MACHINE LEARNING MODEL FOR PREDICTIVE FILE ANALYSIS

Justine A. Cua[1], Reynan B. Awa[1] Leonie A. Cajes[1]
justineacain.cua@my.smciligan.edu.ph[1] r.awa@my.smciligan.edu.ph[1]

[1] St. Michael's College of Iligan

ABSTRACT. PDFs are considered a standard method for document sharing. As technology evolves, individuals took up the advantage to infiltrate personal computers. Beneath the structure of a document, malware are hidden with obfuscated scripts that are undetectable by traditional antivirus programs leading to ineffective malware identification and an increase in false positives. This research aims to develop a machine learning model with a predictive analysis system for identifying malicious PDFs. Specifically, it detects obfuscated JavaScript code, implements a dynamic analysis framework using Docker to observe PDF behavior and evaluates the performance of Machine Learning Algorithms like Random Forest and Logistic Regression. The study utilizes supervised learning with datasets from Contagio, CIC, VirusShare, and Govdocs with a total of 493,238 PDF files. Feature extraction includes text properties, API calls, urls, and obfuscated JavaScript. Experimental results indicate that Logistic Regression outperforms Random Forest in detecting malicious PDFs, achieving 86.67% accuracy, while Random Forest performs better in identifying benign PDFs with 93.33% accuracy. Additionally, this research integrates an Optical Character Recognition model using YOLOv5 and TensorFlow to extract text from both PDFs and embedded images. The TextOCR dataset includes 1 million word annotations across 25,119 images, with the model achieving 96.67% accuracy on images with white backgrounds and 91.73% on complex backgrounds. For flexible implementation, a mobile application with offline capability is suggested, as many individuals rely on mobile phones for document access and security. This study focuses on securing data sharing between computers through PDF and image analysis.

245 words

Keywords*: PDF, Machine Learning, Malware Detection, Obfuscated JavaScript, Dynamic Analysis, Docker, Random Forest, Logistic Regression, YOLOv5, OCR*

Introduction

Malware intended to damage computer systems is a serious concern today due to its ability to alter, remove, or steal data for negative purposes. These programs rely on known signatures to find threats. However, if new malware doesn't match any known signatures this method often fails to spot them [1]. One way to spread malware is to attach it in PDF files. PDF stands for Portable Document Format. It was developed in 1993 by Adobe Inc. It includes executable code, images, text, and other attachments [2]. Researchers at Forcepoint have found that hackers are using these dangerous PDFs to target people who purchase tickets online [3]. Another exploit found by EdgeSpot where almost all antivirus engines failed to detect was an obfuscation technique with steganography. The sample includes two layers of obfuscation. The code pretends to be harmless pictures labeled as icons but is actually decoded and run using JavaScript in the PDF [4].

There are two ways of analyzing malware in PDF files. These are static analysis and dynamic analysis. Static analysis is struggling to find malware with obfuscated code. However, dynamic analysis which tracks behavior in real time, static analysis looks at the code without running it and focuses on features. These programs have trouble with packaged malware that encrypts its code in order to avoid detection. Static analysis fails to detect obfuscated code that is only revealed during execution [5]. The researchers want to make PDF files safer by improving how we detect hidden malware especially with obfuscated code. They'll use dynamic analysis, which watches how PDF files behave in real time. These effective

methods include using Docker and machine learning. The system focuses on PDF files where users can upload in a web-based system and a browser extension to identify malware and analyze files.

Objectives of the Study

The study aims to create a machine learning model for predictive file analysis This study specifically seeks to detect if a pdf file is malicious or benign with an extraction feature to identify obfuscated JavaScript code. Develop a dynamic analysis framework using Docker to simulate PDF file behavior and structural analysis. Evaluate machine learning methods such as Random Forest and Logistic Regression to identify malicious PDF files.

Methods

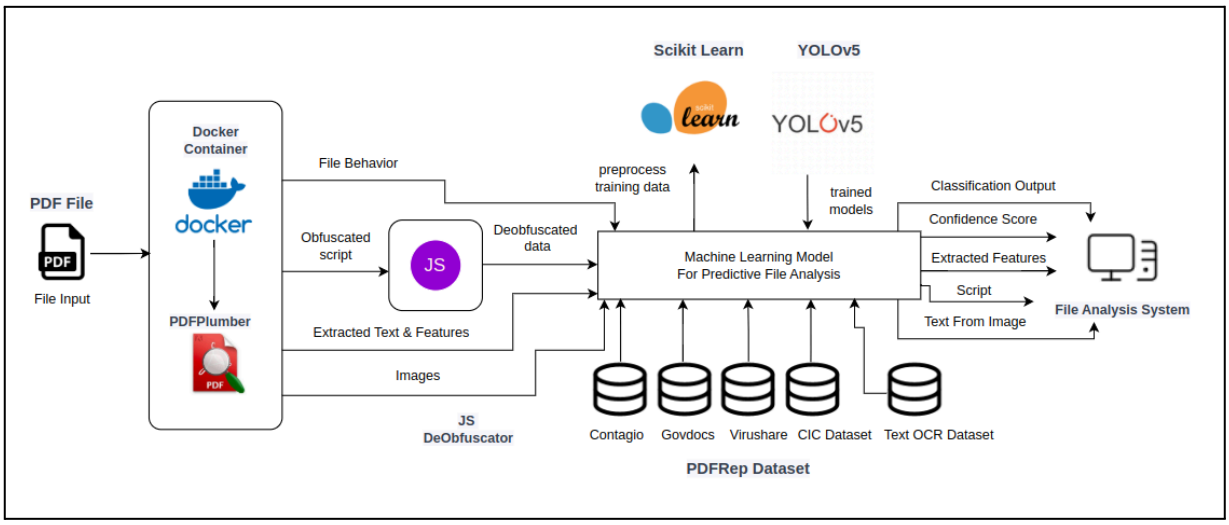


Figure 1. Conceptual Framework

A conceptual framework represents a structure between methods used in a study. It demonstrates how these components work together. Firstly, a user can upload a PDF file in the system. Then the file is opened inside a Docker container to analyze its behavior utilizing a safer environment. The system extracts features from the PDF, including its API calls, structure, content, images and embedded JavaScript code, using PDFMiner. If JavaScript is found, another tool is used to make the code easier to understand so the system can figure out what it does. Next, the system cleans and organizes the collected information to get it ready for a machine learning model. This model used machine learning algorithms like Scikit Learn and YOLOv5 to study the data and make predictions about the PDF's behavior. The model is trained using known datasets such as PDFRep, Contagio, Govdocs, Virushare, and the CIC Dataset to help it recognize harmful files.The model then gives a result showing whether the PDF is safe or malicious. It shows a confidence score which tells how sure the system is about its answer. The system also provides extra information like the features found, any scripts, and text from images.

