Princess Sumaya جامعـــة
University الأميـرة سميّـة
for Technology للتكنولوجيا

Princess Sumaya University for Technology

King Hussein School for Computing Sciences

Project Title:

PDF Malware Analysis

Prepared by:

Sara Abbas 20190716
Maisa Bilal 20190142
Kholoud Qubbaj 20190116
Abdulrahman Shabaneh 20190546

Software Engineering Course Project

Tutor Name: Dr.Ammar Odeh

Spring Semester - 2022/2023

Github repository link which includes the source code of our system:
https://github.com/Sarabbas/PDFimgMalwareAnalysis.git

List of Abbreviations

PDF: Portable Document Format.
CIA: Confidentiality, Integrity, Availability.
MFO: Moth-Flow-Optimization.
DBN: Deep Belief Network.
RBM: Restricted Boltzmann Machine.
SVM: Support Vector Machines.
MLP: MultiLayer-Perceptron.
LR: Logistic Regression.
RF: Random Forest.
SVC: Support Vector Classifier.
DT: Decision Tree.
SGB: Stochastic Gradient Boosting.
RF: Random Forest.
IWO: invasive Weed Optimization.
LSTM: Long Short Term Memory.
IWO-S-LSTM: Invasive Weed Optimization with Stacked Long Short Term Memory.
ML: Machine learning.
SIFT: Scale-Invariant Feature Transform.
ANN: Artificial Neural Networks.
CNN: Convolution Neural Networks.
ReLU: Rectified Linear Unit.
LBP: Local Binary Pattern.
KNN: K-Nearest Neighbor.
IEEE: Institute of Electrical and Electronics Engineers.
UI: User Interface.
FR: Functional Requirements.
NFR: Non-Functional Requirements.
VGG: Visual Geometry Group.
TP: True Positive.
TN: True Negative.
FP: False Positive.
FN: False Negative.
MPD: Makrov Plot Diagram
SFCM: Space Filling Curve Mapping

Table of Contents

Table of Figures

1. Phase One: Problem Definition

1.1 Introduction

On May 5th 2000, a malware called ILOVEYOU disrupted the world. The file looks to be from a secret admirer, but once opened, the full file extension is revealed. It's a visual basic script that uses an exploit in early computer systems that allow it to run system code simply by being opened.

Once the damage is done, the worm enters outlook and scans for emails then sends itself using the originator's email address. In total, the estimated damages were said to exceed 7.8$ billion in wipe files and another 15$ billion for cleanup.

Knowing how a malware code works is the foundation for developing effective protection and detection technologies because it helps you understand the context that the malware seeks to reach, the intended audience of the attack, the gathered information, and the usage and destination of this malware.

When discussing Malware Analysis, we can say that there are two approaches for analysis, Static Analysis and Dynamic Analysis.
Static Analysis - which is frequently the initial stage in malware analysis, studies the code or structure of a malware program to understand its function.
Dynamic analysis (known widely as "runtime" analysis) - is primarily dependent on behavior, it creates information that may aid in understanding the cyber danger in issue, supporting intelligence development from a heuristic that permits the identification of the artifact and enhancing detection efficacy.

1.1.1 Introduction for PDF files

PDF files extend the process since malicious code can be encoded and compressed inside the file's streams, all the while, these files are routinely being utilized for legal business in companies, both internally and externally. Furthermore, alert fatigue might lead to missed alarms and increase the time it takes to evaluate incoming files.

The PDF structure is hierarchical, with four major parts:
1. **Header** - Identifies the PDFs version number.
2. **The body** - The section of the document that contains all of the information, including text and other components such as images, links, and so on.
3. **Cross-reference table** - Specifies the offset from the beginning of the file to each item in the file, allowing the PDF reader to locate them without having to load the entire document.
4. **Trailer** - Provides information about the cross-reference table so that the PDF reader can locate the table and other items.

Our end goal for this project is to be able to detect and analyze malicious PDF files which can be misused to deliver malwares.

1.2 System Description

This paper's proposed system is a machine learning-based approach to analyzing PDF files by extracting their physical visual features, then classifying the PDF files as malicious or benign. The general system pipeline consists of (1) user input: which is the suspected PDF file along with the chosen methodology of machine learning-based detection model; either KNN, SVM, Decision tree, or RandomForest (2) data pre-processing: ranging from getting image dimensions removing noise,

transforming the image to byteplot (3) feature engineering: extracting and selecting the image features of the PDF file, then (4) classification: finally inputting these features into a classifier which differentiates malicious PDFs from non-malicious ones and (5) user interface and result display: outputting the result for the user.

