Enhancing malware detection reliability in non-executable files using confidence score prediction

Rasoul Rezvani-Jalal[a], Morteza Zakeri[b], Saeed Parsa[a,*], Amin Hasan-Zarei[a]

[a]*School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.*
[b]*School of Computer Science, Institute for Research in Fundamental Sciences (IPM), P. O. Box 19395-5746, Tehran, Iran.*

Abstract

Malware attacks targeting widely used non-executable formats, namely Microsoft Office and PDF files, have become a prevalent threat. These files, which encompass a broad spectrum of data types are classified as complex files. Existing malware detection models currently lack transparency, providing only binary labels without confidence scores. Incorporating confidence score enhances interpretability and detection accuracy. This article proposes a learning-based malware detection approach including two complementary parts. The first part involves the development of binary classifiers, on an enriched dataset of related files, with an extended feature set to achieve high accuracy. The second methodology employs regression models to ascribe a confidence score to each sample. A reliability score is assigned to various antiviruses to accurately label samples with confidence scores. By completion of the detection process, a pair consisting of x and y is provided, where x is the binary classifier output and y is the regressor output, showing the confidence score. Our findings demonstrate an enhancement compared to existing malware detection classifiers, with improvements of approximately 2.44% for PDF files and 2.27% for MS Office. Using confidence score along with binary classification boosts detection accuracy to 99.74% for PDFs and 99.77% for office files.

Keywords:
Malware detection, non-executable file, portable document format, Microsoft Office document, confidence score, machine learning.

1. Introduction

PDF files, known for their versatility in incorporating images, multimedia, and executing `JavaScript`, possess a hierarchical structure consisting of various interconnected objects with different types. This intricacy enables them to store extensive information but also introduces vulnerabilities, such as the `CVE-2024-4367` in PDF.js, which allows arbitrary `JavaScript` execution, potentially affecting millions of websites. The vulnerability stems from an oversight in handling fonts, demonstrating how malicious actors can exploit PDF files to launch malware or perform unauthorized actions, highlighting the need for constant vigilance and updates to mitigate such risks. Based on this characteristic [1], they are called complex files. Furthermore, despite their wide usage and versatility, these file types are not executable, meaning they cannot run malicious code without assistance.

Microsoft Office documents also renowned for their rich content including `VBScript`, links, texts, images, and multimedia, present a complex landscape for both creators and attackers [2]. This complexity, while enabling sophisticated document designs and functionalities, also introduces vulnerabilities. A prime example is `CVE-2024-20677`, which highlighted the potential for arbitrary code execution through the insertion of `FBX` files into Office applications. This vulnerability underscores the security challenges posed by the intricate nature of Office documents, making them attractive targets for malicious entities seeking to exploit their rich object types for illicit purposes.

*Corresponding author
Email addresses: `rasoul_rezvanijalal@comp.iust.ac.ir` (Rasoul Rezvani-Jalal), `zakeri@imp.ir` (Morteza Zakeri), `parsa@iust.ac.ir` (Saeed Parsa), `amin_hasan@comp.iust.ac.ir` (Amin Hasan-Zarei)

The abovementioned characteristic makes them a prime target for malware, as attackers must embed malicious code within the file to exploit vulnerabilities. Malware embedded in PDF or Office files can exploit the file's capabilities to execute malicious actions. For instance, a PDF file can contain other file types, including `HTML`, JavaScript, `SWF`, `XLSX`, `EXE`, Microsoft Office files, or even another PDF file. Hence, the detection of malicious activities within non-executable files often relies on identifying patterns involving scripts, links, images, and the communication between different scripting languages like `JavaScript` and `VBScript`. Despite the prevalence of these types of files and the potential for malware, their structure remains a challenge for effective detection and prevention mechanisms. So, understanding the intricate structure and capabilities of these files is important in developing robust defenses against malware. In a report published by Kaspersky [3], a notable increase in attacks related to malicious files utilizing Microsoft Office document formats was observed in 2023. Kaspersky's detection systems reported a 53% surge in attacks employing malicious Microsoft Office documents alongside other prevalent formats such as PDFs. Furthermore, as reported by Avira Corporation [4], Microsoft Office documents are the second most frequently used file format by malware targeting Windows systems, consistently appearing in Malspam campaigns. Therefore, it is imperative to implement stringent measures to mitigate the risk of malicious file execution within the victim's system.

Many studies [1, 2], utilize binary classification models for malware detection, assigning labels of 0 or 1 to samples based on antivirus opinions. For instance, Koutsokostas et al. A research by koutsokostas et al. [2] use `JSON` files from the Virus Total [5] database to label samples, with a label of 1 assigned if any antivirus flags a file as malicious. This approach, however, raises questions about the reliability of learning models. It may inadvertently group files flagged by a low-reputation antivirus with those recognized by well-known antiviruses, potentially leading to misclassification. This issue complicates the interpretability of malicious files and suggests that files with low confidence scores, showing their low likelihood of being malicious, may require manual review. This manual review process can be time-consuming, highlighting the need for more nuanced labeling and confidence scoring methods to improve malware detection accuracy and reliability. In general, the major factor in the accuracy of performance and as a result the success of learning models is the feature space that is used to search for its purpose. Therefore, the effectiveness of detecting malicious activities in non-executable files, such as Office documents, hinges on recognizing patterns involving scripts, links, images, and the interaction between JavaScript and VBScript.

This article aims to enhance the detection capabilities of PDF and MS Office malware, ensuring user trust in the generated outputs for accurate analysis. It contributes by compiling a dataset from various file types, including `pdf`, `doc`, `docx`, `xls`, `xlsx`, `ppt`, and `pptx`, and introducing novel features to cover a broad spectrum of attacks. The contributions continue by creating binary classification models from this dataset, fine-tuning hyperparameters for optimal detection accuracy, and introducing the concept of a confidence score. This approach not only improves detection rates but also enhances the interpretability of detection outcomes, thereby facilitating informed decision-making by malware analysts.

We introduce an efficient approach to malware detection by assigning a confidence score to each sample, addressing the limitations of binary classification. A confidence score is generated for each malicious sample in our dataset by averaging the results of different anti-viruses after scanning the file. Involving confidence scores along with binary labels (benign or malicious) enhances the malware detection performance of the existing malware classification models. A floating-point confidence score in the range of $(0, 1]$, indicates the likelihood of maliciousness for each target file. The closer the confidence score is to 1, the higher the confidence in maliciousness, and vice versa. This dual output allows for more informed decision-making by human analysts, prioritizing samples with low confidence scores for manual review. Our approach not only improves the accuracy of malware detection but also optimizes human analysts' time by focusing on the most suspicious files, those with a confidence score close to 0. In summary, the main contributions of this research are as follows:

1. We introduced an innovative mechanism in terms of a mathematical model to weigh different antiviruses for computing confidence scores corresponding to each malicious file.

2. We developed regression models to determine the confidence score for non-executable complex files efficiently without any need to use different antivirus and improved the corresponding malware detection compared to binary classification alone.

3. Through the study of complex file formats and massive feature engineering, we strategically selected

distinct and diverse features for each file type, enhancing the detection of related malware in comparison to previous endeavors.

4. By conducting systematic tests on a range of hyperparameters across various models and utilizing k-fold cross-validation, we pinpointed the most effective hyperparameter configurations, guaranteeing the dependability of our model's outputs.

5. We compiled a large and quality collection of malicious and benign files for both PDF and Office formats. By utilizing a wide array of resources, we not only enhanced but also expanded the existing datasets. This way training is conducted on an extensive dataset, yielding superior binary classification models' outcomes, and increasing the reliability of the outcomes.

In the subsequent sections, we offer a review of existing research concerning malware, explicitly focusing on office documents and PDF files. This review delves into the examination of their structural elements and the detection techniques utilized. Subsequently, we delve into the methodology employed in this study, detailing the data collection, validation, feature extraction, and learning processes in Section 3. Section 4 provides an overview of our proposed dataset, including an exploratory analysis of its content and a discussion of dataset statistics. Section 5 analyzes our findings and experiments, while Section 6 provides possible threats to validation. Finally, Section 7 summarizes our contributions and potential future research directions that could extend the scope of this work.

2. Related Work

This section offers readers a foundational understanding and overview of the related work that underpins our study. Initially, we delve into the structure of MS Office documents and PDF files, providing a comprehensive background. Subsequently, we outline the defensive methods employed against PDF and MS Office malware, distinguishing between the threats.

2.1. PDF files

Before delving into the examination of tasks pertinent to detecting malware, it is imperative to familiarize oneself with the selection process, with PDF files as a prime instance The general structure of the PDF files has been shown in Figure 1. This study underscores the complexity of PDF files, highlighting their logical structure, which can be illustrated as a directed graph. As depicted in Figure 2, each node within this graph is identified by a numerical value representing the object's sequence in the file. This conceptualization is further elaborated upon in a study [6], which posits that each node in this graph symbolizes an individual object within the PDF file. For a more detailed exploration of this graph, refer to Figure 3 in the study by Šrndic et al. [7], which provides a proper visualization of the PDF file's structure.

Owing to their intricate and adaptable structure, as well as their capacity to incorporate a variety of content types, documents such as PDF files are classified as complex files. A significant component of the mentioned content is JavaScript code, which enhances the capabilities of PDF files. Consequently, numerous studies have focused on detecting malicious PDF files by identifying JavaScript code. A study by Jiaxiang Gu et al. introduced a set of tokens derived from JavaScript code variables as file attributes, achieving an accuracy of 96.93% using an SVM model [8]. Similarly, a related study [9] employed an algorithm named PJscan to translate JavaScript code into tokens, achieving an accuracy of 85%. Moreover, recognition power has been enhanced by incorporating additional general features related to JavaScript, achieving a recognition power of 99.89% using the stack learning technique with a meta-learner Logistic Regression (LR) [1]. This approach demonstrated superiority by addressing the oversight of preprocessing in previous studies.

A recent study [10] has been conducted on the CIC-Evasive-PDFMal2022 dataset [11], using the optimizable decision tree (O_DT) algorithm, achieving a recognition accuracy of 98.84%. Furthermore, different sets of features for PDF files have been proposed [7], converting each file into a set of paths or bag-of-paths, where each path represents a feature of the file. This methodology, utilizing SVM and DT models, achieved high accuracy in malware detection.

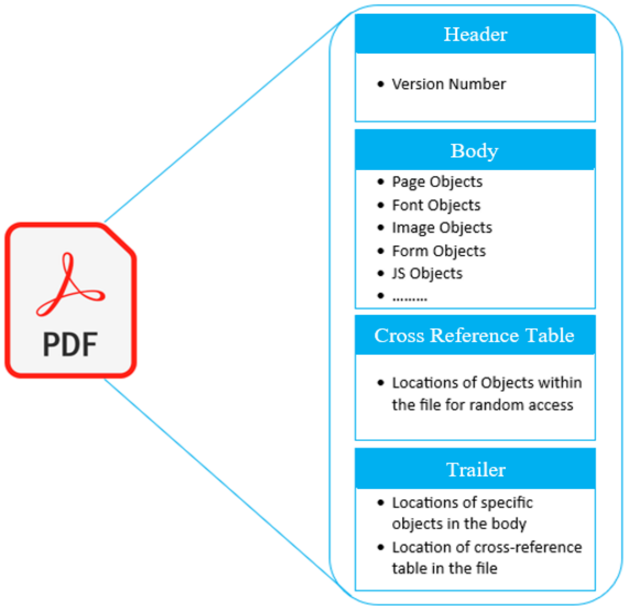

PDF
Header
Version Number
Body
Page Objects
Font Objects
Image Objects
Form Objects
JS Objects
....
Cross Reference Table
Locations of Objects within the file for random access
Trailer
Locations of specific objects in the body
Location of cross-reference table in the file

Figure 1: PDF internal structure.

2.2. Office documents

A deep understanding of the structural composition of office documents is imperative in document analysis. This necessity is underscored by the scholarly endeavors in this domain, such as those conducted by Koutsokostas et al. [2] and Singh et al. [6], which have provided detailed descriptions of the hierarchical organization of these file types. As depicted in Figure 4, it is evident that office documents exhibit a tree-like structure akin to PDF files. This structural similarity underscores the complexity and interconnectedness inherent in both document types.

Many studies and activities have been undertaken to identify and analyze office documents, emphasizing examining macro codes. For instance, VBA code has been explored utilizing the Olevba library in Python [12], although the used model did not undergo a specific evaluation based on machine learning models. Another research [13] employed NLP-based algorithms, precisely the Sparse Composite Document Vectors (SCDV) algorithm, to construct feature vectors for each sample, achieving an f-score of 93% with the SVM model. The challenge of detecting obfuscated VBA macros has been addressed in a study [14], employing various static features to achieve a 90% accuracy rate. In their study, Ravi et al. [15] explored obfuscated macros using a method known as 'obfuscated_word2vec.' They extracted features such as API calls and obfuscated variables, achieving an accuracy rate of 82.65%. Koutsokostas et al. [2] expanded their study on the analysis of `VBA` macros, incorporating dynamic data exchange (`DDE`), a feature facilitating data exchange between different office documents, among its features. This study achieved a 97.5% detection accuracy with the Random Forest (RF) model.

Additionally, a method has been developed [16] that combines image analysis with VBA code analysis to diagnose documents. While highly effective, this approach is limited to samples containing images, which may not be present in all cases. The study done by Nissim et al. [17] diverged from the previous studies by focusing on file paths as file attributes, introducing features representing the path from the root to each leaf in the file structure. Using the SVM model, this method successfully identifies malicious docx files with high detection power. However, about 98% of the files were benign, and only about 2% were malicious, which can impact learning.

3. Proposed Approach

In this research, we aim to enhance the analysis of complex non-executable files and achieve high-quality diagnosis by employing two primary methodologies. The first involves the use of classifiers to categorize files as either benign or malicious while the second methodology focuses on assigning a confidence score to each

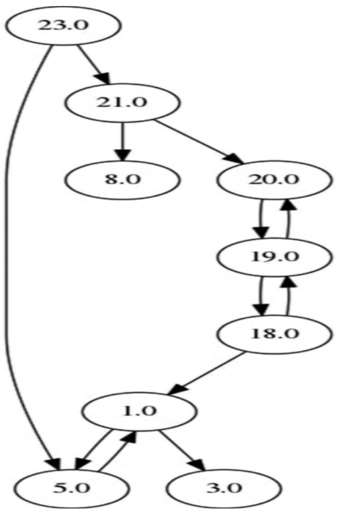


Figure 2: PDF file as a directed graph [6].

file under review. This score indicates how confident one can be that a sample is malicious, which can reduce the risk of incorrect detection. These methodologies are elaborated upon in sections 3.2 and 3.3, respectively.

