PDF Malware Detection Using Ensemble Learning

by

G.M. Sakhawat Hossain

Roll No: 19MCSE005P

A thesis submitted for the partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering(CSE)

CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Chattogram-4349, Bangladesh

September 10, 2024

CERTIFICATION

The thesis titled **"PDF Malware Detection Using Ensemble Learning"** submitted by **G.M. Sakhawat Hossain**, Roll No **19MCSE005P**, Session **2019-2020** has been accepted as satisfactory in partial fulfillment of the requirement for the degree of MASTER OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING on

BOARD OF EXAMINER

1. _____ Dr. Kaushik Deb Professor Department of Computer Science & Engineering Chittagong University of Engineering & Technology Chattogram- 4349, Bangladesh.	Chairman (Supervisor)
2. _____ Dr. Abu Hasnat Mohammad Ashfak Habib Professor & Head Department of Computer Science & Engineering Chittagong University of Engineering & Technology Chattogram- 4349, Bangladesh.	Member (Ex-Officio)
3. _____ Dr. Muhammad Ibrahim Khan Professor Department of Computer Science & Engineering Chittagong University of Engineering & Technology Chattogram- 4349, Bangladesh.	Member (Internal)
4. _____ Dr. Asaduzzaman Professor Department of Computer Science & Engineering Chittagong University of Engineering & Technology Chattogram- 4349, Bangladesh.	Member (Internal)
5. _____ Dr. Md. Nazrul Islam Mondal Professor Department of Computer Science and Engineering Rajshahi University of Engineering & Technology Rajshahi- 6204, Bangladesh	Member (External)

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the materials used, references to the literature, and the work of others have been referred to. This thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Author: G.M. Sakhawat Hossain .....
(signature)

Date:

It is my genuine gratefulness and warmest regard that

I dedicate this work to my beloved

Father and Mother

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Terms and Abbreviations

PDF	Portable Document Format
Maldoc	Malicious Document
Malicecontent	Malicious Content
AdaBoost	Adaptive Boosting
DNN	Deep Neural Network
GBC	Gradient Boosting
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
ROC	Reciever Operating Characteristic

Acknowledgement

I am grateful for being able to convey my appreciation to those whose unwavering assistance and support enabled me to successfully complete my thesis on PDF malware detection using ensemble learning. I am very pleased to express my sincere respect and profound gratitude to my supervisor, Prof. Dr. Kaushik Deb, for his intellectual direction, instruction, helpful feedback, valuable recommendations, supervision, encouragement, tireless support in all stages of the research work, and in the preparation of the report manuscript. Without his pedagogic supervision, sincere assistance, unwavering focus, and constructive critique, this report would not have been completed. Also, I express immense gratitude to Prof. Dr. Iqbal H. Sarker for his exceptional guidance, instruction, constructive feedback, unconditional support, and invaluable insights. I am deeply appreciative of the esteemed examination board members' invaluable recommendations, constructive comments, and heartfelt sentiments.

I am immensely appreciative of the faculty members and personnel of the Department of Computer Science and Engineering at CUET for their assistance. I would like to express my gratitude to my friends and several students of CUET for their invaluable aid in making this effort possible.

I want to extend my heartfelt respect and genuine gratitude to my parents for their invaluable collaboration, insightful recommendations, selflessness, and warm encouragement throughout my professional journey.

Finally, all glory to the Almighty since He is the one who gives me the strength and courage to complete the task and reach the objective.

Abstract

Since Portable Document Format (PDF) is one of the most common file types, scammers put damaging code into people's PDF files to get into their computers. Standard solutions and methods for identifying PDF malware are often insufficient to stop it completely. This is because PDF malware is very flexible and does not rely on a single set of traits. This work mainly aims to identify PDF malware quickly and effectively so that the problems can be addressed. To achieve the objective, initially various benchmark datasets were combined to generate an extensive dataset consisting of 15958 PDF samples, which considers the many types of behaviours exhibited by the samples, including non-malevolent, fraudulent, and deceptive behaviours. Three widely recognized PDF analysis tools (PDF-PARSER, PDFINFO, and PDFiD) were employed to extract notable attributes from the PDF samples in the recently compiled merged dataset. Additionally, several types of derivations of traits were developed that have been shown through experimentation to help diagnose PDF malware. A technique was devised for constructing a highly effective and comprehensible set of characteristics by conducting a thorough empirical analysis of the retrieved and deduced features. Various standard machine learning models were examined which showed that the Random Forest classifier when using the final feature set, achieved an increase in accuracy of approximately 2%. In addition, the model's explainability was showcased through the creation of a decision tree that produces rules that can be easily understood by humans. Finally, a comparative analysis was conducted with prior research that helped to highlight several significant findings.

Keywords: PDF Malware, Machine Learning, explainable AI, Data Analytics, Cybersecurity, Decision Rule, Human Interpretation.

Chapter 1

Introduction

In the modern era of technology, most of our activities are performed online, thus rendering it crucial to safeguard our data, information, and applications from various cyber attackers who constantly create new malicious programs and attacks to disrupt the system [1]. Although security measures have been continuously enhanced, PDF files continue to be a popular method for adversaries to disseminate malware and carry out their attacks [2]. Malware is a malicious program that is primarily designed to harm computer resources. Likewise, PDF malware refers to a PDF that is infected with malicious code. PDF malware can cause various harmful actions, such as establishing hidden doors, stealing passwords, deploying spyware, compromising web browsers, leaking data, engaging in manipulating others, and perpetrating scams. Identifying malicious PDFs is one of the major challenges in the contemporary world. Attackers create many types of malware in this format, and its characteristics are constantly evolving. Malware can be identified primarily through signature-based detection and behavior-based detection. The distinctive characteristics of the underlying object are utilized to create an individual hallmark in the technique based on signatures. The method efficiently identifies the presence of a digital signature by examining the object. However, by employing machine intelligence and other methodologies, the behaviour-based approach can partially identify and comprehend unknown and intricate malware, despite the intricacy of the procedure. Furthermore, machine learning models can be interpreted or explained to discover underlying causes for identifying PDF malware. This process refers to the explainability analysis or model interpretation. In other words, explainability pertains to the insights and reasoning a model provides to make its operation apparent or simple to comprehend [3].

The basic framework of a PDF document consists of a header, body, xref (cross reference) table, and trailer, as depicted in Figure 1.1. The header of the PDF specifies the version that will be utilized in parser format. The body of a PDF document contains images, file-specific metadata, typefaces, and text blocks. It also identifies the substance of the document [4]. The contents of a PDF can be classified into four categories: booleans, streams, strings, and numbers [5]. The PDF file contains a cross-reference table that

provides information about the byte offset or position of each item in the file. This table allows for fast random access to specific objects, making it easier to explore and retrieve content from the document. A PDF reader or parser can navigate and get various elements within the file by utilizing the trailer, which provides essential details such as the root object, the size of the PDF, information about metadata, details of encryption, and the PDF file's unique identifier. PDF malware is typically generated by embedding malevolent content or programs inside the core components of a PDF file.

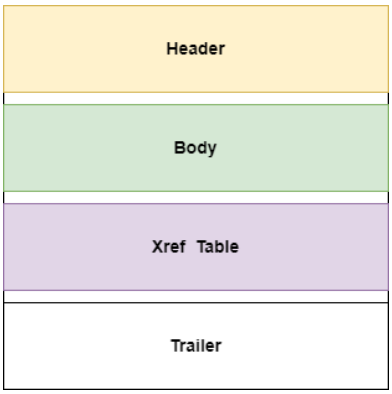

Figure 1.1. A sample PDF's structure.

1.1 Objective

The primary goal of this study is to detect PDF malware from web sources with interpretable machine-learning modeling. To accomplish the goal, the following research objective is set:

- **Research Objective**: To develop an explainable approach for detecting PDF malware by extracting and selecting an effective feature set.

1.2 Motivation

As machine learning (ML) grows in prominence for the detection of malware, there has been a tendency to seek adversarial instances which may decrease its efficacy. Criminals have taken advantage of the ubiquity and flexibility of PDF files to create multiple assaults against reader software, causing significant harm to many individuals. PDF virus is currently providing a serious danger to cybersecurity and is likely to do so in the future as well. According to SonicWall [6], a private network security firm, over 47,000 new PDF-related threats were found in 2020, with 73,000 of those assaults found in March 2019, alone. Furthermore, according to the findings of [7], 66% of the malware attacks

in recent years were transmitted through PDF files. Figure 1.2 depicts the cyberattacks carried out by the fraudsters during the last year. It was observed that, using PDF files, 22% of attacks were executed by attaching malicious PDFs to victims' emails. Because of the massive number of assaults that have recently been carried out using PDF, it is critical to address such threats by building an efficient and comprehensible mechanism. Therefore, the key concern of this work was to prevent such danger by creating an effective feature set, designing an approach for identifying PDF malware, and explaining the model's capacity by presenting decision rules for malicious PDF identification.

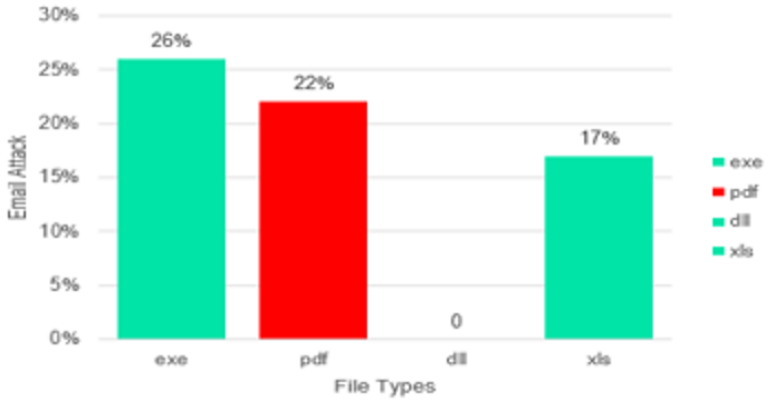

Figure 1.2. Cyberattacks based on various file types throughout the year 2023 [8].

1.3 Application

This study is critical in many industries since it improves security and ensures that documents are handled correctly. In cybersecurity, it may aid by protecting enterprises from dangerous assaults encoded in PDF documents, which are frequently exploited as a means for malware and phishing delivery. Malicious PDF detection in document management systems protects sensitive data by stopping contaminated files from spreading throughout a company's database, ensuring secure and trustworthy file storage. For financial companies that constantly exchange files such as agreements, declarations, and reports, PDF malware detection guarantees that these files stay safe, preserving critical financial data from unwanted access or alteration. likewise, government and law enforcement agencies depend largely on protected file exchange. Identifying PDF malware in these areas is crucial for combating illicit activity, breaches of information, and interference with legal or sensitive information, which supports national security and the confidentiality of legal systems.

1.4 Challenges

PDF malware can be examined by applying a static, dynamic, or hybrid technique, as stated by the source cited [9]. The static approach examines malware without enacting the

software it contains, while the dynamic strategy examines malware through running its code[10], [11]. Static evaluation is vulnerable whenever sophisticated techniques are employed to conceal malicious operation patterns. Given the current cybersecurity landscape, relying exclusively on static examination is generally insufficient as a determined attacker would conceal and encrypt their code, rendering it typically undetectable through static assessment [4]. Contrarily, dynamic approaches exhibit more resilience to coding deceit, making them a superior protection against sophisticated bugs [12]. Dynamic evaluation is frequently characterized by its sluggishness and complexity, while static inspection is typically known for its speed. Combining the two ways leads to hybrid evaluation, which can be more efficient in countering sophisticated malware compared to using each method individually. However, this strategy requires more time and involves an additional difficult analytic method.

Existing techniques for identifying malware often rely on selecting feature sets based on a manual examination of malicious PDF files and are influenced by the knowledge and skills of an expert. Selected aspects may be unique to deceptive files, attracting attackers to potentially manipulate the appearance and characteristics of a malicious file, while avoiding detection. The incidences of reverse mimicry and mimicry have worsened because program designers rarely provide detailed information about the procedures they use to ensure the integrity and resistance to the dangers of their programs. Furthermore, the data that developers can access, including both benign and malicious samples, datasets, vulnerabilities, payloads, and attack vectors used in them, significantly limits their work. These circumstances lead to the solutions being outdated far sooner than anticipated by the conscientious creators.

Machine learning programs have reached a level of advancement where they are capable of safeguarding systems against threats and assisting cybersecurity experts in their experiments by identifying potentially harmful PDF files [13]. Nevertheless, elusive tactics have become proficient in subverting threat document analyzers. Many machine-learning-based detection technologies are vulnerable because they may incorrectly identify well-designed evasive scenarios [14], [15].

Several assessment or detection techniques have been developed to identify particular occurrences, but the prompt danger presented by elusive attacks has not yet been reduced. Furthermore, it had been noticed that most of the work for malicious PDF detection was based on a fixed set of features, and limited consideration was found for PDF content-related features. Additionally, there is very little consideration of the combined use of

evasive samples together with pure benign and malicious PDF samples, which might make the detection mechanism more robust. Also, a lack of combined use of various benchmarked datasets was observed throughout the studies. Most importantly, the lack of explainability analysis of the machine learning model developed to tackle PDF malware was encountered. This involves determining which features are essential for the model's decision-making, understanding how these specific features contribute to the final decision, examining how the proposed model utilizes these features to create decision rules for distinguishing between malicious and benign samples, and ultimately, explaining how these rules can be interpreted in a way that is easily understandable to humans, thus facilitating comprehension of the decision-making process employed by our approach for identifying PDF malware [1], [4], [16].

Creating enhancements to the process of feature engineering and developing effective classifiers for dangerous PDFs by incorporating their adversarial behaviors is a difficult but essential task that holds great potential for making a substantial contribution to the field. Addressing the aforementioned challenges is crucial to mitigating the current vulnerabilities regarding PDF malware detection. Therefore, the key concern was enhancing the identification method for PDF malware by implementing three key steps. Firstly, leveraging the combined use of comprehensive datasets that include evasive attributes of suspicious PDFs and a collection of clean and harmful PDF samples. Secondly, extracting the distinctive features from these PDF samples. Lastly, combining the most significant features to create a powerful feature set that can be inputted into a classifier results in higher accuracy in identifying PDF malware. The key attributes recognized for detecting PDF malware were thoroughly examined and analyzed, as was the proposed classifier's effectiveness.

1.5 Research Questions

To address the challenges observed in PDF malware detection, we developed the following research questions:

- **Research question 01**: How can we extract and select an effective feature set for PDF malware detection?
- **Research question 02**: How can we develop an approach to provide explainability analysis with rule extraction for detecting PDF malware?

1.6 Contributions

By mitigating the challenges as well as addressing the research questions identified in PDF malware detection, this study provides the following contributions:

- Several standardized datasets were combined to create an extensive dataset of 15,958 PDF samples. This dataset includes 7666 malicious PDFs, 7500 clean PDFs, and 792 evasive PDFs, taking into account the legitimate, fraudulent, and elusive characteristics of the PDF samples leveraging three widely used PDF analysis tools, namely PDFINFO [17], PDFiD [18], and PDF-PARSER [19].
- A technique for constructing an easily understandable feature set was devised by considering the features' attributes and their importance scores.
- A framework for identifying fraudulent PDF files was developed, and the performance of various machine-learning classifiers was evaluated to determine their effectiveness in different scenarios.
- A decision tree that creates decision rules for human interpretation to showcase the model's explainability regarding malicious PDF detection was highlighted.
- A comprehensive array of empirical analyses was conducted and juxtaposed the findings with those of prior investigations regarding PDF malware detection. Additionally, certain significant findings from the study were emphasized.

1.7 Organization of the thesis

The subsequent chapters of the study are structured in the following manner: Chapter 2 offers a comprehensive and well-structured summary of the latest research in the same academic area. Chapter 3 outlines the prescribed methodology for detecting malware in PDF files. Chapter 4 discusses the implementation setup employed to conduct this work. Chapter 5 provides an assessment of our discoveries, together with the data obtained from the experiments. Also, offers a comprehensive analysis while highlighting a few notable observations. Ultimately, concluding statements are provided in Chapter 6.

Chapter 2

Literature Review

PDFs have become increasingly popular for distributing harmful materials and malware. To address the significant and consequential increase in hazardous PDF advancements, several impactful works were conducted to develop efficient methods for identifying and classifying malware and other hazardous documents [20]. The methods developed in recent years vary significantly in their level of generality and simplicity to their specificity and complexity. Some methods aim to identify discrepancies by thoroughly examining the entire file [21]. Another technique involves searching a target file for similarities to common patterns observed in malicious PDF files [22]–[26]. Other suites of tools focused on acquiring, assessing, and discovering attack techniques, such as identifying JavaScript-based exploits [27]–[33]. The majority of these methods rely substantially on machine learning techniques, such as Random Forests, one- and two-class Support Vector Machines, and decision trees.

The work described in [34] aims to build a method for identifying a set of traits obtained from existing instruments, as well as creating a fresh set of characteristics to enhance the recognition of malicious PDF documents and extend the effectiveness of presently available detection and evaluation techniques. The significance of the generated characteristics was evaluated by employing a wrapped function that utilized three essential supervised learning techniques along with a feed-forward deep neural network. Afterwards, a new classifier was built that greatly enhanced the effectiveness of categorization while reducing the period required for training. This was achieved by utilizing attributes that had the most importance. The findings were corroborated using extensive datasets obtained from VirusTotal [35].

The researchers in [36] examined the layout of PDFs and the JavaScript code embedded within them, covering all aspects from start to finish. They created a diverse set of visual and metadata characteristics, such as the byte rate, encoding technique, phrases, object labels, and readable characters in JavaScript. Furthermore, due to the sensitivity of AI computations to even little alterations, it is difficult to create adversarial frameworks whenever

the characteristics differ. In order to mitigate the potential for detrimental assaults while preserving the integrity of frameworks and data attributes, a model for categorization was devised utilizing discovery-type models. They developed an elusive strategy to adopt the proposed perspective. The PDF was summarized in [37], and modern cyberattacks on PDF malware were conducted utilizing dependable schemes of attack derived from nature. An illustration was provided on utilizing programming abilities to do an in-depth evaluation of a PDF document in order to detect indications of embedded malware. They examined many innovative AI-driven technologies designed to detect PDF malware. These tools can aid in computational scientific assessments and identify suspicious documents prior to publishing a comprehensive statistical assessment. The individuals examined the limitations imposed by the PDF format, as well as a few outstanding problems, particularly focusing on how these weaknesses could potentially be exploited to divert accurate inspections. Ultimately, they provided guidance on enhancing the resilience of those constructions against assaults and outlined a potential evaluation.

The researchers in [38] devised a means of identification that relied on behavioural discrepancies across several platforms utilizing a software engineering concept, as PDF documents exhibit uniform behaviour on different devices. Conversely, an illicit file will exhibit varying behaviour based on its operation platform. The research conducted in [39] highlighted the use of malware embedded in PDF documents as a prominent illustration of modern intrusions. The researchers initiated the procedure by systematically categorizing the many methods of producing PDF viruses. They employed a validated antagonistic AI system to thwart PDF malware detection that depends on machine learning. This approach, for example, enabled the identification of pre-existing vulnerabilities in PDF malware catchers focused on learning, as well as the detection of new potential vulnerabilities that could jeopardize these systems, together with the probability of implementing preventative measures.

The researchers in [40] presented a novel method for identifying data issues in an ensemble learner. The ensemble classifier's anticipated outcome was incorrect when a sufficient number of votes from different classifiers conflicted throughout the detection process. The ensemble classifier consensual assessment technique was used to identify different types of system evasions without requiring supplementary extrinsic ground truth. The authors conducted experiments to evaluate the proposed method leveraging PDFrate, a tool for detecting malware in PDF files. The results demonstrated that numerous assertions may be inferred by employing an enhanced ensemble classifier that considers the complete dataset of the network.

The researchers in [26] illustrated how to analyze the worst-case performance of a malware classifier based on specific intensity characteristics. Furthermore, it was found that developing models incorporating legitimately validated effective characteristics could increase the cost of preventing unrestricted intruders by bypassing basic onslaught escape strategies. The researchers proposed a different method of measuring distance based on PDF files' hierarchical structure. They also found two distinct categories of prominent characteristics: deletions and subtree insertion.