Table 1
Extracted Features of PDF

Feature Name	Description
Text Length	Total number of characters in the PDF file
Text Entropy	The entropy of the text in the PDF file
Font Size Variance	Variance of font sizes used in the PDF file
Font Style Count	Number of different font styles used in the PDF file
Image Count	Number of images in the PDF file.
JavaScript Length	Total number of characters in the JavaScript code
JavaScript Entropy	Entropy of the JavaScript code
Obfuscated Code	Obfuscated code in the JavaScript
PDF Version	PDF version used to create the file
Creation Date	Creation date of the PDF file
Modification Date	Modification date of the PDF file
Author	Author of the PDF file
Page Count	Number of pages in the PDF file
Page Size	Size of each page in the PDF file
Attachment Count	Number of attachments in the PDF file
Attachment Type	Type of each attachment
Encryption Presence	Presence of encryption in the PDF file
Encryption Type	Type of encryption used
API Calls	Total number of API calls made by the PDF file
API Call Frequency	Frequency of API calls made by the PDF file
URLs	URLs from the PDF file

The table describes the features extracted from PDF files, which are used to train the machine learning model for predictive file analysis. The features cover various aspects of the PDF file, including its textual content, font properties, image content, JavaScript code, metadata. These features can be used to identify potentially malicious or suspicious files. The JavaScript is examined for any dangerous elements including methods for disguising destructive code. If an Obfuscated code exists, it will be converted to an understandable script using JS DeObfuscator.

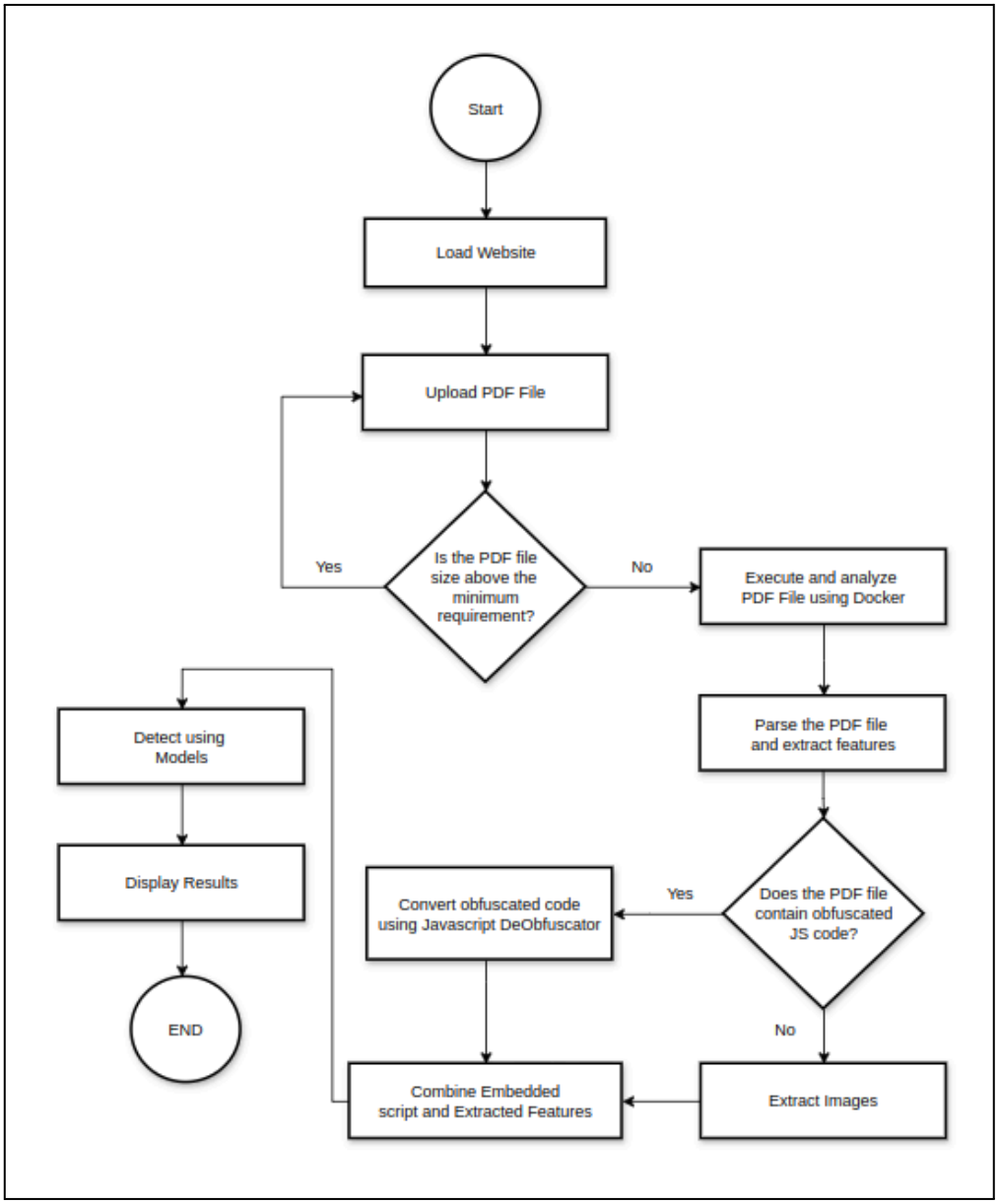


Figure 2. System Process Flow Diagram

Figure 2 which is the process flow diagram shows how to analyze PDF files for hidden JavaScript code and determine if the file is malicious or secure. It starts when the user uploads the PDF file to the website. The file is validated to ensure it meets the minimum size requirements. If the document is eligible, it will be executed first in a safe environment to investigate its behavior. After analyzing the file, it checks if there is a hidden JS code embedded in it. The detected code is converted to a readable layout using JavaScript Obfuscator for the user if it exists. The machine decides whether the PDF document is harmful or safe through the trained ensemble model.