3.1. Overview

Previous research efforts in detecting malware within PDF or MS Office files have typically produced binary outputs, such as "Yes" or "No", or numerical values like 0 or 1, to indicate whether a given sample is malicious or benign based on the learning models employed. In contrast, this study introduces a novel approach that generates outputs consisting of a pair of values, denoted as (X, Y). In this two-parameter structure, X is a binary value (0 or 1) that signifies whether the file is benign or malicious. Additionally, Y represents a decimal value ranging from 0 to 1, which serves as a confidence score indicating the degree of similarity between the sample and known malicious files. By evaluating both X and Y simultaneously, this methodology allows for a more nuanced and accurate determination of a file's status, enhancing the overall effectiveness of malware detection.

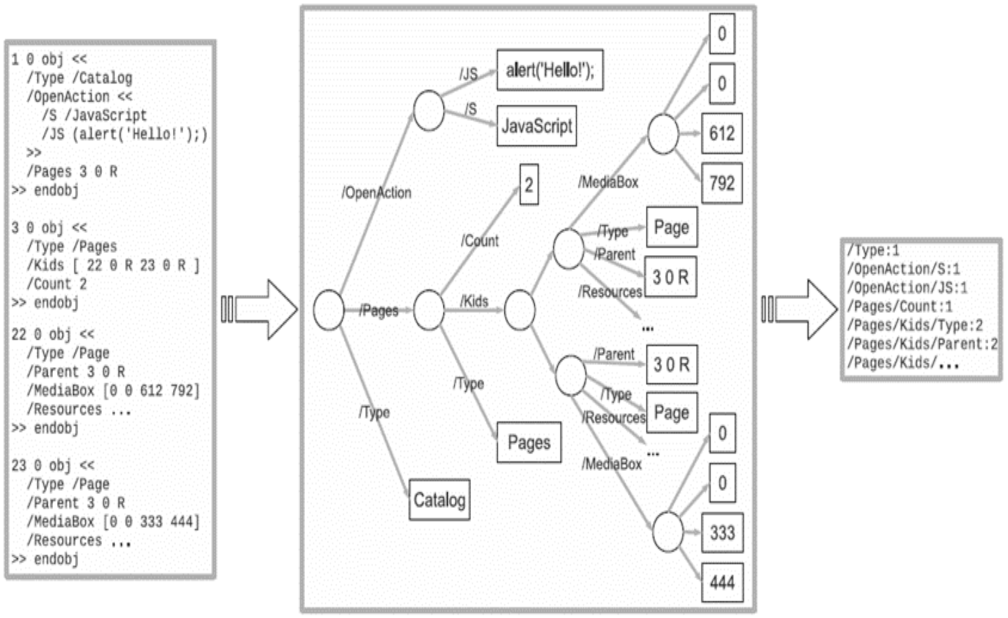
The overall detection process of the current study is shown in Figure 5. In this process, first, the type of the file under investigation is determined. Thereafter, each sample receives a binary label, malicious or benign resulting from the binary classifier under the X variable and a decimal score representing the confidence score assigned by the regressor under the Y variable. According to the received information, the analyst can determine whether each sample is benign or harmful based on the values of the X and Y variables and a threshold of 0.1, which was established in this study based on experimental results. In this recognition process, the human agent can quickly tag a large number of files. In other words, this process has helped the human factor make the right decision at the right time.

3.2. Classification

In the context of classifying malicious files of non-executable complex types, such as PDF and MS Office files, the development of learning models is a critical process. This process is depicted in Figure 6 and Figure 7, which outline the methodology for each file type. The approach comprises four distinct phases: data collection, reliable labeling, feature extraction, and preprocessing, followed by the creation of classifier models. Each phase encompasses subsections and anticipated outputs, which are elaborated upon in the subsequent subsections of this study.

3.2.1. Data collection

This section acquired a substantial collection of related files to facilitate accurate analysis in subsequent phases. Initially, we accessed a resource for acquiring files, MalwareBazaar [18], which packages malware files into daily bundles from 2020 to the present. Specifically, we obtained daily bundles for 2021, 2022, and 2023, encompassing a significant volume due to the vast array of files within each bundle, including executable



```
1 0 obj <<
  /Type /Catalog
  /OpenAction <<
    /S /JavaScript
    /JS (alert('Hello!');)
  >>
  /Pages 3 0 R
>> endobj

3 0 obj <<
  /Type /Pages
  /Kids [ 22 0 R 23 0 R ]
  /Count 2
>> endobj

22 0 obj <<
  /Type /Page
  /Parent 3 0 R
  /MediaBox [0 0 612 792]
  /Resources ...
>> endobj

23 0 obj <<
  /Type /Page
  /Parent 3 0 R
  /MediaBox [0 0 333 444]
  /Resources ...
>> endobj
```

```
/Type:1
/OpenAction/S:1
/OpenAction/JS:1
/Pages/Count:1
/Pages/Kids/Type:2
/Pages/Kids/Parent:2
/Pages/Kids/...
```

Figure 3: Physical layout vs. logical structure of PDF file [7].

EXE, ELF, PDF, and a wide range of office documents. Given the project's focus on Office and PDF files, we employed a Python script to identify and segregate relevant files from the rest.

To augment the dataset for this project, we incorporated examples of related files from prior research. For PDF files, we utilized the CIC-Evasive-PDFMal2022 dataset [11], which encompasses both benign and malicious files, and the IUST Deep Fuzz dataset [19], which comprises solely benign files, as illustrated in Figure 6. To augment the dataset with office documents, we developed a crawler to gather appropriate data, including doc, docx, xls, xlsx, ppt, and pptx formats. Additionally, we integrated datasets explained in related work [2]. The culmination of these efforts yielded a dataset comprising 52,304 PDF files and 38,576 high-variety Office documents, with a notable addition of 2,237 Office documents containing RTF objects absent from the related dataset [2]. The fourth section elaborates on the statistics detailing the acquisition of files.

3.2.2. Labeling

In the second phase, the critical task of ascertaining the maliciousness of the files initially collected is paramount. To this end, a JSON file, related to every file from the collected source, is taken from Virus Total [5]. To assign a 0 or 1 label to each sample for training binary classification models, we process the JSON file obtained from VirusTotal for the sample. But this way of labeling is enhanced with a concept called confidence score which is explained in the following sections.

3.2.3. Feature extraction

In the third phase of our analysis, we meticulously extracted a set of features. Specifically, due to their inherent differences in structure and content, extracting features from PDF documents and Office documents necessitated the application of distinct methodologies, as shown in Figure 6 and Figure 7.

For PDF documents, extracting textual and image-related features was facilitated using tools such as PymuPDF. In addition, we employed PeePDF to delve into the more intricate aspects of PDF files. This tool was instrumental in extracting features related to embedded objects, JavaScript codes, variables, and metadata, essential for understanding PDF documents' broader context and functionality. Additionally, the feature extraction process from Office documents required different tools. Aspose, olefile, Python-docx, pptx, msofficecrypto, zipfile, and Exiftools were utilized to extract metadata and text content features. Furthermore, to extract features based on VBA codes, which are often used for automating tasks within Office documents, we employed oletools.

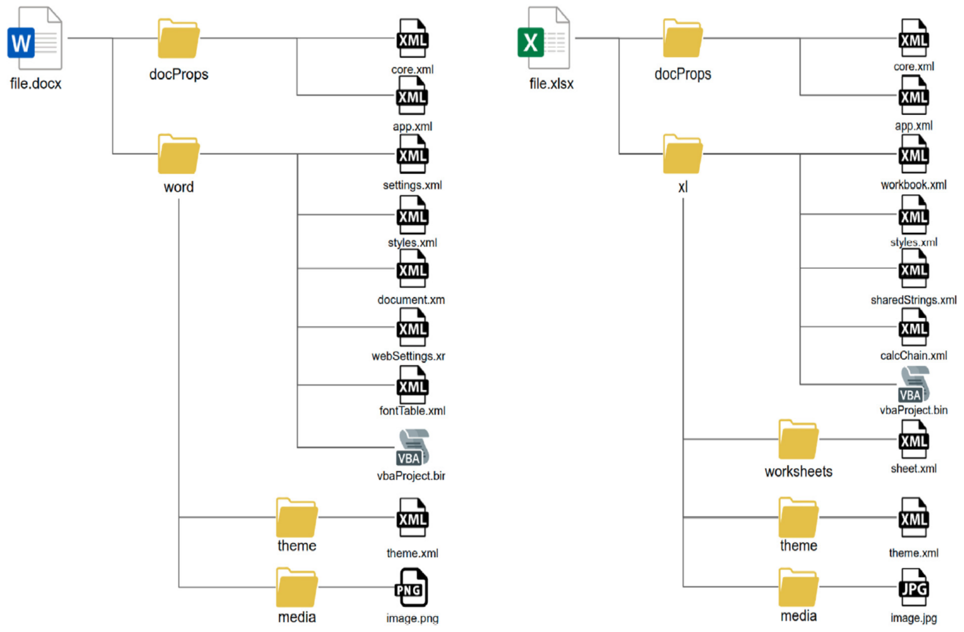


Figure 4: The structure of office documents. (a) MS Office docx file, (b) MS Office xlsx file [2].

The detailed list of features extracted for each type of PDF and Office file has been documented in Table 1 and Table 1, providing an overview of the features extracted from each document. In these tables, features unique to our study are highlighted in bold, while the remaining entries are obtained from prior research. Specifically, the unbolded elements in Table 1 from Koutsokostas et al.'s study [2] and those in Table2 2 from Issakhani et al.'s study [1] have been incorporated. More precisely, as shown in the relevant tables, in this process we extracted 34 features for office files and 66 features for PDF files.

Table 1: MS Office features used in this study.

Notation	Data Type	Description
FILE_TYPE	**Categorical**	**The file type or the MIME of the file is specified.**
FILE_EXTENSION	**Categorical**	**The file extension which is different in some cases**
NUMBER_OF_CHARS	**Numerical**	**Number of characters in office file**
NUMBER_OF_WORDS	**Numerical**	**Number of words in office file**
FILE_SIZE	**Numerical**	**File volume**
NUMBER_OF_PAGE	**Numerical**	**Number of file pages**
NUMBER_OF_LINKED_ OLE_OBJECTS	**Numerical**	**The number of dynamic links created between content in the document and content created in another Microsoft Office program**
NUMBER_OF_NORMAL_ OLE_OBJECTS	**Numerical**	**The number of embedded objects within the document**
NUMBER_OF_IMAGES	**Numerical**	**Number of different images within the content**
NUMBER_OF_URLS	**Numerical**	**Number of different URLs within the content**
HAS WRONG_STRUCTURE	**Categorical**	**An attacker can sometimes implement a vulnerability by changing the file structure and not following the storage protocol. This feature checks whether the file structure is healthy or faulty.**
IS_ENCRYPTED	**Categorical**	**File encryption makes any analysis impossible.**
HAS_MACROS	**Categorical**	**Number of different normal macros**
HAS_XLM_MACROS	**Categorical**	**Number of different XLM macros**
HAS_EXTERNAL_ RELATIONSHIPS	**Categorical**	**A connection or link between an Office file and external resources. This feature allows the office file to reference or use the content of these external sources.**
NUMBER_OF_FLASH	**Numerical**	**The number of different types of multimedia content that can be embedded in a document to add interaction or visual elements such as animation.**

Table 1: MS Office features used in this study.

Notation	Data Type	Description
FILE_ENTROPY	**Numerical**	**The amount of confusion and complexity of the file**
VBA_SIZE	**Numerical**	**Number of words within VBA**
VBA_ NUMBER_OF_CHARS	**Numerical**	**Number of chars within VBA**
VBA_ENTROPY	**Numerical**	**The amount of confusion and complexity of VBA**
VBA_SUSPICIOUS_SIZE	**Numerical**	**Number of suspicious words in VBA defined by oletools**
VBA_ NUMBER_OF_MACROS	**Numerical**	**Number of different macros within written in VBA**
IS_RTF	**Categorical**	**Using RTF within a document makes it hard to analyze**
VBA_NUMBER_OF_KILL	Numerical	The VBA code contains the kill command to delete files.
VBA_ NUMBER_OF_SHELL	Numerical	VBA uses the shell keyword to execute a shell command.
VBA_ NUMBER_OF_CALL	Numerical	VBA uses the call statement to transfer control to another procedure.
VBA_ NUMBER_OF_OPEN	Numerical	VBA uses open to manipulate a file.
VBA_ NUMBER_OF_RUN	Numerical	VBA code uses the run function to run a macro or call a function.
VBA_ NUMBER_OF_WRITE	Numerical	VBA code uses the write function to write a file.
VBA_NUMBER_OF_AUTOOPEN	Numerical	VBA code has macros named AutoOpen to execute them automatically.
VBA_NUMBER_OF_OUTPUT	Numerical	VBA code leverages the output function to write to a file
VBA_IS_BASE64	Categorical	VBA code is in base64 form for obfuscation
VBA_ NUMBER_OF_VBHIDE	Numerical	VBA code uses the vbhide parameter to hide the execution window.
VBA_ NUMBER_OF_ENVIRON	Numerical	VBA uses the ”environ” keyword to collect OS environment variables.

Table 2: PDF features used in this study.

Notation	Data Type	Description
Pdf_size	Numerical	Size of the PDF file extracted by *getsize()* method in Fitz library.
Title_characters	Numerical	The number of characters in the PDF title.
Encryption	Categorical	Check if the PDF file is encrypted or not.
Metadata_size	Numerical	Size of the metadata section in the file.
Page_number	Numerical	The number of pages in a PDF file.
Header	Categorical	Determining the format of PDF file.
Image_number	Numerical	Number of images in PDF file.
Text	Categorical	Check if the PDF file has textual content or not.
Xref_length	Numerical	Length of xref table within PDF file.
Font_objects	Numerical	The number of all fonts (directly or indirectly) referenced by the page.
Numbers_of embedded_files	Numerical	Number of embedded files in PDF file.
Number_of_Stream_keyword	Numerical	Number of “streams” in PDF file.
Number_of_endStream_keyword	Numerical	Number of “endstreams” in PDF file.
Number_of_startxref_keyword	Numerical	Number of “startxref” in PDF file.
Number_of_Trailer_keyword	Numerical	Number of ”trailers” in PDF file.
Number_of_Xref_keyword	Numerical	Number of “xref” in PDF file.
Number_of_ObjStm_keyword	Numerical	Number of “/ObjStm” in PDF file.
Number_of_JS_keyword	Numerical	Number of “/JS” in PDF file.
Number_of_JavaScript_keyword	Numerical	Number of “/JavaScript” in PDF file.
Number_of_AA_keyword	Numerical	Number of “/AA” in PDF file.
Number_of_OpenAction_keyword	Numerical	Number of “/OpenAction” in PDF file.
Number_of_AcroForm _keyword	Numerical	Number of “/AcroForm” in PDF file.
Number_of_JBIG2Decode_keyword	Numerical	Number of “/JBIG2Decode” in PDF file.
Number_of_RichMedia_keyword	Numerical	Number of “/RichMedia” in PDF file.
Number_of_Launch_keyword	Numerical	Number of “/Launch” in PDF file.
Number_of_XFA_keyword	Numerical	Number of “/XFA” in PDF file.
Trailer	**Numerical**	**Length of trailer section located at the end of PDF file structure.**
Signature	**Categorical**	**Check if the PDF file has SigFlag or not.**
Number_of_Obj_keyword	**Numerical**	**Number of “obj” in PDF file.**
Number_of_endObj_keyword	**Numerical**	**Number of “endobj” in PDF file.**
Number_of_Page_keyword	**Numerical**	**Number of “/Page” in PDF file.**
Number_of_Encrypt_keyword	**Numerical**	**Number of “/Encrypt” in PDF file.**

Table 2: PDF features used in this study.