The authors introduced Lux0R, also known as "Lux 0n discriminant References," in their study [33]. Lux0R is an innovative and versatile method for identifying fraudulent JavaScript files. The suggested approach relied on explaining the code for JavaScript employing API information, involving features that an API of JavaScript framework may easily understand, such as objects, variables, operations, features, strategies, and expressions. The suggested strategy utilized machine learning to identify and segregate potentially dangerous content within a specific segment from API references. This technique was then applied to detect JavaScript infection. The author's main objective in this dissertation was to detect possibly hazardous JavaScript programs in PDF documents. The contributors of [41] discovered the flaws in existing PDF feature extraction programs by assessing them and examining the structure of the fake files. Afterwards, the authors created FEPDF (feature extractor-PDF), an advanced feature extractor that could recognize traits that regular techniques for extraction would overlook and accurately capture data about the components of the PDF. To examine the latest protection systems and sequence extractors, the authors generated multiple new malicious PDFs as specimens. The findings suggest that some current antivirus apps failed to detect the newly hazardous PDFs. Yet, FEPDF successfully extracted the crucial elements for enhanced categorization of hazardous PDFs.

The study [42] proposed a blended identification approach to monitor the execution behaviour of JavaScript programs and identify obfuscation-related characteristics. These traits involve covering particular keywords using ASCII hexadecimal when multiple enlargement adjustments are applied, as well as the presence of illegal objects. The study described in reference [43] focused on the differences in developing hazardous and clean documents. The researchers utilized methods for acquiring the feature set by using the document architecture or structural path. According to the authors, a tree was created by considering the document structure to identify hazardous and clean files by examining certain pathways. Nevertheless, in order to address the current dangers presented by PDF malware and overcome the difficulties given by the elusive nature of PDFs, it becomes imperative to develop a powerful classifier that utilizes a comprehensive set of features capable of explaining the various behaviours exhibited by PDFs. This study involved

combining benchmarked datasets to develop a comprehensive dataset that encompasses the attributes of both benign and malicious PDF files, as well as a brief overview of the evasive tactics employed by PDFs. In addition, three popular PDF assessment programs were leveraged to pull out valuable features from the PDFs, which allowed the identification of a clarified feature set. Also, discovered a highly efficient machine learning classifier that utilizes a freshly created feature set to accurately identify PDF malware with enhanced precision. Ultimately, a comprehensive elucidation of the classifier's performance was offered by delineating a decision tree constructed from one of the classifier's estimators and extracting a few pivotal decision rules for efficiently detecting PDF malware. The following chapter presents a comprehensive discussion of the methodology employed in this research.

Chapter 3

Proposed Methodology

PDF documents are widely utilized globally. Nevertheless, cybercriminals can exploit PDF documents, which are typically considered harmless, to add safety hazards through illicit code, similar to the way they're capable of with Jpeg content, dot-com content, and Bitcoin [4]. Consequently, PDF threat emerges, necessitating methods for distinguishing between harmful and harmless files. This part presents the suggested identification method for examining and classifying PDF documents as either harmless or malicious. The figure labelled as Figure 3.1 illustrates the comprehensive schematic structure of the suggested method employed in this study.

In the recommended methodology, the process began by collecting an overall of 29,901 unprocessed PDF files from the sources: [44], [45], and [35]. These documents were initially obtained via the VirusTotal and Contagio Data Dump. The PDF files are categorized into Safe, Fraudulent, and adversarial (Evasive) groups based on their preassigned labels, as stated in sources [44], [46]. Subsequently, a total of 15958 PDF files were selected, which were categorized as safe, fraudulent, and evasive out of the initial pool of 29901 unprocessed files. This dataset was then utilized to conduct the investigations. Three modern PDF evaluation programs, namely PDFINFO [17], PDFiD [18], and PDF-PARSER [19], were employed to obtain three distinct standard feature sets F_1, F_2, and F_3 based on the actual PDF files in our experimental dataset. Furthermore, seven additional features were obtained by closely analyzing the attributes of PDF examples from the feature set F_1 in addition to the existing collection of features. The model selection procedure comprised the conventional feature sets F_1, F_2, and F_3, as well as a set of standard machine learning classification algorithms. The objective of the model selection step was to choose the most efficient model among the standard detectors utilized in this study, by evaluating their efficacy with each feature set.

The retrieved derived features were combined with the standard feature set F_1, F_2, and F_3 to create the derived feature set F_1', F_2', and F_3', respectively. Our objective was to

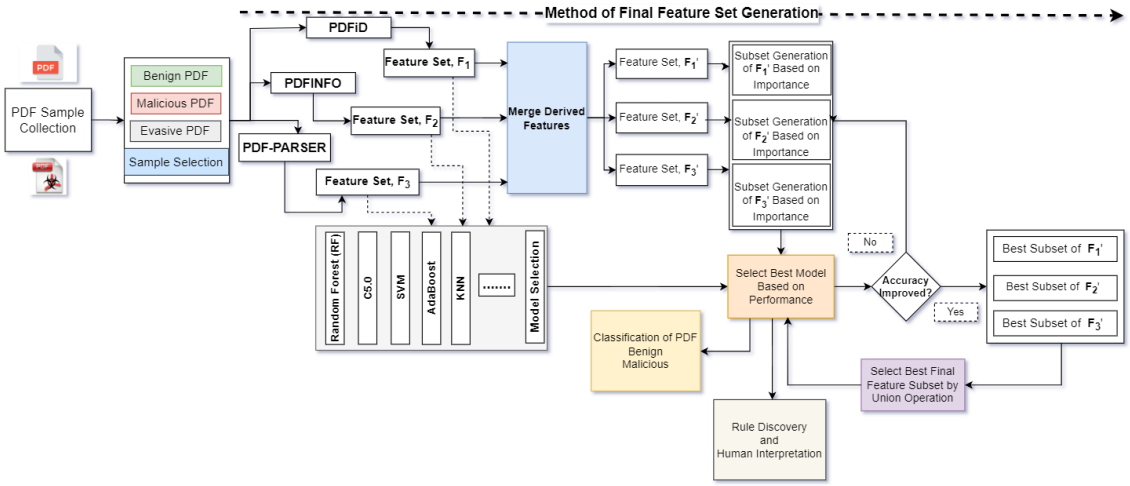

Figure 3.1. Proposed architecture for malicious PDF detection.

create subsets of features from F_1', F_2', and F_3' by considering their relevance. Then, the most effective ones were identified using these subsets in the best-performing model. Ultimately, a union operation was performed on the three most optimal subsets obtained from F_1', F_2', and F_3' in order to create the ultimate feature set utilized for identifying fraudulent PDF files. Our proposed model's effectiveness was measured by employing a range of performance indicators, including f1-measure, recall, precision, and accuracy. In addition, the influence of the ultimate collection of features and its contribution to the classification tasks was emphasized further by capitalizing on the capabilities of the top-performing model, an elucidation to enhance its comprehensibility for humans. This involves extracting significant decision rules that contribute to the classification process. The specific components of our suggested methodology for detecting fraudulent PDF files are outlined in the subsequent subsections.

3.1 Dataset Development

The current datasets may not encompass the whole spectrum of detrimental PDFs. Creating a new dataset may incorporate a wider variety of samples, which allows us to capture the methods and strategies used by adversaries while creating their attacks. The field of cybersecurity is in a perpetual state of flux, with attackers consistently developing novel methods to avoid detection. The utilization of an additional dataset permits the documentation of distinct circumstances that might have been omitted from previous collections. The goal is to create a comprehensive dataset that includes clean and fraudulent PDF samples, as well as a few adversarial PDFs that exhibit features in contrast to their assigned class. This will aid in the development of an effective malicious PDF classifier.

Table 3.1. Gathered PDF documents considered in this study.

Source	Malicious	Benign
VirusTotal	20000	-
CIC-Evasive -PDFMal2022	392 (Evasive)	400 (Evasive)
Contagio	-	9109
Total	20392	9509

3.1.1 PDF Sample Collection

In order to conduct the tests, a substantial corpus of unprocessed PDF files was collected from [44], comprising a total of 29,901 PDFs. The PDF files originate from two reputable websites, namely VirusTotal and Contagio Data Dump. From the Contagio Data Dump, a total of 9109 PDF files were gathered that were classified as benign. Additionally, 20000 PDF files from VirusTotal were acquired that were identified as dangerous. According to the reference [44], a total of 792 PDF files were obtained that exhibit evasive behaviour. The distribution of these files was malignant evasive-392 and clean evasive-400, respectively. Table 3.1 presents the arrangement of the gathered PDF documents, together with their origins and preassigned labels.

3.1.2 PDF Sample Selection

This study's main objective was to create a highly efficient malicious PDF detector by utilizing a well-selected and easily understandable set of features collected from the PDF files. Consequently, the primary focus was on selecting samples to construct our dataset, taking into account a limited number of restrictions. Attackers mostly utilize PDF files, including JavaScript, to create malicious PDF documents. These PDFs were selected based on their demonstration of malicious behaviours. Additionally, PDF files were examined that exhibit contrasting behaviour to malicious ones. For instance, legitimate PDF files typically lack JavaScript-related functionalities that could potentially harm the user's systems. However, it is still feasible for a safe PDF document to be created utilizing harmless JavaScript features. Moreover, using an ML classifier that relies exclusively on these types can result in a model that is very specialized for identifying malicious PDFs, which may not perform well on other types of data. Furthermore, JavaScript is not the sole defining property of malicious PDFs; there are other activating elements as well. Examples of traits that may suggest potential harmful activity in a PDF include OpenAction and Additional Action (AA) [34]. In addition, incorporating excessive variations in both of these categories has the potential to distort the weights and significance of the features, leading to reduced accuracy in the classification. Based on the aforementioned rationale, the next step was to carefully choose a small number of PDF files that display malicious actions but are falsely classified as harmless (benign evasive), as well as PDF files that

Table 3.2. Chosen PDF samples for our functional dataset.

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

exhibit harmless characteristics yet incorrectly categorized as harmful (fraudulent evasive). This study aims to create a functional dataset consisting of 15958 PDF files. The dataset includes 7666 malicious files, 7500 clean (benign) files, 400 evasive benign files, and 392 evasive malicious files. The scope of the experiment is limited to ensure prompt and reliable findings. Table 3.2 displays the distribution of the chosen PDF files for the experimental investigation.

3.2 Extraction of Standard Feature Set

Three tools were leveraged, namely PDFINFO, PDFiD, and PDF-PARSER, to extract features from PDF files. While they have a common purpose, they yield distinct outcomes that can be utilized to generate three distinct sets of features. The tools extract a feature set that can be classified into the following groups:

i Structure Related Features: The term "structural feature" pertains to the specific components employed in the creation of a PDF document. This function establishes an internal connection and demonstrates the hierarchical structure between different elements within PDF documents. In the functional dataset, we investigated several structural traits, such as "obj," "endobj," "%EOF," "start ref," and "trailer."

ii Content-Related Features: Insights on the PDF's textual and visual content can be gleaned from content-related elements. In the operational dataset, it can be seen that the following characteristics are linked to content, such as */Image*, */Font*, */ProcSet*, and so on.

iii Triggering Features: Triggering features are specific characteristics or elements inside the PDF document that have the potential to induce different behaviours or actions, some of which may be detrimental. Adversaries may utilize these functionalities to disseminate malicious software, run scripts, or carry out other nefarious actions. Within the functional dataset, a thorough analysis was done for

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

Figure 3.2. A sample screenshot of PDFiD features.

the triggering attributes such as /OpenAction, /AA (Additional Action), /Launch, /JavaScript, /JS, and others.

iv Metadata Features: The metadata elements of a PDF document offer crucial data about the document, such as its title, creator, production date, and further details. Within our functional dataset, we have analyzed metadata attributes such as "File-size_kb", "/ID", "/CreationDate", "/ModDate", "pages", and so on.

3.2.1 Features of PDFiD

PDFiD is a Python utility [18] that scans PDF documents and identifies specific features and qualities that may signal dangerous behaviour. PDFiD doesn't run any programs within the PDF; rather, it analyzes the PDF's components and structure to reveal its properties. All of the PDF files of the dataset were meticulously evaluated and the PDFiD application was used to extract 22 distinct characteristics, as shown in Figure 3.2. The characteristics retrieved were added to the standard feature collection, labelled as F_1. Below is a concise explanation of the features:

- PDF Header: The inclusion of a PDF header is necessary for programs and apps to recognize and understand PDF documents accurately. The PDF header contains a version code that comes after the identification "%PDF". The string "%PDF-1.3" indicates the PDF file's version which is 1.3. This code sends notifications to programs and PDF readers indicating that the file format is PDF.

- obj: PDF files consist of various components, including text, fonts, photos, forms, and more. The attribute "obj" denotes the initiates of the definition of an object. The feature offers a comprehensive count of the terms that may be detected within the PDF format.
- endobj: The phrase "endobj" denotes the conclusion of the object description. PDFiD identifies the frequency of the endobj term within the PDF layout.
- stream: In PDF documents, a stream object is used to store binary data, such as fonts, pictures, or additional binary content, inside the page. The attribute calculates the total count of stream terms present in the PDF document.
- endstream: This keyword indicates that the stream's binary data component has been finished. Within the framework of PDFiD, this characteristic provides a measure of the number of endstream terms that may be discovered throughout a PDF file.
- xref: The xref (cross reference) table facilitates the maintenance of relationships across the hierarchical items represented in PDF documents. PDFiD determines the quantity of xref tables present within a PDF document.
- trailer: The trailer serves as the ultimate element of the PDF document and includes vital information regarding the byte position to the commencement of the cross-reference (XRef) table. PDFiD's functionality reveals the number of trailers in a PDF file's architecture.
- startxref: The term "startxref" indicates the precise point where the Xref table of the PDF begins. This attribute determines the frequency of occurrences of the "startxref" keyword within a PDF document.
- /Page: This function displays the overall page count of a PDF document.
- /Encrypt: The functionality calculates and displays the count of /Encrypt keywords found in the PDF structure.
- ObjStm: The count of object streams is determined with the /ObjStm command. The ObjStm has the capacity to store more objects, which makes it valuable for concealing items.
- /JS: The count of items that include the /JS keyword, indicating the presence of JavaScript code within those items.
- /JavaScript: This feature highlights the number of entities that encompass JavaScript code, which is a widely used and widespread method of concealing information.
- /AA: This characteristic indicates the count of /AA (Additional Action) keywords detected within a PDF document.
- /OpenAction: The /OpenAction directive indicates an automatic operation when a page or file is viewed. This attribute reveals the number of /OpenAction tags included within the layout of a PDF file.
- /AcroForm: The characteristic represents the count of /AcroForm phrases included in a PDF document. Intruders may take advantage of the Acrobat forms utilized in

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

Figure 3.3. A sample screenshot of PDFINFO features.

PDF documents.

- /JBIG2Decode: This function provides information on the quantity of /JBIG2Decode tags included in the organization of a PDF document. The characteristic indicates whether the PDF employs JBIG2 compression, but it does not directly indicate malicious intent. Further analysis is necessary to determine if the PDF is malicious.
- /RichMedia: The function displays the quantity of /RichMedia keywords present in the PDF format, which serves as an indicator for flash files.
- /Launch: This function returns the count of /Launch keywords included in the PDF document.
- /EmbeddedFIle: This denotes the quantity of "EmbeddedFile" terms that are present within the framework of a PDF.
- /XFA: XFAs, found in certain PDF documents, are XML Form designs that offer programming possibilities that may be maliciously exploited. This functionality generates the count of /XFA terms present within a PDF document.
- /Colors: This feature represents the total count of distinct colors employed in the PDF format.

3.2.2 Features of PDFINFO

One of the most prominent applications for extracting metadata from PDF files is the command-line software known as PDFINFO, which is a component of the Poppler utility package. With the help of the PDFINFO tool, fourteen characteristics were extracted from the operational dataset, as shown in Figure 3.3. The standard feature set, denoted by the letter F_2, was comprised of these fourteen features. In what follows, a description of the characteristics of the feature set F_2 is provided:

- Custom Metadata: This functionality denotes the existence of customized custom metadata within a PDF file. This option returns a binary value, either 'yes' or 'no'.

- Metadata Stream: This functionality indicates whether a PDF file contains a metadata stream, expressed as either 'yes' or 'no'.
- Tagged: This feature indicates whether the PDF file has been marked with tags to enhance accessibility.
- UserProperties: UserProperties refer to additional attributes or information that users might provide in a PDF file for various purposes, such as document management or personal commentary. This attribute provides information on whether the PDF document includes any UserProperties or not.
- Suspects: This lets to know whether a PDF file has any potential flaws or errors in it.
- Form: This feature generates a list of the Form types that are used in the PDF files. From this feature, we have noticed three possible outputs: XFA, AcroForm, or none.
- JavaScript: This feature indicates the presence or absence of JavaScript in the PDF file.
- Pages: This feature allows us to determine the exact number of pages in a PDF document.
- Encrypted: This feature indicates if the PDF file is encrypted or not.
- Page size: This characteristic shows the exact measurements for the pages in the PDF file. We encountered PDFs with a variety of page sizes, including A4, Letter, A3, and other less common sheet formats. If the page's outline is uncommon, we designate it as miscellaneous, also known as *Page size_miscsize*.
- Page rot: This attribute offers details regarding the rotation of pages in the PDF document.
- File size: This characteristic provides the exact size of the PDF document, measured in bytes. However, to ensure the experiment is straightforward, we transformed the file size to kilobytes. Therefore, we have designated this characteristic as *Filesize_kb* for the duration of the investigation.
- Enhanced: This attribute indicates if a PDF file has been optimized, for example, through size compression, or not.
- PDF version: This feature allows users to determine the version of a PDF file.

3.2.3 Features of PDF-PARSER

PDF-PARSER is a Python software as well as a toolkit that uses a command-line interface to parse and analyze the layout of PDF files. This program does not have the capability to create or change PDFs. Its purpose is to examine the inner layout and material of prior PDF documents. Although the toolkit does not offer characteristics directly, a total of 27 characteristics were pulled out from the processed architecture of the PDF, as displayed in Figure 3.4. The features referred to are mostly the phrases that are commonly found in the decoded layout of the PDF. These phrases are regarded as the typical set of features F_3. At first, every one of the PDFs within the functional dataset was systematically examined to

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

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

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


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

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


obj 3 0
 Type:
 Referencing: 5 0 R

  <<
    /JavaScript 5 0 R
  >>


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

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

Figure 3.4. A sample screenshot of PDF-PARSER features.

obtain the processed architectures. Next, a lookup for particular phrases was conveyed, such as attributes, within the processed frameworks. This search assisted in generating the feature set F_3. Next, a concise overview of these features will be provided:

- /Size: Count of /Size terms included in the processed architecture of a PDF. It denotes the overall quantity of items contained within the PDF file.
- /JS: Count of /JS phrases found in PDF's parsed structure.
- /JavaScript: Count of /JavaScript terms in PDF's parsed structure.
- /Producer: Count of /Producer phrases in PDF's parsed structure. This indicates the specific program or software that was used to create the PDF.
- /ProcSet: Count of /ProcSet terms PDF's parsed structure. This specifies the precise set of processes or procedures that must be used when displaying visual contents

or a page within a PDF document. While this term does not explicitly identify the fraudulent nature of a PDF, it has often been found inside the parsed layouts of non-malicious PDF documents.

- startxref: Count of startxref terms in PDF's parsed structure.
- %EOF: Count of %EOF terms in PDF's parsed structure. It serves as an indicator that signifies the conclusion of the PDF file.
- /S: Count of /S keywords in PDF's parsed structure. It specifies the specific category or type of different items or actions, such as links or text remarks.
- /ID: The /ID keyword count refers to the number of occurrences of /ID terms identified in the PDF's parsed structure. It provides the file identification information, essential for confidentiality and safety of the file. This also indicates the hostile activity with the file.
- /CreationDate: Count of /CreationDate terms in PDF's parsed structure.
- xref: Count of xref terms in PDF's parsed structure.
- obj: Count of obj terms in PDF's parsed structure.
- «: Count of « terms in PDF's parsed structure. Within a PDF file, the '«' symbolizes the initiation of a dictionary object.
- »: Count of » terms in PDF's parsed structure. In a PDF file, the '»' symbol denotes the ending of a dictionary object.
- /Font: Count of /Font terms in PDF's parsed structure.
- /ModDate: Count of /ModDate terms in PDF's parsed structure. The PDF file's alteration date and time are indicated using the /ModDate keyword.
- /XObject: Count of /XObject terms in PDF's parsed structure. It is used to specify and encompass additional pictorial content.
- /Widget: Count of /Widget terms in PDF's parsed structure. Widget tags are gadgets that engage users to put input data.
- Comment: Count of Comment terms in PDF's parsed structure.
- /Info: Count of /Info terms in PDF's parsed structure. It refers to the PDF's information dictionary.
- /XML: Count of /XML terms in PDF's parsed structure.
- /FontDescriptor: Count of /FontDescriptor terms in PDF's parsed structure.
- Referencing: Count of Referencing terms in PDF's parsed structure.
- /Length: Count of /Length terms in PDF's parsed structure. It indicates the content stream's precise size associated with a PDF object, measured in bytes.
- /Rect: Count of /Rect terms in PDF's parsed structure.
- /Image: Count of /Image terms in PDF's parsed structure.
- /Action: Count of /Action terms in PDF's parsed structure.