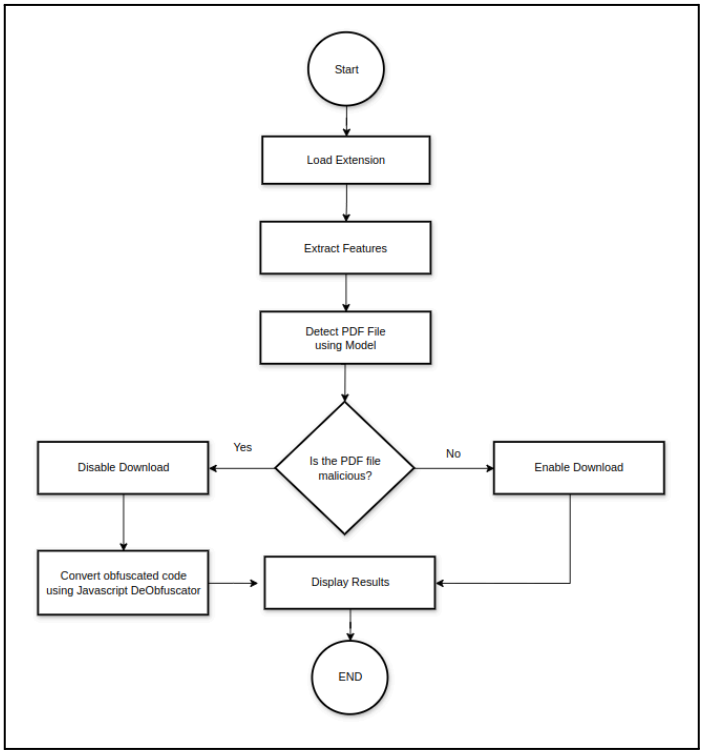


Figure 3. Browser Extension Process Flow Diagram

The flowchart shows how the system checks if a PDF file is safe or malicious in a browser extension setting. It begins by loading the browser extension which collects important details from the PDF. These details are analyzed by a model. If the PDF is safe the system allows the user to download it. If the PDF is malicious the system blocks the download and tries to convert any hidden code using a JavaScript deobfuscator. The system then displays the results

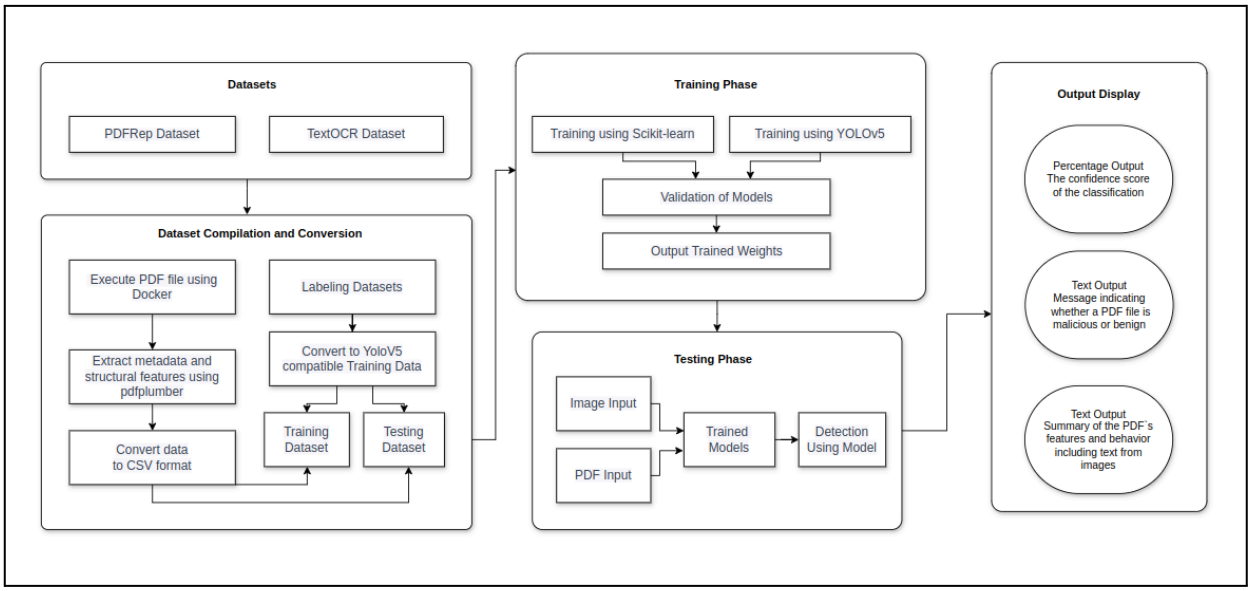


Figure 4. System Design

The system relies on two main datasets: the PDFRep Dataset which contains data related to PDF files and the TextOCR Dataset which includes text extracted from various documents using Optical Character

Recognition.. These datasets are compiled and processed to prepare them for machine learning. PDF files are executed inside Docker containers. The datasets are then labeled for supervised learning. Using the pdfplumber library, metadata and structural features are extracted from the PDF files. The data is converted into a format that works with the YOLOv5 model and also transformed into CSV format for easier handling and analysis.

During the training phase, machine learning models are trained using two approaches. The Scikit-learn library is used for traditional model training, while YOLOv5 is used for detecting objects within the datasets. Once training is complete, the models are validated to check their accuracy and performance, and the resulting trained weights are saved for future use. In the testing phase, the system can accept both image and PDF file inputs. These inputs are analyzed using the trained models to detect and classify their contents. The final output is presented in three ways: a percentage score that shows the model's confidence in its classification, a simple message stating whether the PDF is safe or malicious and a summary of the file's features and behavior including any text extracted from images.