Notation	Data Type	Description
Number_of_EmbeddedFile_keyword	**Numerical**	**Number of "/EmbeddedFile" in PDF file.**
Number_of_version	**Numerical**	**The number of different versions of the PDF file.**
Max_Tree_Depth	**Numerical**	**Maximum depth of structural Tree of PDF file.**
Number_of_urls_in_objecs_with_Js	**Numerical**	**The number of URLs in all objects contains JavaScript code.**
Number_of_unscaped_byte_in_objects_with_Js	**Numerical**	**The number of unescaped bytes in all objects contains JavaScript code.**
Number_of_Eval_keyword_in_objects_with_Js	**Numerical**	**The number of "eval" in all objects contains JavaScript code.**
Average_depth_of_tree	**Numerical**	**The average depth of the tree for all versions.**
Sd_depth_of_tree	**Numerical**	**The standard deviation of the depth of the tree for all versions.**
Number_of_leaf_of_tree	**Numerical**	**All leaves are in the object of the structural tree of all versions.**
Average_number_of_leaf_Tree	**Numerical**	**The average of the leaf nodes of each version in the structural tree of objects tree.**
Sd_number_of_leaf_Tree	**Numerical**	**The standard deviation of the leaf nodes of each version in the structural tree of objects tree.**
Number_of_keywords	**Numerical**	**Number of keywords in all JavaScript objects.**
Number_of_operators	**Numerical**	**Number of operators in all JavaScript objects.**
Avg_sum_header_byte	**Numerical**	**A list containing the sum of bytes of PDF components for each version is extracted, then the average sum of headers is calculated for all versions.**
Avg_sum_objects_byte	**Numerical**	**A list containing the sum of bytes of PDF components for each version is extracted, and then the average sum of objects is calculated for all versions.**
Avg_sum_xref_byte	**Numerical**	**A list containing the sum of bytes of PDF components for each version is extracted, and then the average sum of xref is calculated for all versions.**
Avg_sum_trailer_byte	**Numerical**	**A list containing the sum of bytes of PDF components for each version is extracted, and then the average sum of the trailer is calculated for all versions.**
Avg_sum_eof_byte	**Numerical**	**A list containing the sum of bytes of PDF components for each version is extracted, and then the average sum of eof is calculated for all versions.**
Avg_sum_compressed_byte	**Numerical**	**A list containing the sum of bytes of PDF components for each version is extracted, and then the average sum of compressed is calculated for all versions.**
Avg_avg_header_byte	**Numerical**	**A list containing the average of bytes of PDF components for each version is extracted, and then the average of headers average is calculated for all versions.**
Avg_avg_objects_byte	**Numerical**	**A list containing each version's average bytes of PDF components is extracted, and the average of objects is calculated for all versions.**
Avg_avg_xref_byte	**Numerical**	**A list containing the average bytes of PDF components for each version is extracted, and then the average of the xrefs is calculated for all versions.**
Avg_avg_trailer_byte	**Numerical**	**A list containing the average of bytes of PDF components for each version is extracted, and then the average trailer average is calculated for all versions.**
Avg_avg_eof_byte	**Numerical**	**A list containing the average of the bytes of PDF components for each version is extracted, and then the average of the average of eof is calculated for all versions.**
Avg_avg_compressed_byte	**Numerical**	**A list containing each version's average bytes of PDF components is extracted, and then the average compressed average is calculated for all versions.**
Avg_sd_header_byte	**Numerical**	**A list containing the standard deviation of PDF components' bytes for each version is extracted, and the average header standard deviation is calculated for all versions.**

Table 2: PDF features used in this study.

Notation	Data Type	Description
Avg_sd_objects_byte	**Numerical**	**A list containing the standard deviation of PDF components' bytes for each version is extracted, and the average standard deviation of the objects is calculated for all versions.**
Avg_sd_xref_byte	**Numerical**	**A list containing the standard deviation of PDF components' bytes for each version is extracted, and the average of the xref standard deviation is calculated for all versions.**
Avg_sd_trailer_byte	**Numerical**	**A list containing each version's standard deviation of PDF components' bytes is extracted, and the average trailer standard deviation is calculated for all versions.**
Avg_sd_eof_byte	**Numerical**	**A list containing the standard deviation of bytes of PDF components for each version is extracted, and then the average of the standard deviation of eof is calculated for all versions.**
Avg_sd_compressed_byte	**Numerical**	**A list containing the standard deviation of PDF components' bytes for each version is extracted, and then the average compressed standard deviation is calculated for all versions.**
Number_of_object_contain_JS	**Numerical**	**Number of objects containing JavaScript. code**
Number_of_error	**Numerical**	**It gives the number of errors in each PDF version; errors may occur that our decoder cannot decode objects or other things.**

Despite the provision of explanations within the description section in Table 2, the rationale behind the selection of numerous features may appear unclear. Consequently, the criteria for choosing features from various sections of the PDF file, based on their significance, are elucidated below:

- Header: Antivirus software primarily examines file headers to identify potential threats. Malicious PDF files often feature obfuscated and improperly formatted headers, a characteristic not typically found in benign files [20].

- Metadata: Metadata, which includes details such as the file's creation date or description, is often manipulated by attackers to conceal parts of their shell code within different file sections. These hidden sections are then referenced in the attackers' malicious payloads [21].

- JavaScript: JavaScript is considered the riskiest feature enabled in PDFs. It is often used when a user clicks a link or button or interacts with a form within the document. Attackers frequently exploit This feature due to its potential to connect to harmful URLs for malware distribution or to execute malicious shell code [22].

- Streams: Streams are primarily used to store binary data, such as images or page composition objects, and are often compressed to save space [23]. Attackers frequently conceal malicious JavaScript code within these streams because they do not have a length limit, making them an ideal hiding spot for such code.

- `ObjStm`: This keyword indicates object streams, essentially objects embedded within streams [24]. Attackers exploit this feature to hide objects containing malicious code by enveloping them within object streams [25].

- `Encoding` filters: Encoding or compression filters are applied to PDF streams to minimize their size or safeguard sensitive information. However, attackers often employ these filters to conceal malicious content [26].

- Action class: This class contains elements that respond to various events, such as clicks or drags. Attackers can easily manipulate these events to execute malware [27].

- `OpenAction/AA` object type: This object tag is a part of the Action group that represents an action or script that automatically executes when the file is opened. This feature, when combined with the execution of malicious JavaScript, has been observed in numerous exploited PDFs [28].

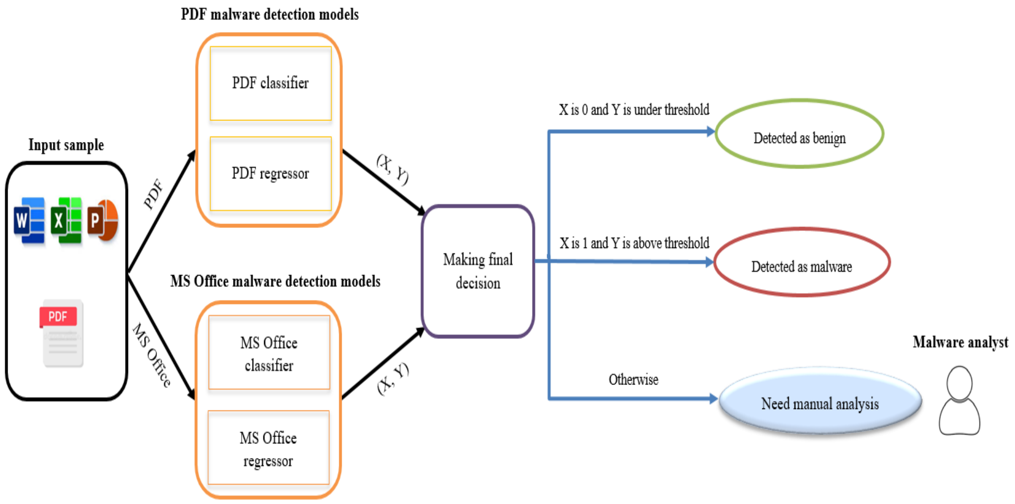


Figure 5: Overview of the detection process.

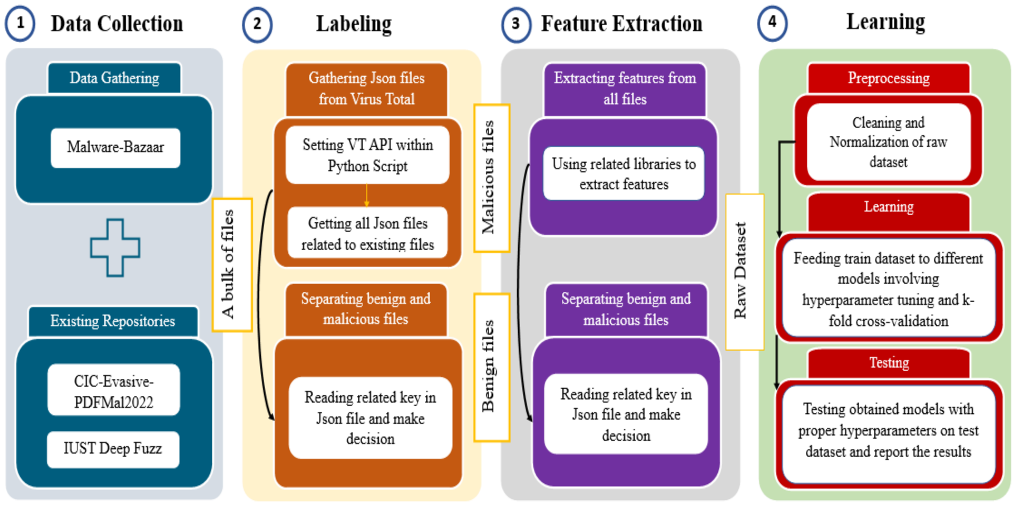
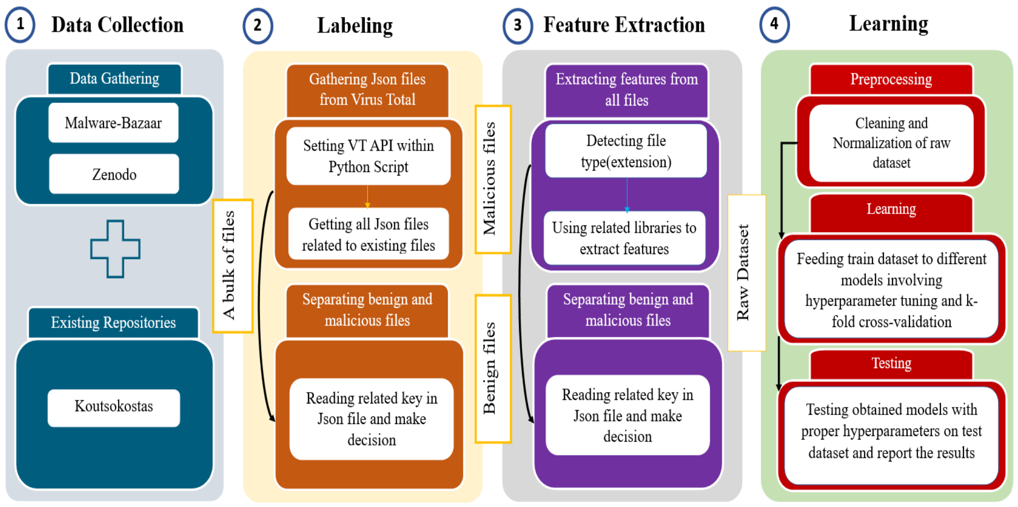

1 Data Collection
Data Gathering
Malware-Bazaar
Existing Repositories
CIC-Evasive-PDFMal2022
IUST Deep Fuzz
A bulk of files
2 Labeling
Gathering Json files from Virus Total
Setting VT API within Python Script
Getting all Json files related to existing files
Separating benign and malicious files
Reading related key in Json file and make decision
Malicious files
Benign files
3 Feature Extraction
Extracting features from all files
Using related libraries to extract features
Separating benign and malicious files
Reading related key in Json file and make decision
Raw Dataset
4 Learning
Preprocessing
Cleaning and Normalization of raw dataset
Learning
Feeding train dataset to different models involving hyperparameter tuning and k-fold cross-validation
Testing
Testing obtained models with proper hyperparameters on test dataset and report the results

Figure 6: Malicious PDF file detection methodology.

- Embedded files: PDFs can incorporate various file types, including documents like `DOCs`, `XML` files, and executable files such as `EXEs`. Malicious actors can exploit this feature to embed harmful files within a PDF document [29].
- RichMedia: PDFs can incorporate various media files and flash objects within them, which attackers can exploit to embed harmful media content into the document cui2020mmpd.
- PDF Obfuscation: PDFs can employ various obfuscation techniques, such as "Name Obfuscation", where elements within the PDF can be represented in different forms. For instance, Hexadecimal Encoding allows a JavaScript tag to be represented as `"J#53"` or a `URI` tag as `"#55RI"`. Attackers often use this method to alter the representation of malicious code to avoid detection by antivirus software [1].

3.2.4. Learning

This phase is designed to identify the optimal model for the task, focusing on binary classification. Before applying machine learning models, it is imperative to preprocess and normalize the final dataset. Initially, the dataset was split into training and testing subsets using a 0.75 to 0.25 ratio. Following this division, preprocessing steps were applied to both datasets. This preprocessing stage includes tasks such as removing duplicate rows, where each row represents a unique feature vector of a sample, and ensuring that all columns contain single-valued entries, each corresponding to a distinctly defined feature. Following preprocessing,



1 Data Collection
Data Gathering
Malware-Bazaar
Zenodo
Existing Repositories
Koutsokostas
A bulk of files
2 Labeling
Gathering Json files from Virus Total
Setting VT API within Python Script
Getting all Json files related to existing files
Separating benign and malicious files
Reading related key in Json file and make decision
Malicious files
Benign files
3 Feature Extraction
Extracting features from all files
Detecting file type(extension)
Using related libraries to extract features
Separating benign and malicious files
Reading related key in Json file and make decision
Raw Dataset
4 Learning
Preprocessing
Cleaning and Normalization of raw dataset
Learning
Feeding train dataset to different models involving hyperparameter tuning and k-fold cross-validation
Testing
Testing obtained models with proper hyperparameters on test dataset and report the results

Figure 7: Malicious MS Office file detection methodology.