3.3 Derived Features

Derived features can be defined as synthetic features generated through empirical observation from the existing features of the operational dataset. To capture the data insights or patterns or relationships that may not be readily apparent in the existing features alone, derived features can play a crucial role. By closely examining the traits of the PDFs in the conventional feature list F_1, an additional seven traits were discovered. The introduced derivative features are located at the points as follows:

- **Headercorrupt**: This binary attribute takes into account the PDF's version. Whenever the PDFs do not have correct syntax such as %PDF-1.X, where X is a number between 0 and 7, then the value of the feature is assigned as 1; alternatively, it is assigned as 0.
- **Headerlength**: This characteristic considers the total number of characters of the file's name.
- **Small content**: This trait is obtained by meticulously observing the quantity of objects in a file. If a file contains objects that are fewer than or equal to 14, then the attribute is assigned a value of 1; contrary, it is assigned a value of 0. The average score categorized by category (fraudulent and harmless) was tracked regarding the quantity of items found in the documents. The average score for dangerous documents is 14.1, while for the innocuous ones, the average is 85.6. In addition, 6277 harmful files were detected, making up for 77.90% of the total hazardous document samples that satisfy or are below the threshold value of 14. Conversely, we came across 1709 harmless PDFs, accounting for 21.6% of the overall innocuous PDF samples, that additionally meet this exact limitation. Furthermore, we detected 22.10% of PDF files as dangerous and failing to match the requirements. After careful examination, the minimum amount for this binary trait as 14 was established.
- **Stream corrupt**: Whenever the total amount of endstreams and streams are not equal, then the value is 1; alternatively, it is set to 0.
- **Content corrupt**: Whenever the total amount of endobj and obj are not equal, then the value is 1; alternatively, it is set to 0.
- **Malicecontent**: This binary characteristic provides a value of 1 if at least single instances of /AA (Additional Action), /Launch, /JS, /JavaScript, or /OpenAction are observed in two of the aforementioned features and alternatively put to 0. These attributes highlight possible concerns as these are being frequently observed to inject into PDFs and launch malware attacks.
- **Hidden File**: The PDFiD application detects the presence of any concealed file that attackers have put throughout a PDF file. This binary characteristic is assigned a value of 1 if any concealed file is detected from the document, and 0 if no such file is located.

In order to assess the efficacy of the derived traits in detecting malevolent PDFs, those derived characteristics were combined with the standard feature sets F_1, F_2, and F_3 to create the derived feature sets F_1', F_2', and F_3', respectively.

3.4 Selection of Machine Learning Model

The objective of this study was to create a robust data-driven method using machine learning to identify fraudulent PDF files. In order to choose an efficient machine learning model, we begin by initializing the conventional feature set F, which includes the feature sets F_1, F_2, and F_3. For our experimental study, a set of M traditional machine learning algorithms were chosen, which include AdaBoost, Deep Neural Network (DNN), Gradient Boosting (GBC), Random Forest, SVM, J48, C5.0, and KNN. An iterative process was used where the assessment of each feature set in F was performed and every classifier was evaluated in M using 10-fold cross-validation. This allows us to provide classification metrics for each classifier and feature set combination. We evaluated the classification metrics obtained for each feature set and selected the most effective model for detecting PDF malware, depending on the report.

Algorithm 1: Algorithm For Generating Final Feature Set

Input: Derived Feature Sets, $F' = \{F1', F2', F3'\}$
Output: Final Subset

```
foreach f ∈ F' do
    Determine the feature importance score of f;
    Take the importance score to sort the features in f in descending order;
    Take top features from f to generate subsets S = {S_1, S_2, ...};
    Initialize best_subset = ∅;
    foreach s ∈ S do
        Apply subset s to the best ML model for PDF malware detection;
        Analyze classification metrics;
        if subset performance is better than best_subset performance then
            Set best_subset = s;
    Apply a union operation to append best_subset to Final_Subset;
```

3.5 Final Feature Set

Once the derived features were combined with the standard feature sets F_1, F_2, and F_3 to create the derived feature sets F_1', F_2', and F_3' correspondingly, Then, the focus shifted to constructing the final feature set. The rationale for constructing the ultimate feature set is to generate a highly efficient composite feature set by examining all the derived feature sets. Consequently, the significance of the characteristics was assessed and arranged for each set of derived features, and subgroups were created according to the ranking of the

features. Each feature subset was executed by applying the best-performing model to determine its effectiveness. We sequentially cycled through the derived feature sets F_1', F_2', and F_3' to generate subsets by selecting the most important characteristics from each set. Once the iteration for generating subsets and evaluating them using the top-performing model was finished, the model's performance across the subsets of each feature set was analyzed to determine the optimal feature subset. The most optimal subset of features from each derived feature set F_1', F_2', and F_3' were selected and then combined to create the final feature set. The recently constructed feature set was used to perform the final classification tasks for detecting PDF malware. Algorithm 1 outlines the overarching methodology for generating the final feature collection. In the subsequent chapter, we present the implementation details of this study pertaining to the detection of PDF malware.

Chapter 4

Implementation

The purpose of this study was to achieve four main objectives. Firstly, to develop a feature set that is both efficient and enhanced in order to identify PDF malware accurately. Secondly, empirically investigating different baseline machine learning classifiers to determine the most effective one. This classifier will be able to utilize the newly created feature set and improve the accuracy of PDF malware detection. Thirdly, to look at how the attributes in the final feature set help the classifier recognize fraudulent PDF documents. Finally, to derive a small number of crucial decision rules from the classifier that can be readily comprehended and interpreted by humans. These guidelines will aid in detecting possible harmful intent in PDF files.

The experimental setup employed in this study comprised the subsequent hardware and software configurations:

- Processor: Intel(R) Core(TM) i5-10300H 2.50 GHz
- RAM: 16 GB
- GPU: NVIDIA GeForce GTX™ 1650
- OS: Windows 10 Home
- Python: Python 3.9.13
- Scikit-learn: Scikit-learn Version 1.4
- Matplotlib: Matplotlib 3.8.3
- SHAP: SHAP v0.42.1

The following chapter elaborates on each result obtained through our investigation and, where applicable, draws comparisons to other studies of similar scope.

Chapter 5

Experimental Result Analysis and Evaluation

In order to achieve the objectives, the investigations that are based on the following scenarios were conducted:

- *CASE I*: The primary goal in this case is to determine the responses to the accompanying inquiries: 1) How do the typical feature sets F_1, F_2, and F_3 contribute to PDF malware detection? 2) Which machine learning technique is appropriate for detecting malware in PDF files?
- *CASE II*: The primary objective in this situation is to disclose the results of the subsequent inquiries: 1) How do the derived features impact detecting PDF malware? and 2) What do the resulting feature sets F_1', F_2', and F_3' do to help detect PDF malware?
- *CASE III*: In the present scenario, the responses to the following inquiries are looked into: 1) Which attributes are chosen for the final feature set? Furthermore, 2) Does the final set of features enhance the classifier's performance?
- *CASE IV*: In this specific situation, the results of the subsequent inquiries are disclosed: 1) In what way does the feature set comprising F_1, F_2, F_3, as well as derived features denoted by F_1' U F_2' U F_3' contribute to the identification of fraudulent PDF files? 2) Is the optimal feature subset obtained by considering the technique mentioned in *CASE III* different from the final set of features obtained in *CASE III*? 3) Does the optimal feature subset generated in this case have a favourable or unfavourable effect on the classifier's performance? and finally, 4) How well does the classification work while no derived characteristics are used, like when only F_1 + F_2 + F_3 are used?
- *CASE V*: The primary aim in this instance is to clarify how the recently developed final set of features aids the classifier in the detection of PDF malware. In addition, another concern in this stage is to conduct an examination of how the attributes of the final set of features are dispersed throughout the functioning dataset in order to pinpoint several potential avenues that could significantly contribute towards the detection of PDF malware.

Furthermore, the power of the classifier to uncover a handful of crucial decision criteria was harnessed that may be easily comprehended and employed by humans to identify potentially hazardous PDF content.

5.1 Performance Metrics

In this work, to assess the effectiveness of the machine-learning models, the following performance metrics are used:

- Accuracy: Accuracy quantifies the proportion of correctly identified test samples out of the overall number of test samples.

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

  where TP represents True Positive, FP indicates False Positive, TN denotes True Negative, and FN means False Negative.
- Precision: Precision is a quantitative metric that assesses the ratio of accurately predicted positive cases (true positives) to the total number of positive instances predicted, encompassing erroneously labeled positive instances (false positives).

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

- Recall: This can be expressed as True positive rate (TPR) or sensitivity which can be measured by the following:

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

- F1-Score: This presents the balanced average of recall and precision for a particular approach.

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

5.2 CASE I

This phase analyzes the influence of three feature sets, namely F_1, F_2, and F_3, together with conventional machine learning approaches, to identify the most effective ones for identifying malicious PDFs. Table 5.1 presents the results of different conventional machine-learning approaches for detecting PDF maldocs. The evaluation is based on the standard feature set F_1, F_2, and F_3 using 10-fold cross-validation. Evidently, the Random Forest classifier demonstrates superior performance in identifying PDF malware when compared to the baseline classifiers across all standard feature sets. Scikit-Learn, a renowned open-source Python package for machine learning, was leveraged to create the traditional

classification approaches. In order to control for randomization, 100 estimators and a random_state value of 42 were considered while developing the Random Forest classifier. Conversely, The AdaBoost, SVM, C5.0, KNN, and J48 classifiers were constructed using their Scikit-Learn module's default hyperparameters.

In addition, a Deep Neural Network (DNN) model was created using an MLPClassifier. The model consists of a hidden layer with 100 units and a random_state value of 42. The DNN model was run for a total of 100 epochs. The Gradient Boosting classifier (GBC) was presented with 100 estimators and a random_state value of 42. During the implementation of the standard feature set F_1, it is evident that the Random Forest classifier outperforms the other models employed in this study in PDF maldoc identification. An accuracy of 96.82% is observed from the Random Forest, making it the top-performing model.

After analyzing the performance of the other classifiers, the second-highest accuracy of 96.59% is noticed for the C5.0 classifier in identifying malicious PDFs. Nevertheless, the SVM classifier exhibited relatively inferior performance in recognizing the fraudulent PDFs. Conversely, the DNN, KNN, J48, GBC, and AdaBoost models achieved accuracy rates of 96.20%, 95.77%, 96.57%, 96.49%, and 95.92%, correspondingly in detecting malicious PDFs. In a similar vein, it can be found that when employing the standardized feature sets F_2 and F_3, respectively, the Random Forest model outperforms the conventional models with an accuracy of 96.53% and 97.19% for malicious PDF detection. Figures 5.1, 5.2, and 5.3 clearly demonstrate the comparability of the ROC curves for the classifiers used in this work, based on the standard feature sets F_1, F_2, and F_3 respectively.

In all of the usual set of features, the highest AUC score is exhibited for the Random Forest model in comparison to the other classifiers utilized for malicious detection. Upon evaluating the performance of the model, it is determined that feature set F_3 is the most efficacious among the three conventional feature sets. To validate the performance of the Random Forest model, the model was leveraged directly to assess the significance of the standard feature sets. The standard feature sets F_1, F_2, and F_3 are listed in Table 5.2 along with their respective feature importance when using the Random Forest model for PDF malware identification. It can be determined the main characteristics of the Random Forest model that let it perform better than the other baseline classifiers from Table 5.2. startxref, /JS, and JavaScript are discovered as the most crucial features from F_1, while the features Optimized, Metadata Stream, and Filesize_kb, are found as the most crucial features from F_2. Furthermore, this can be determined that the three most significant features from the feature set F_3 are /JS, /JavaScript, and /Producer. Based on the performance of the Random

Table 5.1. A 10-fold cross-validation study of machine learning algorithms on the standard feature sets F_1, F_2, and F_3.

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

Table 5.2. Importance score of standard feature sets F_3, F_2, and F_1.

Feature Set, F3 (PDFPARSER)	**Importance**	**Feature Set, F2 (PDFINFO)**	**Importance**	**Feature Set, F1 (PDFiD)**	**Importance**
/JS	0.17806	Filesize_kb	0.385827	/JS	0.189384
/JavaScript	0.174966	Metadata Stream:	0.14866	startxref	0.133327
/Producer	0.098267	Optimized:	0.132621	/JavaScript	0.128817
/Size	0.079436	Pages:	0.112362	obj	0.092926
startxref	0.053516	JavaScript:	0.071808	xref	0.080607
/ProcSet	0.046	Page size:_miscsize	0.054809	endobj	0.075699
%EOF	0.043771	Tagged:	0.028214	stream	0.052136
xref	0.03762	Page size:_A4	0.0246	trailer	0.047182
/Font	0.028149	Page size:_letter	0.015741	/OpenAction	0.046888
Referencing	0.024605	Form:_none	0.009629	endstream	0.037798
/ID	0.024562	Form:_XFA	0.007045	/XFA	0.037407
/S	0.02106	Custom Metadata:	0.00654	/Page	0.023114
/Rect	0.02038	Page rot:	0.001469	/AcroForm	0.019413
s/Widget	0.020207	Encrypted:	0.000674	/EmbeddedFile	0.016478
	0.016886	UserProperties:	0	/ObjStm	0.011918
/XObject	0.016549	Suspects:	0	/AA	0.002771
/CreationDate	0.016449			/Launch	0.001869
obj	0.016161			/RichMedia	0.001168
	0.015845			/Colors	0.000437
/Image	0.015456			/JBIG2Decode	0.000386
Comment	0.010286			/Encrypt	0.000274
/ModDate	0.009622				
/FontDescriptor	0.008315				
/Length	0.008113				
/Action	0.006537				
/Info	0.005279				
/XML	0.003905				

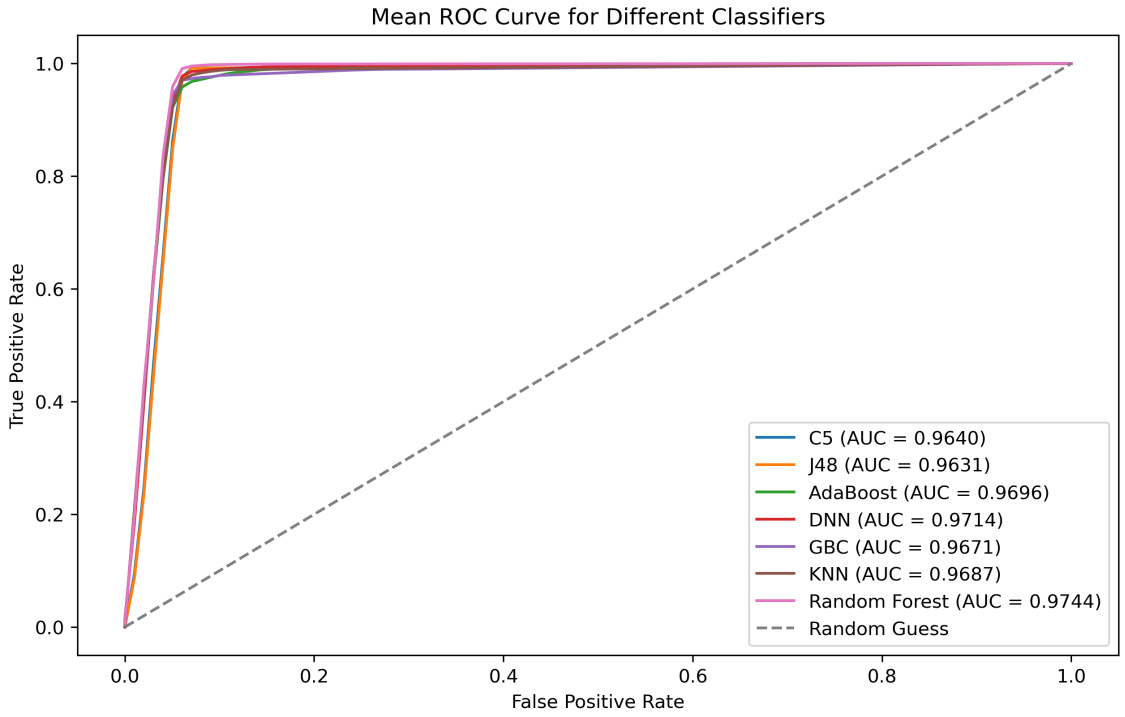

Figure 5.1. Random Forest model ROC curve comparison on the usual feature set F_1.

Forest model in detecting PDF malware, it is considered to be the best model due to its ability to make decisions based on ensemble learning, its effectiveness in high-dimensional feature space, and its observation of the previously mentioned empirical analysis using standard feature sets. Therefore, in order to carry out our intended experiments, we solely use the Random Forest model in future case studies.

5.3 CASE II

This instance demonstrates the effect that the derivative set of traits has on the efficiency of the classifier, in addition to determining the extent to which the resulting characteristics aid in the detection of PDF malware. The results of the Random Forest classifier's evaluation using derived feature sets, as determined by 10-fold cross-validation, are presented in Table 5.3. The results unequivocally indicate that the performance of the model was significantly enhanced by the derived feature set in comparison to the conventional feature sets. A marginal improvement of approximately 2% in the classifier's accuracy is observed while the derived feature set F_1' is employed in lieu of the typical feature set F_1. In a similar fashion, the performance enhancement of the classifier, as exemplified by the other derived sets of features F_2' and F_3', remains consistent. Conversely, the classification report for the derivative set of features F_3' is the most accurate, with the classifier achieving 98.90% precision in identifying PDF malware.

In order to evaluate the relevance of the derived features, the classifier was leveraged to

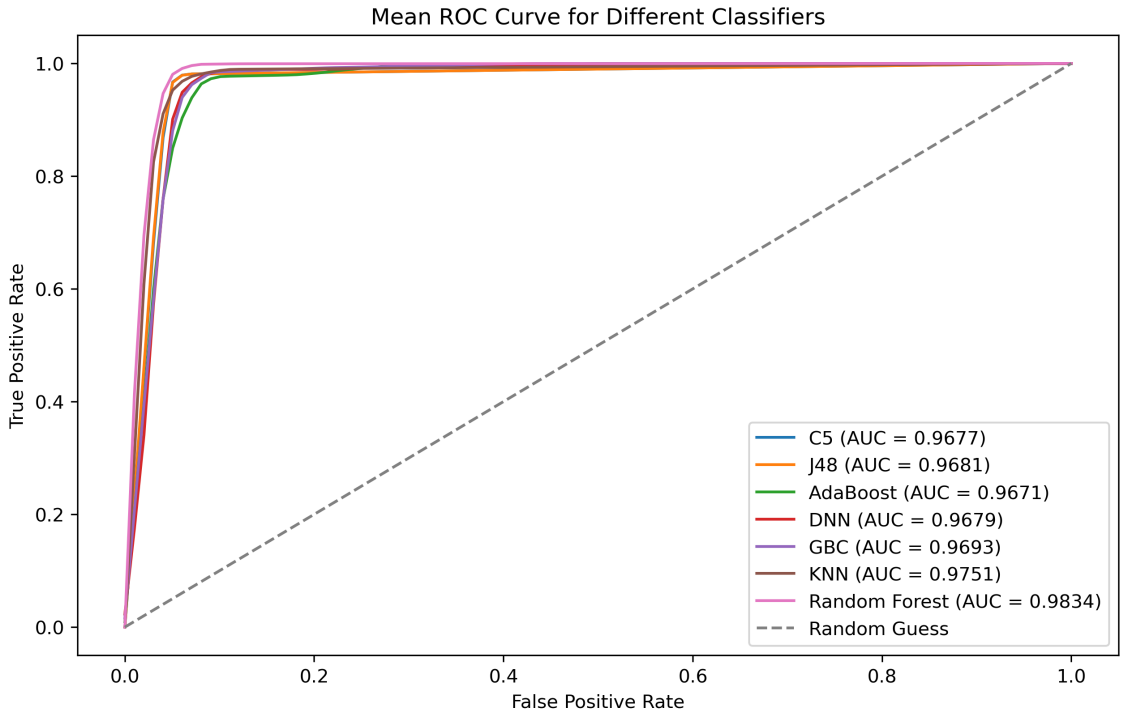

Figure 5.2. Random Forest model ROC curve comparison on the usual feature set F_2.

calculate the importance score of the features for each derived feature set F_1', F_2', and F_3', respectively. The importance score of the derived feature sets is highlighted in Table 5.4, which plainly demonstrates the value of the derived features and their contributions to the detection process. The feature *Headerlength* has the highest contribution among all the generated feature sets, as can be noticed from the findings. Specifically, the attribute *Headerlength* assists 28.19%, 34.15%, and 30.85% respectively, when F_1', F_2', and F_3' are used for classification activities.