The system consists of three main subsystems which are divided based on the role chosen by the user. One is for the standard user, they can upload their PDF and get the malware classification result. Another subsystem is made for researcher users, they can upload their PDFs, and compare the results of different machine learning algorithms by choosing which algorithm they want to test. Moreover, the last subsystem was made for the data engineer, since their role allows them to add records to the database and regularly clean the data.

1.2.1 Image Analysis

This paper focuses on image properties that could lead to finding abnormal image areas aiding in malware detection. The features consist of keypoint descriptors extracted by the SIFT algorithm, usually used for object detection and pattern recognition.
The benefit of taking PDFs as images to perform malware analysis lies in the fact that a lot of PDF malware attacks mimic the structure of a benign PDF file, therefore some visual properties can lead us to a different and more effective way of malware detection.

1.2.2 Dataset Description

The dataset utilized in the proposed system is the Contagio dataset containing 10,980 malicious and 9,000 benign PDF files. This dataset was used to create byte plot images from each PDF sample. We ended up with a balanced image dataset of 15,000 files. 7,5000 of which are malicious and 7,5000 are benign.

1.3 System Purpose

PDFs are one of the most used files nowadays, people use them to share information via emails, social media, academic platforms, business, and legal correspondences, etc…. Unfortunately, there are many ways for malware to hide inside a PDF, and this implies risking computers being harmed, and data being exposed to hackers. Setting up a highly efficient system to check PDFs for malware is needed. This research aims to develop a system to detect malicious PDF files using machine learning models and feature engineering, characterized by low complexity and high accuracy. Decreasing the chances for type 2 error or false negative is also important, so using f1-score besides accuracy is needed.

1.4 Problem Statement

The losses incurred due to the malware attacks on the PDFS are very heavy to many sectors. The attackers use the malicious PDF to expose the user's confidentiality, integrity, and availability. The need to develop a detection and protection technique is a vital purpose that implements the machine learning models. This approach needs a thorough understanding of the mechanism of the malicious PDF works and its features that enable a quick and accurate process of detection and prevention to be created. In Figure [1] below a typical attack scenario is shown, where the attacker

sends an email with a malicious PDF, the user receives it and just by downloading it, the attacker will have the ability to CIA breach.