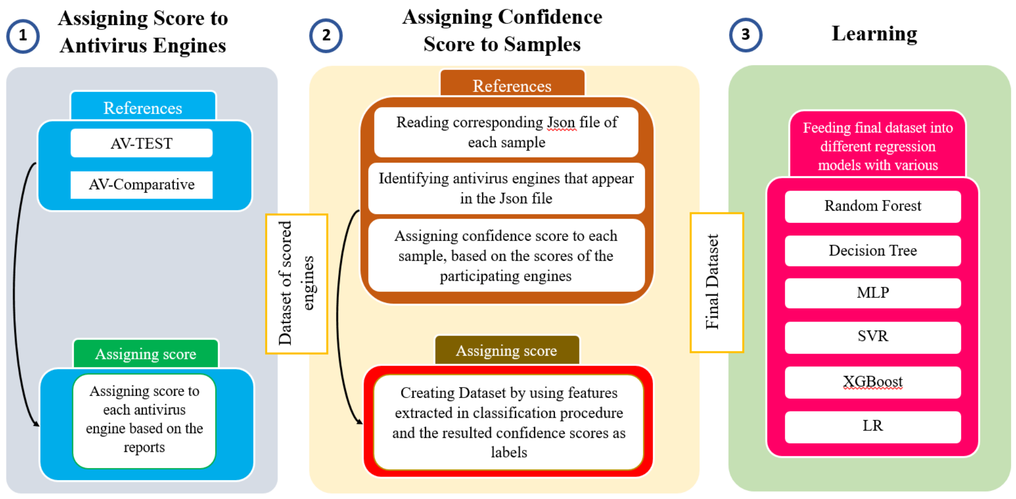
Z-score normalization is performed on numerical features to ensure that they are on a similar scale. In this investigation, a 5-fold cross-validation approach was employed to identify the optimal hyperparameters. Specifically, the grid search method was utilized through the `gridsearchcv` library in Python. This involves specifying a range of potential values for each hyperparameter, including the variable associated with the cross-validation fold count (k=5 in this case), and passing these to `gridsearchcv`. After computation, the library identifies the best hyperparameters. Importantly, these computations are executed on the training set, and the selected hyperparameters are then applied to the test set to evaluate model performance. The results of these hyperparameter tuning efforts are documented, with the optimal hyperparameters for each model being identified.

The final phase of our study thus culminates in an evaluation of the models, where the performance of each model, as determined by its optimal hyperparameters, is reported. In addition to the individual model evaluations such as Gradient Boosting [30], Multi-Layer Perceptron (MLP) [31], Random Forest [32], K-Nearest Neighbor [33], Support Vector Machine [34], Logistic Regression [35], AdaBoost [36], and Decision Tree [37], this phase also includes the application of an ensemble learning approach, explicitly employing a voting classifier. The voting classifier is designed to enhance the predictive power of the models by aggregating their predictions. This ensemble method leverages the collective strengths of the individual models to improve overall performance, thereby contributing to a more robust and accurate classification outcome. To enhance the comprehension of the findings presented in the fifth section of this study, it is essential to highlight the employment of two distinct types of voting classifiers within the research framework. Specifically, the study incorporates a voting classifier, referred to as `Voting Classifier-model3`, which is constructed through the amalgamation of three individual models: Gradient Boosting, MLP, and Random Forest. Additionally, the study introduces another voting classifier, designated as `Voting Classifier-model5`, which integrates five models: Gradient Boosting, MLP, Random Forest, KNN, and SVM.

The fifth part of our study will delve into a detailed analysis of the results obtained from each model, including the voting classifier. This analysis will provide insights into each model's performance characteristics under the binary classification task, focusing on identifying the model(s) that exhibit the highest predictive accuracy. Through this evaluation, we aim to provide a clear and actionable understanding of each model's strengths and limitations, thereby facilitating informed decision-making in applying these models to real-world problems.

3.3. Confidence score prediction

This research broadens its investigation beyond simply classifying non-executable files as either benign or malicious and further investigates an important subsequent step which is determining the degree of confidence for each file by assigning a confidence score. These scores reflect the collective consensus of antivirus software on the file's maliciousness, a significant advancement over previous methodologies that relied on binary labeling based on a single antivirus vote. By recognizing the variability in antivirus opinions, this research introduces



1
Assigning Score to Antivirus Engines
References
AV-TEST
AV-Comparative
Assigning score
Assigning score to each antivirus engine based on the reports
Dataset of scored engines
2
Assigning Confidence Score to Samples
References
Reading corresponding Json file of each sample
Identifying antivirus engines that appear in the Json file
Assigning confidence score to each sample, based on the scores of the participating engines
Assigning score
Creating Dataset by using features extracted in classification procedure and the resulted confidence scores as labels
Final Dataset
3
Learning
Feeding final dataset into different regression models with various
Random Forest
Decision Tree
MLP
SVR
XGBoost
LR

Figure 8: Confidence score determination methodology.

an approach to maliciousness classification, offering a quantifiable measure of confidence in labeling a file as malicious. This method not only addresses the issue of data labeling disparities but also provides a valuable tool for malware analysts, enabling them to prioritize their efforts based on the likelihood of potential threats.

Given the vast volume of files processed by antivirus software, this quantitative assessment is pivotal for managing and prioritizing the review of files. This research also outlines a methodology in Figure 8 to facilitate a more nuanced understanding of file benign status. The subsequent explanation will detail the procedural aspects of each phase, offering an overview of the methodology employed.

3.3.1. Assigning scores to antivirus engines

In the initial phase of the proposed methodology, it is imperative to compile reports and evidence concerning various antivirus software's operational effectiveness and detection capabilities. This process involves assigning a numerical value or a suitable weight to each antivirus software based on its performance. For this purpose, two reputable and impartial sources [38, 39], are utilized. These sources evaluate antivirus software against different parameters, comprehensively assessing their effectiveness. `AV-TEST` evaluates antivirus software based on three primary criteria: Protection, Performance, and Usability. This evaluation methodology assesses the software's ability to protect against malware, its impact on system performance, and user-friendliness.

Conversely, `AV Comparative` focuses on criteria such as Online Detection, Offline Detection, and Online Protection rates. These metrics aim to determine the software's effectiveness in detecting and blocking malware in online and offline environments. The current study intends to score antivirus software based on these criteria, particularly focusing on detection capabilities to identify malware. The scoring process systematically evaluates all antivirus software listed in `AV-TEST` and `AV Comparative` sources. The `AV-TEST` score for each software is calculated as the average of the three criteria mentioned. If a software is not listed in `AV-TEST`, its score from this source is utilized. For antivirus software not listed in either source, the average score of the rated antivirus software is calculated and used as its score.

A weighted scoring system is employed to aggregate the scores from both sources and assign a final score to each antivirus software. The `AV Comparative` score is multiplied by the `AV-TEST` score, resulting in a final score ranging from 0 to 6. This scoring system allows for an evaluation of antivirus software, with a score of 6 indicating exceptional reliability and performance and 0 indicating poor performance. This methodology scores 43 antivirus software engines based on the criteria from `AV-TEST` and `AV Comparative` sources shown in Table 3. The scale of these scores is normalized to facilitate comparison and analysis. Antivirus software not listed in the initial evaluation is addressed through a policy that will be detailed in the subsequent phase of the study.

3.3.2. Computing confidence score

In the subsequent phase following the assessment of antiviruses, our attention turns to the examination of individual `JSON` files. We identify the participating antiviruses within these files to assign confidence scores to

Table 3: Antiviruses Scores.

Antivirus name	AV-TEST average score	AV Comparative average score (%)	Final score
AhnLab	5.83	94.75	5.52
Avast	6.00	97.27	5.83
AVG	6.00	97.27	5.83
Avira	6.00	96.82	5.80
Baidu	3.50	75.33	2.63
Bitdefender	6.00	96.50	5.79
BullGuard	6.00	96.17	5.77
CheckPoint	6.00	96.17	5.77
Comodo	6.00	96.17	5.77
Cylance	3.16	72.50	2.29
EmsiSoft	5.66	93.33	5.28
Eset	5.83	94.95	5.53
F-Secure	5.83	97.80	5.70
Fortinet	5.33	90.58	4.82
GData	6.00	97.40	5.84
Heimdal	3.00	71.17	2.13
K7 Antivirus	5.33	93.20	4.96
Kaspersky	6.00	89.90	5.39
LavaSoft	4.50	83.67	3.76
Malwarebytes	5.33	92.15	4.91
McAfee	6.00	92.85	5.57
Microsoft	6.00	87.57	5.25
MicroWorld	5.66	93.33	5.28
Northguard	6.00	96.17	5.77
Norton	6.00	95.40	5.72
Panda	5.83	76.07	4.43
PC Matic	5.83	94.75	5.52
PC Tools	4.25	81.58	3.46
Total AV	5.33	96.55	5.14
Qihoo 360	2.50	67.00	1.67
quick	5.83	94.75	5.52
Sophos	5.33	90.58	4.82
Surf Shark	4.50	83.67	3.76
Tencent	5.25	89.92	4.72
Threat track	5.50	92.00	5.06
Total Def	5.79	96.50	5.58
Trend	5.83	69.02	4.02
Vipre	5.83	94.90	5.53
Webroot	2.50	67.00	1.67
DrWeb	4.66	85.00	3.96
Google	3.83	77.58	2.97
Symantec	6.00	96.17	5.77
Crowdstrike	5.33	90.58	4.82

each target file. Leveraging the previously assigned point values for each antivirus, we establish a confidence score for the malware corresponding to the `JSON` file. Note that if any attended antivirus in the `JSON` file is not found in a prepared dataset of scored antiviruses, a trivial point (1 out of 6) is assigned to that AV. To this end, a related formula has been used, illustrated in Eq. (1). In Eq. (1), if all antiviruses comment on a file's maliciousness for the file f and none of them comments to be benign, then the value of n will equal the value of m (n=m). In essence, n represents the count of antivirus programs within the `JSON` file that have deemed the sample to be malicious. m signifies the total count of antivirus engines present in the `JSON` file, regardless of their votes.

$$confidence\ score(f) = \frac{\sum\limits_{i=1}^{n} AV_i Score(f)}{\sum\limits_{i=1}^{m} AV_i Score(f)} \tag{1}$$

This process is based on the premise that if multiple antivirus engines agree on a file's maliciousness, it will likely be an important threat. Subsequently, we construct a dataset that encompasses the features extracted during the third phase of our prior methodology. However, in this iteration, the point allocations associated with each malware serve as the confidence score, effectively serving as the label. For each file, this curated dataset is now poised for utilization by machine learning models in the subsequent third phase.

3.3.3. Prediction models

In the concluding phase of this methodology, the dataset, prepared in the preceding phase, is subjected to regression analysis. This dataset encompasses features and corresponding labels integral to the learning models. Throughout this phase, an array of renowned regression models is trained using a selection of well-regarded datasets. This approach mirrors the learning phase of the prior methodology, which focused on binary classification and employed a broad spectrum of hyperparameters. To optimize the performance of these models, grid search is employed to identify the most effective hyperparameters. Upon completion of this phase, the most effective hyperparameters and the corresponding outcomes for each model are documented.

These details are provided in the fifth section of this study, ensuring transparency and thoroughness in the methodology's execution. It is important to note that, following the preprocessing actions applied to the final dataset in binary classification methodology, this methodology did not undertake further preprocessing actions. Instead, the focus was on updating the labels for the regression models, ensuring that the dataset remains in its final, prepared state for the subsequent learning phase.

3.4. Implementation details

Python programming language has been used to implement and evaluate the proposed approach on the collected dataset. After gathering samples via a Python script, we obtained `JSON` files for each sample from the Total Virus website using multiple API keys acquired through different email addresses and saved them in a text file. This strategy allowed us to collect extensive data efficiently within a short period, considering Total Virus's limitations. Utilizing the virustotal-python library and our API keys, we accessed the SHA hashes of repository files with the hashlib library. We then fetched and stored this file information in `JSON` format, preserving the file's hash name.

In the process of feature extraction, the PyMuPDF and peepdf libraries were employed to extract the first 13 features as outlined in Table 2, while the subsequent 52 features were extracted utilizing the peepdf tool. However, the extraction of features from Microsoft Office files presents a more complex scenario due to the varied structure of these files. Despite the commonality of the defined features across all types of Office files, the necessity arises to execute the extraction process for each file type with specific libraries. To address this, a design approach was implemented that involves the creation of an interface. This interface, upon identifying the file type, directs the extraction process to a specialized class tailored for extracting data from a particular file type, such as docx, xls, or pptx. Each of these specialized classes returns a list of features along with their corresponding values. During the learning phase, a suite of libraries including NumPy, pandas, scikit-learn, and scipy were utilized to execute preprocessing operations, thereby preparing the final dataset for subsequent utilization by various learning models. Additionally, the scikit-learn and Keras libraries were employed in the construction of learning models, encompassing both regression and classification tasks.

3.5. Running example

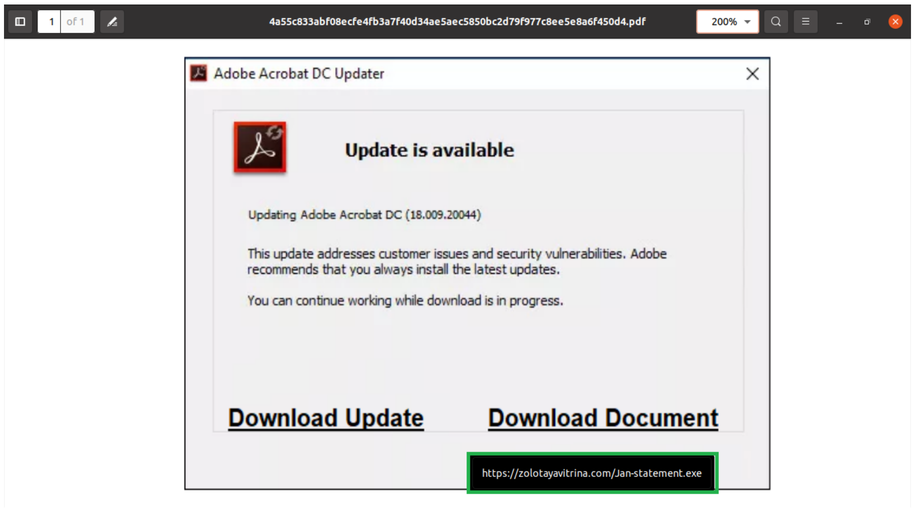
In this segment, we aim to demonstrate the superiority of our proposed method over existing approaches and current state-of-the-art techniques through a practical example. Specifically, we examine a PDF file associated with the related hash value[1]. This malware is identified by the models within our study, yet undetected by comparable methodologies found in related works. As it is obvious in Figure 9 it employs an AcroForm object to generate a graphical form, thereby inducing victims to initiate software updates. Consequently, it downloads an EXE file type of malware from a predefined location.