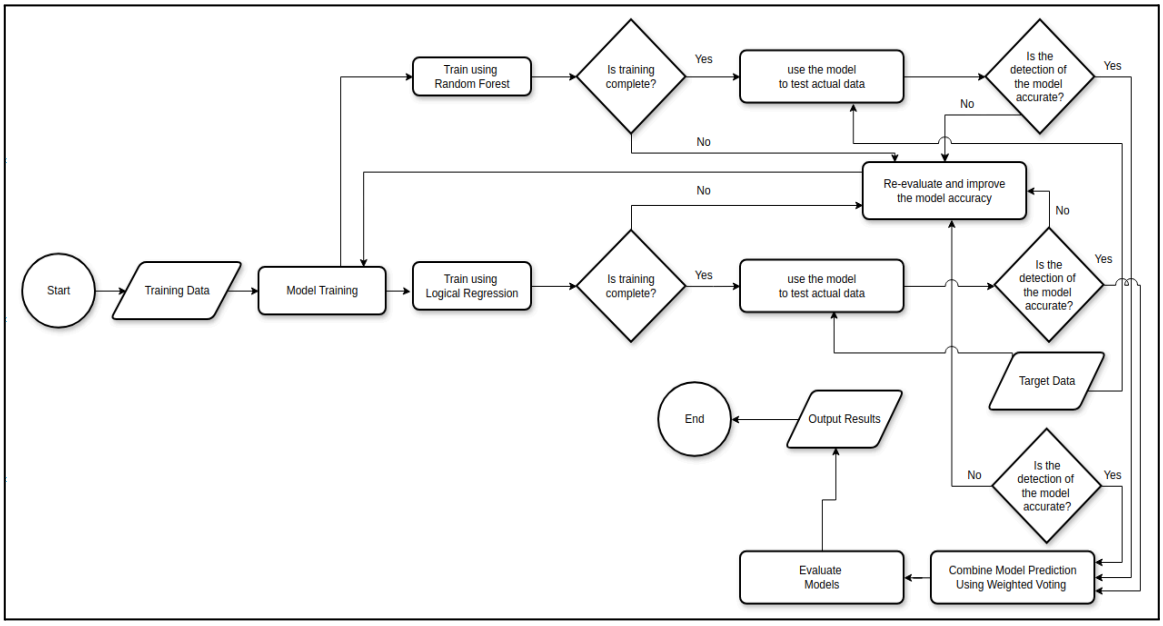


Figure 5. Scikit Learn Flowchart for Training Procedure

Figure 5 shows the flowchart for training procedure of the model. It begins with training data to train the model. Methods like Random Forest and Logistic Regression will be used. After training, a testing phase will occur to test if the results are accurate enough. If it doesn't, the researchers need to adjust and retrain the model. Once the accuracy is high enough, it is now time to predict. Predictions from different methods will be combined and get the final result.

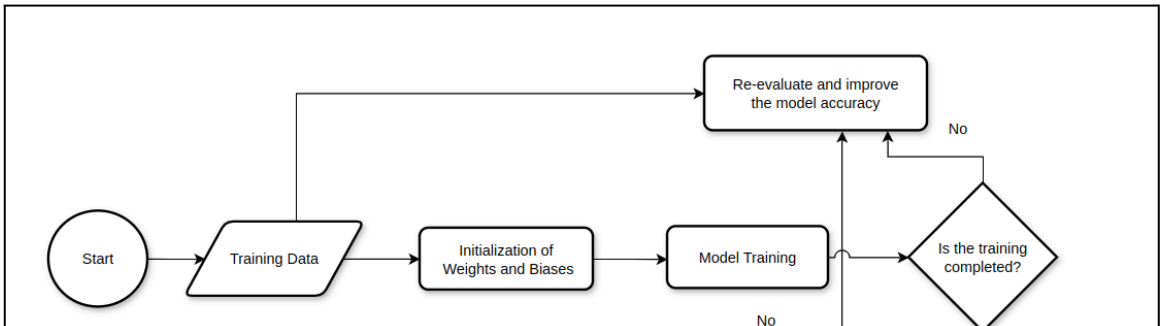


Figure 6. YOLOv5 Flowchart for Training Procedure

The process starts with training data. The system sets weights and biases to help the model learn. Then, it trains the model. It checks if the training is finished. If not, it keeps training. When training is done, the model tests real data. It checks if the model works well. If yes, it shows the results and ends. If not, it goes back to improve the model and repeats the steps.

Results and Discussions

Table 2
Results for Benign PDF Input from Random Forest, SVM and Gradient Boosting

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF Classification	SVM Benign PDF Classification	Gradient Benign PDF Classification
1	1.65	N	Y	Y
2	3.88	N	N	N
3	2.98	Y	Y	N
4	5.43	Y	Y	N
5	4.65	N	N	Y
6	2.20	N	N	N
7	3.04	Y	Y	Y
8	3.22	N	N	N

Table 2
Results for Benign PDF Input from Random Forest, SVM and Gradient Boosting

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF	SVM Benign PDF	Gradient Boosting Benign PDF

		Classification	Classification	Classification
9	4.34	Y	Y	Y
10	2.31	Y	Y	Y
11	1.91	Y	N	Y
12	4.62	Y	Y	N
13	2.66	N	Y	N
14	3.37	Y	N	N
15	5.05	N	Y	N
16	5.41	Y	N	N
17	3.87	N	Y	Y
18	4.23	Y	N	N
19	4.12	N	N	N
20	6.75	N	Y	Y
21	3.41	Y	Y	Y
22	4.40	Y	Y	N
23	1.77	N	Y	Y
24	3.96	N	N	Y
25	5.01	Y	Y	N
26	3.12	Y	Y	N
27	2.5	Y	N	Y
28	2.95	Y	Y	N
29	6.75	N	N	Y
30	4.71	N	Y	N

Average time: 3.63 seconds

The researchers tested 10, 025 records of PDF files with three different algorithms before using another dataset. These three algorithms are Random Forest, SVM and Gradient Boosting.

