2008, the International Organization of Standardization (ISO), adopted the PDF reference 1.7, with minor adjustments, as ISO 32000-1 standard [3].

PDF 2.0 standard was published by ISO in December 2020, named ISO 32000-2:2020. In April 2023, ISO, Adobe, Apryse, and Foxit collaborated to make the standard available worldwide at no cost [3].

B. PDF FILE ARCHITECTURE

The architecture of the PDF file is shown in Figure 1. Typically, a PDF file is composed of the following parts:

1) Header: The header, located in the first 10 bytes of the file, includes the PDF version. This version number identifies the standard reference used in building the file, such as 1.0, 1.2, or 2.0. The version is written in ASCII format such as `%PDF-1.7%`.

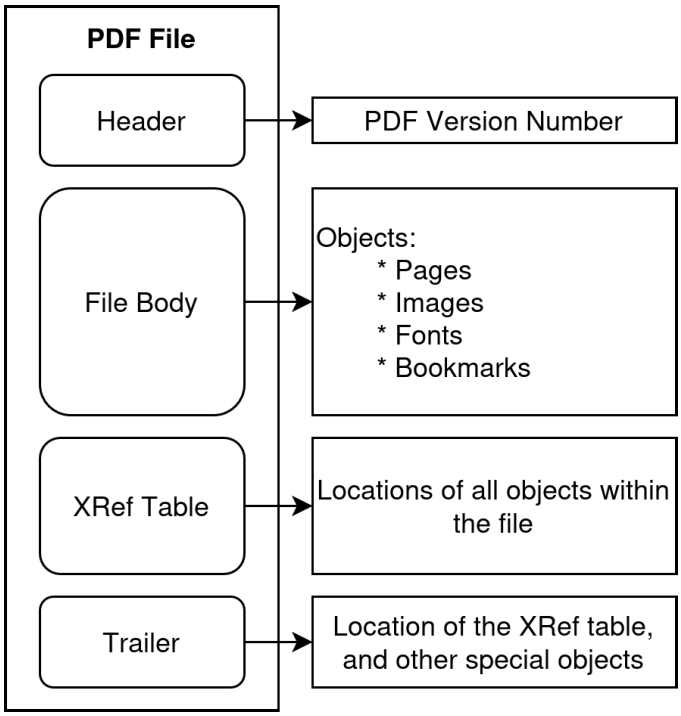


FIGURE 1. PDF file architecture.

2) File body: Located immediately after the file header, and contains objects. The beginning of each object is identified with `obj.≪`, while the end of the object is denoted with `≫.endobj`. These objects can be pages, images, fonts, or bookmarks. Practically, any type of file, including portable executable (PE), can be embedded as an object. The goal of the capability is to have the document viewed in a predetermined way regardless of the platform, operating system, or device on which it is viewed.
3) Crossreference table (XREF Table): A table that identifies the locations of all objects within the document. The table starts with a `.xref` marker and then lists the objects and their locations within the file.
4) Trailer: The trailer's only job is to identify the location of the XREF table, and any other special object. The trailer starts with a `.trailer.≪`, and contains a pointer to the XREF table location `≫.startxref`, and the trailer ends with `%%EOF.` marker.

A reader is expected to start reading the file from the trailer to identify the location of the XREF table, and then start extracting and rendering these objects one by one.

For extensibility purposes, the PDF architecture allows editing the file to add or remove objects from the file. The capability of incremental updates means that the file will keep its original content, but add new body, xref table, and trailer. In these incremental updates, the editor can remove parts of the existing object or update them by attaching new parts at the end of the file. If a file undergoes one update, then it will have the header, original body, original XREF table, original trailer, body update 1, XREF table update 1, trailer update 1, in that sequence. The updates will mark the deleted objects as "Deleted" so that the reader would not load them. This enabled PDF files to serve as forms and other fillable documents. It also enabled PDF files to be signed.

C. MALICIOUS USE OF PDFS

PDF files rely on "commands", and "keywords" that enable them to perform certain actions, such as clearing the fillable content in a PDF form. While most of these commands are commonly used in benign PDF files, many can be misused by malicious actors. Some of these commonly misused commands/keywords include the following:

- /JavaScript and /JS can be used to embed malicious JavaScript to perform system exploits, session hijacking, or information stealth, among many other possible attacks.
- /OpenAction and /AA can be used to trigger an action to be executed automatically when the document is opened. This can be used to trigger downloading of other malicious content, or run malicious commands.
- /Launch can be used to open another document or launch another program.
- /URI can be used to connect to a malicious URL to download malciious content, or perform other types of phishing.
- /EmbeddedFile can be used to embed malicious files such as binary executable files.
- /XFA can be used to embed scripts through XML Forms Architecture.

The commands/keywords mentioned above is not a comprehensive list. There were other malicious keywords identified in our analysis that were later used to extract features from PDF files for detection purposes. Additionally, malicious actors rely on additional evasion techniques to avoid detection such as obfuscation of commands or keywords, or using deliberately corrupted XREF tables.

Kaspersky Labs published a report addressing the spread of PDF-based malware in 2025 [11]. The report mentioned the following means by which PDF malware can spread:

1) Form-based: A form of malware where the PDF file contains a form where the user is asked to type private information. The form then transfers this information to the malicious actor.
2) Fake CAPTCHA Redirects: A common type of malicious PDF where the user is tricked into a fake CAPTCHA verification process to be eventually redirected to a malicious website.
3) Static images that look like videos: A static image with a ''Play'' button on top that is designed to trick the user into pressing play, while the user gets redirected to a malicious website or to download malicious content. This type of PDF could lead to attacks such as Cross-Site Request Forgery (CSRF).
4) Fake attachments: PDF files are widely used in phishing campaigns. The malicious actor creates a PDF file that pretends to be an invoice or any other type of document to trick the user into opening the PDF. The PDF, in such scenario, is usually filled with embedded malicious code or malware that gets activated once the PDF is document is opened by a browser or any type of reader.

Malware detection can be categorized into signature-based and anomaly-based detection. In signature-based detection, the detector uses a set of predefined signatures to identify malware. These signatures could be part of the malware code, hashes, or even behaviors that distinctly identify the malware. Once there is a match, the file is flagged as malware [12]. While signature-based detection is highly accurate at detecting known-malware, it performs very poorly in detecting unknown-malware, because new malware does not match any existing signatures. In addition, malicious actors work extensively to evade detection by altering malware signatures using different obfuscation techniques.

In anomaly-based detection, the detector ''learns'' normal behavior or features to be able to flag any ''abnormal'' behavior or features. Although it might not necessarily provide the high accuracy achieved by signature-based detection, it performs much better, comparatively, in detecting unknown-malware [13]. Better detection of unknown-malware is due to the fact that it doesn't rely on signatures, but rather on whether the file, or behavior aligns with what is normal. In that context, machine learning presents itself as a suitable candidate for anomaly-based detection systems due to its superior capability to identify patterns in data and behavior. Such machine learning-based detection systems need reliable benchmark datasets to be used for training and testing.

III. RELATED WORKS

One of the oldest and most used datasets in PDF malwar research is Contagio malware dump, which was released in 2013 [14]. This dataset contained 11,152 raw malicious PDF files along with 9,000 non-malicious PDFs. The dataset, being raw files, was used in many research publications in different methods. Corum et al. proposed an image-based PDF malware detection method using the Contagio dataset [15]. Additionally, authors used a supplementary dataset of 1,500 reverse mimicry attack samples that were sourced from a publicly available repository. The proposed Markov plot based detector delivered an accuracy of 0.9939 and F_1 score of 0.9736.