This malware serves as a downloader capable of exfiltrating sensitive information and introducing various auxiliary files into the compromised system. Subsequently, the command and control (C&C) server at the designated endpoint receives this data and distributes additional malicious payloads across the victim's system, facilitating further illicit activities. Such PDF-based malware typically infiltrates systems via email attachments, exploiting the unsuspecting user's trust. Our research underscores the effectiveness of our approach in identifying and mitigating threats like this.

As mentioned, this malware traps the victim by creating a fake form to continue the work by clicking on it. The structure of this form and code is shown in Figure 10:

This study enhances malware detection by integrating both statistical and structural file features. We focus on features like `Avg_sum_xref_byte` and `Avg_sum_header_byte`, which consider the essence of the file, alongside structural features such as `Average_depth_of_tree` and `Sd_number_of_leaf_Tree`. These additions aim to maximize malware detection rates. Despite the significance of `AcroForm`, previous works [1, 10] overlooking these features missed detecting certain malware. Our approach, by including these features, successfully identifies malicious files, demonstrating improved detection capabilities.

[1]Hash Value: a55c833abf08ecfe4fb3a7f40d34ae5aec5850bc2d79f977c8ee5e8a6f450d4


1
of 1
4a55c833abf08ecfe4fb3a7f40d34ae5aec5850bc2d79f977c8ee5e8a6f450d4.pdf
200%
Adobe Acrobat DC Updater

Update is available

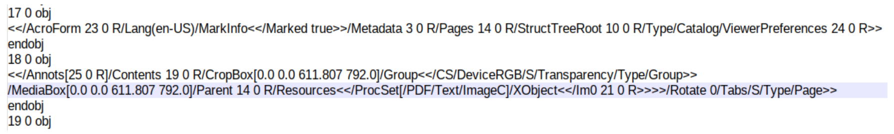
Updating Adobe Acrobat DC (18.009.20044)

This update addresses customer issues and security vulnerabilities. Adobe recommends that you always install the latest updates.

You can continue working while download is in progress.

Download Update
Download Document
https://zolotayavitrina.com/Jan-statement.exe

Figure 9: Confidence score determination methodology.



```
17 0 obj
<</AcroForm 23 0 R/Lang(en-US)/MarkInfo<</Marked true>>/Metadata 3 0 R/Pages 14 0 R/StructTreeRoot 10 0 R/Type/Catalog/ViewerPreferences 24 0 R>>
endobj
18 0 obj
<</Annots[25 0 R]/Contents 19 0 R/CropBox[0.0 0.0 611.807 792.0]/Group<</CS/DeviceRGB/S/Transparency/Type/Group>>
/MediaBox[0.0 0.0 611.807 792.0]/Parent 14 0 R/Resources<</ProcSet[/PDF/Text/ImageC]/XObject<</Im0 21 0 R>>>>/Rotate 0/Tabs/S/Type/Page>>
endobj
19 0 obj
```

Figure 10: Confidence score determination methodology.

4. Dataset

In this project, our primary objectives were to design learning models capable of distinguishing between malicious and benign files and to develop models that could estimate the confidence score of target files. To accomplish these goals, a specific set of files was required, and these files were of the office and PDF type. Furthermore, the physical versions of these files were necessary as they were to be used to extract unique features, the details of which will be explained subsequently. Consequently, many sources that merely provided the hash of the file were disregarded due to this requirement.

The strategy for file collection was predicated on the understanding that malicious files were sourced from the Zenodo and malware-bazaar database and subsequently combined with resources gathered by other related works [1, 2]. This mechanism allowed us to produce the final file collection. On the other hand, for benign files, we searched a reliable website, Zenodo [40], to acquire truly benign files. Furthermore, another project was done by Zakeri-Nasrabadi et al. [19], creating some PDF samples; these samples have also been added to our collection. To form the final file collection, these received files were incorporated into the benign sources of other related works. Details and statistics of collected files are shown in Table 4, Table 5, and Figure 11.

Table 4: Statistics of collected PDF files.

Maliciousness	Resource			
	CIC-Evasive-PDFMal2022 [11]	Malware bazaar [18]	IUST Deep Fuzz [19]	Overall
Benign	9107	38	6109	15254
Malicious	21898	1424	0	23322
Total	31005	1462	6109	38576

5. Evaluation

This segment delves into the experimental phase after elucidating the proposed methodology, the architecture's constituent components, and the dataset's specifics. It outlines the tests conducted, how they were carried out, the outcomes achieved, and the evaluation criteria utilized. This narrative is structured into distinct subsections to ensure clarity.

Table 5: Statistics of collected Office files.

Maliciousness	Resource / File Type	Malware Bazaar [18]	Koutsokostas [2]	Zenodo [40]	Overall
Benign	xls	5	2286	11	2302
	xlsx	22	0	899	921
	doc	56	682	5	743
	docx	0	0	2828	2828
	ppt	2	0	194	196
	pptx	2	0	61	63
Malicious	xls	3188	1831	0	5019
	xlsx	21277	493	0	21770
	doc	4812	11066	0	15878
	docx	1082	1558	0	2640
	ppt	164	14	0	178
	pptx	6	13	0	19
Total	All types	30616	17933	3998	52557

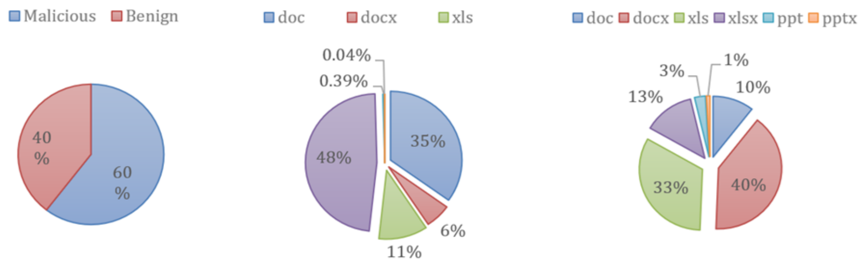


Figure 11: Distribution of file types in the proposed dataset. (a) PDF, (b) Malicious Office, (c) Benign Office.

5.1. Experimentation setup

All experiments were performed on an Ubuntu 18.04 (x64) machine with a 2.8 GHz Intel® Core™ i7 6700HQ CPU and 12 GB RAM. The static metrics extraction and preprocessing of the primary dataset for PDF and Office on this machine took about 170 hours. In the next step, we utilized various implementations of scikit-learn to use a different learning model. The ML-Flow tool was used to manage different versions of the trained model. This step on the mentioned machine took 82 hours on the PDF dataset and 69 hours on the Office dataset.

5.2. Research questions

In advancing knowledge and understanding in the field of study, it is important to formulate clear, focused research questions that guide the investigation. Based on this, we have formulated the following five main research questions as a guideline for our experiments:

- RQ1: Which machine learning classifier model most effectively detects non-executable complex malware?
  - RQ1.1: what is the best model to detect PDF malwares?
  - RQ1.2: what is the best model to detect Office malwares?
- RQ2: What are the most important features for distinguishing non-executable complex malicious files?
  - RQ2.1: what features are the most significant in detecting PDF malwares?
  - RQ2.2: what features are the most significant in detecting Office malwares?
- RQ3: What is the effectiveness of our approach in classification models compared with other related work?
- RQ4: Which machine learning regression model most effectively determines the confidence score of non-executable complex files?
  - RQ4.1: what is the best model for PDF malwares?
  - RQ4.2: what is the best model for Office malwares?

- RQ5: How effective is our approach in regression models compared with the existing services that report the confidence score?
- RQ6: How does incorporating confidence scores enhance the detection accuracy of non-executable, complex malicious files?

5.3. Evaluation metrics

Establishing a comprehensive set of criteria is imperative for evaluating each model's performance across various tests and experiments. This necessity is not exclusive to machine learning models but is a fundamental requirement for assessing the effectiveness of binary classification and regression models. In this study, both binary classification and regression models are employed, necessitating a detailed examination of the evaluation criteria applicable to each.

Regression models predict a continuous outcome variable based on one or more input features. The evaluation of these models is often conducted using metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and Median Absolute Error (MedAE). These metrics provide a quantitative measure of the model's prediction accuracy and the magnitude of its errors. Evaluating the performance of Binary classification models involves metrics such as accuracy, precision, recall, and the F1-score, which provide insights into the model's ability to correctly classify instances and its sensitivity to false positives and negatives [41]. To explain these criteria better, the definitions of essential components in the confusion matrix, which are true positive, true negative, false positive, and false negative, are utilized. Based on these definitions, the examined criteria for binary models are as follows:

- Accuracy is a metric used to evaluate the performance of a machine-learning model. It is the ratio of correct predictions to the total number of predictions. Here is the formulation of the accuracy metric:

$$Accuracy = \frac{CorrectPredictions}{AllPredictions}$$

Or:

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

- Precision is the number of true positive results divided by the number of all positive results, including those not identified correctly. The formula below illustrates precision formulation:

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

- Recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Recall Formulation has been exhibited below:

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

- F-score, also known as the F1-score or F-measure, is a metric used to evaluate the performance of a machine learning model, particularly in binary classification systems. It is the harmonic mean of the model's precision and recall. The formulation of the F-score has been shown in the following formula:

$$F\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Or in basic form:

$$F\ score = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

As mentioned, some other criteria related to regression models exist, which are brought in the following paragraphs. Before examining them, some variables used in the related formulation of criteria should be defined. Hence, we have them as follows:

- N: Number of observations,

- y_i: The i'th expected value,
- $\hat{y}_i$: The i'th predicted value.

- MSE: Mean Square Error (MSE) is the average of the squared differences between the actual and predicted values. The formulation related to MSE has been illustrated in the following formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- MedAE: The median absolute error (MedAE) is a metric used to evaluate regression models' performance. It is the median of the absolute differences between the actual and predicted values.

$$\text{MedAE} = \text{median}\left(|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|\right)$$

- MAE: The Mean Absolute Error (MAE) is a metric used to evaluate the performance of regression models. It is the average absolute difference between the actual and predicted values. The following formula is associated with MAE criteria:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- RMSE: The Root Mean Square Error (RMSE) is a popular metric used to evaluate the performance of regression models. It is the square root of the Mean Squared Error (MSE), the average of the squared differences between the actual and predicted values. The RMSE provides a measure of how spread out the residuals are. It quantifies the average difference between the predicted and actual values, revealing how tightly the observed data clusters around the predicted values. A lower RMSE indicates that the model fits the data well and has more precise predictions. RMSE formulation has been brought in below:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

5.4. Results

In this part, we respond to the research questions from section 5.2 by sharing the evidence and documents we found from our experiments. At the end of each section, we raise the related research question and answer based on our presentation.

5.4.1. Detection models performance

Concerning RQ1, the effectiveness of each learned model was measured with standard metrics used for evaluating the performance of classification models, including accuracy, precision, recall, and finally, F-score. Table 6 shows the evaluation metrics for all models learned using the provided dataset of PDF files. The best value obtained for each evaluation metric is bolded in this table. According to the evaluation results given in this table, it concluded that:

- The gradient-boosting classifier performs better than all other models in accuracy, precision, recall, and f-score. However, voting classifier3 shows the same accuracy and precision outcome as gradient boosting.
- The combination of multiple models, such as voting classifier3 and voting classifier5, has not performed better than a singular gradient boosting model.

On the other hand, to satisfy RQ1 completely, the abovementioned metrics for the PDF dataset should also be explained for the Office dataset. Therefore, related results of different classification models learned with the Office dataset have been reported in Table 7. The best value obtained for each evaluation metric is bolded, the same as the PDF table in Table 7. According to the evaluation results given in this table, it inferred that:

Table 6: Best outcomes of each model related to the PDF dataset.

Model name	Evaluation criteria			
	Accuracy	F-score	Precision	Recall
Random Forest	0.991	0.989	0.992	0.987
Gradient Boosting	**0.993**	**0.992**	**0.994**	**0.990**
Decision Tree	0.960	0.952	0.993	0.913
MLP	0.989	0.987	0.991	0.984
KNN	0.980	0.976	0.991	0.963
AdaBoost	0.986	0.983	0.985	0.981
Logistic Regression	0.964	0.958	0.979	0.938
SVM	0.977	0.973	0.987	0.960
Voting Classifier-model3	**0.993**	0.991	**0.994**	0.989
Voting Classifier-model5	0.991	0.989	0.993	0.985

Table 7: Best outcomes of each model related to the Office dataset.

Model name	Evaluation criteria			
	Accuracy	F-score	Precision	Recall
Random Forest	**0.994**	**0.996**	**0.997**	**0.995**
Gradient Boosting	0.992	0.995	**0.997**	0.994
Decision Tree	0.982	0.989	0.993	0.986
MLP	0.989	0.994	0.992	**0.995**
KNN	0.986	0.992	0.992	0.991
AdaBoost	0.987	0.992	0.991	0.993
Logistic Regression	0.969	0.982	0.990	0.973
SVM	0.984	0.991	0.989	0.993
Voting Classifier-model3	**0.994**	**0.996**	**0.997**	**0.995**
Voting Classifier-model5	0.990	0.994	0.993	**0.995**

- The gradient random forest and voting classifier3 perform better than all other models in accuracy, precision, recall, and f-score. Although gradient boosting shows the same outcome as random forest in terms of precision and MLP and voting classifier5 in terms of recall.
- The combination of multiple models, such as voting classifier3 and voting classifier5, has not performed better than the singular random forest model.

As discussed in the fourth section, the imbalance in the dataset, particularly with Office files and PDFs, poses a challenge that could impact the results. Despite this, the F-score values exhibit promising figures, suggesting minimal influence from the unbalanced data on the models. However, to substantiate this claim further, it's essential to incorporate additional supporting evidence. Consequently, the ROC curve, which provides insight into the models' performance and reliability, is presented in Figure 12 and Figure 13.

In the third section, the explanation of hyperparameters for each model was given. This process involved defining a set of hyperparameters and systematically employing a grid search technique to explore the parameter space. The grid search methodology is a well-established approach in machine learning for hyperparameter tuning, allowing for identifying the most suitable hyperparameter values for each model. The most effective configurations were determined and documented by systematically testing various combinations of hyperparameters. These configurations are presented in Table 8 and Table 9. Table 8 outlines the optimal hyperparameters for learning models on PDF datasets, while Table 9 details the best parameters for models trained on Office datasets. This process ensures the models are configured with the most suitable settings, enhancing their predictive accuracy and reliability.

Table 8: Best configuration of each model for the PDF dataset.

Model	Best Configuration
Random Forest	`{'criterion': entropy, ' max_depth ': 20, 'max_features': None, ' n_estimators ': 350}`

Table 8: Best configuration of each model for the PDF dataset.