Similarly, an additional significant developed feature, *Malicecontent*, ranks among the top three most significant features among those in the derivative set of features F_1' and F_2', contributing 12.6‘% and 17.10%, respectively, to the classification effort. Upon analysis, it can be found that the feature *Malicecontent* is in the top two significant attributes, contributing 9.7% to the identification of PDF malware when the classifier uses F_3'. In addition to the *Malicecontent* and *Headerlength* derived features, it can be rarely seen any significant assistance from other sets of derived features. However, the feature *small content* provides some assistance compared to the other derived features to the classifier.

5.4 CASE III

This phase carefully observes the feature sets F_1', F_2', and F_3' to determine the characteristics chosen for the final set of features. Also, the impact of incorporating the final set of traits

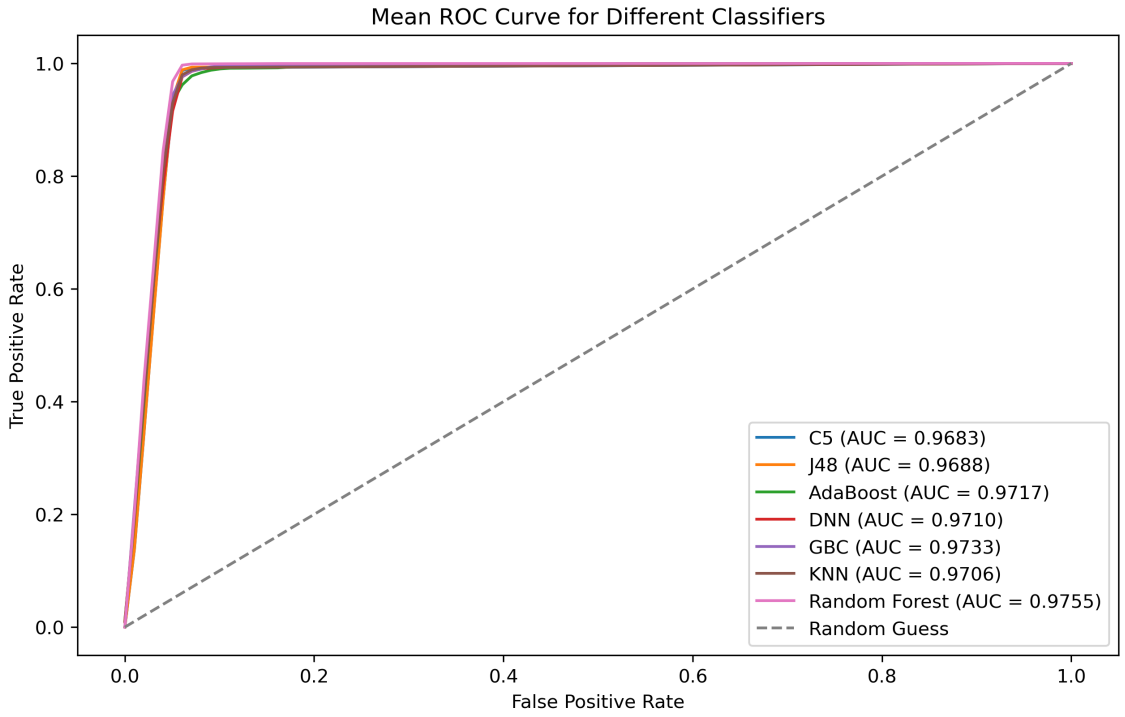

Figure 5.3. Random Forest model ROC curve comparison on the usual feature set F_3.

Table 5.3. A 10-fold cross-validation-based investigation to show Random Forest classifier's performance improvement while using derived feature sets.

Feature Set	F1-Score	Precision	Recall	Accuracy
F1 (Standard)	0.9682	0.9690	0.9682	0.9682
F2(Standard)	0.9653	0.9656	0.9653	0.9653
F3(Standard)	0.9719	0.9727	0.9719	0.9719
F1'(Derived)	0.9868	**0.9868**	**0.9868**	**0.9868**
F2'(Derived)	0.9824	**0.9825**	**0.9824**	**0.9824**
F3'(Derived)	0.9890	**0.9891**	**0.9890**	**0.9890**

on the classifier's effectiveness is investigated. Moreover, the importance of the ultimate collection of features is assessed in detecting malicious PDFs.

So that the best feature groups can be found from the generated sets of features, the procedures outlined in Algorithm 1 were adhered to. The results presented in Table 5.4 are utilized, which displays the estimated importance score of features for each set of derived features, ranked in order of their significance. To build subsets from the sets of derived features F_1', F_2', and F_3', the features that have a feature importance score of at least 1% were only considered. Therefore, the top 15 characteristics from the set F_1' were identified. The initial feature subsets are generated by selecting the top 18 features from F_3' and the top 10 features from F_2' that meet the specified requirement. Subsets are generated by incrementally adding features in a sequential manner, starting from the initial feature and

Table 5.4. Importance score of derived feature sets F_3', F_2', and F_1'.

Feature Set, F3' (PDFPARSER)	**Importance**	**Feature Set, F2' (PDFINFO)**	**Importance**	**Feature Set, F1' (PDFiD)**	**Importance**
Headerlength	0.308559	**Headerlength**	0.341565169	**Headerlength**	0.281992628
Malicecontent	0.097292	Filesize_kb	0.192103387	/JavaScript	0.155315511
/JS	0.093437	**Malicecontent**	0.171037759	**Malicecontent**	0.126014543
/JavaScript	0.082273	Metadata Stream:	0.083803166	/JS	0.113431885
/Producer	0.055951	Optimized:	0.057181587	xref	0.052227619
/Size	0.040859	Pages:	0.0405821	startxref	0.051961463
/CreationDate	0.03431	**small content**	0.035097756	obj	0.034711542
%EOF	0.032303	JavaScript:	0.018807386	endobj	0.029616909
startxref	0.029804	Page size:_miscsize	0.016363782	stream	0.026597245
/ProcSet	0.028089	Tagged:	0.014240093	/OpenAction	0.025362229
/ID	0.025948	Page size:_A4	0.007776304	/XFA	0.022755269
xref	0.014015	Page size:_letter	0.006177369	trailer	0.019853837
/Info	0.013927	Custom Metadata:	0.003728307	endstream	0.016040389
/Font	0.013196	contentcorrupt	0.002811481	/EmbeddedFile	0.014248542
/S	0.012968	Form:_none	0.002611641	/AcroForm	0.010383106
Referencing	0.01296	Form:_XFA	0.002316382	/Page	0.006201786
/XML	0.010976	headercorrupt	0.001100524	**small content**	0.004261149
/Rect	0.010309	HiddenFile	0.000976744	/ObjStm	0.003643089
/ModDate	0.009883	streamcorrupt	0.000798319	/AA	0.001284778
obj	0.009521	Page rot:	0.000714061	contentcorrupt	0.001125052
/XObject	0.009186	Encrypted:	0.000206682	streamcorrupt	0.000936493
	0.008798	UserProperties:	0	/Launch	0.000570549
	0.007742	Suspects:	0	HiddenFile	0.000351583
/Widget	0.006901			/RichMedia	0.000341436
small content	0.006212			/Encrypt	0.000328662
/Image	0.006063			headercorrupt	0.000241423
Comment	0.004843			/Colors	0.000112839
/Length	0.004401			/JBIG2Decode	8.8444E-05
/FontDescriptor	0.003952				
/Action	0.002423				
contentcorrupt	0.001105				
HiddenFile	0.00084				
headercorrupt	0.000505				
streamcorrupt	0.000452				

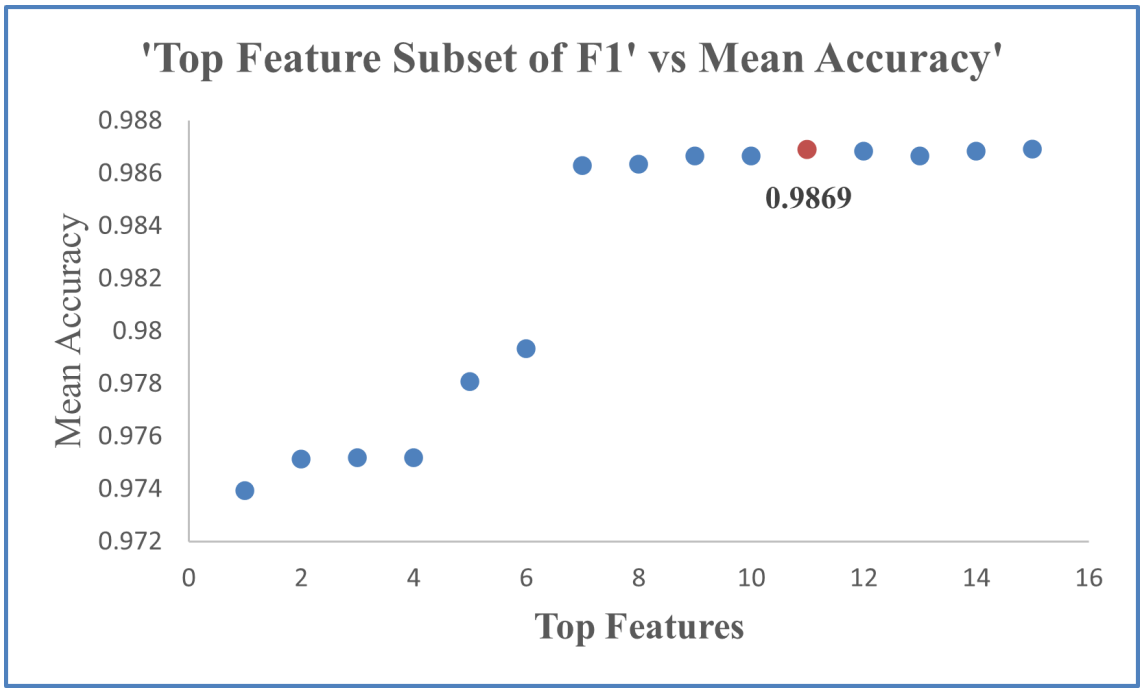

Figure 5.4. Random Forest classifier's mean accuracy vs subset of top features from the derivative set of features F_1'.

continuing until the final feature is evaluated from each set of derived features for feature subset generation.

Consequently, 15, 10 and 18 subsets are constructed from F_1', F_2', and F_3', correspondingly. Additionally, we proceed with the procedure outlined in Algorithm 1 in order to assess the efficacy of every subgroup of features and determine which of them is most suitable for the formation of the final set of features. In Figure5.4, Figure 5.5, and Figure 5.6, the average efficiency is emphasized that was achieved by employing the method described above for every subset of features of F_1'. F_2', and F_3', respectively. The representation in Figure 5.4 shows that the model is most accurate when it uses a particular set of 11 key characteristics from F_1'. The average precision varies slightly among the other subgroups of F_1'. In an identical way, Figure 5.5 shows that the model has the highest score of 97.91% while employing a set of 8 prominent characteristics from F_2'.

The average precision for different subgroups of F_2' varies a lot. Yet, it can be seen in Figure 5.6 that this predictor is most accurate while all 18 significant traits are used for subset construction from F_3'. The best subset of features from F_1' is thus the portion of the set containing the most prominent 11 features, the most effective feature subgroup from F_2' is the collection containing the eight most important features, and the finest subset of features from F_3' is the subgroup containing the best 18 features. The union process is

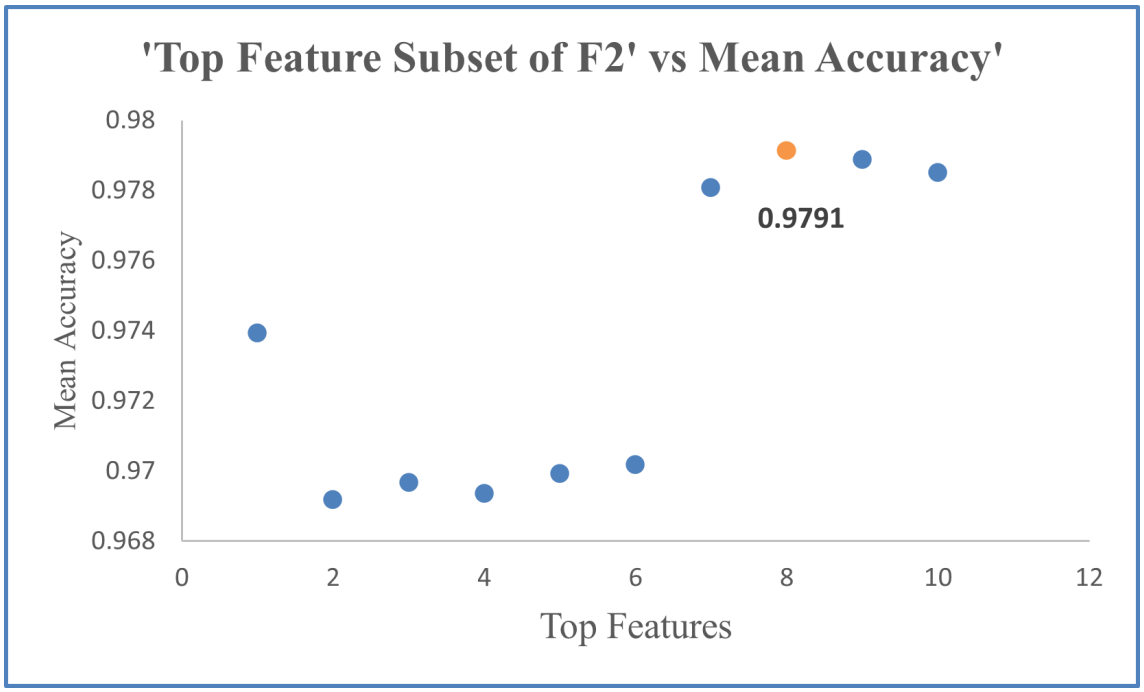

Figure 5.5. Random Forest classifier's mean accuracy vs subset of top features from the derivative set of features F'_2.

undertaken to build the final set of features, taking into account both the uncommonness and the commonality among the freshly found optimal subsets of features. The summary of detected traits, which are eventually taken into consideration for the final set of features, is shown in Table 5.5. Within the final feature set, we find 3 derivative characteristics: *small content*, *Malicecontent*, and *Headerlength*. We only take into account the trait *JavaScript* once (from F'_1), in the list since it appears in each of the three derivative sets of features, F'_1, F'_2, and F'_3.

Since F'_1 and F'_3 have the attributes *xref*, *startxref*, and */JS*, we include these just once (from F'_1) in the list of features of the final set of features. Particularly from F'_1 in the final set of features, we encounter the attributes *stream*, *endobj*,*obj*, */XFA*, and */OpenAction*. The remaining attributes shown in Table 5.5 are exclusively seen from F'_3, with the exception of *MetadataStream*, *Filesize_kb*, *Pages*, and *Optimized*. In order to identify malicious PDFs, we examine how the final set of features affects the proposed model, which is based on 10-fold cross-validation. The classifier's results across the various sets of features employed in the present investigation are displayed in Table 5.7.

While contrasting the final set of features to the usual and derivative set of features, there can be observed a significant improvement in the performance of the model. Using the final set of features, the classifier's greatest efficiency enhancement over the usual set of

Table 5.5. List of selected features of the final set of features.

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

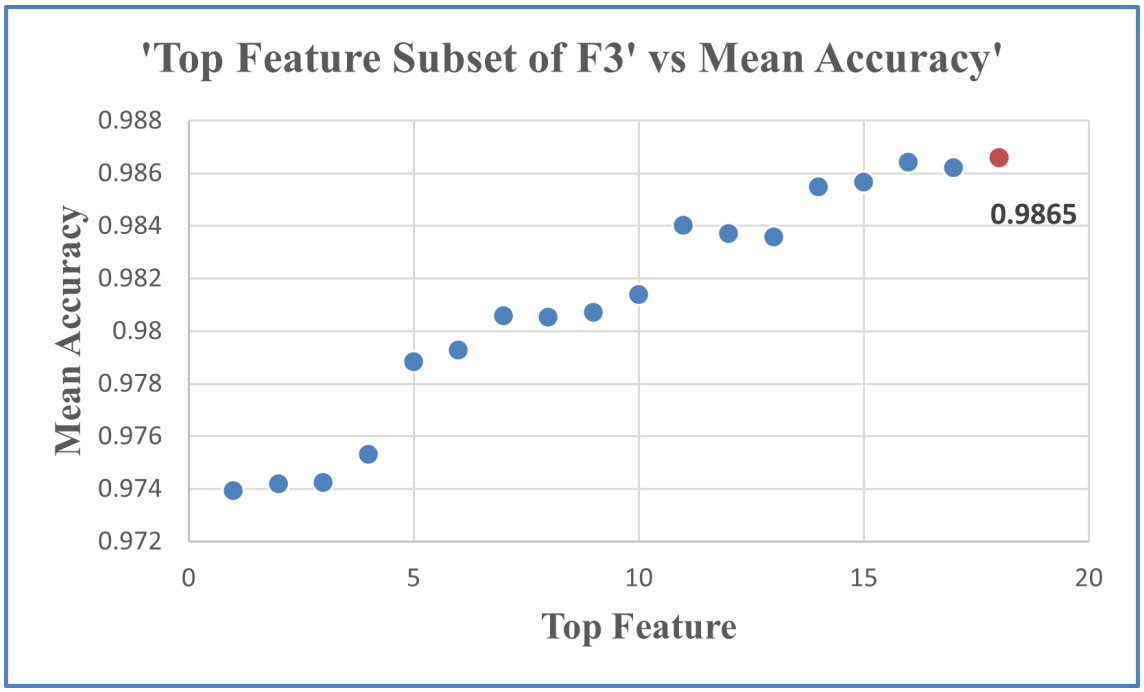

Figure 5.6. Random Forest classifier's mean accuracy vs subset of top features from the derivative set of features F_3'.

features F_2 is 2.71%. In contrast, it can be seen that the classifier's minimal performance increase when employing the final set of features is 0.34% higher than while using the derivative set of feature F_3'. Nevertheless, when it comes to detecting malicious PDFs, the classifier performs noticeably better since it has access to the recently created final set of features.

Figure 5.7 depicts the performance trajectory of the proposed classifier, which was evaluated based on 10-fold cross-validation using the final set of features. Throughout the 6th fold of the 10-fold cross-validation, the model exhibits an efficiency of 99.56%, while on

Table 5.7. A 10-fold cross-validation-based investigation to show Random Forest classifier's performance improvement while using the final set of features for detecting PDF malware.

Feature Set	F1-Score	Precision	Recall	Accuracy
F1 (Standard)	0.9682	0.9690	0.9682	0.9682
F2(Standard)	0.9653	0.9656	0.9653	0.9653
F3(Standard)	0.9719	0.9727	0.9719	0.9719
F1'(Derived)	0.9868	0.9868	0.9868	0.9868
F2'(Derived)	0.9824	0.9825	0.9824	0.9824
F3'(Derived)	0.9890	0.9891	0.9890	0.9890
Final Feature Set	**0.9921**	**0.9924**	**0.9924**	**0.9924**

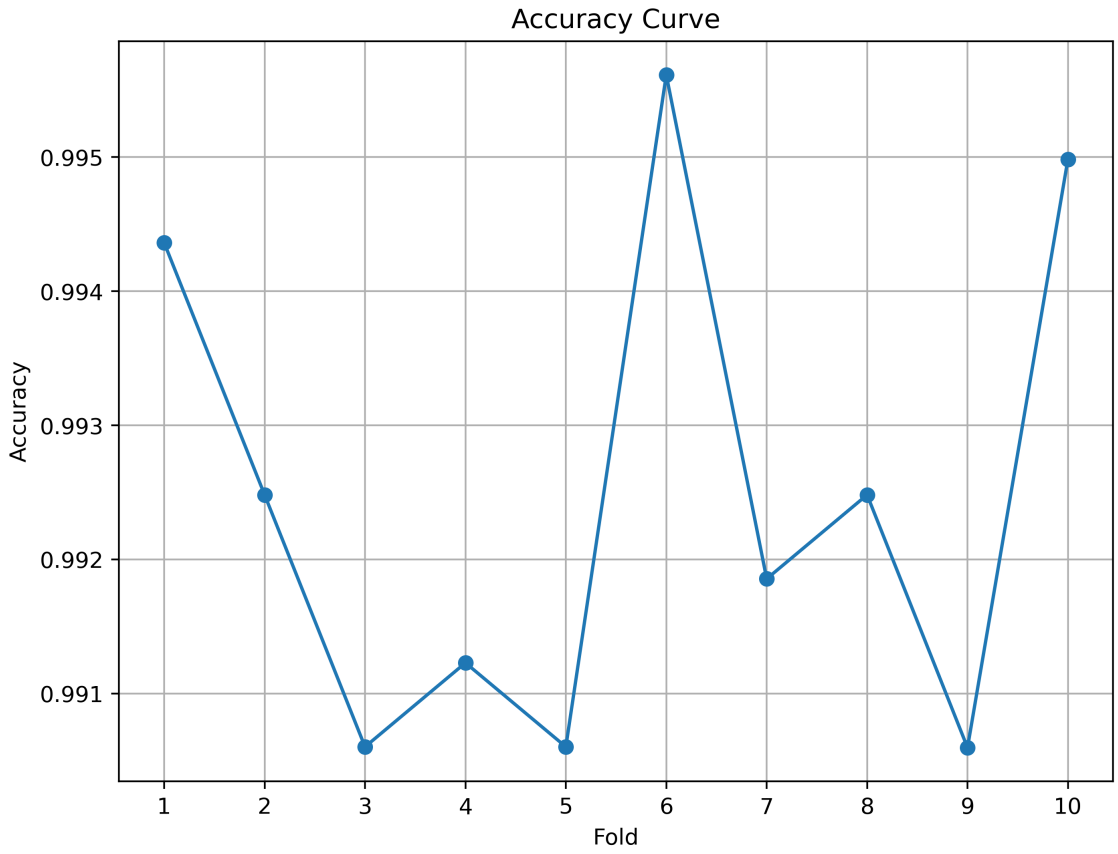

Figure 5.7. Random Forest model's accuracy curve on the final set of features based on 10-fold cross-validation.

the ninth fold, it has the lowest performance of 99.05%. Figure 5.8 shows the model's log loss trajectory at different levels of the 10-fold cross-validation. Throughout the whole process, There is the lowest loss in the initial phase and the largest loss at the third fold. The proposed model's Receiver Operating Characteristics (ROC) curve for the different phases of the validation process on the final set of features is shown in Figure 5.9. It is observed that the proposed model has an area under the curve of 1.00 across the validation process.