Chen et al. presented, in 2020, a detection method that utilizes Contagio dataset [10]. The study was focused on identifying inherent data quality limitations in Contagio dataset that significantly impacts robustness of detectors trained using this dataset. While the research did not present a new dataset, it focused on presenting the dataset in a different approach where Hidost feature extractor is used to extract structural path features. The input features had 3,514 dimensions representing all the distinct path features. The paper used 19,338 samples, where 10,344 were malicious, and 8,994 were benign. The paper identifies the problem of highly-homogenous data within this dataset that results in nearly-perfect accuracy of 0.999.

In 2022, Issakhani et al. introduce Evasive-PDFMal-2022 [8]. The PDF malware dataset containing 10,025 samples, each containing 32 features. The goal of this dataset was to overcome the limitations identified in Contagio repository that was previously used to generate PDF malware datasets. The authors found that Contagio contained 44% duplicate entries and exhibited significant data bias, with 74% of malicious samples exploiting only JavaScript and OpenAction features. This makes the classifiers trained biased towards this specific type of attacks. The authors used an ensemble of base learners (Support Vector Machine, Random Forest, Multi-Layer Perceptron, and AdaBoost), in addition to Logistic Regression as the meta-classifier. Testing results showed an accuracy of 0.9869, and F_1 score of 0.9877.

In 2023, Liu et al. presented PdfRep [9]. A PDF malware dataset designed to address identified representativeness deficiencies in both the Contagio dataset and CIC-PDFMal-2022 (Evasive-PDFMal2022) datasets. The authors performed statistical analysis on real-world samples taken from VirusTotal to identify these representativeness deficiencies. The dataset presented was collected from Govdocs, VirusTotal, Contagio, and VirusShare. The dataset contains a total of 493,238 samples, with 170,479 benign and 322,759 malicious samples. The dataset was built by extracting only 15 features from the samples, which raises questions about the true representativeness of the samples

and whether 15 features are actually capable of capturing the variation of types that the paper aimed to address. The authors attempted using different types of classifiers, and tests showed the best performing classifier to be XGB, with an accuracy of 0.9878 and F_1 score of 0.9825. While this dataset is the largest published so far, it doesn't provide a detailed description of how the features were extracted, nor does it capture structural features that could help clearly distinguish malicious PDFs.

Complementing purely static approaches, behavior-based (dynamic) malware detection analyzes runtime actions such as system calls, API sequences, and thread-level behavior, offering robustness against obfuscation and polymorphism. Recent ensemble techniques have demonstrated that hardening behavioral classifiers against thread-division polymorphism significantly improves the detection of evasive variants [16]. Furthermore, generative adversarial networks (GANs) and related models have proven effective for simulating malicious behaviors, generating synthetic event graphs, and enriching limited malware datasets to counter concept drift and class imbalance [17]. These directions open promising avenues for hybrid static-dynamic pipelines and dataset augmentation that can leverage the structural insights provided by the present benchmark.

IV. DATASET

Malware analysis can be categorized into static analysis and dynamic analysis. Static analysis is performed by examining the malicious file without opening or running the malware [18]. This type of analysis relies on examining the file and extracting information that would help in detecting malware and attempting to understand the malware intent.

Dynamic malware analysis relies on activating the malware inside a sandbox or a safe environment and monitoring its behavior [19]. This technique helps in providing better understanding of the malware intent, and identifying all activities performed on the victim system.

While dynamic analysis provides deeper insights into the malware's behavior, it does not serve the purpose of detection. When detection relies on dynamic behavior, it can only happen after the malware has already been activated, which can cause irrevocable damage in many cases. On the other hand, when the detection relies on static analysis of the malicious file before it is activated, it can help avert the damage. Therefore, in our work, the dataset we presented is based on static features extracted from PDF files without the need to run these files. This can help build the capacity to detect malicious files without the risk of activating the malware first. The steps followed in creating the dataset are shown in Figure 2.

For improved efficiency and to reduce dimensionality without loss of performance, the three zero-variance features (Acroform, Colors, BaseEncoding) can be safely removed prior to training. In the remainder of this section, we will present the data collection and feature extraction methodologies, along with the labeling process.

TABLE 1. Number of samples collected.

Samples	Benign	Malicious	Total
Collected	13,242	11,243	
Corrupted	7	141	
Included in PDFMal	13,235	11,102	24,337

A. DATA COLLECTION

The malicious samples were obtained from VirusTotal [20]. The samples were real-life samples submitted by VirusTotal users, and collected during the period 2017 to 2025. VirusTotal collects samples through public user submissions, automated scanning of internet resources, and submissions by industry partners, in addition to honeypots and internet crawlers.

To collect benign samples, we built a specialized internet crawler to find and download benign PDF files. The crawler was created to automatically move to different websites and download random PDF files with the size range of 25kB-1.5MB. The range of file sizes was chosen to resemble the malicious file corpus obtained from VirusTotal.

Upon detailed examination of the collected files, we found that some of them were corrupted and could not be used for data extraction. Table 1 shows the number of samples collected in each category and the number of corrupted samples.

As shown in Table 1, the total number of usable samples collected was 24,337, out of which 13,235 were benign and 11,102 were malicious. These samples were stored in a sandbox environment to prevent accidental spillage of the malware into the research machines.

B. DATA EXTRACTION

Upon thorough analysis of malicious samples and detailed comparison with benign samples, we built an extraction tool, namely PDFMalExtract [21]. The tool was designed to extract 42 static features from the PDF files. The tool is open-sourced to enable researchers to further expand the tool with the evolving types of malware. PDFMalExtract is written in Python with the capacity to extract features from a large number of files in large batches, and it has the capability of labeling the extracted samples manually. Using the pyMuPDF library, the tool can read PDF files and extract features from these files [22]. A complete list of the extracted features along with their descriptions can be found in Table 2.

As shown in Table 2, the features extracted can be split into two types:

1) Meta data: Features such as the size of the file, metadata size, number of pages, PDF version, number of embedded files and number of images.
2) Strings: Features counting the number of occurrences of specific strings that are often seen associated with malicious PDF files, such as ''/JavaScript'', ''/Registry'', and ''/Launch''.

The meta data, such as the file size, number of images, number of embedded files, and PDF version were included

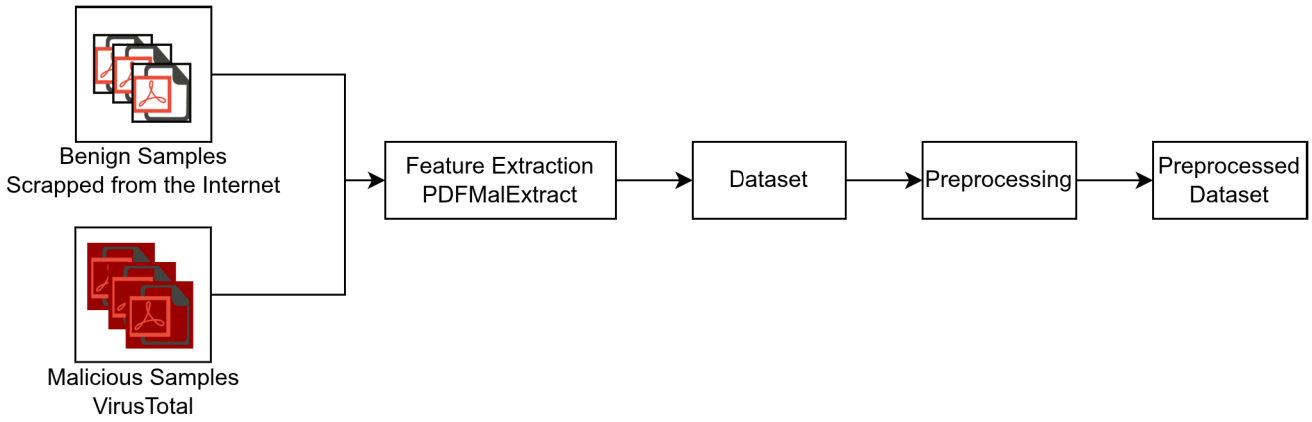


FIGURE 2. Dataset creation steps.