Benign Random Forest results for 10, 025 PDF Dataset

$Accuracy = \frac{16}{30} \cdot 100 = 53.33\%$

$ErrorRate = \frac{14}{30} \cdot 100 = 46.67\%$

$Precision = \frac{16}{16+14} \cdot 100 = 53.33\%$

Benign SVM results for 10, 025 PDF Dataset

$Accuracy = \frac{18}{30} \cdot 100 = 60.00\%$

$ErrorRate = \frac{12}{30} \cdot 100 = 40.00\%$

$Precision = \frac{18}{18+12} \cdot 100 = 60.00\%$

Benign Gradient Boosting results for 10, 025 PDF Dataset

$Accuracy = \frac{13}{30} \cdot 100 = 43.33\%$

$ErrorRate = \frac{17}{30} \cdot 100 = 56.67\%$

$Precision = \frac{13}{13+17} \cdot 100 = 43.33\%$

These files were evaluated using three different machine learning algorithms: Random Forest, Support Vector Machine and Gradient Boosting. As shown in Table 2, the performance of these algorithms varied. SVM performed the best, identifying 60% of benign PDFs while Random Forest reached 53.33%, and Gradient Boosting only achieved 43.33%. Despite SVM having the highest accuracy among the three, none of the models showed strong or consistent results, and the precision rates also indicated room for improvement.

Table 3
Results for Malicious PDF Input from Random Forest, SVM and Gradient Boosting

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF Classification	SVM Benign PDF Classification	Gradient Boosting Benign PDF Classification
1	3.63	Y	Y	Y
2	6.48	Y	Y	N
3	4.67	N	N	N
4	4.65	Y	Y	Y

Table 3
Results for Malicious PDF Input from Random Forest, SVM and Gradient Boosting

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF Classification	SVM Benign PDF Classification	Gradient Boosting Benign PDF Classification

5	4.24	N	N	Y
6	5.36	Y	Y	N
7	5.33	Y	N	N
8	6.02	Y	Y	N
9	6.42	Y	Y	N
10	6.39	Y	N	N
11	2.23	Y	N	N
12	1.81	N	Y	Y
13	5.85	Y	N	Y
14	3.94	Y	Y	Y
15	4.98	N	Y	Y
16	6.03	Y	Y	Y
17	4.86	N	N	N
18	5.32	Y	Y	N
19	1.82	N	N	Y
20	4.28	Y	N	N
21	3.37	Y	Y	N
22	1.71	Y	N	N
23	3.01	N	N	N
24	2.58	Y	Y	N
25	4.95	N	N	Y
26	4.55	Y	N	Y
27	3.29	Y	N	N

Table 3
Results for Malicious PDF Input from Random Forest, SVM and Gradient Boosting

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF Classification	SVM Benign PDF Classification	Gradient Boosting Benign PDF Classification

28	6.31	N	N	Y
29	2.45	Y	N	N
30	3.24	N	Y	N

Average time: 4.26 seconds

Malicious Random Forest results for 10, 025 PDF Dataset

$Accuracy = \frac{20}{30} \cdot 100 = 66.67\%$

$ErrorRate = \frac{10}{30} \cdot 100 = 33.34\%$

$Precision = \frac{20}{20+10} \cdot 100 = 66.67\%$

Malicious SVM results for 10, 025 PDF Dataset

$Accuracy = \frac{14}{30} \cdot 100 = 46.67\%$

$ErrorRate = \frac{16}{30} \cdot 100 = 53.34\%$

$Precision = \frac{14}{14+16} \cdot 100 = 46.67\%$

Malicious Gradient Boosting results for 10, 025 PDF Dataset

$Accuracy = \frac{12}{30} \cdot 100 = 40.00\%$

$ErrorRate = \frac{18}{30} \cdot 100 = 60.00\%$

$Precision = \frac{12}{12+18} \cdot 100 = 40.00\%$

Random Forest performed best with an accuracy of 66.67%. SVM achieved an accuracy of 46.67% but had a higher error rate of 53.34% which means more mistakes in classification. Gradient Boosting had the lowest with an accuracy of 40.00% making it the least accurate model. The average classification time across all models was 4.26 seconds. The researchers decided to use a new dataset of 400,000 records because the first dataset of 10,025 PDF files had low accuracy. Random Forest had the best accuracy at 66.67% for detecting malicious PDFs, but it still made mistakes with an error rate of 33.34%. SVM and Gradient Boosting had lower accuracies of 46.67% and 40.00%. Due to these unsatisfactory results, the researchers decided to find and use additional datasets beyond the CIC dataset. Although the CIC dataset contained over 10, 025 PDF records, its diversity and quality were limited. A dataset lacking variety in features or real-world malicious behaviors can hinder a model's ability to learn and generalize effectively. Therefore, the researchers incorporated other datasets like PDFRep, Contagio, Govdocs, Virushare, and the TextOCR Dataset to increase the range of examples and improve the model.

Table 4

Results for Benign PDF Input from Random Forest Model and Logistic Regression for 400,000 PDFs

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF Classification	Logistic Regression Benign PDF Classification
1	1.88	Y	Y
2	6.03	Y	N
3	7.21	Y	Y
4	9.84	Y	Y
5	6.44	Y	Y
6	10.84	Y	Y
7	9.05	Y	Y
8	9.88	Y	Y
9	8.1	Y	Y
10	10.42	Y	Y
11	9.3	Y	Y
12	5.60	Y	Y
13	8.04	Y	Y
14	2.85	Y	Y
15	3.0	Y	Y
16	11.92	N	Y
17	4.6	Y	Y
18	6.04	Y	Y
19	8.23	Y	Y
20	7.00	Y	Y
21	6.22	Y	Y

Table 4

Results for Benign PDF Input from Random Forest Model and Logistic Regression for 400,000 PDFs

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF Classification	Logistic Regression Benign PDF Classification

22	4.49	Y	Y
23	3.05	Y	N
24	6.04	Y	Y
25	5.67	Y	Y
26	8.6	Y	Y
27	7.2	Y	Y
28	1.53	N	Y
29	10.40	Y	Y
30	7.98	Y	Y

Average time: 6.91 seconds

The table presents the results of testing two classification models Random Forest and Logistic Regression on benign PDF documents. They were evaluated to classify benign PDFs, with the classification results presented as "Y" for correct and "N" for incorrect. Both models showed strong performance, but Logistic Regression achieved slightly higher accuracy, while Random Forest demonstrated a higher precision.