5.5 CASE IV

Within this instance, the outcomes of a PDF malware detection implementation utilizing a combined set of features (i.e., F_1 + F_2 + F_3 plus derivative features) are shown. The optimal subgroup of attributes is determined based on the merged feature set using the methodology described in *CASE III*. The discrepancy between this particular subset and the final set of features is then determined. Furthermore, the effect that the most effective set acquired within this instance has on the performance of classification is investigated. Furthermore, the classifier's performance is examined in the absence of the derivative set of features when attempting to detect malicious PDFs.

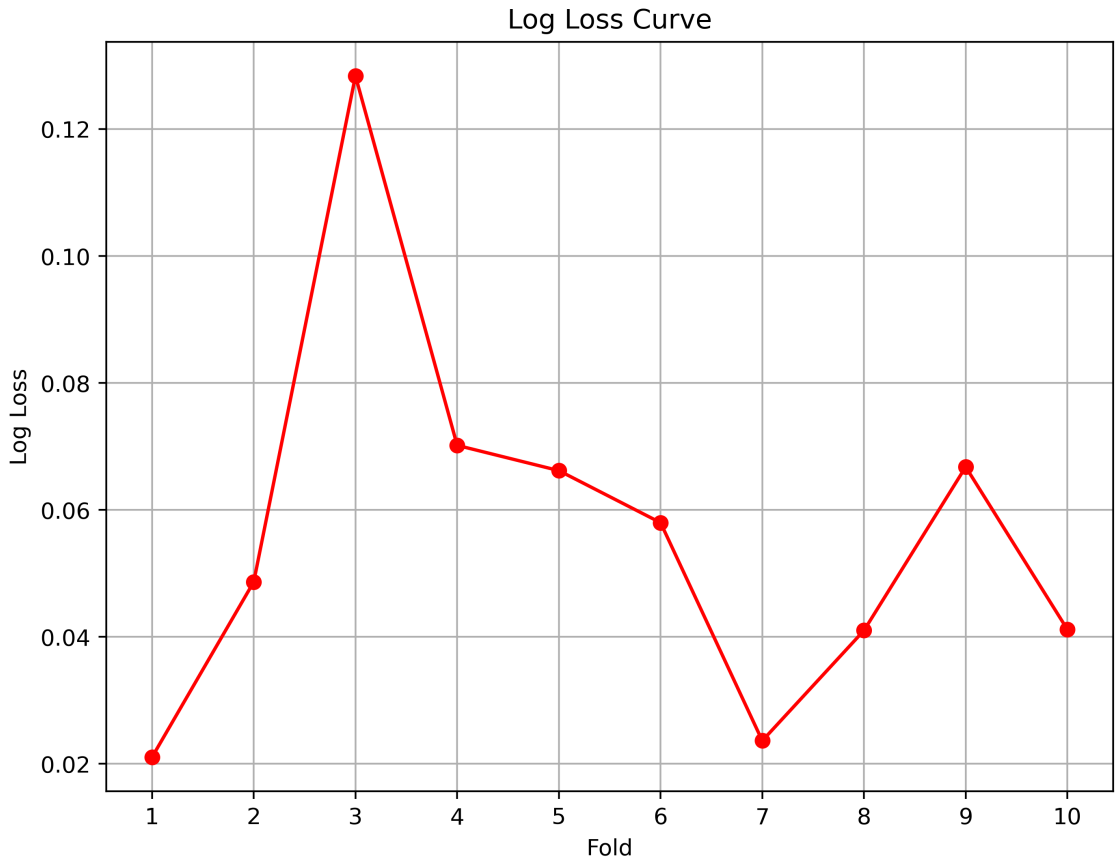

Figure 5.8. Random Forest model's loss curve on the final set of features based on 10-fold cross-validation.

Table 5.8 provides an explicit and obvious representation of the proposed classifier's performance using both the combined set of features with the derivative set of features and the merged set of features without the derivative set of features. Additionally, the table compares the effectiveness of the classifier with the final set of features. When using the combined set of features with the derivative set of features (i.e., F_1 + F_2 + F_3+ derived features) to detect malicious PDFs, it can be found that the model achieved a 99.19% detection rate. However, with no derivative set of features, the model produced a detection rate of 97.28% for the merged set of features, or F_1 + F_2 + F_3. Therefore, it is evident that the derived set of features significantly improves the classifier's performance by a minimum of 1.91 % in the detection of malicious PDFs. On the contrary, this can be observed that the classifier performed far more effectively when the final set of features was applied to detect PDF malware. This is due to the fact that the combined set of features comprises a significantly greater quantity of attributes in comparison to the final set of features. In high-dimensional spaces, where the amount of attributes is greater, the classifier can, at times, identify sequences in the noise instead of authentic patterns, resulting in performance that is marginally less effective. In addition, by removing superfluous characteristics, the classifier's capacity to accurately classify and generalize PDF malware can be enhanced.

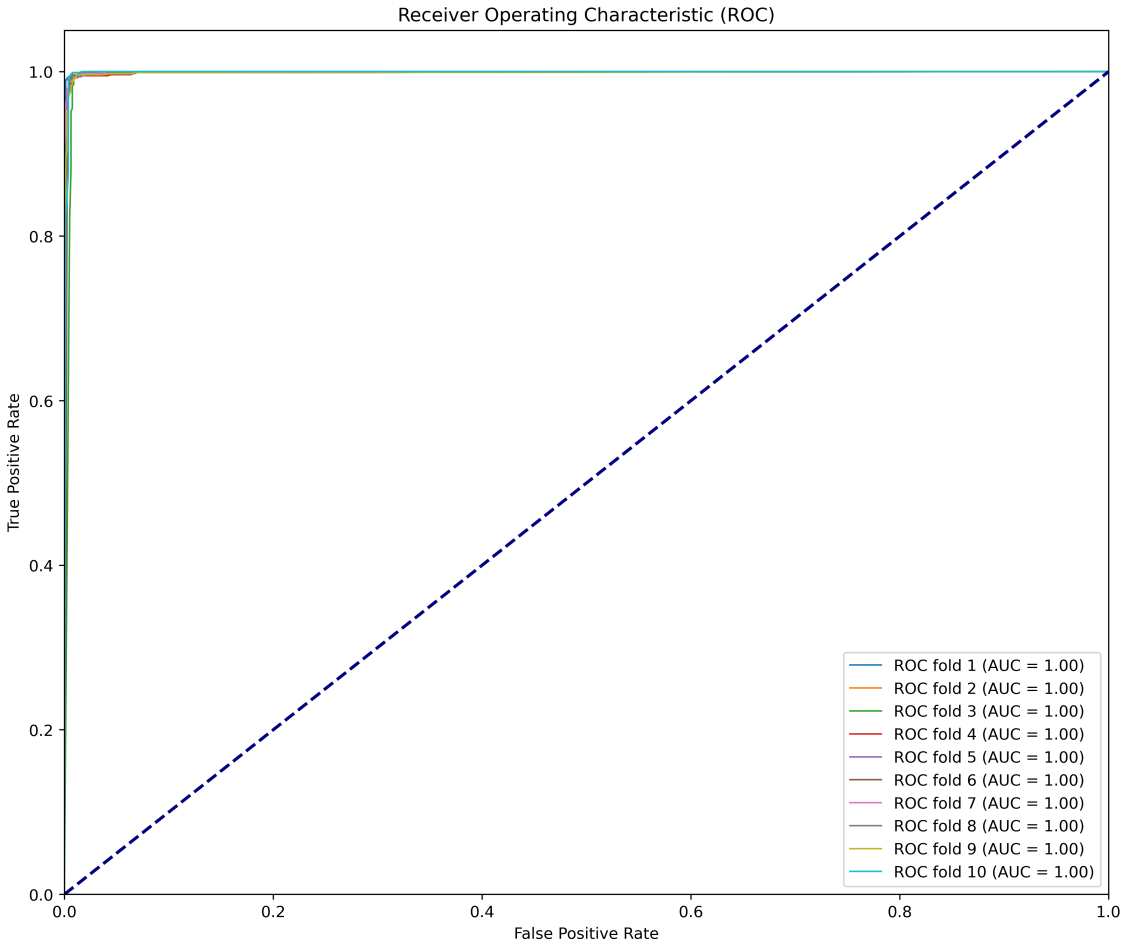

Figure 5.9. Random Forest model's ROC curve on final feature set based on 10-fold cross-validation.

Therefore, the final set of features achieved greater precision compared to the combined set of features by incorporating the most significant characteristics according to their importance.

The significance of the most important traits of the combined set of features is displayed in Table 5.9, which was acquired by leveraging the classifier's strength. In order to build the subset (as specified in *CASE III*), we began by selecting the features from the table that have a feature importance score of at least 1%. Hence, it was only taken into account the top ten attributes from Table 5.9 in order to get the ideal subgroups from the merged set of features.

Next, the method described in *CASE III* was used for these attributes in order to construct 10 subsets. The average accuracy achieved by using the classifier on these subgroups is shown in Figure 5.10. The subgroup made up of the top 10 traits was discovered to have the greatest average efficiency of 98.53%. Hence, it can be determined that the feature subset consisting of the following characteristics (/Info, Headerlength, /Producer, /JavaScript,

Table 5.8. Effect on the performance of the Random Forest classifier for PDF malware detection on both the feature set devoid of derived features (i.e. F1+ F2+ F3) and the combined feature set, as determined by 10-fold cross-validation.

Feature Set	F1-Score	Precision	Recall	Accuracy
F1 (Standard)	0.9682	0.9690	0.9682	0.9682
F2(Standard)	0.9653	0.9656	0.9653	0.9653
F3(Standard)	0.9719	0.9727	0.9719	0.9719
F1'(Derived)	0.9868	0.9868	0.9868	0.9868
F2'(Derived)	0.9824	0.9825	0.9824	0.9824
F3'(Derived)	0.9890	0.9891	0.9890	0.9890
Final Feature Set	**0.9924**	**0.9924**	**0.9924**	**0.9924**
F1+F2+F3+Derived (Combined)	**0.9919**	**0.9919**	**0.9919**	**0.9919**
F1+F2+F3	**0.9728**	**0.9729**	**0.9728**	**0.9728**

Filesize_kb, Malicecontent, /S, startxref, /ProcSet, /ID) is the most optimal subset from the combined set of features. Evidently, there is a clear distinction in both the optimal subset derived from the merged set of features and the final set of features. Nevertheless, it is worth mentioning that the optimal subset obtained in this scenario does not enhance the accuracy of the classifier in comparison to its performance with the final set of features.

Additionally, the CFS (Correlation-based Feature Selection) approach was employed to produce the most suitable subset of features from the merged set of features in order to verify the efficacy of the final set of features. The *CfsSubsetEval* feature selection technique was utilized, employing the Best First Search algorithm, through the widely-used tool *Weka 3* [47]. The approach produces the subsequent traits as output in the best subset: The following terms are included: Headerlength, contentcorrupt, /Encrypt, Malicecontent, /Colours, /Action, Page size:_miscsize, Metadata Stream, Optimized, /Size, and Page size:_A4. In a similar vein, the technique for choosing traits known as "ReliefFAttributeEval" (with Ranker approach) was applied in order to generate the best possible subset from the merged set of features utilizing *Weka 3*. Attributes with a minimum merit score of 1% were taken into account, and the subset of features was developed and evaluated using the methodology described in *CASE III*. Lastly, using this method, it can be determined that which subset is optimal and has the subsequent characteristics: xref, /CreationDate, /Action, /XML, Form:_XFA, %EOF, /ModDate, /AcroForm, /Producer, small content, /Font, /XFA, /FontDescriptor, Form:_none, Custom Metadata, /EmbeddedFile, Tagged, Metadata Stream, Malicecontent, Optimized, Headerlength. These sets of features were utilized with the classifier effectiveness and contrasted the outcomes in order to assess the

Table 5.9. Top feature's importance score of the combined set of features in reverse order.

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

final feature set's strength. It was discovered that the model produced a success rate of 97.59% for the subset of *CfsSubsetEval* and 98.75% for the subset of *ReliefFAttributeEval*. It is worth mentioning that the model achieved its maximum efficiency when employing the final attribute set to identify malicious PDFs.

In general, It was discovered that the model from *CASE I* yielded the best accuracy of 97.19% on the usual set of features F_3 whereas the classifier provided the best detection rate of 98.90% in *CASE II*, utilizing the derived set of features F_3'. The model results in the greatest score of 99.24% using the final set of features from *CASE III*. The classifier delivers the maximum success rate of 99.19% on the combined set of features with the derived set of features from *CASE IV*.

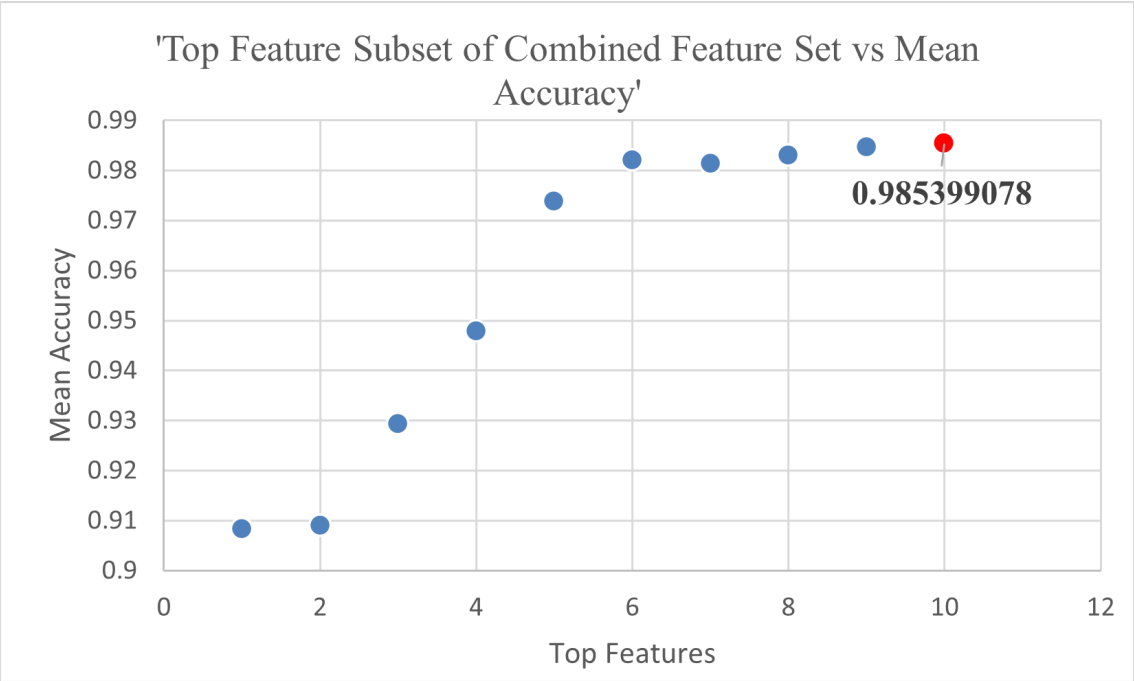

Figure 5.10. Random Forest classifier's mean accuracy vs top subset of features of the combined set of features.

5.6 CASE V

This phase explains how the newly developed final set of features enhances the model's ability to recognise PDF malware. To assess the efficacy of the final set of features, the Random Forest classifier was utilized to estimate the significance of the top features. This process is represented in Figure 5.11. This image highlights the significant characteristics and their respective contributions to the activities of the classification process. The image demonstrates that the properties known as "Headerlength" and "Malicecontent" play a significant role in identifying PDF malware. However, the properties of /JS and /Javascript are equally essential for detecting fraudulent PDFs.

Every attribute within the recently constructed final set of features was scrutinized in order to investigate the characteristics of each group of PDFs with respect to the aforementioned characteristics within our functioning dataset. As shown in Figure 5.12, the mean score of the derivative set of features *Headerlength* for innocuous files is 16.50, accompanied by a typical variance of 12.45. In contrast, the mean score for malevolent files is 42.48, with a typical variance of 7.28. Additionally, the illustration depicts that 75.83% of the safe files contain titles which are similar to or shorter compared to the mean title length of the harmless PDFs. Conversely, 89.16% of the malicious PDFs have a length of the title which is equal to or shorter than their average title length. Nevertheless, this observation clarifies that the average length of titles in the benign PDFs is significantly shorter than that of the harmful ones in our operational sample. This discovery offers a possible clue for detecting

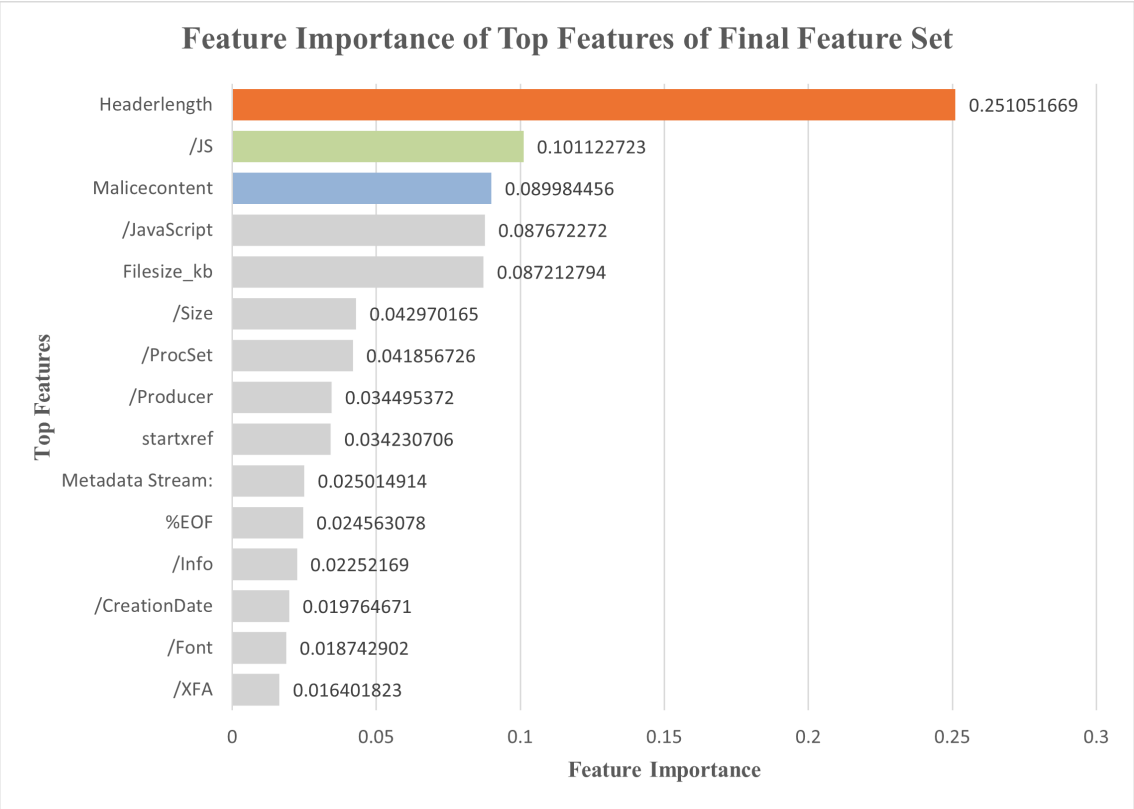

Figure 5.11. Feature importance score of best features from the set of final features.