TABLE 2. Description of features extracted using PDFMalExtract.

Feature Name	Description
hash	The SHA256 hash of the PDF file. This should be removed before training the classifier. It was kept only to help detect duplications in the pdf files.
pdfsize	Size of the PDF file in kilobytes.
metadata size	Size of metadata extracted from the PDF file, in bytes.
pages	Number of pages in the PDF file.
xref length	xref length as extracted from the file header.
title length	Number of characters in the file title.
isEncrypted	1 If the file is encrypted, and 0 otherwise.
embedded files	Number of embedded files inside the PDF as extracted from the PDF header
images	Number of images in the PDF file. This will return -1 if the file is corrupted and doesn't show the number of images.
contains_text	Does the file contain a certain number of characters of text? (the default threshold is 50 characters). This feature shows 1, if the threshold is met, 0 if it wasn't met, and -1 if the file is corrupted and the text couldn't be read.
pdf_ver	Version of the PDF file, as extracted fro mthe file header.
obj	Number of occurrences of the string "obj" in the PDF file.
endobj	Number of occurrences of the string "endobj" in the PDF file.
stream	Number of occurrences of the string "stream" in the PDF file.
endstream	Number of occurrences of the string "endstream" in the PDF file.
trailer	Number of occurrences of the string "trailer" in the PDF file.
xref	Number of occurrences of the string "xref" in the PDF file.
startxref	Number of occurrences of the string "startxref" in the PDF file.
page_command	Number of occurrences of the string "/Page" in the PDF file.
Encrypt	Number of occurrences of the string "/Encrypt" in the PDF file.
ObjStm	Number of occurrences of the string "/ObJStm" in the PDF file.
JS	Number of occurrences of the string "/JS" in the PDF file.
JavaScript	Number of occurrences of the string "/JavaScript" in the PDF file.
AA	Number of occurrences of the string "/AA" in the PDF file.
OpenAction	Number of occurrences of the string "/OpenAction" in the PDF file.
Acroform	Number of occurrences of the string "/Acroform" in the PDF file.
JBIG2Decode	Number of occurrences of the string "/JBIG2Decode" in the PDF file.
RichMedia	Number of occurrences of the string "/RichMedia" in the PDF file.
Launch	Number of occurrences of the string "/Launch" in the PDF file.
EmbeddedFile	Number of occurrences of the string "/EmbeddedFile" in the PDF file.
XFA	Number of occurrences of the string "/XFA" in the PDF file.
Colors	Number of occurrences of the string "/Colors > 2^24" in the PDF file.
URI	Number of occurrences of the string "/URI" in the PDF file.
BaseEncoding	Number of occurrences of the string "/BaseEncoding" in the PDF file.
Encoding	Number of occurrences of the string "/Encoding" in the PDF file.
ProcSet	Number of occurrences of the string "/ProcSet" in the PDF file.
Registry	Number of occurrences of the string "/Registry" in the PDF file.
Resources	Number of occurrences of the string "/Resources" in the PDF file.
www	Number of occurrences of the string "/www" in the PDF file.
server	Number of occurrences of the string "/server" in the PDF file.
Root	Number of occurrences of the string "/Root" in the PDF file.
BitsPerComponent	Number of occurrences of the string "/BitsPerComponent" in the PDF file.
Label	The label of the entry

because they can be used by the classifier to find certain patterns that can help in classifying malicious files properly. Based on our analysis of a large number of samples, the strings used in feature extraction were chosen to incorporate

features that often exist in malicious samples, and less likely to exist in benign samples.

The tool was built to extract all features in numeric format to minimize the need for preprocessing. The sample's hash was included for reference purposes to identify repetitions if two samples are the same file but have different names. The hashes are considered unique identifiers and are expected to be removed before using the dataset for training or testing machine learning-based detectors to avoid overfitting.

C. LABELING AND COMPILATION

For labeling purposes, we chose binary labeling, 0 for benign samples and 1 for malicious samples. While PDF malware can sometimes be categorized into sub-types, in most scenarios, it is used as a *dropper*. A dropper can be defined as an early-stage malware used to install other types of malware, such as worms and backdoors [23].

At the compilation stage, we merged the benign and malicious extracted samples after properly labeling them to produce the final version of the dataset with 24,337 samples (13,235 benign and 11,102 malicious). Each sample carries 42 features (including a unique file hash). All features are in numerical form, with no empty or missing values.

D. DATASET ANALYSIS

Upon compiling the dataset, we analyzed it to detect any anomalies. The mean, variance, minimum, and maximum were calculated for each feature in the dataset, as shown in Table 3.

As shown in the Table 3, there were three features showing values of 0 consistently across both classes. These features, namely Acroform, Colors, and BaseEncoding, do not have any effect on the classifier's decision.

Another interesting finding is that the maximum value of the pdf_ver feature was 5, which is not a valid PDF version. The highest valid PDF version is 2.0. When we explored this further, we found that many malicious samples use such invalid values for the PDF file version. Interestingly, most PDF readers do not flag this as invalid behavior.

Additionally, we found that some malicious PDF files have a page count of 0. Such files are deliberately corrupted by malicious actors, as explained earlier in Section II-C. The same finding was observed in the "contains text" feature, where several values of -1 were found. Such values are used by our data extraction software, PDFMal-Extract, to indicate a corrupted file where this particular feature was not extractable.

Figure 3 shows the features correlation heatmap. The figure also shows that Acroform, Colors, and BaseEncoding features appear as white lines. This means that these features were consistently non-varying and therefore do not impact the classifier's decision.

As shown in Figure 3, high correlation was detected in features that are logically relevant to each other, as expected, such as obj with endobj and stream with endstream. No unexpected levels of correlation were found among the dataset features.

V. MACHINE LEARNING-BASED DETECTION

A. IMPLEMENTATION ENVIRONMENT

All experiments, data collection and extraction, training, and testing were performed on a computer with the specifications listed in Table 4.

B. PERFORMANCE METRICS

Using the four basic performance measures of True Positive(TP), True Negative (TN), False Positive (FP), and False Negative (FN), the following metrics were calculated in our experiments:

1) Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2) Precision

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

3) Recall

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

4) F_1 score

$$F_1 score = 2 * \frac{\frac{TP}{TP+FN} * \frac{TP}{TP+FP}}{\frac{TP}{TP+FN} + \frac{TP}{TP+FP}} \tag{4}$$

In addition, we measured the time required for the inference of a single sample to help differentiate models by efficiency, not just accuracy.