Model	Best Configuration
Adaboost	{’algorithm’: SAMMER, ’n_estimators’: 350}
Decision Tree	{’criterion’: ’entropy’, ’max_depth’: 5, ’splitter’: best}
MLP	{’activation’: ’tanh’, ’hidden_layer_sizes’: (32, 64, 32), ’learning_rate’: ’constant’, ’max_iter’: 550, ’solver’: ’adam’}
SVM	{’C’: 4, ’cache_size’: 100, ’degree’: 1, ’gamma’: auto}
Logistic Regression	{’max_iter’: 200, ’penalty’: None, ’solver’: newton-cholesky}
KNN	{’algorithm’: auto, ’leaf_size’: 10, ’n_neighbors’: 2}
Gradient boost	{’criterion’: friedman_mse, ’loss’: exponential, ’max_depth’: 7, n_estimators: 250}
Voting classifier model3	[(’Model_1’, GradientBoostingClassifier(loss=’exponential’, max_depth=7, n_estimators=250, random_state=0)), (’Model_2’, MLPClassifier(activation=’tanh’, hidden_layer_sizes=250, max_iter=550, random_state=17)), (’Model_3’, RandomForestClassifier (criterion=’entropy’, max_depth=20, max_features=None, n_estimators=350, random_state=17))]
Voting classifier model5	[(’Model_1’, GradientBoostingClassifier(loss=’exponential’, max_depth=7, n_estimators=250, random_state=0)), (’Model_2’, MLPClassifier(activation=’tanh’, hidden_layer_sizes=250, max_iter=550, random_state=17)), (’Model_3’, RandomForestClassifier (criterion=’entropy’, max_depth=20, max_features=None, n_estimators=350, random_state=17)), (’Model_4’, KNeighborsClassifier(leaf_size=10, n_neighbors=2)), (’Model_5’, SVC(C=4, cache_size=100, degree=1,gamma=’auto’, probability=True, random_state=17)]

Table 9: Best configuration of each model for the Office dataset.

Model	Best Configuration
Random Forest	{’criterion’: gini, ’ max_depth ’: 30, ’max_features’: sqrt, ’ n_estimators ’: 300}
Adaboost	{’algorithm’: SAMMER, ’n_estimators’: 300}
Decision Tree	{’criterion’: entropy, ’max_depth’: 5, ’ splitter’: best}
MLP	{’activation’: relu, ’hidden_layer_sizes’: (32, 64, 32), ’learning_rate’: constant, ’max_iter’: 100, ’solver’: adam}
SVM	{’C’: 4, ’cache_size’: 100, ’degree’: 1, ’gamma’: auto}
Logistic Regression	{’max_iter’: 150, ’penalty’: None, ’solver’: newton-cg}
KNN	{’algorithm’: auto, ’leaf_size’: 10, ’n_neighbors’: 6}
Gradient boost	{’criterion’: squared_error, ’loss’: exponential, ’max_depth’: 7, n_estimators: 150}
Voting classifier model3	[(’Model_1’, GradientBoostingClassifier(criterion=’squared_error’, loss=’exponential’, max_depth=7, n_estimators=150, random_state=0)), (’Model_2’, MLPClassifier(hidden_layer_sizes=(32, 64, 32), max_iter=100, random_state=17)), (’Model_3’, RandomForestClassifier(max_depth=30, n_estimators=300, random_state=17))]
Voting classifier model5	[(’Model_1’, GradientBoostingClassifier(criterion=’squared_error’, loss=’exponential’, max_depth=7, n_estimators=150, random_state=0)), (’Model_2’, MLPClassifier(hidden_layer_sizes=(32, 64, 32), max_iter=100, random_state=17)), (’Model_3’, RandomForestClassifier(max_depth=30, n_estimators=300, random_state=17)), (’Model_4’, KNeighborsClassifier(leaf_size=10, n_neighbors=6)), (’Model_5’, SVC(C=4, cache_size=100, degree=1, gamma=’auto’, probability=True, random_state=17))]

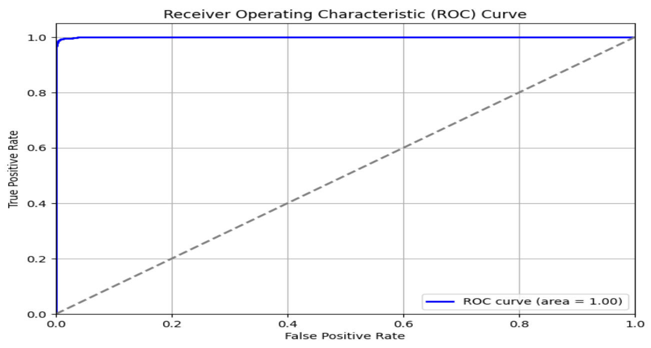


Figure 12: ROC curve of best model related to PDF files.

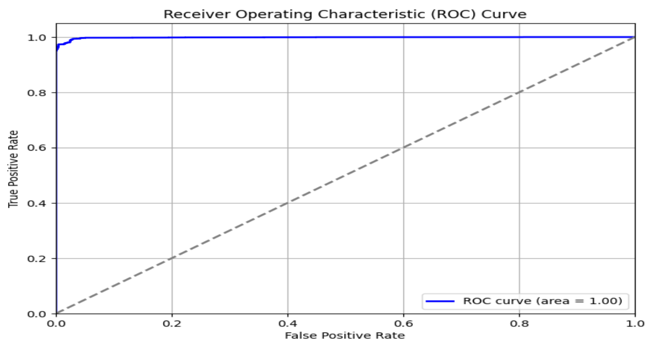


Figure 13: ROC curve of best model related to MS Office documents.

RQ1: *Which machine learning classifier model most effectively detects non-executable complex malwares?*

- **RQ1.1**: *What is the best model to detect PDF malwares?*
- **RQ1.2**: *What is the best model to detect Office malwares?*

Answer to RQ1: *The models are evaluated using the best configurations for each model, as detailed in Table 6 and Table 7. The Gradient Boosting model performs best on the PDF dataset across multiple criteria, including Accuracy, F-score, Precision, and Recall. Conversely, the Random Forest and Voting Classifier-model 3 perform best on the Office dataset across all defined criteria. This analysis is presented to support the research questions RQ1.1 and RQ1.2, which aim to compare the performance of these models on different datasets.*

5.4.2. Determining the most significant features

The significance of elucidating the functionality of machine learning models cannot be overstated, given that these models often operate as "black boxes," posing challenges in understanding their inner workings. Descriptive learning methods, such as the SHAP (Shapley Additive Explanations) method, are instrumental in disclosing the influence of various features on the model's output. SHAP [42], a method rooted in cooperative game theory, is particularly adept at enhancing the transparency and interpretability of machine learning models.

Based on the explanations provided, it is necessary to mention in this study the features that have had the most significant impact on the learning of the best model, which is gradient boost for the PDF dataset and also random forest for the Office dataset, to increase the interpretability and understanding of these models. For this reason, the SHAP diagram for these two models shown in Table 6 and Table 7 are presented in Figure 14 and Figure 15, respectively.

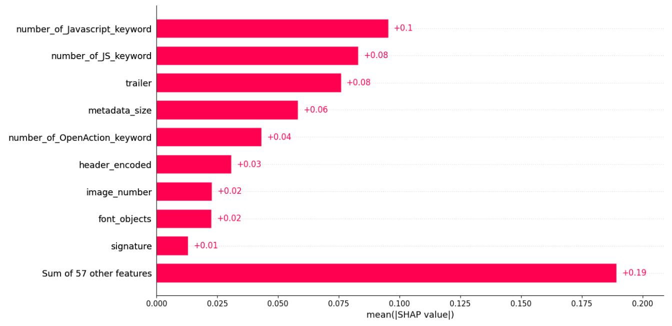


Figure 14: Significant features in PDF files.

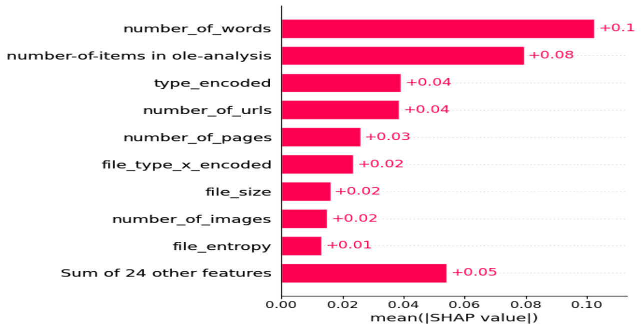


Figure 15: Significant features in MS Office files.

RQ2: *What are the most important features for distinguishing non-executable complex malicious files?*

- **RQ2.1**: *What features are the most significant in detecting PDF malwares?*
- **RQ2.2**: *What features are the most significant in detecting Office malwares?*

Answer to RQ2: *As shown in Figure 14, the most effective and important features in PDF malware detection are number$_o f_j$avascript, number$_o f_j s_k$eyword, trailer, and metadata$_s$ize, respectively, which indicates the importance of javascript codes to analysis in the PDF files (RQ2.1). On the other hand, based on what is shown in Figure 15, the most essential features in the detection of MS Office malware are number$_o f_w$ords, number$_o f_i$tems in ole-analysis, type$_e$ncoded and number$_o f_u$rls respectively, which shows the importance of analyzing dangerous links and the number of Suspicious words in the VBA code (RQ2.2). This study presents a comprehensive enhancement to document processing, introducing 66 features for PDF files, including 40 new additions, alongside 34 features for Office files, with 23 being newly developed, marking advancements in handling these file types.*

5.4.3. Comparison with state-of-the-art detection models

In malware detection, specifically targeting Office and PDF files, various studies have been conducted in recent years to evaluate and demonstrate the effectiveness of different methodologies. These studies have predominantly focused on identifying a singular file type, PDF or Office. Conversely, the current research endeavors to encompass both file types. This approach facilitates a nuanced evaluation for both PDF and Office files, taking into account the datasets, feature selections, and learning models employed in each study, along with their resultant outcomes as detailed in Table 10 and Table 12.

Comparing our proposed method with various related works presents a formidable challenge due to the absence of complete measurements and difficulty accessing the exact datasets used for training and testing models. Nevertheless, we have briefly summarized some articles that reported measurable outcomes and

Table 10: Comparison of different approaches for detecting malicious PDF files.

Reference	Dataset	Dataset availability	Features	Method(s)	Available Hyperparameters	Outcomes
[9]	Collection of PDF files obtained from Virus Total for a total of 65942	NO	A set of JavaScript-related features	One-Class Support Vector Machine (SVM)	NO	Best Accuracy 85%
[8]	Collection of 22196 malware and 40441 benign files	NO	A set of JavaScript-related features	One-Class Support Vector Machine (SVM)	NO	Best Accuracy 96.93%
[1]	A refined version of the Contagio dataset [43] integrated with Virus Total is called the Evasive-PDFMal2022 dataset, which has 5557 malicious and 4468 benign entries.	YES	File structure, File content, and JavaScript for a total of 28 features	Stack Learning with base learners RF, MLP, SVM, AdaBoost, and meta learners LR, KNN, DT	NO	The Best F-score with LR is 99.86%, recall is 99.88%, precision is 99.84%, and accuracy is 99.89%.
[10]	The evasive-PDFMal2022 data set mentioned in the [11]	YES	Same as [1] Features	Optimized Decision Tree(O-DT)	NO	Best Accuracy 98.84%
Our Approach	Benign samples collected from malware-Bazaar integrated with Evasive and IUST Deep fuzz samples and malicious samples collected from malware-Bazaar integrated with Evasive samples, for a total of 15254 benign and 23322 malicious	YES	File structure, File content, and JavaScript for a total of 64 features	SVM, RF, Gradient Boosting, MLP, DT, AdaBoost, KNN, Logistic Regression, Voting Classifier-model3, Voting Classifier-model5	YES	Best Accuracy 99.3%, F-score 99.2%, Precision 99.4%, and Recall 99% with Gradient Boosting

Table 11: Comparison between the robustness and stability of the results.

Method	Dataset	
	Evasive-PDFMal2022	Our dataset
The best model of the Issakhani et al. study [1] (Linear Regression)	Accuracy of 99.89%, F-score of 99.86%, Precision of 99.84%, and Recall of 99.88%	Accuracy of 97.26%, F-score of 94.11%, precision of 95.68%, and recall of 92.58%
Best model of our study (Gradient Boosting)	Accuracy of 99.95%, F-score of 99.95%, precision of 100%, and recall of 99.91%	Accuracy of 99.3%, F-score of 99.2%, Precision of 99.4%, and Recall of 99%

compared their findings in Table 10. Our approach has demonstrated superior test accuracy compared to other experiments. Although the research conducted by Issakhani et al. [1] exhibited better results, we conducted experiment to prove the superiority of our research, considering the dataset differences. The results illustrated in Table 11 indicate that our model achieves better results across both datasets, demonstrating robustness and resistance to dataset size variations.

Conversely, the model provided in mentioned study [1] showed diminished outcomes on a larger dataset with its best meta-learner which is Linear Regression. Furthermore, our research stands out for its transparency, including the availability of the dataset source and the disclosure of selected hyperparameters for the learning models, alongside the implementation details. These elements underscore the clarity of our experiments, offering a valuable reference for future researchers.

As evidenced in Table 12, this article presents distinct advantages over related works in detecting MS Office malware. The research conducted in this study not only offers a more comprehensive and varied dataset incorporating both .ppt and .pptx files but also achieves superior outcomes across all four introduced criteria: accuracy, precision, f-score, and recall. Furthermore, this study uniquely endeavors to identify each learner model's optimal parameters and implement fine-tuning, a feature not commonly observed in other studies. Additionally, the availability of the implemented implementations and the collected datasets further distinguishes this research from its counterparts.

Table 12: Comparison of different approaches for detecting malicious Office files.