PDF malware just based on the length of the PDF title. However, it is important to note that this characteristic alone does not definitively indicate the presence of dangerous content within a PDF. In practical situations, a clean PDF document may frequently have a lengthy title.

The distribution of a secondary attribute called *Malicecontent*, which is created by examining the malice attributes in both benign and malicious PDFs, is shown in Figure 5.13. In our pilot investigation, it can be seen that 7246 PDFs with malicious content were detected using this feature, whereas 651 PDFs without malicious content were also classified as such. This illustrates that dangerous PDFs mostly consist of triggering elements in contrast to safe ones, which also indicates a possible approach for detecting malicious PDFs in real-world scenarios. The */JavaScript* attribute, which is commonly utilized by criminals to construct malicious PDFs, is depicted in Figure 5.14 based on a thorough examination of our operational dataset. The study reveals that over 92% of clean PDFs have minimal or no JavaScript characteristics, whereas, among the malicious PDFs in our dataset, roughly one-third possess this feature. The existence of this characteristic in a PDF indicates a high likelihood of malicious behaviour. Just like the */JavaScript* feature, it can be found that the */JS* feature is rarely present in safe files. However, for the malicious files, it can be noticed that this term appears frequently.

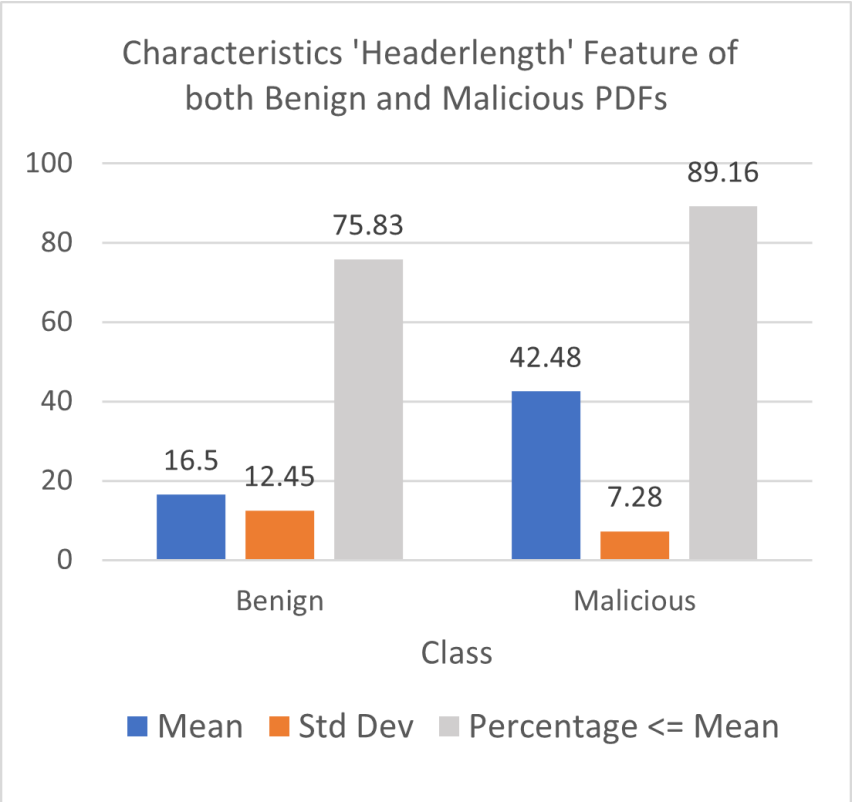

Figure 5.12. Headerlength attribute distribution in the functional dataset for both malicious and benign PDFs.

In Figure 5.15, the characteristics of the *FIlesize_kb* attribute are illustrated, which clearly indicates that the mean value of the safe PDFs is significantly more than that of the hazardous PDFs. A significant variation is observed in the safe PDF file size. Additionally, it is pertinent to highlight that a mere 68.77 percent of the pristine PDFs possess a file size shorter or equal to their mean size. Furthermore, there can be observed a significant level of variability in the file sizes of the malicious PDFs, with approximately 10% of them deviating from the average size. Also, benign PDF's mean value for the keyword */Size* is 2.27 whereas for the fraudulent is 1.00. This highlights the criminal's intention to insert their payload within PDFs rather than inserting the actual contents. The metadata attributes */Info*, */CreationDate*, */ID*, and */Producer* are frequently present in clean PDFs but are hardly found within the fraudulent PDFs. Similarly, the clean files exhibit a higher frequency of the *MetadataStream* feature compared to the malicious files.

This discovery suggests that hazardous PDFs may have a lower number of metadata characteristics compared to clean PDFs. Figure 5.16 shows the breakdown for the object attribute between the two PDF groups. For the safe PDF files, the average amount of objects is 85.61, while the typical variation is a sizable 169.81. The risky PDFs, on the other

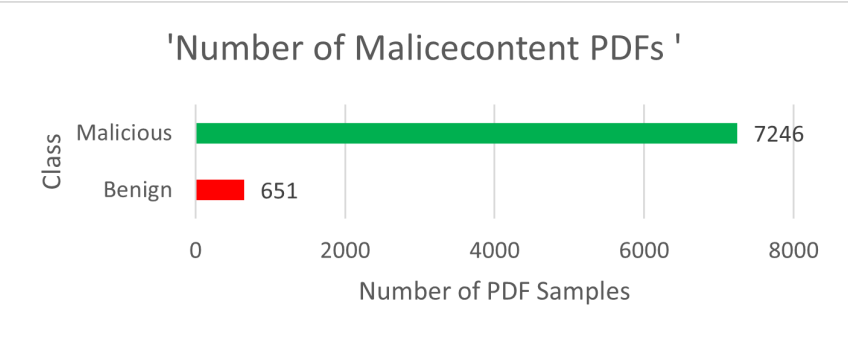

Figure 5.13. Malicecontent attribute distribution in the functional dataset for both malicious and benign PDFs.

hand, have a variance of 21.89 and a relatively small average of 14.11. This observation underscores the tendency for illicit files to contain a reduced quantity of items, as the objective of the perpetrator is to efficiently execute their assault by constructing harmful code using minimal resources. A consistent pattern is evident with respect to the endobj feature, which dictates that every object definition ought to be followed by an endobj. With a mean score of 16.50, however, there is a modest disparity in the mean size of the endobj and obj features in harmful PDFs. Regarding the feature*stream* it has been observed that harmful files employ a reduced quantity of streams when compared to legitimate files. Attackers utilize the *xref* and *startxref* functions to evade identification. A PDF document is shown and rendered by a reader program in the following manner: The EOF (End Of File) marker functions as the initial reference point for reading the file and is located at the closing of the document. The *startxref* will be followed by the cross-reference table's offset just above it. The root dictionary's offset, which acts as the initial reference point for the hierarchical structure (PDF file), is stored in the cross-reference table. This offset allows for the retrieval of all items. If either the *startxref* or the *xref* components are absent, meticulous parsers and readers will deem the document as defective. On the contrary, a versatile reader will have the ability to explore the dictionary of the root and display the file, similar to modern readers. Attackers can exploit the properties of %EOF to carry out malicious activities.

From this study's dataset, it was discovered that the traits */OpenAction* and */XFA* are more prevalent in illicit PDFs compared to safe ones. Upon analysis, it was found that clean PDFs are larger in size and have a greater number of elements than hazardous ones. The attributes such as /S (subtype of objects or tasks), /Rect, XML, Referencing, /Font and set of procedures or /ProcSet are more commonly observed in the safe documents as opposed to the malicious PDFs.Furthermore, there are 5.79 mean pages in files that are harmless and 1.44 mean pages in dangerous files. This shows that the harmful files found within the operating data are mainly small, with one or two pages that are usually

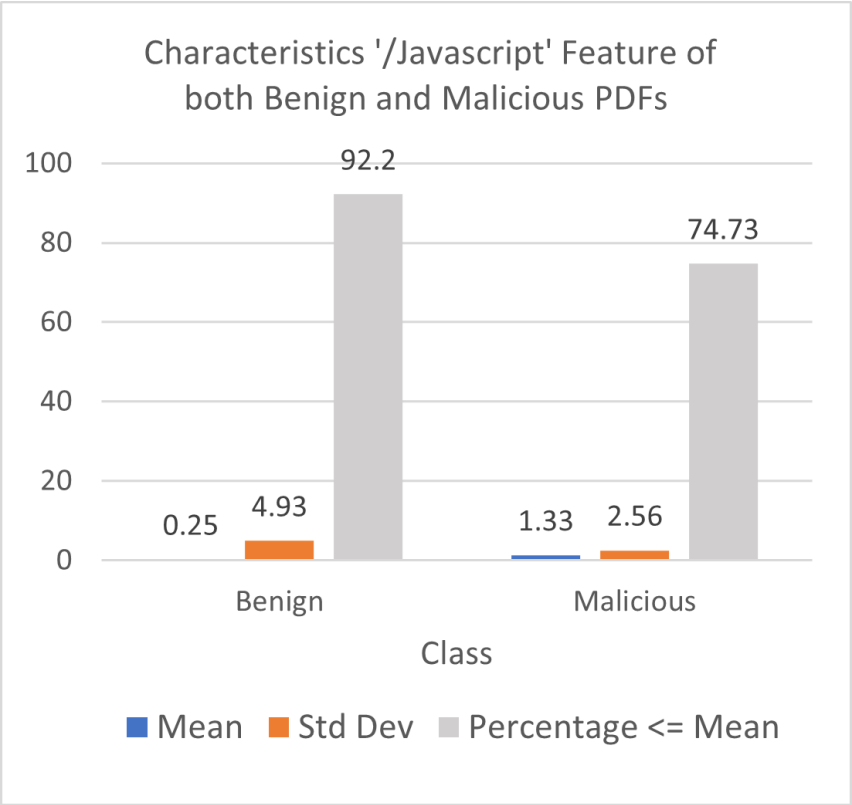

Figure 5.14. /JavaScript attribute distribution in the functional dataset for both malicious and benign PDFs.

blank or have very little content. This finding suggests a possible method for spotting questionable PDF documents. To recognize fake PDFs, the restricted relevance of the characteristic "Optimized" was assessed and the associated feature "small content" over our whole operating dataset.

5.7 Human Interpretation and Rule Discovery

The explanation regarding the results of the Random Forest model is investigated at this phase, specifically its ability to forecast illicit PDF files using the final set of features.This is accomplished by creating a decision tree using one of the classifier's estimators and acquiring several crucial decision rules derived from the classifier's implementation for detecting PDF malware. In addition, explanations of the decision rules are offered to facilitate understanding. Figure 5.17 shows one of the decision trees from our Random Forest classifier's 100 estimators to analyze its effectiveness. The created decision tree explains the classifier's performance by reflecting the requirements of the different characteristics from the final set of features in its vertices for recognizing benign or malicious PDFs. Upon inspection of the resulting tree, it becomes apparent that every vertex delineates an attribute accompanied by an acceptance criterion that divides the data points. Additionally, every vertex provides the proportion of samples that have been processed and reached that particular node. Also, in addition to indicating the eventual class name that has received

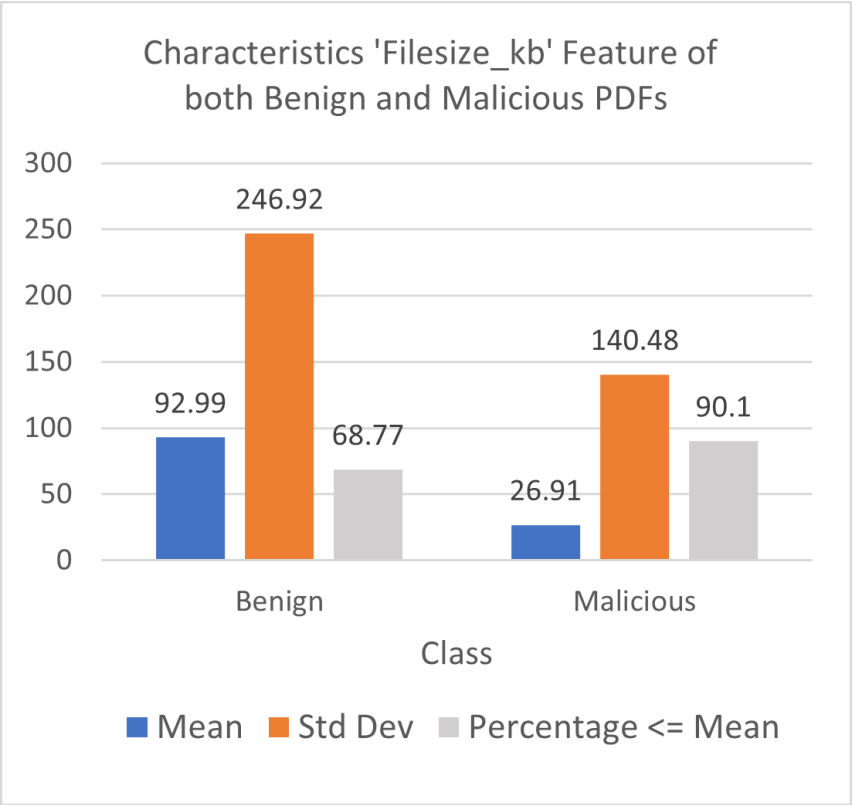

Figure 5.15. Filesize_kb attribute distribution in the functional dataset for both malicious and benign PDFs.

the largest number of votes vote, every vertex offers the relative distribution of classes of the data points that made it to that node. The key component of the tree is indicated by the attribute at the root node. The control moves to the left child of the root node if the condition in the root node is true, and to the right child in the opposite case. This procedure is repeated for all nodes until the leaf node is encountered, which signifies a decision rule that specifies a particular class designation.

The graphical representation of the tree's structure reveals that the *Malicecontent* feature is situated in the root node. This attribute establishes a criterion with the value $Malicecontent <= 0.5$ that is utilized to partition the PDF examples. The criterion specifies that the very first choice stage of the tree assesses if the current value of the *Malicecontent* trait is equal to or below 0.5. It is evident that the base node processes all of the data that is taken into account for the tree. The numbers enclosed in square brackets represent the category dispersion at this node in a proportional fashion. The initial number represents 49% safe instances, while the subsequent value denotes 51% malevolent class presence across the examples that arrived at the root point. At this node, the overwhelming preference is for the malevolent class. The hues intensify as the node approaches a state of total benignity or malice. Likewise, the hues get brighter when the node has a more uniform distribution of the samples within it.

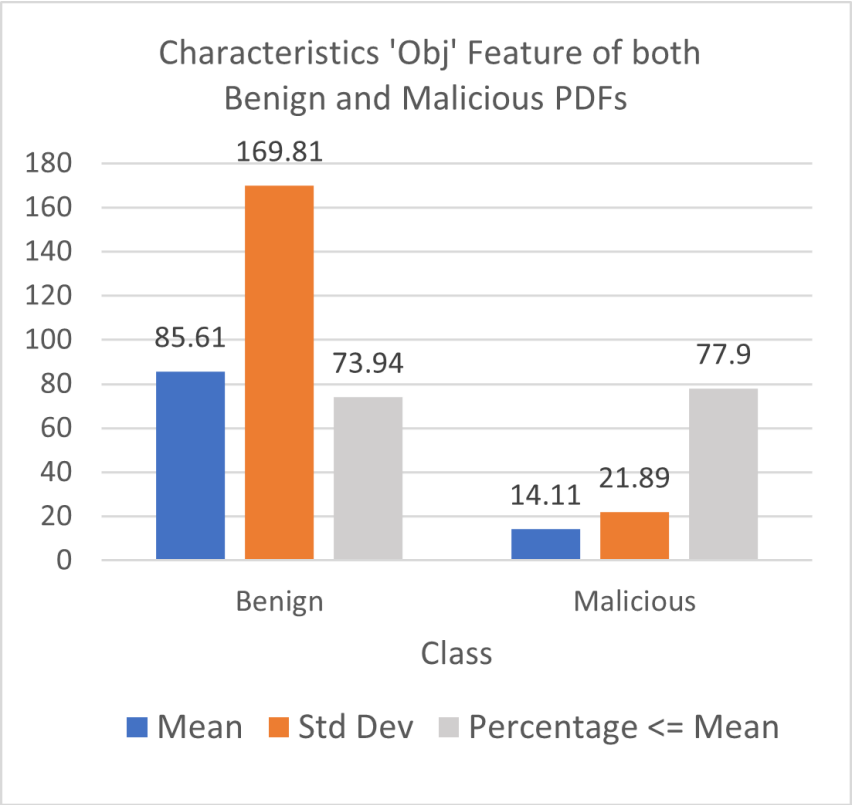

Figure 5.16. Obj attribute distribution in the functional dataset for both malicious and benign PDFs.

One of the decision rules in Figure 5.17 states that if the values of Malicecontent, /ID, and Headerlength satisfy the conditions (Headerlength <= 39.5 and /ID > 0.5 and Malicecontent <= 0.5), then the sample is classified as Benign. This suggests that in the event that a PDF example that comes from our functioning dataset has a title length of 39.5 or less, has /ID metadata attribute and does not contain the Malicecontent attribute, the sample is categorized as benign. An example of a decision plot produced by the Random Forest classifier is illustrated in Figure 5.18.

This plot represents one case that meets the aforementioned decision rule and is classified as benign. The decision plot was generated using the SHAP library, which serves to improve the understanding of the method of decision-making by highlighting the significance of individual features in relation to the results generated by the classification model.

The plot underscores the attributes that impact the classifier's ability to categorize a specific occurrence as detrimental or beneficial. Two distinct areas comprise the plot. The characteristics denoted by the blue area situated to the vertical line's left are those that impact the classifier's prognosis with respect to the benign category. The crimson region, situated to the right of the vertical line, highlights the characteristics that impact the classifier's prognosis with respect to the malevolent class. Upon analysis of Figure

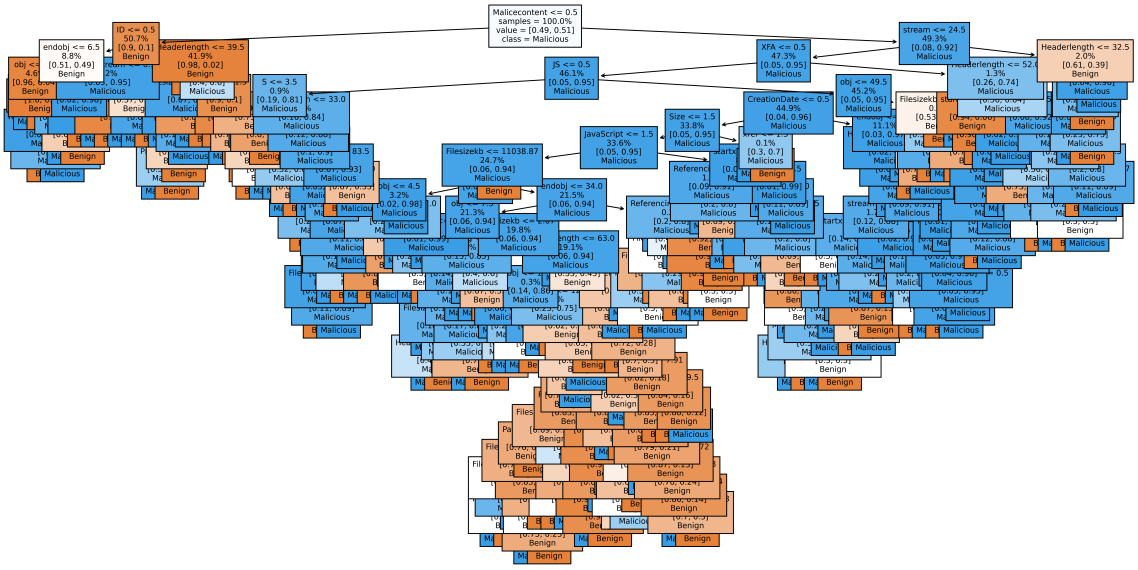

Figure 5.17. A decision tree constructed from the Random Forest classifier's estimators to detect malicious PDF

5.18, it becomes apparent that certain attributes, namely *Malicecontent*, *Headerlength*, and */ID*, exert a substantial impact on the classifier's prognosis, thereby providing a preference for the benign class.