C. IMPLEMENTATION PLAN

The different steps of the implementation plan are shown below:

1) *Data collection*: In this step, malicious and benign PDF samples are to be collected, as explained in Section IV-A.
2) *Feature extraction*: In this step, we built a python script to extract 42 features from each PDF file. The complete list of features and their descriptions can be found in Table 2.
3) Data labeling and compilation: In this step, the data extracted from malicious and benign samples is labeled (1 for malicious and 0 for benign) and compiled into a single dataset. The dataset can be found in [24]
4) Train-Test data split: The data was randomly split into 75% training and 25% testing subsets to ensure that there is no data spill between the training and testing subsets. The split was performed with stratification to ensure that the same class distribution is maintained in both the training and the testing subsets.
5) ML pipeline creation: To find the best performing classifier, we create a pipeline with the following classifiers:

TABLE 3. Per-Feature statistics.

Feature	Mean	Variance	Minimum	Maximum
pdfsize	800.93	22774872.63	0	468246
metadata size	319.70	1542151.37	176	77185
pages	8.26	1343.50	0	2096
xref length	2454.95	248276623.09	0	263987
title length	45.23	1150837.32	0	76993
isEncrypted	0.01	0.01	0	1
embedded files	0.02	0.03	0	8
images	31.03	664712.50	0	98560
contains_text	0.55	0.25	-1	1
pdf_ver	1.42	0.09	0	5
obj	137.79	880410.44	0	72292
endobj	138.44	895850.32	0	72291
stream	54.27	94097.44	0	15182
endstream	54.53	95892.22	0	15182
trailer	1.16	1.41	0	91
xref	1.13	1.46	0	91
startxref	1.36	2.41	0	156
page_command	9.68	2137.51	0	2789
Encrypt	0.02	0.04	0	11
ObjStm	2.36	306.80	0	1294
JS	0.64	268.05	0	2469
JavaScript	0.69	267.41	0	2469
AA	0.72	445.41	0	3197
OpenAction	0.39	0.25	0	4
Acroform	0.00	0.00	0	0
JBIG2Decode	0.29	134.74	0	1310
RichMedia	0.01	0.13	0	32
Launch	0.04	9.74	0	443
EmbeddedFile	0.34	2.35	0	19
XFA	0.06	0.06	0	2
Colors	0.00	0.00	0	0
URI	7.69	77614.06	0	39112
BaseEncoding	0.00	0.00	0	0
Encoding	4.93	1346.36	0	2769
ProcSet	8.69	4515.16	0	7491
Registry	1.15	21.17	0	307
Resources	15.20	19737.78	0	11142
www	4.87	18245.69	0	19775
server	0.03	0.17	0	10
Root	1.43	3.65	0	157
BitsPerComponent	18.28	31438.67	0	10186

TABLE 4. Implementation environment.

Hardware	
Processor	AMD Ryzen9-5950X
RAM	128GB
Software	
Python	3.11
PyTorch	2.6
SciKit-Learn	1.8
pyMuPDF	1.26.7

- Random Forest (RF)
- Logistic Regression (LR)
- Decision Tree (DT)
- Gaussian Naive-Bayes (GNB)
- eXtreme Gradient Boost (XGB)

6) Classifier training and testing: The five classifiers created in the previous step were trained using the training subset and then tested using the testing subset to obtain a selected group of performance metrics described in Section V-B. Figure 4 shows the layout of the proposed detection system.
7) Cross-Validation: To ensure the robustness of the obtained results and to confirm that the proposed system can generalize beyond its training subset, the best performing classifier was subjected to 10-fold cross-validation. In this validation, the dataset is randomly split into 10 folds. One fold is used for testing, while the other nine folds are used for training. This process is repeated ten times such that each fold is used for testing only once and for training nine other times. The detailed steps of 10-fold cross-validation are shown in Algorithm 1.
8) Model's Explainability: Shapley additive explanation is used to calculate the explainability of the best performing model. More details about SHAP can be found in Section VI.

D. RESULTS

The testing results of the classifiers' pipeline are shown in Table 5. As shown in the results, RF, and XGB outperform DT with a small margin, while outperforming LR and GNB by

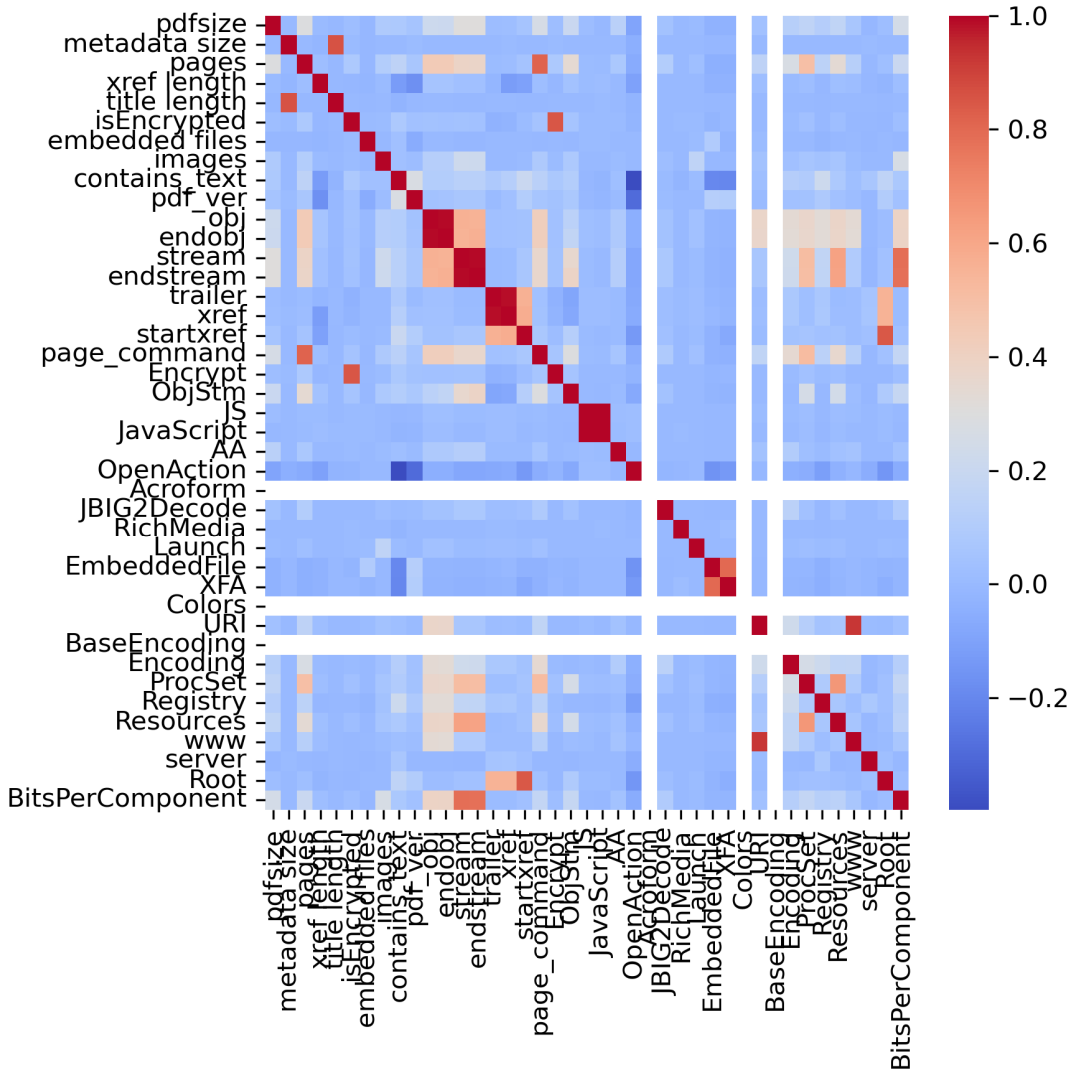


FIGURE 3. Features correlation heatmap.