Reference	Dataset	Dataset availability	Features	Method(s)	Hyperparameter availability	Outcomes
[14]	A random collection of office files, collected according to VirusTotal analysis, included 2,537 files and 4,212 macros (sometimes more than one per file), of which 877 were obfuscated.	NO	A set of 15 lexicographical and function call features	SVM, RF, MLP, LDA, BNB	NO	The different machine-learning approaches report accuracies of around 90% in the task of identifying obfuscated macros. MLP was the most prominent with a 92% $F2$ score.
[13]	7,145 samples, including macros retrieved from VirusTotal	NO	Different language processing-related features, including SCDV, LSI, Doc2vec, Bag-of-words	SVM	NO	The best $F1$ score reported is 93%.
[2]	Benign samples with macros collected from official sites and malicious samples collected from VirusTotal AppAny, Virusign, and Malshare, for a total of 2736 benign and 15,571 malicious	YES	Lexicographical, VBA statistics, and function call analysis (LOLBAS, PowerShell, PSDecode)	SVC, RF, XGBoost, MLP	YES	The best F1 score was above 98%, and best accuracy score was 97.5% with RF, and recall was above 97.5% with RF. In the F2 score, 98% with RF was obtained.
Our Approach	Benign samples with macros collected from Zenodo and malware-Bazaar integrated with related samples [2] and malicious samples collected from malware-Bazaar integrated with related samples [2] , for a total of 7053 benign and 45504 malicious	YES	File structure File content, VBA structure, VBA content, external relationships, and RTF-related features	SVM, RF, Gradient Boosting, MLP, DT, AdaBoost, KNN, Logistic Regression, Voting Classifier-model3, Voting Classifier-model5	YES	Best F-score 99.6% with RF and Recall 99.5% and Precision 99.7% and Accuracy 99.4% with RF

RQ3: *What is the effectiveness of our approach in classification models compared with other related work?*
Answer to RQ3: *As demonstrated across multiple tables, the learning models introduced in this study effectively identified non-executable complex malware. Specifically, for PDF malware detection, our gradient boosting model achieved an impressive 99.3% accuracy, surpassing the performance of the random forest model used in related studies. Similarly, in detecting MS Office malware, our model achieved a 99.4% accuracy, outperforming the random forest model in this context. The enhancement observed in the performance of the constructed models relative to previous models stems from the variety of features chosen. These features effectively address numerous attack scenarios commonly encountered in such files. Additionally, another contributing factor to this improvement is the substantial increase in the volume of collected data, surpassing that of comparable studies. .*

5.4.4. Confidence score prediction models performance

To address research question 4, it is needed to evaluate the performance of various regression models in this domain for both PDF and Office datasets. The outcomes of the regression models applied to the PDF and Office datasets are illustrated in Table 13 and Table 14, respectively.

In the 3rd section and, to a lesser extent, in section 5.4.1, the process of optimizing learning models was detailed. In section 5.4.1, Table 8 and Table 9 displayed the optimal hyperparameters for each classifier model employed. Given the focus on regression models in this study, it is required to list the best hyperparameters for each regression model. These hyperparameters are the foundation for the results presented in Table 13 and Table 14, which are reflected in Table 15 and Table 16, showcasing the optimal hyperparameters for each regression model used for the learning process on PDF and Office datasets, respectively.

Table 13: The outcomes of regression models for PDF based on the best configuration of them.

Model	Evaluation Criteria			
name	MSE	MedAE	MAE	RMSE
Random Forest	**0.0006**	**0.0005**	**0.0096**	**0.0256**
XGBoost	0.0047	0.0063	0.0272	0.0689
Decision Tree	0.0080	0.0068	0.0331	0.0896
MLP	0.0064	0.0280	0.0478	0.0806
SVR	0.0205	0.1333	0.1276	0.1432
Linear Regression (LR)	0.0224	0.0688	0.1045	0.1498

Table 14: The outcomes of regression models for office based on the best configuration of them.

Model	Evaluation Criteria			
name	MSE	MedAE	MAE	RMSE
Random Forest	**0.0006**	**0.0005**	**0.0096**	**0.0256**
XGBoost	0.0047	0.0063	0.0272	0.0689
Decision Tree	0.0080	0.0068	0.0331	0.0896
MLP	0.0064	0.0280	0.0478	0.0806
SVR	0.0205	0.1333	0.1276	0.1432
Linear Regression (LR)	0.0224	0.0688	0.1045	0.1498

RQ4: *Which machine learning regression model most effectively determines the confidence score of non-executable complex files?*

- **RQ4.1**: *What is the best model for PDF malwares?*
- **RQ4.2**: *What is the best model for Office malwares?*

Answer to RQ4: *Table 13 reveals that the Random Forest model surpasses all others across various metrics, including Mean Squared Error (MSE), Median Absolute Error (MedAE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). Hence, the results indicate that the Random Forest model is the most effective on the PDF dataset, as supported by RQ4.1. Additionally, for Office dataset the Random Forest model maintains its superior performance across all criteria, outperforming other models as defined in Table 14. Therefore, this supports the conclusion that the Random Forest model is the most effective for the Office dataset, as indicated by RQ4.2.*

5.4.5. Confidence score model verification

In Section 5.4.3, it was elucidated that to substantiate the efficacy of a study and to demonstrate enhancements compared to prior research, the methodology employed must be juxtaposed with existing works. Consequently, this section aims to unveil the efficacy of regression models and, more broadly, assign a confidence score to each related sample. Given the absence of similar methodologies in the extant literature, the outcomes were contrasted with alternative approaches distinct from those discussed in Section 5.4.3. After an exhaustive review, the Hybrid Analysis [44] was identified as a comparative benchmark. This platform collaborates with reputable malware analysis entities, including Virus Total, Meta Defender, and Crowdstrike Falcon, to assign a confidence score to each file under examination by each agent.

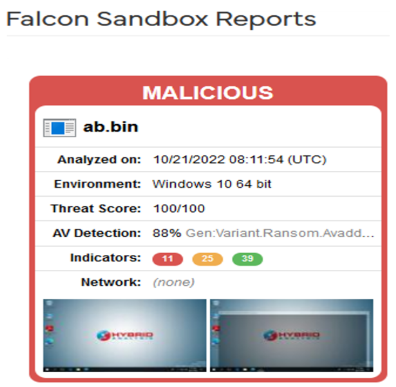
The scoring mechanism generates an output akin to that depicted in Figure 16 and Figure 17. In Figure

Table 15: Best configuration for each regression model on the PDF dataset.

Model	Best Configuration
Random Forest	`{'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}`
XGBoost	`{'gamma': 0, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 300}`
Decision Tree	`{'criterion': 'friedman_mse', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 6}`
MLP	`{'activation': 'tanh', 'alpha': 0.01, 'hidden_layer_sizes': (50, 50, 50), 'learning_rate': 'constant', 'max_iter': 200, 'solver': 'adam'}`
SVR	`{'C': 100, 'epsilon': 0.2, 'kernel': 'rbf'}`
Linear Regression (LR)	`{'fit_intercept': Flase}`

Table 16: Best configuration for each regression model on the Office dataset.

Model	Best Configuration
Random Forest	{'max_depth': 20, 'min_samples_split': 5, 'n_estimators': 100}
XGBoost	{'gamma': 0, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 300}
Decision Tree	{'criterion': 'friedman_mse', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 8}
MLP	{'activation': 'relu', 'alpha': 0.001, 'hidden_layer_sizes': (50, 100, 50), 'learning_rate': 'adaptive', 'max_iter': 200, 'solver': 'adam'}
SVR	{'C': 100, 'epsilon': 0.2, 'kernel': 'rbf'}
Linear Regression (LR)	{'fit_intercept': True}


Falcon Sandbox Reports

MALICIOUS

ab.bin

Analyzed on: 10/21/2022 08:11:54 (UTC)
Environment: Windows 10 64 bit
Threat Score: 100/100
AV Detection: 88% Gen:Variant.Ransom.Avadd...
Indicators: 11 25 39
Network: (none)

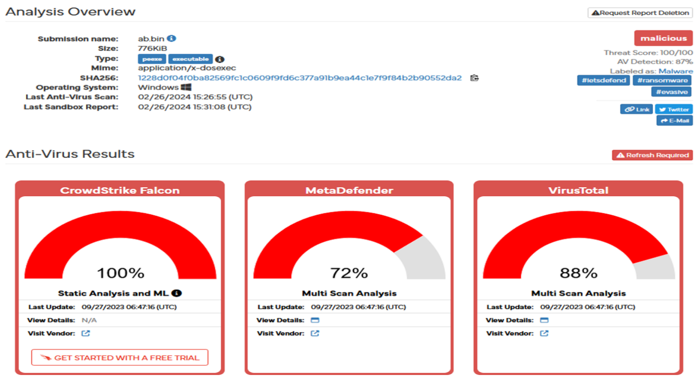
Figure 16: Score of falcon dynamic for a given file in Hybrid-Analysis [44].

17, an exemplary file under examination is illustrated, showcasing the output of four distinct sources, each providing a confidence score. Specifically, the scores assigned by Falcon Static, MetaDefender, VirusTotal, and Hybrid Analysis are 100, 72, 88, and 87, respectively. The Hybrid Analysis score is prominently displayed at the top of the image under the heading "AV Detection Score." Additionally, the published report includes the Falcon Dynamic score, derived from sandbox results, which is presented in Figure 16. For this particular instance, the Falcon Dynamic score is 88.

Based on the provided explanations, we compiled a dataset comprising 100 malicious and benign files, ensuring balanced diversity in confidence scores, which the regression model produces. For each file, we assigned scores corresponding to analyses from CrowdStrike Falcon (both static and dynamic), MetaDefender, VirusTotal, Hybrid Total, and a Confidence Score. Additionally, a Classifier score was assigned to each file and we talk more about it in the next paragraph. This dataset was constructed for PDF and Office file types, resulting in a dataset of 100 records across seven columns. These columns are ordered: Confidence Score, Falcon Static, Falcon Dynamic, VirusTotal, MetaDefender, Hybrid-Analysis AV-Detection (Hybrid Total), and Classifier score.

The Hybrid-Analysis AV-Detection column which is illustrated as Hybrid Total is specific to the Hybrid-Analysis server, as depicted in Figure 17. The approach identified following the investigations represents an average of the other methodologies provided by the website, encompassing Falcon Static, Falcon Dynamic, Meta Defender, and Virus Total. As depicted in Figure 18 and Figure 19, a notable correlation exists between the confidence score derived from the regression model developed in this study and the Hybrid Total AV-detection approach, as indicated in the heatmap columns. Specifically, the correlation values are 95% for PDF files and 76% for MS Office documents.

Notably, the Classifier column is populated with a floating point generated by a binary classifier model for each sample, ranging from 0 to 1. In other words, the classifier model assigns a label of 0 or 1 to each sample based on its learning known as probability score. The probability score, which serves as a measure of confidence indicating the likelihood of a file being malicious or benign, exhibits a low correlation with other columns, as depicted in Figure 18 and Figure 19. This correlation weakness underscores its inadequacy as a substitute for confidence scores generated by alternative methodologies. The discrepancy arises from the training of classifier models on binary labels (0 and 1), where the significance of antivirus weights is disregarded in the labeling process. In contrast, regression models incorporate floating point labels derived



Analysis Overview

Submission name:	ab.bin
Size:	776KiB
Type:	peexe executable
Mime:	application/x-dosexec
SHA256:	1228d0f04f0ba82569fc1c0609f9fd6c377a91b9ea44c1e7f9f84b2b90552da2
Operating System:	Windows
Last Anti-Virus Scan:	02/26/2024 15:26:55 (UTC)
Last Sandbox Report:	02/26/2024 15:31:08 (UTC)

malicious
Threat Score: 100/100
AV Detection: 87%
Labeled as: Malware
#letsdefend #ransomware #evasive

Anti-Virus Results

CrowdStrike Falcon		MetaDefender		VirusTotal	
100%		72%		88%	
Static Analysis and ML		Multi Scan Analysis		Multi Scan Analysis	
Last Update:	09/27/2023 06:47:16 (UTC)	Last Update:	09/27/2023 06:47:16 (UTC)	Last Update:	09/27/2023 06:47:16 (UTC)
View Details:	N/A	View Details:		View Details:	
Visit Vendor:		Visit Vendor:		Visit Vendor:	

Figure 17: Scores of different agents for a given file in Hybrid-Analysis [44].

from the weighted opinions of antiviruses, enhancing the accuracy of the confidence score. The classifier models selected for this experiment are those that achieved the highest performance outcomes as evidenced by Table 6 and Table 7. Specifically, for PDF files, the Gradient Boosting model was identified as the most effective, while for Office documents, the Random Forest model demonstrated superior performance.

To assess the correlation between different confidence score approaches, the Spearman rank-order correlation coefficient method was employed. The outcomes of this analysis are illustrated in Figure 18 and Figure 19, which pertain to the PDF and Office datasets, respectively.

It is worth noting that the absence of a Falcon Dynamic column in the Office dataset stems from the infrequency of CrowdStrike Falcon's inability to evaluate MS Office files. Consequently, the Office dataset was excluded from the analysis, with no representation in the heat map depicted in Figure 19.

This conclusion is supported by the analysis presented in Figure 18 and Figure 19, as it was mentioned the classifier model output does not correlate strongly with other columns that determine the confidence score floating point. Based on the figures presented, the correlation between the classifier model column and all other approaches has been recorded as less than 70% for PDF files and less than 50% for MS Office documents. Conversely, the regression model, when compared across all other approaches, exhibits a correlation value nearing 90% for PDF files and values close to 80% for MS Office documents.

Regarding the correlation between the confidence score derived from this study and other approaches, the strongest correlation is observed with Virus Total, with a correlation value of 99% for PDF files and 98% for MS Office documents. This high correlation is attributed to the regression model's foundation on `JSON` file reports obtained from Virus Total, with the data source originating from Virus Total itself. Nevertheless, it is noteworthy that other approaches also exhibit acceptable correlations with the confidence score. Consequently, it can be considered that the numbers assigned by the classifier model do not reflect the actual confidence score, emphasizing the necessity to employ our approach for predicting the confidence score.

The hybrid analysis platform, which assigns confidence scores, employs various methodologies for this purpose. In comparison to such platforms, the approach presented in this project delivers outputs with high correlation more rapidly than existing methods, as the developed model does not require waiting for outputs from dynamic analyses like Falcon Dynamic or Hybrid Total. Instead, it produces results immediately, aligning with the project's objective of static analysis. Static analysis is characterized by its superior speed compared to dynamic analysis, a characteristic evident in this project. Furthermore, this project stands out for its open-source nature, distinguishing it from existing industrial works such as hybrid analysis. This open-source project makes it easier for other researchers in the malware detection area and other related areas to adopt and use it, serving as a valuable resource. Additionally, this study introduces a novel mechanism for scoring antiviruses, with the open-source nature of the project allowing for the reuse of these scores in subsequent studies.

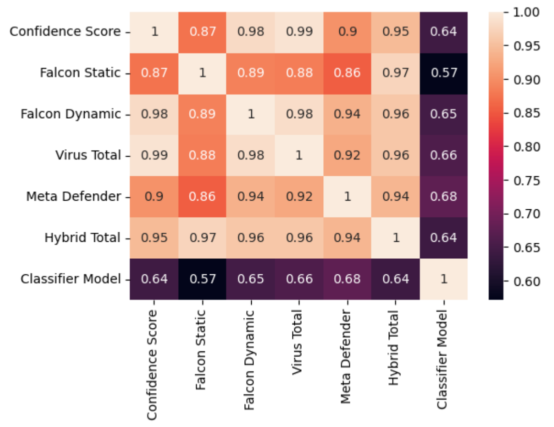


Figure 18: Relationship between various confidence score approaches on the PDF dataset.