Benign Random Forest results

$Accuracy = \frac{28}{30} \cdot 100 = 93.33\%$

$ErrorRate = \frac{2}{30} \cdot 100 = 6.67\%$

$Precision = \frac{28}{28+2} \cdot 100 = 93.33\%$

The Random Forest model shows solid performance in classifying benign PDFs resulting in an accuracy of 93.33%. This model had a low error rate of 6.67%, indicating only 2 incorrect classifications. The average classification time was 6.91 seconds showing a reasonable speed.

Benign Logistic Regression results

$Accuracy = \frac{27}{30} \cdot 100 = 90\%$

$ErrorRate = \frac{3}{30} \cdot 100 = 10\%$

$Precision = \frac{27}{27+3} \cdot 100 = 90\%$

The Logistic Regression model performed well in identifying benign PDF files. It correctly classified 27 out of 30 tests, with 3 mistakes. This gives it an accuracy of 90% and an error rate of 10%. Both Random

Forest and Logistic Regression models demonstrated reliable performance in classifying malicious and benign PDF files. For malicious PDFs, Logistic Regression outperformed Random Forest with higher accuracy 86.67% compared to 80% and a lower error rate 13.33% compared to 20%. For benign PDFs Random Forest achieved slightly better accuracy 93.33% compared to 90%, while Logistic Regression maintained consistent performance across both categories. These results suggest that Logistic Regression is more effective for malicious file detection, while Random Forest excels in benign file classification. Both models can be more optimized to improve accuracy and reduce classification times.

Table 5
Results for Malicious PDF Input from Random Forest Model and Logistic Regression for 400,000 PDFs

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF Classification	Logistic Regression Benign PDF Classification
1	6.41	Y	Y
2	10.48	N	Y
3	4.5	Y	Y
4	3.76	Y	Y
5	1.42	Y	Y
6	4.34	N	Y
7	3.21	Y	N
8	7.76	Y	Y
9	3.15	Y	Y
10	1.65	Y	Y
11	7.50	Y	Y
12	4.44	N	Y
13	1.06	Y	Y

Table 5
Results for Malicious PDF Input from Random Forest Model and Logistic Regression for 400,000 PDFs

Test No.	Time (seconds, milliseconds)	Random Forest Benign PDF Classification	Logistic Regression Benign PDF Classification
14	1.22	Y	Y

15	4.55	Y	Y
16	3.4	Y	N
17	2.04	Y	Y
18	2.98	Y	Y
19	5.89	Y	Y
20	6.07	N	Y
21	6.4	Y	Y
22	10.5	N	Y
23	7.89	Y	Y
24	5.06	Y	Y
25	3.87	Y	Y
26	8.03	Y	Y
27	1.5	Y	N
28	1.87	N	Y
29	5.3	Y	Y
30	3.06	Y	Y

Average time: 4.36 seconds

Table I shows the test results of Random Forest and Logistic Regression classifiers in detecting malicious PDF files. The tests recorded the time taken for classification and the detection results Y for a correct detection and N for an incorrect detection. Random Forest had 24 correct detections and 6 incorrect detections, while Logistic Regression achieved 28 correct detections and 2 incorrect detections.

Malicious Random Forest results

$Accuracy = \frac{24}{30} \cdot 100 = 80.00\%$

$ErrorRate = \frac{6}{30} \cdot 100 = 20.00\%$

$Precision = \frac{24}{24+6} \cdot 100 = 80.00\%$

The Random Forest model correctly identified malicious PDFs in 24 out of 30 tests, giving it an accuracy of 80%. It made 6 mistakes, leading to an error rate of 20%. The model's precision was also 80%, showing it was fairly good at spotting malicious files. However, there is room for improvement in

reducing errors and speeding up detection since it took an average of 4.47 seconds to process each file

Malicious Logistic Regression results

$$Accuracy = \frac{26}{30} \cdot 100 = 86.67\%$$

$$ErrorRate = \frac{4}{30} \cdot 100 = 13.33\%$$

$$Precision = \frac{26}{26+4} \cdot 100 = 86.67\%$$

The Logistic Regression model performed better, correctly identifying malicious PDFs in 26 out of 30 tests. This gave it an accuracy of 86.67% and an error rate of just 13.33%. Its precision was also 86.67%.

Results for Image Text Input with white background for OCR Model

$$Accuracy = \frac{29}{30} \cdot 100 = 96.67\%$$

$$ErrorRate = \frac{1}{30} \cdot 100 = 3.33\%$$

$$Precision = \frac{29}{29+1} \cdot 100 = 96.67\%$$

The OCR model demonstrates strong performance in extracting text from images with white backgrounds. With an accuracy of 96.67%, the model produced only 1 incorrect classification out of 30 samples resulting in a low error rate of 3.33%. The model maintains a high precision of 96.67%. The average classification time across all samples was 4.49 seconds

Results for Image Text Input with complex background for OCR Model

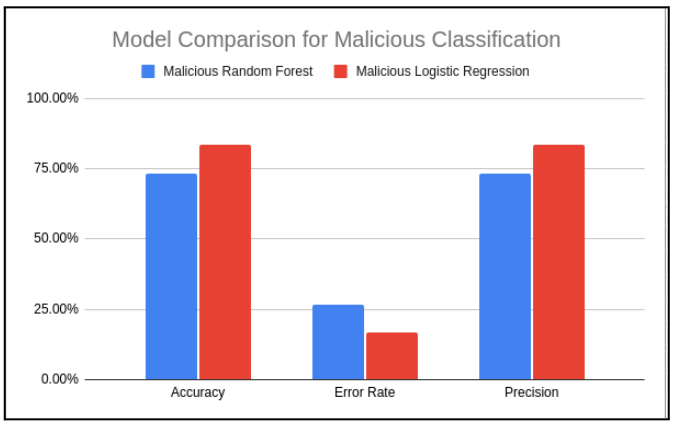
$$Accuracy = \frac{27}{30} \cdot 100 = 91.73\%$$