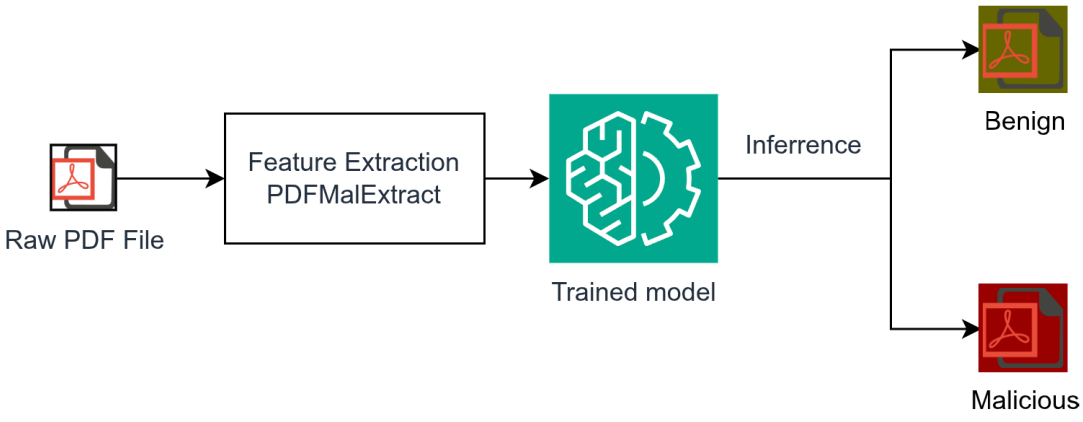


FIGURE 4. Proposed PDF Malware detector.

a large margin. XGB slightly outperforms RF by delivering an accuracy of 0.9949, and F_1 score of 0.9948. While this small difference in accuracy is negligible, XGB outperforms RF significantly in terms of inference time. While RF's single sample inference time was 3.92ms, XGB's single sample inference time was 0.564ms. That is merely 14% of the time

Algorithm 1 10-Fold Cross-Validation

```
Input: Dataset
Output: testing results for 10 runs; results
Array ← Dataset
Split Array into 10-folds randomly to produce fold(1to10)
model = MLClassifier
for i = 1to10 do
    TestingData = fold(i)
    TrainingData = Null
    for j = 1to10 do
        if j <> i then
            TrainingData = TrainingData.Append(fold(j))
        end
    end
    train model(TrainData)
    test model(TestData) to produce results(i)
end
print results
```

needed for RF. Therefore, We consider XGB as the best performing classifier due to its extremely high accuracy, and F_1 score, combined with comparatively low inference time.

As shown in Table 5, GNB presented the poorest performance with accuracy of 0.6218, and F_1 score of 0.5906 only.

Figure 5 shows the confusion matrix plots for the five classifiers. As shown in these plots, XGB shows a false-positive rate of 0.18%, and a false-negative rate of 0.9% outperforming all the other classifiers. While RF delivers excellent false-positive rate of only 0.09%, it has a noticeably higher false-negative rate of 1.19%. While these numbers do not indicate poor performance, they fail in comparison to XGB.

As shown in Figure 5, GNB shows a high true-positive rate of 98.41%, but fails to detect true-negatives, with only a 31.79% success rate. This means that 68.21% of benign files are being flagged as malicious, i.e, false-positives.

Figure 6 shows the Receiver Operating Characteristic (ROC) Curve Area Under Curve (AUC) for the five classifier. The plot confirms our earlier finding the XGB and RF slightly outperform DT, while noticeably outperforming LR classifier. The classifier with the lowest AUC was GNB, as deduced earlier.

Table 6 shows the results of the 10-fold cross-validation process. The small standard deviation shown in Table 6 proves that the results obtained earlier are reliable, and that the classifier can generalize well beyond its training subset.

VI. MODEL'S EXPLAINABILITY

The black box nature of machine learning models makes these models less desirable to cybersecurity practitioners. Many systems we use ubiquitously in our daily lives rely significantly on the security, and availability of these systems. This makes the risks associated with such assets extremely high, and the security of these systems is of imperative importance. This requires the highest possible transparency of machine learning-based cybersecurity systems to have a better understanding of how their decisions are made. Especially with the long history of biases in machine learning based systems [25].

Explainability, in the machine learning context, can be used for debugging models, user interpretation validation, and, most importantly, for building trust in these models and the decisions they make. While many classical models, such as decision trees, are considered inherently interpretable, they face challenges in generalizing this interpretability to the highest predictive performance models [26]. Such challenges emphasize the need for model-agnostic explainability. Establishing explainability without direct relevance to the model being analyzed provides a holistic approach that gives explanations for any machine learning model.

SHAP was introduced in 2017 as a model-agnostic technique of explainability based on game theory [27]. SHAP calculates the impact of each feature on each decision made by comparing the model's performance with the feature, to the model's performance without the feature. This helps in identifying the specific impact of lower or higher values of this feature on the classifier's decision, as well as measuring the importance of this feature in the decision-making process overall. The SHAP values were calculated for the XGBoost model presented in Section V.