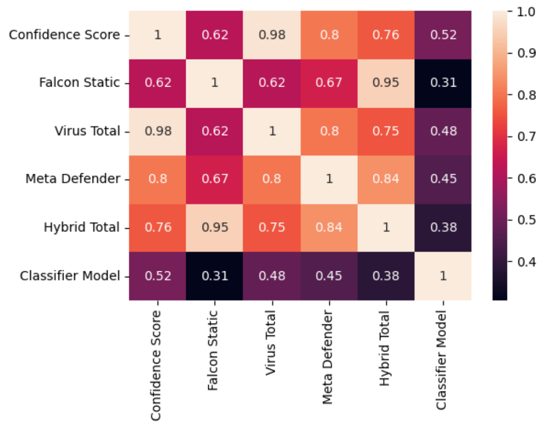


Figure 19: Relationship between various confidence score approaches on the Office dataset.

RQ5: *How effective is our approach in regression models compared with the existing services that report the confidence score?*
Answer to RQ5: *As depicted in Figure 18 and Figure 19, the numerical outputs generated by the regression models employed in this study exhibit a robust correlation with the reliability scores provided by esteemed service providers such as VirusTotal, Meta Defender, Hybrid Analysis, and Falcon. This correlation underscores the validity and reliability of the results obtained, minimizing the risk associated with their application. Specifically, in the heatmap analysis related to PDF files, the most significant correlation with the confidence score is observed with the VirusTotal service provider, achieving a correlation value 0.99. Similarly, the strongest correlation for Office files is also with VirusTotal, albeit slightly lower at 0.98. .*

5.4.6. The effect of confidence score

As demonstrated in the aforementioned experiments, the binary classifiers initially faced challenges in accurately distinguishing between malicious and benign files. However, this section reveals an exciting breakthrough: by leveraging the proposed methodology, we significantly enhanced the confidence and accuracy of the binary classifiers' results through the innovative application of Random Forest regressor.

For a second time, the two datasets used to learn the binary classifiers were used to train and test Random forest regressor for the PDF and MS Office files, respectively. In this iteration, according to Equation (1), the label for each sample was calculated as a decimal value, representing the percentage of the Virus Total antiviruses identifying each sample as malicious. Consequently, the response variables of this regressor indicated the degree of certainty regarding the misclassification of samples as malicious or benign. As evidenced

Table 17: The number of samples in each part correctly recognized by the confidence score.

	Misdiagnosis samples		
Dataset	False Positive	False Negative	Total
PDF	24	5	29
MS Office	20	5	25

in Table 17, this approach enabled us to improve the accuracy of our binary classifiers from 99.3% to 99.74% for PDF files and from 99.4% to 99.78% for Office Word files. In effect, the regressor models were used to compute the percentage of the confidence in the samples which were diagnosed as malicious or benign, incorrectly. The confidence degrees helped us to determine the percentage of the accuracy of the incorrectly labeled samples, as benign and malicious. As a result, as shown in 17, among 33 false positive and 13 false negative samples in 6553 PDF samples, 29 samples were detected correctly. Also, among the 28 false positive and 12 false negative samples in 6657 Ms-Office samples, 25 samples were detected correctly.

RQ6: *How does incorporating confidence scores enhance the detection accuracy of non-executable, complex malicious files?*
Answer to RQ6: *As shown in Table 17, by using the confidence score, we were able to correctly identify 29 cases from the false set detected by the PDF binary classifier and 25 cases from the false set detected by the Office binary classifier. Considering the accuracy formula and adding the correct detection rate to the correct detection set by the mentioned binary models, we were able to reach a detection accuracy of 99.78% for the office dataset and 99.74% for the PDF dataset. Therefore, the use of the confidence score could significantly enhance the detection accuracy of these types of malware. .*

6. Threat to Validity

A significant drawback identified in comparable studies within this domain pertained to the focus on labeling, highlighting the potential resolution through the introduction of confidence scores. However, this approach also introduces challenges. Given that the analyzed files predominantly spanned the years 2022 and 2023, the evaluations from those years were leveraged to assign weights to antivirus software such as `AV-TEST` and AV-Comparative. Consequently, models developed based on these assessments might not accurately address malware emerging in subsequent years. Thus, it becomes imperative to recalibrate antivirus scores annually based on current reports.

Furthermore, potential risks associated with constructing learning models encompass data set imbalance. Despite an unequal distribution of benign and malicious samples in our study, the f-score and ROC curve analyses indicated that the outcome was unaffected by this disparity. Nonetheless, if future evaluations fail to meet these standards or reveal the impact of data imbalance, stratified sampling would be necessary. Additionally, the labeling strategy employed here, relying on VirusTotal's findings, treats any sample deemed malicious by at least one antivirus as inherently harmful. While this conservative stance facilitates cautious learning, it contrasts with the aggressive scrutiny typically required in malware analysis, where a suspicious eye toward all samples is advisable. Moreover, expert opinions might diverge significantly, suggesting alternative labels that could influence model development differently. Indeed, discrepancies between human-assigned labels and those provided by platforms like VirusTotal underscore the variability inherent in labeling, potentially leading to slight deviations in model robustness compared to the approach taken in this study.

Moreover, the hyperparameter optimization conducted in this research involved setting a particular range of values for each hyperparameter and selecting a specific number for k-fold cross-validation to identify optimal settings. Expanding this range and experimenting with varying k values could potentially enhance model performance and improve the accuracy of detecting malicious files.

7. Conclusion and Future Works

In this study, an endeavor was undertaken to analyze PDF and Office files simultaneously. This project aimed to enhance and expand upon previous works by amassing a substantial collection of files in PDF and various MS Office formats. Features were extracted from these datasets statically, with an effort to include the selection of features to extract all common and essential ones in PDF and Office formats. Model training yielded improved outcomes in detecting malicious PDF and MS Office files, demonstrating the study's success in enhancing file detection capabilities. Generally, by applying the mentioned processes and the extraction of features from a dataset encompassing 38,576 PDF files, the study achieved an accuracy of 99.3% with the gradient boosting model. This accomplishment was achieved through the execution of the specified procedure on a dataset comprising 52,557 MS Office files, wherein the highest accuracy of 99.4% was attained utilizing the random forest model.

However, the study's contributions extend beyond the improvement of file detection. A novel concept, assigning the confidence score to complex file types, which is not seen in any of the previous works, was introduced in this project. In organizational antivirus architectures, numerous malicious files, including zero-day threats, pose a challenge to analysts, necessitating swift evaluation to prevent their proliferation. The confidence score aids in categorizing malicious files based on their likelihood, enabling more effective investigation and response strategies. As previously noted, this contribution addresses the limitations found in prior studies that relied on binary (0 and 1) labeling for learning. Essentially, it minimizes the uncertainty associated with label assignments, facilitating more informed decision-making based on the outcomes achieved.

To this end, by applying related algorithms to the prepared dataset, the study achieved the lowest Mean Squared Error (MSE) of 0.0006 for PDF files utilizing the Random Forest model. Conversely, by applying this procedure to the MS Office dataset, the Random Forest algorithm achieved an MSE of 0.003. Furthermore, by comparing the confidence, it was demonstrated that this study achieved a correlation coefficient of 0.99 to the scores assigned by Hybrid-Analysis in the context of confidence scores for Virus Total, specifically for malicious PDF files. Moreover, a correlation coefficient of 0.98 was identified for MS Office files. Our experiments show when using both the binary class and the confidence score, the detection accuracy increased to 99.74% for PDF and 99.77% for Office.

Looking ahead, our future work focuses on evaluating the severity of file damage to determine appropriate antivirus responses, categorize and cluster PDF and Office malicious files to identify malicious behavior patterns and enhance file classification processes to support efficient signature generation. These steps are important for identifying and mitigating malicious files.

Declaration of competing interest

The authors affirm that they possess no conflicts of interest or personal connections that could potentially sway the outcomes of the research presented in this paper.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used "phind.com" to correct the sentences and increase the readability of the text. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

CRediT authorship contribution statement

Rasoul Rezvani-Jalal: Conceptualization, Methodology, Software, Investigation, Data Curation, Visualization, Writing – Original Draft, Writing – Review & Editing. Morteza Zakeri: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Data Curation, Writing – Review & Editing, Project administration. Saeed Parsa: Supervision, Methodology, Resources, Writing – Review & Editing. Amin Hasan-Zarei: Software, Investigation, Validation, Visualization.

References

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

[2] V. Koutsokostas, N. Lykousas, T. Apostolopoulos, G. Orazi, A. Ghosal, F. Casino, M. Conti, C. Patsakis, Invoice# 31415 attached: Automated analysis of malicious microsoft office documents, Computers & Security 114 (2022) 102582.

[3] Kaspersky, Threats statistics in 2023, [Online]. Available: https://www.kaspersky.com/about/press-releases/2023_rising-threats-cybercriminals-unleash-411000-malicious-files-daily-in-2023 ([Accessed: 2023-12-04]).

[4] Avira, Threats statistics in 2023, [Online]. Available: https://www.avira.com/en/blog/malware-threat-report-q3-2020-statistics-and-trends ([Accessed: 2020-11-10]).

[5] Virus Total ([Accessed: 2023]). [link].
URL https://www.virustotal.com/gui/home/upload

[6] P. Singh, S. Tapaswi, S. Gupta, Malware detection in pdf and office documents: A survey, Information Security Journal: A Global Perspective 29 (3) (2020) 134–153.

[7] N. Šrndic, 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.

[8] J. Gu, R. Kong, H. Sun, H. Zhuang, F. Pan, Z. Lin, A novel detection technique based on benign samples and one-class algorithm for malicious pdf documents containing javascript, in: International Conference on Computer Application and Information Security (ICCAIS 2021), Vol. 12260, SPIE, 2022, pp. 599–607.

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

[10] Q. Abu Al-Haija, A. Odeh, H. Qattous, Pdf malware detection based on optimizable decision trees, Electronics 11 (19) (2022) 3142.

[11] Issakhani M, CIC-Evasive-PDFMal2022 dataset, [Online]. Available: https://www.unb.ca/cic/datasets/pdfmal-2022.html ([Accessed: 2022-04-12]).

[12] B. Sohail, et al., Macro based malware detection system, Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12 (3) (2021) 5776–5787.

[13] M. Mimura, Using sparse composite document vectors to classify vba macros, in: Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, December 15–18, 2019, Proceedings 13, Springer, 2019, pp. 714–720.

[14] S. Kim, S. Hong, J. Oh, H. Lee, Obfuscated vba macro detection using machine learning, in: 2018 48th annual ieee/ifip international conference on dependable systems and networks (dsn), IEEE, 2018, pp. 490–501.

[15] V. Ravi, S. Gururaj, H. Vedamurthy, M. Nirmala, Analysing corpus of office documents for macro-based attacks using machine learning, Global Transitions Proceedings 3 (1) (2022) 20–24.

[16] F. Casino, N. Totosis, T. Apostolopoulos, N. Lykousas, C. Patsakis, Analysis and correlation of visual evidence in campaigns of malicious office documents, Digital Threats: Research and Practice 4 (2) (2023) 1–19.

[17] N. Nissim, A. Cohen, Y. Elovici, Aldocx: detection of unknown malicious microsoft office documents using designated active learning methods based on new structural feature extraction methodology, IEEE Transactions on Information Forensics and Security 12 (3) (2016) 631–646.

[18] Malwarebazaar ([Accessed: 2024]). [link].
URL https://bazaar.abuse.ch/

[19] M. Zakeri Nasrabadi, S. Parsa, A. Kalaee, Format-aware learn&fuzz: deep test data generation for efficient fuzzing, Neural Computing and Applications 33 (5) (2021) 1497–1513.

[20] Y. Li, Y. Wang, Y. Wang, L. Ke, Y.-a. Tan, A feature-vector generative adversarial network for evading pdf malware classifiers, Information Sciences 523 (2020) 38–48.

[21] D. Liu, H. Wang, 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.

[22] I. Corona, D. Maiorca, D. Ariu, 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.

[23] Y.-S. Jeong, J. Woo, A. R. Kang, Malware detection on byte streams of pdf files using convolutional neural networks, Security and Communication Networks 2019 (1) (2019) 8485365.

[24] C. Carmony, X. Hu, H. Yin, A. V. Bhaskar, M. Zhang, Extract me if you can: Abusing pdf parsers in malware detectors., in: NDSS, 2016.

[25] Z. Tzermias, G. Sykiotakis, M. Polychronakis, 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.

[26] R. Brandis, L. Steller, Threat modelling adobe pdf, DSTO Defence Science and Technology Origanisation (2012).

[27] E. Filiol, A. Blonce, L. Frayssignes, Portable document format (pdf) security analysis and malware threats, J. Comput. Virol 3 (2) (2007) 75–86.

[28] J. S. Cross, M. A. Munson, Deep pdf parsing to extract features for detecting embedded malware, Sandia National Labs, Albuquerque, New Mexico, Unlimited Release SAND2011-7982 (2011).

[29] D. Stevens, Malicious pdf documents explained, IEEE Security & Privacy 9 (1) (2011) 80–82.

[30] A. Natekin, A. Knoll, Gradient boosting machines, a tutorial, Frontiers in neurorobotics 7 (2013) 21.

[31] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, N. Mastorakis, Multilayer perceptron and neural networks, WSEAS Transactions on Circuits and Systems 8 (7) (2009) 579–588.

[32] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[33] O. Kramer, K-nearest neighbors. dimensionality reduction with unsupervised nearest neighbors. 13–23 (2013).

[34] S. Suthaharan, Machine learning models and algorithms for big data classification, Integr. Ser. Inf. Syst 36 (2016) 1–12.

[35] C.-Y. J. Peng, K. L. Lee, G. M. Ingersoll, An introduction to logistic regression analysis and reporting, The journal of educational research 96 (1) (2002) 3–14.

[36] T. Hastie, S. Rosset, J. Zhu, H. Zou, Multi-class adaboost, Statistics and its Interface 2 (3) (2009) 349–360.

[37] B. Charbuty, A. Abdulazeez, Classification based on decision tree algorithm for machine learning, Journal of Applied Science and Technology Trends 2 (01) (2021) 20–28.

[38] AV-TEST ([Accessed: 2024]). [link].
URL `https://www.av-test.org/en/`

[39] AV-Comparative ([Accessed: 2024]). [link].
URL `https://www.av-comparatives.org/`

[40] Zenodo ([Accessed: 2024]). [link].
URL `https://zenodo.org/`

[41] G. Canbek, S. Sagiroglu, T. T. Temizel, N. Baykal, Binary classification performance measures/metrics: A comprehensive visualized roadmap to gain new insights, in: 2017 International Conference on Computer Science and Engineering (UBMK), IEEE, 2017, pp. 821–826.

[42] R. Rodríguez-Pérez, J. Bajorath, Interpretation of machine learning models using shapley values: application to compound potency and multi-target activity predictions, Journal of computer-aided molecular design 34 (10) (2020) 1013–1026.

[43] Contagio, Contagio dataset, [Online]. Available: https://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html ([Accessed: 2013]).

[44] Hybrid Analysis ([Accessed: 2023]). [link].
URL https://www.hybrid-analysis.com/