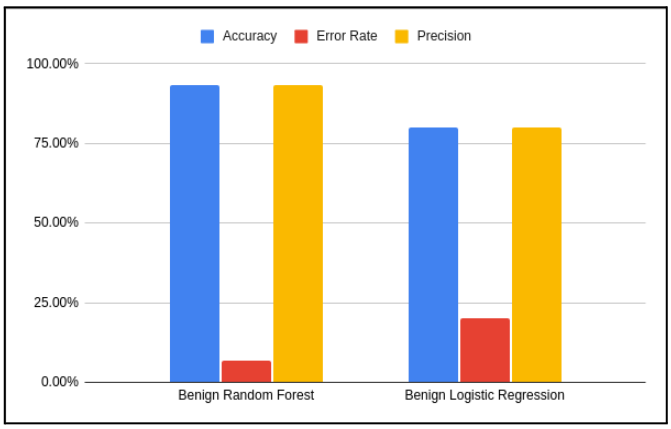
$$ErrorRate = \frac{3}{30} \cdot 100 = 8.33\%$$

$$Precision = \frac{27}{27+3} \cdot 100 = 91.73\%$$

The OCR model performs well on images with complex backgrounds, achieving 91.73% accuracy. It made 3 errors out of 30 samples, resulting in an 8.33% error rate. The model's 91.73% precision shows its ability to extract text reliably. The average classification time was 6.57 seconds.

User Testing





Random Forest achieved an accuracy of 73.33%. Despite correctly identifying the majority of malicious files, it still misclassified 8 files, indicating that the model's performance drops for malicious document detection while Logistic Regression performed better for detecting malicious PDFs, with an accuracy of 83.33%. It made only 5 mistakes, showing it is more effective than Random Forest for malicious document classification. Random Forest demonstrated excellent performance in classifying benign PDF files, achieving an accuracy of 93.33% with only 2 incorrect classifications while Logistic Regression showed decent performance, with an accuracy of 80%, but it had a slightly higher error rate compared to Random Forest for benign file classification.

Malicious Random Forest Results

$$Accuracy = \frac{22}{30} \cdot 100 = 73.33\%$$

$$ErrorRate = \frac{8}{30} \cdot 100 = 26.67\%$$

$$Precision = \frac{22}{22+8} \cdot 100 = 73.33\%$$

Malicious Logistic Regression Results

$$Accuracy = \frac{25}{30} \cdot 100 = 83.33\%$$

$$ErrorRate = \frac{5}{30} \cdot 100 = 16.67\%$$

$$Precision = \frac{25}{25+5} \cdot 100 = 83.33\%$$

Malicious Random Forest Results

$$Accuracy = \frac{28}{30} \cdot 100 = 93.33\%$$

$$ErrorRate = \frac{2}{30} \cdot 100 = 6.67\%$$

$$Precision = \frac{28}{28+2} \cdot 100 = 93.33\%$$

Malicious Logistic Regression Results

$$Accuracy = \frac{24}{30} \cdot 100 = 80\%$$

$$ErrorRate = \frac{6}{30} \cdot 100 = 20\%$$

$$Precision = \frac{24}{24+6} \cdot 100 = 80\%$$

The testing results show that Random Forest excels at benign file classification with a high accuracy of 93.33%, while Logistic Regression proves more reliable for detecting malicious files with an accuracy of 83.33%. Random Forest struggles with a lower accuracy (73.33%) for malicious files, making Logistic Regression a better choice for malicious PDF classification.

Conclusion

This study successfully developed a machine learning model capable of detecting both malicious and benign PDFs focusing on identifying obfuscated JavaScript code. The integration of Random Forest and Logistic Regression models demonstrated strong performance in classifying PDFs, with Logistic Regression proving slightly more effective in detecting malicious files, particularly those with obfuscated code. The dynamic analysis framework using Docker further enhanced the system by simulating PDF behavior during execution. Overall, the combination of machine learning methods and dynamic analysis

allowed for a comprehensive and effective approach to detecting malicious PDF files.

Recommendations

- The researchers suggest further improvement of the models, especially Logistic Regression, to increase accuracy and reduce mistakes. This could involve adjusting settings in the model (hyperparameters) and adding more relevant features for better prediction.
- The researchers suggest focusing on making the models process PDF files faster to ensure the detection system works efficiently in real-time situations.
- The researchers suggest that future work could involve using advanced techniques like deep learning to improve detection accuracy and speed,especially for more complex types of obfuscated threats.
- It would be helpful to develop a plugin or API that allows this detection system to integrate with existing antivirus programs or PDF tools used by businesses and educational institutions for wider use.

References

[1] H. C. B. Y. Q. S. R. &. H. M. Wang, A review of deep learning based malware detection techniques, https://doi.org/10.1016/J.NEUCOM.2024.128010, 2024.

[2] Understanding Adobe PDF. (n.d.), https://helpx.adobe.com/incopy/using/pdf.html, 2021.

[3] T. S. Dutta, Hackers Attacking Online Ticket Booking Users Using PDF Files, https://cybersecuritynews.com/pdf-malware-attack/, 2024.

[4] I. Arghire, Attackers Use Steganography to Obfuscate PDF Exploits - SecurityWeek, https://www.securityweek.com/attackers-use-steganography-obfuscate-pdf-exploits/, 2019.

[5] M. D. M. H. &. I. M. Ijaz, Static and Dynamic Malware Analysis Using Machine Learning. Proceedings of 2019 16th International Bhurban Conference on Applied Sciences and Technology, https://doi.org/10.1109/IBCAST.2019.8667136, 2019.

[6] A. Spadafora, Hackers are now hiding malicious Word documents in PDFs — how to stay safe | Tom's Guide, https://www.tomsguide.com/news/hackers-are-now-hiding-malicious-word-documents-in-pdfs-how-to-stay-safe, 2023.