Additionally, a different decision rule can be identified that states "If the Malicecontent is greater than 0.5, the stream is less than or equal to 24.5, the /XFA is greater than 0.5, and the Headerlength is less than or equal to 52, then the Class is classified as Malicious". This reveals the PDF samples that satisfy the following criteria are deemed malicious: they must contain the Malicecontent feature, have a minimum of 24.5 streams, and have the XFA form with a title length of no more than 52. Figure 5.19 illustrates the decision plot for one instance of the aforementioned decision rule. It is evident that features such as */XFA*, *Headerlength*, *Malicecontent*, and *stream* strongly influence the classifier's prediction for the malicious class.

A comprehensive set of 230 decision rules was extracted out of the drawn Random Forest classifier's decision tree, which serves as the blueprint for PDF malware detection predictions. In light of the fact that the model comprises one hundred estimators, we constructed and deduced every possible decision rule out of the classifier's decision trees in order to dig deeper into the rules. The total number of rules for decision-making identified in all of the decision trees generated by the classifier is illustrated in Figure 5.20. A grand sum of 23183 rules for decision-making are deduced out of the 100 estimators utilized in the algorithm. Additionally, the results indicate that the tree with an index of 53 yields the most amount of decision-making rules (333), while the tree with an index of 07 returns the

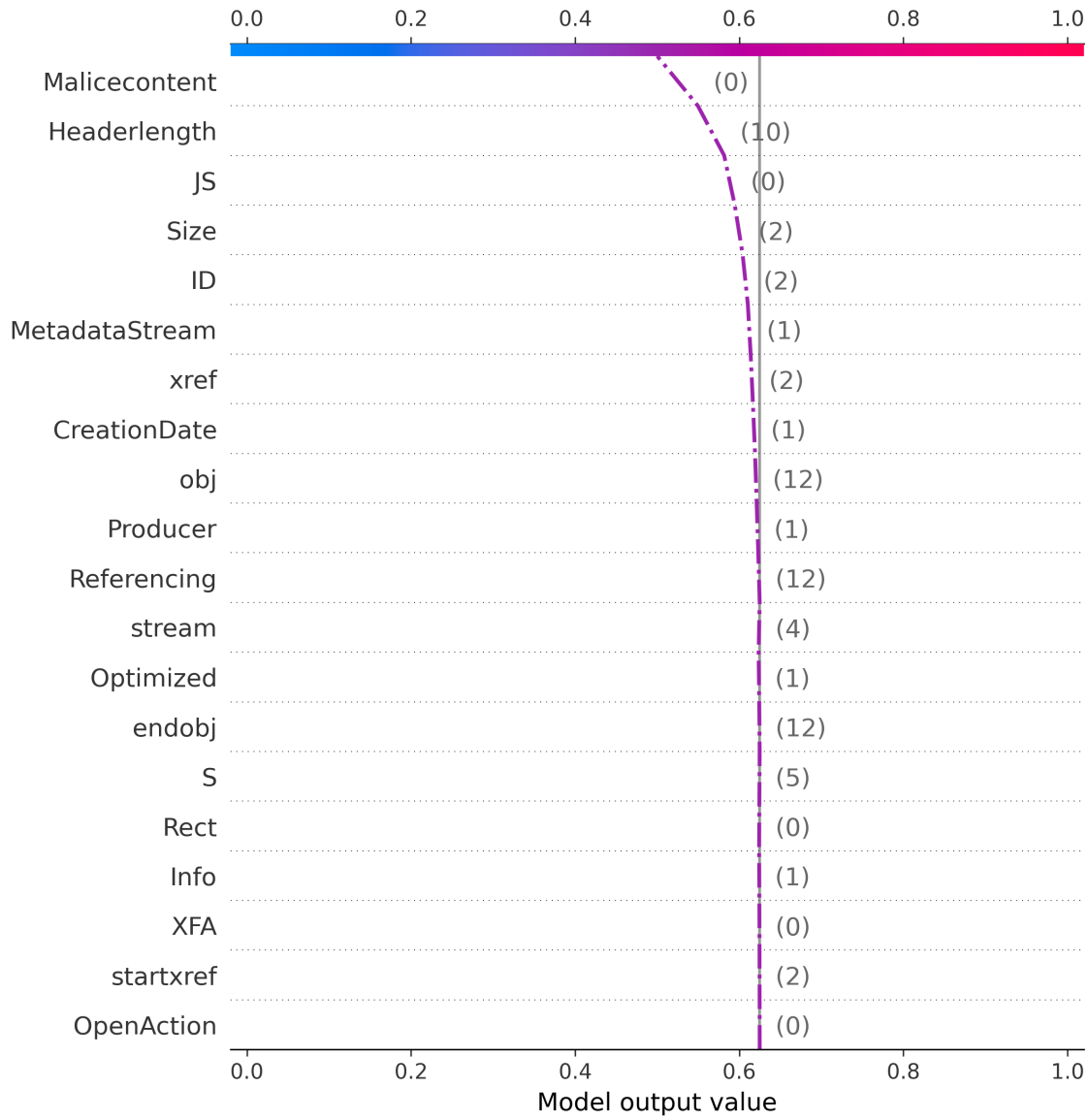

Figure 5.18. This is a visual representation of the decision plot for a specific instance belonging to the Benign Class using SHAP values.

least amount of decision rules (162). The average variation of the total amount of rules for decisions derived from the classifier's decision trees is 34.11, with a mean of 231.83. This indicates that the decision trees employ a mean of 231 rules for decision-making, which facilitates the projections made by the Random Forest classifier.

Tables 5.10 and 5.11 display crucial rules for decision-making for identifying safe and hazardous PDF files, correspondingly. These tables provide information on the rule's conditions, the total sample number that satisfies the rule, and the rule's confidence with the correct predictions, which indicates the frequency at which the rules are verified. Both of these tables offer a thorough elucidation that can be readily understood by humans and assist them in discerning between benign and malicious PDFs. Upon examining Table

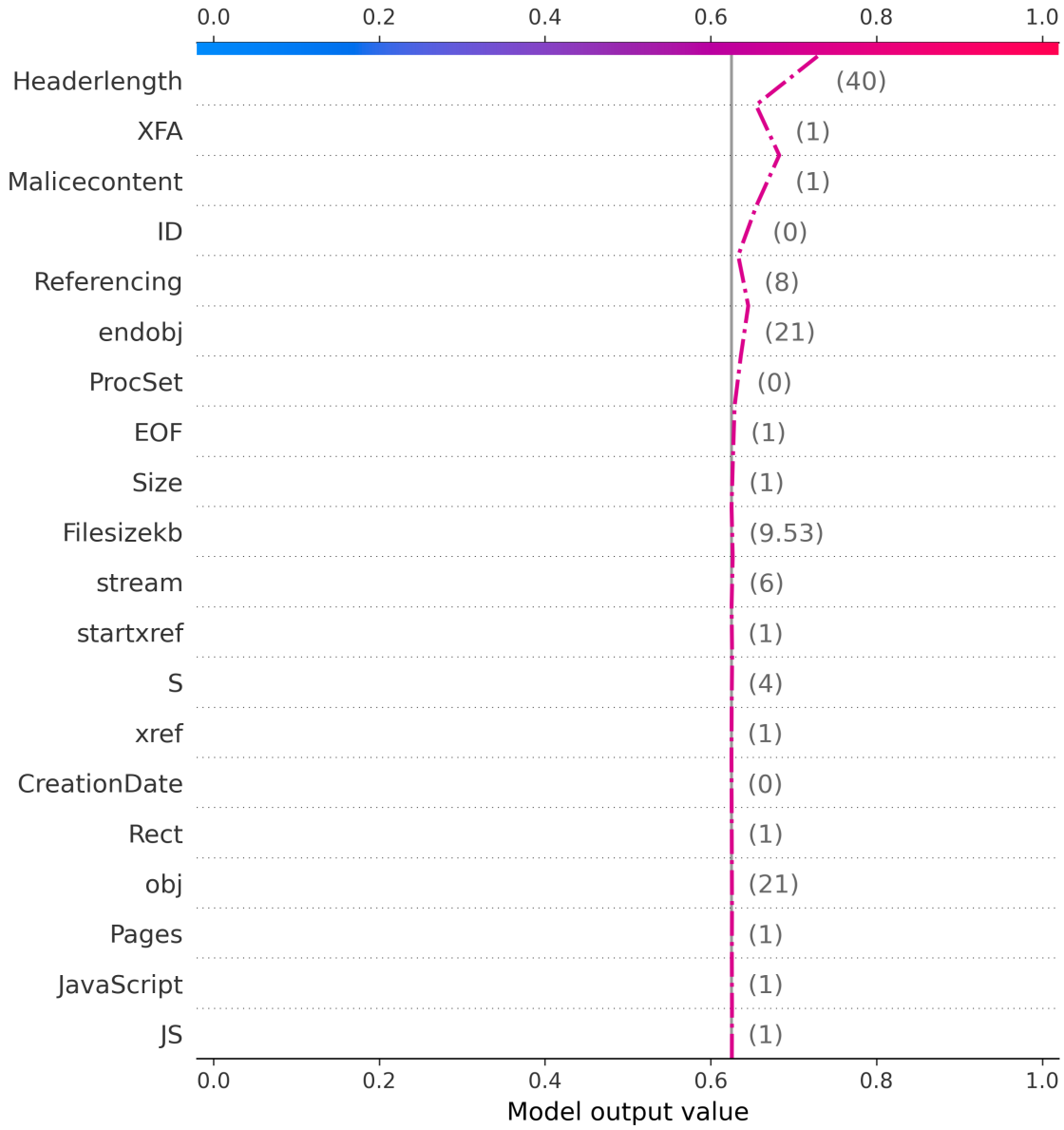

Figure 5.19. This is a visual representation of the decision plot for a specific instance belonging to the Malicious Class using SHAP values.

5.10, it can be observed that the initial decision rule states that if the */Javascript*is not used in a PDF sample and the length of the title is 39.5 or less, then the PDF sample is classified as benign. From our operational dataset, the rule successfully recognizes 7154 PDFs, indicating that the rule has a confidence level of 100%. Just like the initial decision rule, humans can easily explain and interpret all the other rules in the table to accurately identify innocuous PDFs. In addition, we have discovered several robust rules (namely Rule ID 2 to 8) that exhibit a confidence level of 99% to 100% and effectively encompass a diverse set of samples for the purpose of recognizing benign PDFs. In contrast, the rules at the bottom of Table 5.10 (specifically Rule ID 9 to 10) are shown to be less effective. These rules have a confidence level ranging from 92% to 97% and only cover a small

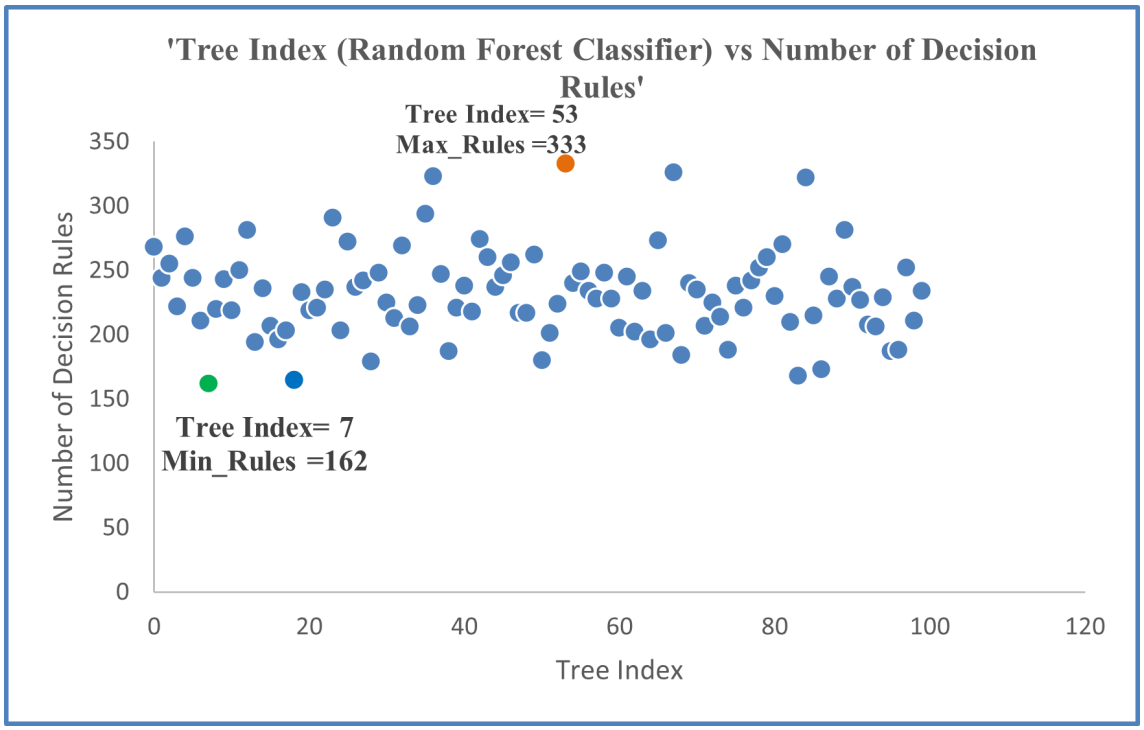

Figure 5.20. Tree index vs number of decision rules.

number of samples, making it evident that they are meant to detect clean files.

Examining Table 5.11, it is evident that for the detection of dangerous PDFs, a significantly greater number of requirements need to be validated compared to the detection of benign PDFs. Although faced with significant limitations, we have identified several robust rules (namely Rule ID 1 to 5) that exhibit almost perfect accuracy in identifying thousands of samples as malicious from the functional dataset. However, as we meticulously analyze the decision rules in the table from 6 to 9, we discover a few less efficient rules that are only applicable to a small subset of samples in our dataset. The decision rules of Table 5.11 can be thoroughly discussed and interpreted, similar to the explanation provided in Table 5.10. The first rule of Table5.11 specifies that a PDF sample is classified as malicious if it does not have the is not optimized, */XML* feature, has a file size smaller than or equal to 46.52 kilobytes but larger than 1.41 kilobytes, has a title length shorter than or equal to 63 but longer than 39, does not include the */CreationDate* feature and contains a cross-reference table that is less than or equal to 1.5 times. The rule accurately detects 5079 fraudulent PDF samples from the dataset.

Furthermore, the remaining decision rules in Table 5.11 can be elucidated and understood to identify malicious PDFs. However, the judgement rules provided in Table 5.10 and Table 5.11 are important for people to accurately categorise benign and harmful PDFs.

Table 5.10. Random Forest classifier's crucial decision rules for identifying safe PDF.

ID	Rule conditions	Total Sample	Correct Predictions	Confidence (%)
1	If (Headerlength <= 38.5 and /ProcSet <= 1.5 and Malicecontent <= 0.5) then	1723	1723	100
2	If (/JavaScript <= 0.5 and Headerlength <= 39.5) then	7154	7154	100
3	If (Headerlength <= 39.5 and Filesize_kb > 10.25 and /XFA <= 0.5 and Malicecontent <= 0.5) then	7133	7133	100
4	If (MetadataStream > 0.5 and /ID > 0.5 and %EOF > 1.5 and Malicecontent <= 0.5 and /Producer > 0.5) then	4530	4530	100
5	If (/Font > 0.5 and Headerlength <= 62.0 and stream > 2.5 and %EOF <= 9.0 and obj > 6.5 and /ID > 0.5 and /XML > 0.5 and MetadataStream > 0.5) then	4434	4433	99.97
6	If (Malicecontent <= 0.5 and Referencing > 36.0 and /Producer <= 0.5) then	85	84	98.82
7	If (/ProcSet <= 1.5 and MetadataStream > 0.5 and Malicecontent <= 0.5 and /Font > 0.5 and %EOF <= 1.5) then	411	409	99.51
8	If (Filesize_kb > 12.82 and xref > 0.5 and /Producer > 0.5 and endobj <= 18.5 and %EOF <= 1.5 and Malicecontent <= 0.5) then	819	817	99.75
9	If (/ProcSet > 0.5 and stream <= 100.5 and /Font > 45.5 and /S <= 163.5 and endobj > 76.0 and Malicecontent > 0.5 and /Producer > 0.5) then	31	30	96.77
10	If (/XFA <= 0.5 and /ProcSet <= 0.5 and /Font <= 140.5 and MetadataStream <= 0.5 and /ID > 0.5 and %EOF > 1.5 and Malicecontent <= 0.5 and /Producer > 0.5) then	13	12	92.30

Table 5.11. Random Forest classifier's crucial decision rules for identifying malicious PDF.

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

5.8 Performance Comparision

In order to evaluate this research on PDF malware detection, a comparative analysis was conducted alongside several prior studies in the identical field of study. The comparison investigation is succinctly outlined in Table 5.12. It assesses our findings across multiple dimensions, involving the origin of the PDF instances, the total count of materials included in the evaluation, the tags applied to the PDFs, the instruments utilized for PDF assessment, and the overall count of PDF attributes taken into account for the purpose of the research. Moreover, the observed accuracy alongside the machine learning models applied during the study. Also, the comparison incorporates decision rules and human explanations regarding malicious PDF identification to verify whether the studies provide these or not. The approach outlined by the researchers of [1] involved the utilization of machine learning models to quantitatively and dynamically assess a provided PDF in order to determine its possibly dangerous content. Using 1200 PDF files as well as the PDF evaluation instrument PhoneyPDF, they determined the fact that the Random Forest classifier provided the most accurate detection rate 98.6% of fraudulent PDFs.

Likewise, the Random Forest classifier was employed by the researchers in [34] to detect fraudulent PDFs. In their pilot inquiry, PeePDF and PDFiD PDF analysis tools were executed to a mere 1000 PDF samples to retrieve the essential elements necessary for the classification assignment. Nevertheless, the proposed technique yielded a peak accuracy of 97.4% in successfully accomplishing the assignment. In addition, the authors in [4] introduced Optimizable Decision Tree, a methodology known as O-DT for the purpose of malicious PDF identification. An accuracy of 98.84% for the proposed technique was acquired by applying a benchmark dataset to conduct their desired tests. A dataset of 10,025 PDF samples was introduced by the study in [46], which were selected based on their elusive attributes. In addition, a stacking learning-based ensemble classifier was adopted by the authors, which achieved a detection accuracy of 98.69% for identifying PDF malware. Our work surpasses the existing studies listed in Table 5.12 by effectively analyzing a significant number of PDF samples. This is achieved through the utilization of three sophisticated tools for extracting features, deriving crucial characteristics, and harnessing the capabilities of the Random Forest classifier. As a result, we achieve a significantly higher accuracy rate of 99.24% in detecting PDF malware. In addition, based on our current understanding, none of the studies listed in Table 5.12 provide a comprehensive human interpretation of the classifier's performance. This includes the visualization of a decision tree and the identification of decision rules for detecting PDF malware.

Table 5.12. Comparing our findings to numerous previous research investigations for malicious PDF detection.

Reference	**PDF Sample Source**	**No. of Samples**	**PDF Labels**	**PDF AnalysisTools**	**No. of Features**	**No. of Derived Features**	**Model**	**Accuracy (%)**	**Rule Discovery and Human Interpretation**
[1]/2022	Internet	1200	Malicious-629 Safe- 571	Phoney PDF	-	-	RF	98.6	No
[4]/2022	Evasive-PDFMal2022	10,025	Malicious Evasive- 5557 Benign Evasive- 4468	-	37	-	O-DT	98.84	No
[46]/2022	Evasive-PDFMal2022	10,025	Malicious Evasive- 5557 Benign Evasive- 4468	-	37	-	Stacking	98.69	No
[34]/2021	PersonalFiles Contagio, VirusTotal,	1000	Malicious-530 Clean-470,	PeePDF PDFiD,	14	05	RF	97.4	No
This Paper	Contagio, VirusTotal, Evasive-PDFMal2022	15,958	Malicious Evasive-392 Benign Evasive-400 Malicious- 7666 Benign- 7500	PDF-PARSER PDFiD, PDFINFO,	28	07	RF	99.24	**Yes**

5.9 Discussion

A dataset was compiled including dangerous, clean, and evasive PDFs for the purpose of malicious PDF identification. Nevertheless, only 792 elusive PDFs were included, which accounts for around 5% of the overall samples. To mitigate the bias of the classifier, evasive probability density samples were introduced. Additionally, the classifier's resilience is enhanced due to the elusive properties of the PDFs for detecting malicious PDFs. In order to address the issue of the classifier being biased towards a specific class, it was made sure to have a roughly equal proportion of benign and malicious PDFs. In order to examine the PDFs and get the pertinent characteristics, three renowned and exceptionally precise tools, PDF-PARSER, PDFINFO, and PDFiD, were employed. The purpose of utilizing these tools was to create a robust set of features for detecting malware in PDF files by examining several effective toolkits that guarantee the standard feature's validity that was extracted from the PDFs. In addition, some significant characteristics were extracted and these were combined with the conventional features to create a consolidated feature set. The final set of features was constructed by creating subsets from the merged set of features and evaluating them using the model's capabilities.