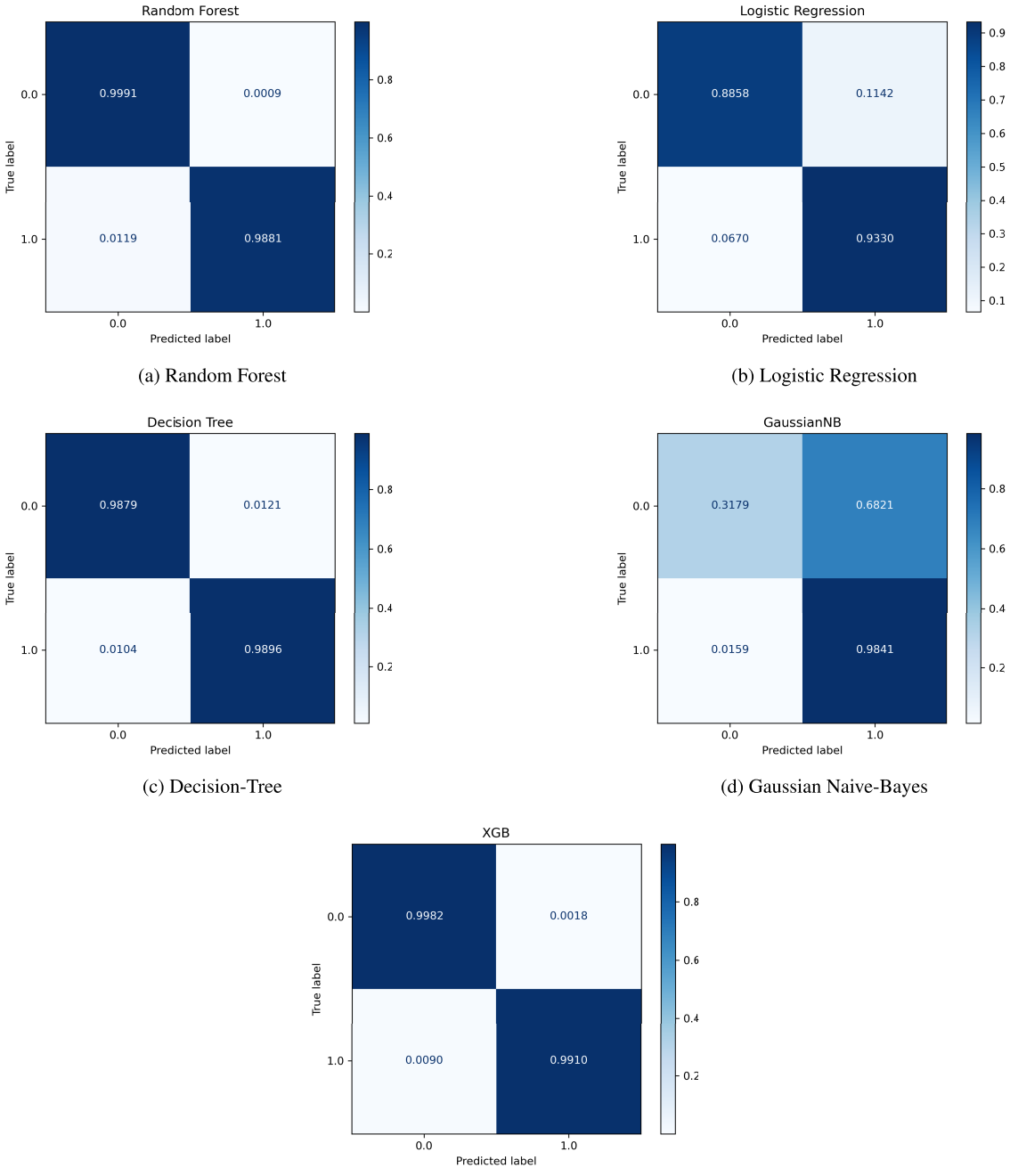
Figure 7 shows the SHAP summary plot for the top ten features. In SHAP summary plots, each line represents one feature, with the features arranged in order of importance, with the highest on top and the lowest in the bottom of the plot. Each dot in the plot represents the impact of the feature on the classifier's decision. Each dot on the right side of the plot pushes the classifier's decision to be higher, in our case pushing the output closer to 1 (malicious PDF file), while each dot on the left pushes the classifier decision to a lower value, which is 0 in our case (benign PDF file). The red color represents a higher value of the feature in that sample, while the blue color represents a lower value of the feature in that sample. The range of colors between red and blue represents the range of values of the features.

According to the plot in Figure 7, JavaScript is the feature with the highest importance. As explained in Table 2, this feature shows the number of occurrences of the /JavaScript string in the PDF file. The figure shows that higher occurrences of the /JavaScript string in the file push the decision towards malicious. This aligns with our experience that higher occurrences of /JavaScript is usually associated with malicious PDF files. The blue dots on the left side mean that lower occurrences of that string are associated with benign PDF files. The same explanation applies to the fourth feature, JS.

The second feature in Figure 7 is contains_text, which is a binary feature that captures if the file contains actual

TABLE 5. Results of all models.

Model	Accuracy	Precision	Recall	F_1 Score	Inference Time (ms)
RF	0.994084	0.994512	0.993603	0.994033	3.923857
LR	0.907313	0.906484	0.909382	0.907021	0.064845
DT	0.988661	0.988427	0.988733	0.988577	0.070683
GNB	0.621857	0.753729	0.651035	0.590655	0.185132
XGB	0.994906	0.995156	0.994590	0.994863	0.564525



(a) Random Forest

(b) Logistic Regression

(c) Decision-Tree

(d) Gaussian Naive-Bayes

(e) Extreme Gradient Boost

FIGURE 5. Confusion matrix plots the classifiers' pipeline.

text (of 50 characters or more). The figure shows almost all blue dots on the right side, which means that 0 value of this feature pushes the classifier's decision towards malicious. Alternatively, the red dots on the left mean that the existence

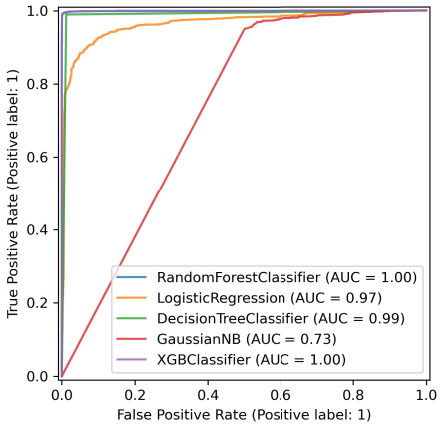


FIGURE 6. ROC AUC Curve plot for all classifiers.

TABLE 6. Results of 10-fold cross-validation for XGB model.

Fold	Accuracy	Precision	Recall	F_1 Score
1	0.994659	0.993476	0.994403	0.993939
2	0.992605	0.993744	0.990205	0.991971
3	0.995070	0.996286	0.992599	0.994439
4	0.993016	0.995483	0.989228	0.992346
5	0.993837	0.998140	0.988029	0.993059
6	0.993837	0.994783	0.992194	0.993487
7	0.995070	0.996491	0.993007	0.994746
8	0.993424	0.995633	0.990443	0.993031
9	0.993424	0.997245	0.988171	0.992687
10	0.993835	0.995345	0.990732	0.993033
Mean	0.993878	0.995663	0.990901	0.993274
St Dev	0.000789	0.001382	0.002012	0.000841

of more than 50 characters of text pushes the prediction toward benign. This is also aligned with our experience that malicious files usually contain very low amount of text, or no text at all, while benign files usually contain more than 50 characters.

The third feature in the figure is pdfsize. Lower file size pushes the prediction of the classifier closer to malicious, while a larger file size pushes it toward benign. This is also aligned with our experience that malicious PDF files tend to be small in size and do not contain a lot of readable content. A similar explanation applies to the pages feature, listed as the tenth in the figure.

An explanation similar to JavaScript and JS features can be found in the figure for the feature named XFA, which captures the occurrences of the /XFA command. The command is used by malicious actors to embed scripts inside XML Forms Architecture. Additionally, the number of URI occurrences is also considered an indication of maliciousness by the classifier.

The calculated feature importance for all features is shown in Table 7. As shown in the table, the feature with the highest importance is the /JavaScript string in the PDF file, as explained earlier. Interestingly, the last seven features; isEncrypted, Acroform, Encrypt, BaseEncoding, Colors, RichMedia, and JBIG2Decode — show 0 importance. This means that the values of these features do not impact the classifier's decision in any way. Such insights can be used for feature selection for future models to reduce the number of features extracted from the PDF files and therefore improve the efficiency of the detection system without sacrificing accuracy. Table 7 also shows that the top five features have much higher importance values in comparison to other features. This can result in a highly effective feature selection process.

TABLE 7. SHAP feature importance for all features.

Feature Number	Name	Importance
21	JavaScript	3.474789
8	contains_text	1.221534
0	pdfsize	1.126533
20	JS	1.126279
35	Registry	0.636202
29	XFA	0.550923
1	metadata size	0.503341
12	stream	0.441044
31	URI	0.377808
2	pages	0.375392
3	xref length	0.364602
11	endobj	0.363546
37	www	0.356451
23	OpenAction	0.318886
36	Resources	0.290378
33	Encoding	0.285145
9	pdf_ver	0.276523
10	obj	0.250006
7	images	0.246687
17	page_command	0.216136
34	ProcSet	0.192889
4	title length	0.189564
13	endstream	0.183275
40	BitsPerComponent	0.148386
28	EmbeddedFile	0.131712
19	ObjStm	0.083332
39	Root	0.075597
14	trailer	0.052374
6	embedded files	0.051439
16	startxref	0.044565
15	xref	0.033615
27	Launch	0.030032
22	AA	0.025128
38	server	0.018971
5	isEncrypted	0.000000
24	Acroform	0.000000
18	Encrypt	0.000000
32	BaseEncoding	0.000000
30	Colors	0.000000
26	RichMedia	0.000000
25	JBIG2Decode	0.000000

The results obtained from SHAP explainability analysis are highly aligned with human experience regarding malicious PDF files. This increases the transparency of the decisions made by the detection system and confirms the importance of the features extracted from our presented dataset.