[7] S. d., Why they are such a popular attack vector, https://safesendsoftware.com/pdfs-why-they-are-such-popular-attack-vector, 2023.

[8] H. K. Saxena, A. K. Singh and V. Sharma, Docker for Multi-containers Web Application, https://www.researchgate.net/publication/340888345_Docker_for_Multi-containers_Web_Application, 2020.

[9] E. D. Gennatas, J. H. Friedman, L. H. Ungar and G. Valdes, Expert-augmented machine learning, https://www.pnas.org/doi/abs/10.1073/pnas.1906831117, 2020.

[10] L. Breiman, Random Forests, 2001: https://link.springer.com/article/10.1023/A:1010933404324.

[11] C. Cortes, G. DeSalvo, C. Gentile, M. Mohri and N. Zhang, Adaptive Region-Based Active Learning, https://arxiv.org/abs/2002.07348?utm_source=chatgpt.com, 2020.

[12] B. K. K. H. K. Y. J. K. H. H. K. T. Y. &. L. H. J. Ndibanje, Cross-Method-Based Analysis and Classification of Malicious Behavior by API Calls Extraction, https://doi.org/10.3390/APP9020239, 2019.

[13] A. Eboka and A. O. Adimabua, Signature-Based Malware Detection Using Approximate Boyer Moore String Matching Algorithm, https://www.researchgate.net/publication/340941735_Signature-Based_Malware_Detection_Usin

g_Approximate_Boyer_Moore_String_Matching_Algorithm, 2019.

[14] A. AlMahadeen and m. alkasassbeh, PDF Malware Detection Using Machine Learning, https://www.preprints.org/manuscript/202301.0557/v1?utm_source=chatgpt.com, 2023.

[15] D. N. Jens Müller, Processing Dangerous Paths – On Security and Privacy of the Portable Document Format, https://www.ndss-symposium.org/ndss-paper/processing-dangerous-paths-on-security-and-privacy-of-the-portable-document-format/, 2024.

[16] T. D. Nicolas Fleury, PDF-Malware: An Overview on Threats, Detection and Evasion Attacks, https://www.researchgate.net/publication/353510544_PDF-Malware_An_Overview_on_Threats_Detection_and_Evasion_Attacks., 2021.

[17] Maryam Issakhani, PDF Malware Detection based on Stacking Learning, https://pdfs.semanticscholar.org/c4e6/1e9545951bf4e7dbefd7796b6f7f050a75f6.pdf, 2022.

[18] S. T. Priyansh Singh, Malware Detection in PDF and Office Documents: A survey, https://www.researchgate.net/publication/339269283_Malware_Detection_in_PDF_and_Office_Documents_A_survey, 2020.

[19] P. Priya and H. P, PDF Malware Detection System based on Machine Learning Algorithm, https://ieeexplore.ieee.org/abstract/document/10029209, 2022.

[20] Y. M. Raphael Fettaya, Detecting malicious PDF using CNN, https://arxiv.org/abs/2007.12729., 2020.

[21] H. K. Singh, J. P. Singh and A. S. Tewari, Static Malware Analysis Using Machine and Deep Learning, https://link.springer.com/chapter/10.1007/978-981-19-0604-6_41, 2022.

[22] N. S.-P. Jianguo Jiang, DETECTING MALICIOUS PDF DOCUMENTS USING SEMI-SUPERVISED MACHINE LEARNING, https://link.springer.com/chapter/10.1007/978-3-030-88381-2_7, 2021.

[23] B. Maundrill, . PDF Malware on the Rise, Used to Spread WikiLoader, Ursnif and DarkGat - Infosecurity Magazine, https://www.infosecurity- magazine.com/news/pdf-malware-on-the-rise/, 2024.

[24] A. Kamboj, P. Kumar, A. K. Bairwa and S. Joshi, Detection of malware in downloaded files using various machine learning models, https://www.sciencedirect.com/science/article/pii/S111086652200072X, 2023.

[25] S. A. Giada Stivala, From Attachments to SEO: Click Here to Learn More about `Clickbait PDFs, https://arxiv.org/abs/2308.01273., 2023.

[26] C.-M. Hsu, C.-C. Yang, H.-H. Cheng, P. E. Setiasabda and J.-S. Leu, Enhancing File Entropy Analysis to Improve Machine Learning Detection Rate of Ransomware, https://ieeexplore.ieee.org/abstract/document/9541344, 2021.

[27] A. F. M. Agarap, Towards Building an Intelligent Anti-Malware System: A Deep, https://arxiv.org/pdf/1801.00318, 2019.

[28] H. T. Neprash, C. C. McGlave and D. A. Cross, Trends in Ransomware Attacks on US Hospitals, Clinics, and Other Health Care Delivery Organizations, https://jamanetwork.com/journals/jama-health-forum/fullarticle/2799961, 2022.

[29] Z. Bala, F. U. Zambuk, B. Y. Imam, A. Y. Gital, F. Shittu, M. Aliyu and M. L. Abdulrahman, Transfer Learning Approach for Malware Images Classification on Android Devices Using Deep Convolutional Neural Network, https://www.sciencedirect.com/science/article/pii/S1877050922017185, 2022.

[30] S. Y. Yerima and A. Bashar, Explainable Ensemble Learning Based Detection of Evasive Malicious PDF Documents, https://www.mdpi.com/2079-9292/12/14/3148, 2023.

[31] D.-S. Jung, S.-J. Lee and I.-C. Euom, ImageDetox: Method for the Neutralization of Malicious Code Hidden in Image Files,

https://www.researchgate.net/publication/365576124_ImageDetox_Method_for_the_Neutralization_of_Malicious_Code_Hidden_in_Image_Files, 2020.

[32] K. K. 1. P. 3. a. S. 1. Gwanghyun Ahn 1, Malicious File Detection Method Using Machine Learning and Interworking with MITRE ATT&CK Framework, https://www.mdpi.com/2076-3417/12/21/10761, 2022.

[33] Ran Liu, Charles Nicholas, December 2023, "PdfRep", IEEE Dataport, doi: https://dx.doi.org/10.21227/7hnq-dc32..