A comprehensive experimental examination was conducted of the final set of features in order to elucidate the characteristics of the feature set. The length of the title in PDF files is a critical factor, as malicious files often have an atypical title length in comparison to clean ones. In addition, clean PDFs often contain metadata and structural features such as /CreationDate, /ID, /ProcSet, /Producer, while harmful PDFs seldom have these elements. Typically, attackers minimize the size of fraudulent files by limiting the information, pages, typefaces, and overall size. This is done to ensure that their assaults are executed quickly and efficiently. It can be found that the main objective of cyber criminals is to embed malicious content, such as OpenAction files and JavaScript code, inside the structure of PDFs in order to cause harm to the victim's systems.

A clear and detailed analysis of the classifier's effectiveness was provided by creating a decision tree using one of its estimators. Additionally, several important decision criteria were emphasized for identifying malicious and clean PDFs. Moreover, robust decision rules can be identified which can accurately classify both types of PDFs with a confidence level of up to 100%. These rules are capable of identifying a huge number of samples. However, we have also observed that certain rules require a substantial number of constraints to be satisfied, which reduces their effectiveness to some extent. Furthermore, these inadequate regulations have the ability to precisely detect a limited number of samples while producing a strong level of certainty. Nevertheless, the decision criteria provide a lucid comprehension

and explanation of how the characteristics might be employed to identify PDF malware.

5.10 Limitations

For the purpose of enhancing the robustness of our classifier, evasive behaviours were incorporated into the experimental dataset in this work. However, the effectiveness of the classifier can be improved to better defend against sophisticated current attacks by incorporating additional elusive PDF characteristics through a thorough examination. In this experiment, three tools were adopted, namely PDF-PARSER, PDFINFO, and PDFiD, to extract features. Although these tools are widely used, they are susceptible to vulnerabilities, such as the PDFiD utility, which can be used in certain attacks. An example of an attack is the parser confusion attack, in which fraudulent content is camouflaged and hidden using various methods to evade detection while still being able to execute and exploit.

Chapter 6

Conclusion

PDF has previously been employed as an increasingly common approach to spreading malware. The document is accessed due to the consumer's trust in this layout, and malicious code is launched as a result of any fragility detected in the application that interprets the file's contents and launches code. In this way, scammers victimize users injecting malicious PDFs into their systems. This study involved a comprehensive analysis of malicious PDF detection. In order to accomplish this, initially, an extensive dataset consisting of 15958 PDF examples was created. The characteristics of these samples were carefully considered, including their non-malevolent, malicious, and evasive properties. In addition, a technique to create a feature set was devised that is both efficient and comprehensible. This was accomplished by extracting key attributes out of the PDF samples from the freshly generated dataset using a variety of PDF analytics applications (such as PDFINFO, PDF-PARSER and PDFiD). In addition, a few derived features were identified that had been experimentally proven to be effective in PDF malware detection. Moreover, a study on various machine learning techniques was conducted and emphasized the efficacy of the Random Forest model, not only for performance evaluation but also for the investigation of explainability analysis by creating decision rules. Furthermore, a comprehensible human interpretation was provided by explaining decision rules as well as illustrating decision plots for identifying malicious PDFs. Besides, the functions of the attributes responsible for identifying PDF malware were elucidated and many significant findings were highlighted that can assist in the identification of dangerous PDF files. Ultimately, the discoveries of various cutting-edge research were compared and emphasized significant insights from our study.

Chapter 7

Future Works

The future efforts will focus on resolving each of the restrictions observed in this study. Complex content, such as JavaScript code, can be thoroughly examined by integrating features from several parsers and analysis methodologies. Moreover, the current set of features is obtained from three extraction methods, and the features utilized by the three programs are based on insights and heuristics provided by their developers. By integrating new sources, such as tools for generating malicious documents or conducting in-depth analysis of harmful PDF documents, a more extensive and comprehensive collection of features can be included. A study that can be conducted is examining the malicious PDFs' internal text content, where attackers may conceal their dangerous program within the text content. Furthermore, evaluating the applicability of our proposed approach across several categories of PDF malware by examining the model's performance against diverse forms of PDF malware is another concern, encompassing both contemporary and sophisticated variants. Additionally, the runtime behavior, dynamic attributes, and phishing web links of the PDF can be utilized to conduct further analysis of suspicious documents.

Furthermore, to validate the practical efficacy, another concern is to apply the suggested approach in simulations or real-world environments in detecting different forms of malicious PDFs. In addition, exploring certificateless signcryption and proxy signcryption as sophisticated approaches to enhance the protection of PDFs can be another future direction. These strategies can provide extra layers of security to mitigate the threats posed by PDF malware. Furthermore, this research can pave the way for future studies that combine cryptographic techniques with malware detection. Furthermore, the presence of adversarial PDF malware continues to represent a significant risk to the security of cyberspace. In order to address such risks in the future, one of the key objectives may be to create a data-driven intelligent method that can successfully handle adversarial PDF viruses. In addition, releasing an extra dataset consisting exclusively of elusive PDF examples that encompass a diverse array of methods used in cyber attacks can lead to another future direction.

Bibliography

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

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

[3] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

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

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

[6] H. Bae, Y. Lee, Y. Kim, U. Hwang, S. Yoon, and Y. Paek, "Learn2evade: Learning-based generative model for evading pdf malware classifiers," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 4, pp. 299–313, 2021.

[7] E. Times. "66% of malware delivered via pdf files in malicious emails: Report." [Accessed: 20-Jul-2023]. (2023), [Online]. Available: %5Curl%7Bhttps://ciso.economictimes.indiatimes.com/vulnerabilities-exploits/66-of-malware-delivered-via-pdf-files-in-malicious-emails-report%7D.

[8] Statista. "Malware - statistics & facts." [Accessed: 20-Jul-2023]. (2023), [Online]. Available: https://www.statista.com/topics/8338/malware/#topicOverview.

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

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

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

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

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

[14] C. Liu, C. Lou, M. Yu, *et al.*, "A novel adversarial example detection method for malicious pdfs using multiple mutated classifiers," *Forensic Science International: Digital Investigation*, vol. 38, p. 301 124, 2021.

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

[16] S. Mejjaouli and S. Guizani, "Pdf malware detection based on fuzzy unordered rule induction algorithm (furia)," *Applied Sciences*, vol. 13, no. 6, p. 3980, 2023.

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

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

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

[20] M. Yu, J. Jiang, G. Li, *et al.*, "Malicious documents detection for business process management based on multi-layer abstract model," *Future Generation Computer Systems*, vol. 99, pp. 517–526, 2019.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[35] *Virustotal*, Available online: https://www.virustotal.com/gui/home/upload, 2023.

[36] A. R. Kang, Y.-S. Jeong, S. L. Kim, and J. Woo, "Malicious pdf detection model against adversarial attack built from benign pdf containing javascript," *Applied Sciences*, vol. 9, no. 22, p. 4764, 2019.

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

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

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

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

[41] M. Li, Y. Liu, M. Yu, G. Li, Y. Wang, and C. Liu, "Fepdf: A robust feature extractor for malicious pdf detection," in *2017 IEEE Trustcom/BigDataSE/ICESS*, IEEE, 2017, pp. 218–224.

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

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

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

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

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

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

[48] N. Livathinos, C. Berrospi, M. Lysak, *et al.*, "Robust pdf document conversion using recurrent neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 15 137–15 145.

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

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

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

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

[53] I. Ullah, N. Ul Amin, M. Zareei, *et al.*, "A lightweight and provable secured certificateless signcryption approach for crowdsourced iiot applications," *Symmetry*, vol. 11, no. 11, 2019, ISSN: 2073-8994. DOI: 10.3390/sym11111386. [Online]. Available: https://www.mdpi.com/2073-8994/11/11/1386.

[54] A. Waheed, A. I. Umar, M. Zareei, *et al.*, "Cryptanalysis and improvement of a proxy signcryption scheme in the standard computational model," *IEEE Access*, vol. 8, pp. 131 188–131 201, 2020.

Appendix A

The appendix presents the findings of various feature selection techniques, such as p-value, that were applied during the experiments. Also, explains the rationale behind taking the feature importance score of at least 0.01 (1%) to select the top features from the feature sets F_1', F_2', and F_3'. Next, the performance curve of our proposed Random Forest classifier is discussed. Finally, a detailed comparative study with a few existing research is presented.

Table A.1 presents the features F_1', F_2', and F_3' along with their corresponding p-value. In this table, the features whose p-value is less than 0.01 (significance less than 1%) were considered. As the p-value suggests, the lower the p-value, the higher the relationship of a feature with the target feature. Thus, we calculated the p-value and identified the top features from the derived feature sets F_1', F_2', and F_3'. Then, we followed Algorithm 1 to generate the final feature set by taking the top features considering p-values.

Initially, we have successfully identified the top 14 features from F_1', the top 07 features from F_2', and the top 17 features from F_3' as the best subset (based on Algorithm 1). Later, we performed a union operation among the best subset of F_1', F_2', and F_3' and finally got 29 features in the selected feature set. The selected features are: 'Headerlength', '/Length', '/FontDescriptor', 'Referencing', '/XML', '/Info', '/ModDate', '/Font', 'xref', 'obj', '/Producer', '/CreationDate', 'startxref', 'small content', 'Malicecontent', '%EOF', '/Size', 'stream', '/OpenAction', 'trailer', '/Page', '/XFA', '/AcroForm', '/ObjStm', 'endstream', 'Page size:_A4', 'Optimized:', 'Page size:_miscsize', and '/JavaScript'. We got 20 features common with our final feature set, which was identified by taking the feature importance score. Furthermore, to justify the effectiveness of the selected feature set (identified by taking p-values), we implemented our proposed model (i.e. Random Forest Classifier) utilizing this feature set. It was observed that the proposed model yielded an accuracy of 98.79%, which is less than the accuracy (99.24%) observed from our final feature set. Hence, it proves the efficacy of our final feature set, which was identified by taking the feature importance score.

Table A.1. p-value of derived feature set F1', F2', and F3' (p-value < 0.01).

F1'	p-value	F2'	p-value	F3'	p-value	F3'	p-value
Headerlength	0	Headerlength	0	Headerlength	0	HiddenFile	2.74149E-76
small content	0	Page size:_A4	0	/Length	0	/Rect	7.48906E-63
stream	0	Malicecontent	0	/FontDescriptor	0	/JavaScript	2.85396E-62
xref	0	small content	0	Referencing	0	/Image	1.2554E-57
startxref	0	Optimized:	0	/XML	0	/JS	1.32942E-41
Malicecontent	0	Page size:_miscsize	0	/Info	0	/ID	7.15099E-37
/OpenAction	0	JavaScript:	0	/ModDate	0	Comment	0.00968731
obj	9.97E-295	Tagged:	0	/Font	0		
trailer	4.4036E-252	Metadata Stream:	0	xref	0		
/Page	2.3826E-226	Custom Metadata:	0	obj	0		
/XFA	6.0491E-145	Pages:	1.0896E-266	/Producer	0		
/AcroForm	2.1592E-132	Filesize_kb	4.5173E-95	/CreationDate	0		
/ObjStm	1.9629E-129	headercorrupt	3.95045E-89	startxref	0		
endstream	4.0494E-127	HiddenFile	2.74149E-76	small content	0		
endobj	1.2249E-107	Page size:_letter	7.78273E-75	Malicecontent	0		
headercorrupt	3.95045E-89	Form:_none	3.64082E-69	%EOF	0		
HiddenFile	2.74149E-76			/Size	0		
/JavaScript	1.6098E-67			/Action	0		
/JS	1.56677E-45				0		
/JBIG2Decode	6.92875E-13				0		
/RichMedia	1.38703E-05			/S	1.8935E-241		
/AA	3.226E-05			/ProcSet	2.6729E-212		
/Colors	0.008775869			/XObject	2.2909E-102		
/Encrypt	0.00969423			/Widget	2.92443E-96		
/EmbeddedFile	0.009566136			headercorrupt	3.95045E-89		

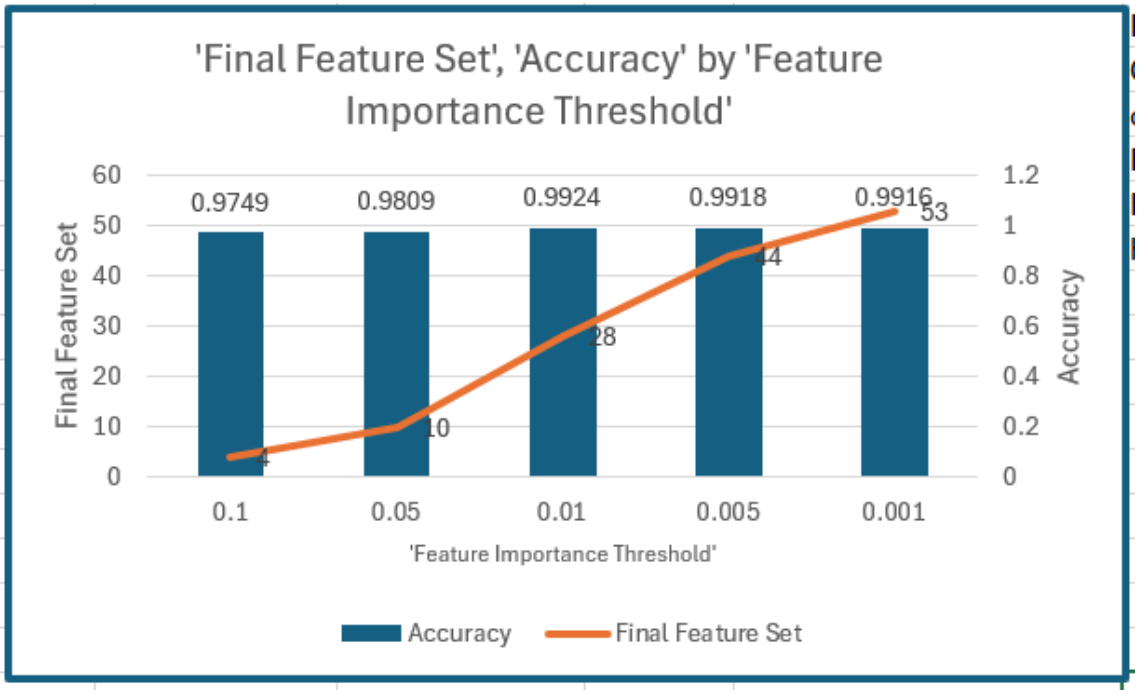

Figure A.1. Final feature set, accuracy by feature importance threshold.

To explain the rationale for taking the feature importance threshold as 0.01, we depicted an illustration in Figure A.1. We empirically analyze the performance of the proposed classifier by taking various thresholds and generating feature sets by following Algorithm 1. From the illustration, it can be observed that whenever the feature importance threshold was considered as 0.01 (i.e. 28 features in the final feature set), the classifier yielded the highest accuracy of 99.24%. However, for the other cases, the feature number and performance decreased whenever we took a greater threshold than 0.01. Nevertheless, when we took a smaller threshold than 0.01, the number of features increased along with the classification accuracy but they did not overpower the performance observed at threshold 0.01. Hence, we discovered the best performance of the classifier at 0.01 and that is why we took it as our final threshold value to select top features.

To further justify the proposed model's performance, we illustrated the learning curve of the model utilizing the final feature set. Figure A.2 depicts the performance curve of the proposed classifier. It can be observed that the classifier obtained a maximum cross-validation accuracy of 99.25% at 14362 examples. Thus, it verifies the effectiveness of our proposed classifier for PDF malware detection.

Finally, to investigate further, we did a detailed experiment to compare our findings with several existing works. We implemented the works of studies [1], [4], [34], [46] ,respectively. We executed these works in two phases: i) using their data with our method

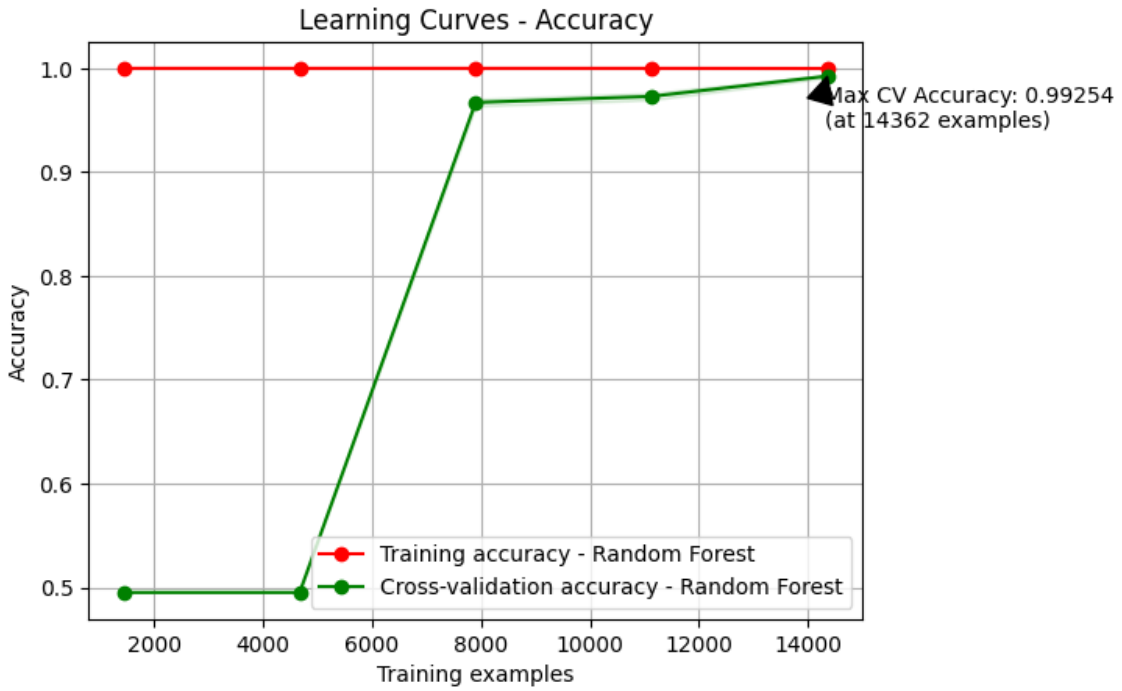

Figure A.2. Learning curve of RF on the final feature set

and ii) using our dataset but with their method.

The following tables (Tables A.2 and A.2) represent the experiments conducted to accommodate the queries as per the reviewer's suggestion. Table A.2 highlights the performance of our proposed approach implemented on the dataset of studies [1], [4], [34], and [46] (papers that are mentioned in the presentation and also cited in the reference section here). On the other hand, Table A.3 represents the performance of the proposed method of studies [1], [4], [34], and [46] on our operational dataset.

As it can be noticed from Table A.2 our proposed approach yielded better accuracy than the studies [1], [4], [34], and [46] when implementing their dataset as compared to their original work for identifying PDF malware. Besides, in Table A.3, we executed the method of studies [1], [4], [34], and [46] on our operational dataset. The increased performance of studies [1], [4], [34], and [46] can be observed as compared to their original result while utilizing our operational dataset. Overall, the above findings prove the efficacy of our proposed approach along with the operational dataset regarding PDF malware detection in both cases.

Table A.2. Implementing our method utilizing the data of [1], [4], [34], [46].

Reference	Dataset	Implementation	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
[1]	Malicious-629, Safe-571	Our Method with data of [1]	98.66	98.69	98.66	98.66
		Original Study [1]	-	-	-	98.60
[4]	Malicious Evasive- 5557, Benign Evasive-4468	Our Method with data of [4]	99.02	99.02	99.02	99.02
		Original Study [4]	98.84	98.80	98.90	98.80
[34]	Malicious-530, Clean-470,	Our Method with data of [34]	98.50	98.50	98.50	98.50
		Original Study [34]	97.40	98.40	96.70	97.50
[46]	Malicious Evasive- 5557, Benign Evasive-4468	Our Method with data of [46]	99.02	99.02	99.02	99.02
		Original Study [46]	98.69	98.88	98.87	98.77

Table A.3. Implementing the method of [1], [4], [34], [46] using our dataset.

Reference	Implementation	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
	Our Dataset with Method of [1]	0.99248	0.99248	0.99248	0.99248
	Original Study [1]	-	-	-	98.60
	Our Dataset with Method of [4]	99.06	99.06	99.06	99.06
	Original Study [4]	98.84	98.80	98.90	98.80
	Our Dataset with Method of [34]	0.99248	0.99248	0.99248	0.99248
	Original Study [34]	97.40	98.40	96.70	97.50
	Our Dataset with Method of [46]	0.99241	0.99242	0.99241	0.99241
	Original Study [46]	98.69	98.88	98.87	98.77