VII. DISCUSSIONS

The main contribution of this work is producing a benchmark dataset for PDF malware that can be used for training machine learning-based detection systems. This includes the collection of a diverse range of samples and the focus on creating a larger number of indicative features that can help

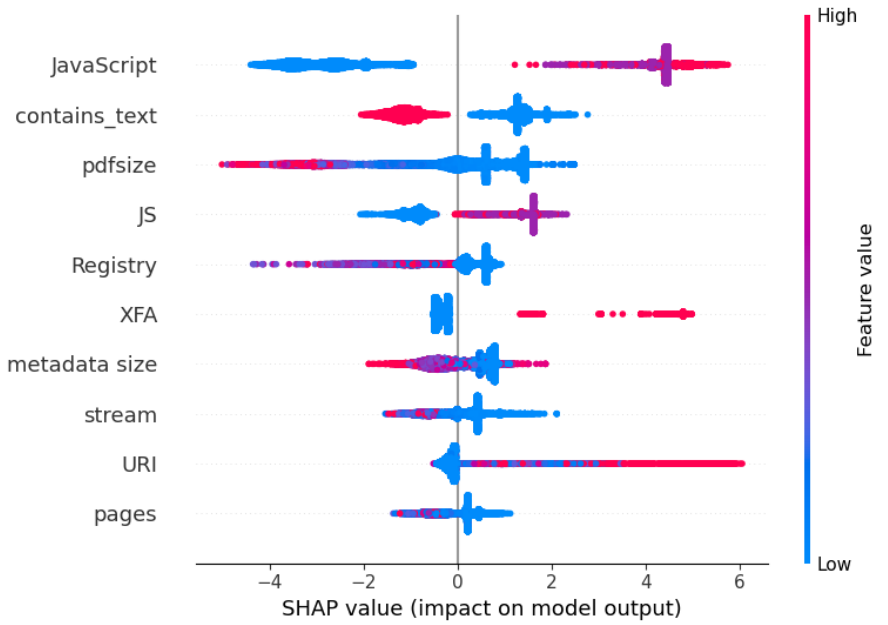


FIGURE 7. SHAP summary plot for the top 10 features.

TABLE 8. Comparison of the proposed detector trained with RIT-PDFMal-2026 with related works.

Reference	Dataset Name	Samples	Malicious %	Benign %	Features	Date	ML Technique	Accuracy	F_1 Score	Inf Time (ms)	Explainable
[15]	Contagio	20,152	55%	45%	128x128	2013	Markov Plot	0.9939	0.9736	-	✗
[10]	Contagio	19,338	53%	47%	3,514	2013	Hidost	0.9990	-	-	✗
[8]	CIC-PDFMal-2022	10,022	55%	45%	32	2022	Stacked-Ens	0.9869	0.9887	-	✗
[9]	PdfRep	493,238	66%	34%	15	2023	XGB	0.9878	0.9825	-	✗
Proposed	RIT-PDFMal-2026	24,337	45%	55%	42	2026	XGB	0.9949	0.9948	0.5645	✓

detection systems capture the essence of malicious content inside PDF files. This is evident in the publication dates of the datasets used in these works. Table 8 shows a comparison of the proposed PDF malware detection system utilizing the proposed dataset with related works.

As shown in Table 8, [10] and [15] both rely completely on Contagio dataset which was published in 2013. Even PdfRep, presented in [9], that has the highest number of samples, includes a large portion of Contagio samples, contains a significnat number of old samples from Contagio and other sources. While PDF malware did exist before 2013, the older malware relied heavily on Javascript, and did not use the obfuscation and multistage techniques disciovered in more recent PDF malware samples.Additionally, PdfRep suffers from low number of features extracted from these samples. The selected features in [9] were mostly statistical and missed many structural features that can help in distinguishing malicious from benign files.

In terms of class balancing, none of the reviewed works suffered from significant imbalance, except [9] that had a higher imbalance comparatively.

CIC-PDFMal-2022, presented in [8] is the most comparable to our presented dataset, due to its newer samples, along with the variation in the types of malware captured in the dataset. However, our proposed dataset contains more than double the number of samples, along with a significant jump in the number of features as well. RIT-PDFMal-2026 contains 24,337 samples with 42 features, while CIC-PDFMal-2022 contains only 10,022 samples, with 32 features. If we ignore the ‘‘hash’’ feature, which can not be used for training, the nine other feature difference was carefully selected based on detailed analysis of the PDF malware samples to select these additional 9 strings. The difference is clearly shown in two out of these nine features, namely Registry and URI, appearing in the top ten in SHAP feature importance.

As we examine the detection model’s explainability, we find that the use of malicious Javascript is still a noticeable theme in malicious PDFs. Additionally, we notice a rise in the use of URIs and URLs to trigger more malicious downloads. This emphasizes the role of PDF malware as a dropper rather than a carrier of the final payload in most scenarios.

A limitation of the current static-only approach is its potential vulnerability to advanced evasion techniques, such as heavy obfuscation or incremental updates that alter structural features while preserving malicious intent. The RIT-PDFMal-2026 dataset, with its diverse recent samples

and rich feature set, was designed to provide an foundation for hybrid static-dynamic detectors. By combining the static features extracted here with behavioral traces (e.g., system-call sequences or event graphs obtained through controlled execution in a sandbox), researchers can build more resilient models aligned with recent advances in polymorphism-aware behavioral classification [16].

VIII. CONCLUSION AND FUTURE WORK

In this paper, we presented a curated benchmark dataset for malicious PDF documents. The research work focused on collecting a reliable real-life pool of malicious samples and combine it with realistic benign PDF documents. The work also presented a carefully selected set of features that can help build highly accurate machine learning-based malicious PDF detection systems.

The built dataset was used in training and testing a large pool of classifiers to ensure its usability, reliability, and correctness [24]. The built detection system was successful in detecting malicious PDFs with an accuracy of 0.9949, F_1 score of 0.9948, FPR of 0.18%, and FNR of 0.9%. The results were validated using 10-fold cross-validation and showed a minimal standard deviation of 0.000789 in accuracy and 0.000841 in F_1 score. This proves that the proposed model generalizes well beyond its training dataset and proves that the results obtained were not caused by overfitting.

Additionally, the best performing model, XGB, was explained using SHAP. Model explainability provided deeper insights into the impact of each feature on the classifier's decision. The obtained explanations were highly aligned with human experience. This increases trust in such systems.

Future research directions include expanding the dataset with additional features as different types of malware families evolve. A related research direction is to explore the capabilities of other types of classifiers, such as deep neural networks and transformers.

Another research direction is to explore the use of Artificial Intelligence (AI) agents in attribution and extraction of malicious content in PDF files. In particular, the RIT-PDFMal-2026 dataset can serve as high-quality training data for generative models (e.g., GANs or VAEs) to synthesize realistic malicious PDF feature vectors or even augmented PDF objects, thereby addressing data scarcity, class imbalance, and concept drift in evolving PDF malware campaigns [17]. Another promising direction is the development of hybrid static-dynamic detection pipelines that first apply the fast XGBoost static classifier for initial screening and then trigger lightweight behavioral analysis on suspicious samples, leveraging ensemble techniques robust to polymorphic thread-division attacks [16]. The dataset also enables adversarial robustness studies (e.g., generating evasive PDF variants) and continual-learning benchmarks, positioning RIT-PDFMal-2026 as a versatile resource for the broader malware-detection community.

REFERENCES

[1] F. Duarte, "Amount of data created daily (2026)," *Exploding Topics*, p. 1, Feb. 2026. [Online]. Available: https://explodingtopics.com/blog/data-generated-per-day

[2] CloudFiles. (May 11, 2026). *How Many Digital Files Exist Globally? 15 Trillion Estimate*. [Online]. Available: https://www.cloudfiles.io/blog/how-many-files-are-there-in-the-world

[3] P. Association. (May 11, 2026). *About the Portable Document Format—PDF Association*. [Online]. Available: https://pdfa.org/about-us/the-portable-document-format

[4] SentinelOne. *CVE-2025-1918: Google Chrome Buffer Overflow Vulnerability*. Accessed: May 11, 2026. [Online]. Available: https://www.sentinelone.com/vulnerability-database/cve-2025-1918

[5] A. Security. (Feb. 2019). *Adobe Security Bulletin*. Accessed: May 11, 2026. [Online]. Available: https://helpx.adobe.com/security/products/acrobat/apsb18-09.html

[6] S. Liu, J. Ming, G. Zhou, X. Liu, J. Fu, and G. Peng, "Analyzing PDFS like binaries: Adversarially robust PDF malware analysis via intermediate representation and language model," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2025, pp. 1334–1348.

[7] N. Nissim, A. Cohen, J. Wu, A. Lanzi, L. Rokach, Y. Elovici, and L. Giles, "Sec-lib: Protecting scholarly digital libraries from infected papers using active machine learning framework," *IEEE Access*, vol. 7, pp. 110050–110073, 2019.

[8] M. Issakhani, P. Victor, A. Tekeoglu, and A. Lashkari, "PDF malware detection based on stacking learning," in *Proc. 8th Int. Conf. Inf. Syst. Secur. Privacy*, 2022, pp. 562–570.

[9] R. Liu, R. Joyce, C. Matuszek, and C. Nicholas, "Evaluating representativeness in PDF malware datasets: A comparative study and a new dataset," in *Proc. IEEE Int. Conf. Big Data (BigData)*, Dec. 2023, pp. 3017–3024.

[10] Y. Chen, S. Wang, D. She, and S. Jana, "On training robust PDF malware classifiers," in *Proc. 29th USENIX Secur. Symp. (USENIX Security)*, 2020, pp. 2343–2360.

[11] Kaspersky. (Nov. 2025). *Can PDFs Contain Viruses? What to Do If You Opened a PDF Scam*. Accessed: Feb. 13, 2026. [Online]. Available: https://www.kaspersky.com/resource-center/threats/pdf-viruses

[12] M. M. Alani and A. I. Awad, "PAIRED: An explainable lightweight Android malware detection system," *IEEE Access*, vol. 10, pp. 73214–73228, 2022.

[13] M. Alshoulie and A. Mehmood, "Deep learning approaches for malware detection: A comprehensive review of techniques, challenges, and future directions," *IEEE Access*, vol. 13, pp. 118652–118677, 2025.

[14] Contagio. *16,800 Clean and 11,960 Malicious Files for Signature Testing and Research*. Accessed: May 15, 2026. [Online]. Available: https://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html

[15] A. Corum, D. Jenkins, and J. Zheng, "Robust pdf malware detection with image visualization and processing techniques," in *Proc. 2nd Int. Conf. Data Intell. Secur. (ICDIS)*, Jun. 2019, pp. 108–114.

[16] L. Mauri and E. Damiani, "Hardening behavioral classifiers against polymorphic malware: An ensemble approach based on minority report," *Inf. Sci.*, vol. 689, Jan. 2025, Art. no. 121499.

[17] G. Gebrehans, N. Ilyas, K. Eledlebi, W. T. Lunardi, M. Andreoni, C. Y. Yeun, and E. Damiani, "Generative adversarial networks for dynamic malware behavior: A comprehensive review, categorization, and analysis," *IEEE Trans. Artif. Intell.*, vol. 6, no. 8, pp. 1–23, Aug. 2025.

[18] S. S. R, R. Varuneshan, and H. G. C, "Designing a static malware analysis framework for detecting malicious malware code with ghidra," in *Proc. 3rd Int. Conf. Self Sustain. Artif. Intell. Syst. (ICSSAS)*, Jun. 2025, pp. 1696–1701.

[19] B. B. H. Kang and A. Srivastava, "Dynamic malware analysis," in *Encyclopedia of Cryptography, Security and Privacy*, New York, USA: Springer, 2025, pp. 713–714.

[20] *VirusTotal–Home*. Accessed: Apr. 27, 2026. [Online]. Available: https://www.virustotal.com/

[21] M. M. Alani. (Apr. 2026). *PDFMalExtract*. Accessed: May 22, 2026. [Online]. Available: https://github.com/Mo-Alani/PDFMalExtract

[22] Artifex, "PyMuPDF documentation," May 2026, [Online; accessed 5. May 2026]. https://pymupdf.readthedocs.io/en/latest

[23] M. M. Alani, A. Mashatan, and A. Miri, "XMal: A lightweight memory-based explainable obfuscated-malware detector," *Comput. Secur.*, vol. 133, Oct. 2023, Art. no. 103409.

[24] M. M. Alani. (Jun. 2026). *RIT-PDFMal-2026*. Accessed: Jun. 22, 2026. [Online]. Available: https://github.com/Mo-Alani/RIT-PDFMal-2026
[25] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Comput. Surveys*, vol. 54, no. 6, pp. 1–35, Jul. 2022.
[26] C. Molnar, "Interpretable machine learning: A guide for making black box models explainable," 2020. [Online]. Available: Lulu.com
[27] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.

MOHAMMED M. ALANI (Senior Member, IEEE) received the Ph.D. degree in computer engineering with a specialization in network security. He is currently a Professor of cybersecurity with Rochester Institute of Technology of Dubai. Previously, he was a professor and a research fellow at many organizations in Canada and Middle East. He has delivered many invited talks and keynote speeches in international conferences around the world. His current interests include applications of ML in cybersecurity and ML security. He is a Senior Member of ACM. He currently serves as an ACM Distinguished Speaker.

ERNESTO DAMIANI (Senior Member, IEEE) received the Ph.D. degree in computer science from the Università degli Studi di Milano, Italy. He was the Senior Director of the Center for Cyber Physical Systems (C2PS), Khalifa University, United Arab Emirates. He is currently a Full Professor with the Department of Computer Science "Giovanni degli Antoni," Università degli Studi di Milano, where he leads the SEcure Service-Oriented Architectures Research (SESAR) Laboratory. He has authored more than 800 scientific publications and several patents in cybersecurity, artificial intelligence, machine learning, big data analytics, behavior-based malware detection, and the application of generative models to simulate malicious behaviors and enrich malware datasets. He is an ACM Distinguished Scientist and has received multiple international awards, including the Doctorate Honoris Causa from INSA Lyon, France. He holds the rank of the Officer of the Star of Italy Order for his contribution to international collaboration in the field of AI.