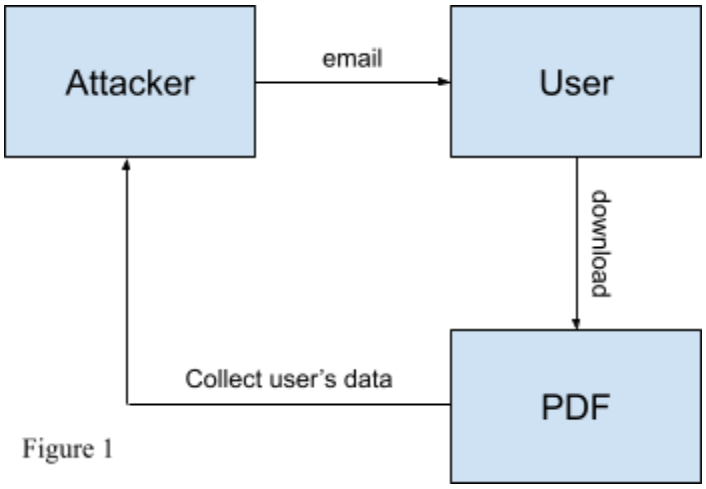

Fig [1]. Displays an Attack Scenario

1.5 The System Context View

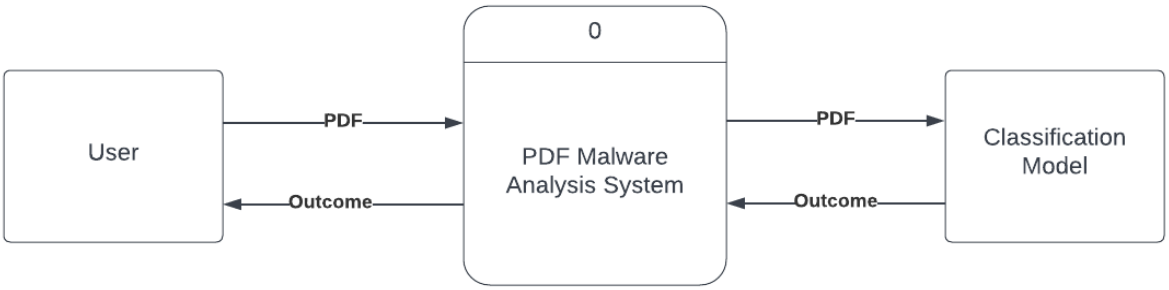

Fig [2]. Displays the system context view

1.6 Literature Review

The intersection of machine learning and cybersecurity has proven effective in many aspects, especially in malware detection. For this paper, different researches and implementations were studied to identify the best approaches and models to use for the proposed PDF malware analysis system. Some methodologies based on the structural properties of PDFs were looked into, as well as others that employed image analysis techniques to classify PDFs.

Two of the most utilized machine learning frameworks for PDF malware analysis are known as Mimicus, which works with the physical features of a PDF, and Hidost which focuses on logical features. Mimicus extracts features like metadata features, does some feature engineering to manipulate them, then generates new malware-infected PDFs and tests them against a random forest classifier. Hidost focuses on the logical file structures of PDFs, extracting absolute file path features by parsing the PDF file path. Just like Mimicus, the Hidost framework tests generated malicious PDFs on a random forest classifier.

Researchers in [1] deployed three commonly known PDF malware attacks which are Mimicry, Mimicry+, and Reverse Mimicry as white-box attacks to study the vulnerabilities of Mimicus and Hidost. A Mimicry attack mimics metadata and the physical structure of a benign PDF, while Mimicry+ is an enhancement on Mimicry. Reverse Mimicry creates new PDFs based on existing benign PDFs with malware injected into them. This allows the malicious PDFs to bypass structure-based detection models as there is a minor difference in PDF structure between the original and malicious PDFs. According to [1] findings, Mimicry, Mimicry+, and Reverse Mimicry can go undetected in Mimicus models. On the contrary, Hidost models yield effective results in recognizing Mimicry and Mimicry+ but are vulnerable to Reverse Mimicry.

Authors in this paper [2] proposed an algorithm to classify PDF files into benign and malware using an optimizable Decision Tree. The PDF samples used in this literature were from the Evasive-PDFMal2022 dataset. The authors performed some data preprocessing methods that deal with null values and duplicates. In order to train and test the model the data was split into (70%) training, (20%) testing, and (10%) validation. The decision tree was built using Adaboost optimizer, it's a boosting algorithm used in classification to increase the efficiency of the model. Based on the evaluation metrics the accuracy of the model was 98.7% after three iterations.

In this paper [3] the authors suggested a methodology to detect malware using static analysis and deploy a machine learning model to classify the data. They separated the dataset into three different datasets based on the features extracted. So many deep learning methods were used in implementing the model: Artificial neural network, convolutional neural network, SVM, and Random forest. Moreover, they used the “Adam” boosting method and ReLu as an activation function. The highest accuracy that was recorded was 98.91% when they selected 100 features from each dataset with the Random forest classifier.

Authors of [4] implemented a support vector machine model using the Contagio dataset. They used feature selection and ended up with 11 features that got them a 99.7% accuracy score. Then the authors moved on to try tricking the model using evasion attacks, using naive attacks, and gradient-descent attacks, they were able to oppose the attacks using a threshold, a smarter feature selection, both together, or adversarial learning, these techniques were able to avoid 99.99% of these attacks.

In paper [5], the authors proposed a unique algorithm for detecting malware in PDFs. They have used 9000 samples from the Contagio dataset to extract features from the files. In the processing stage, they traversed through the PDF structure and extracted significant features such as children, tags, and contents. Then they used the SimHash algorithm to compare similarities between the images, and based on the results an RGB image was generated. A dataset including the features was generated and used to train the VGG19 model. The VGG19 is a built-in model in Tensorflow that was proposed by the Visual Geometry Group from Oxford University. In conclusion, the model scored well in the performance metric since the accuracy was 97.3% and the f1-score was 97.5%.

Researchers in paper [6], proposed a visualization-based algorithm to detect malware in portable executable files.A dataset containing 1500 malicious and 12000 benign samples was used in this research. In the feature extraction phase, they used three extraction methods to generate feature sets which are Intensity-based, Gabor-based, and Wavelet-based. These features were then used in to train an SVM model and an accuracy of 95.95% was achieved when using all features in training the model.

In paper [7], the authors used 2 methods of malware visualization which are the Markov Plot Diagram and the Space Filling Curve Mapping to be used in deep convolutional neural networks to detect malware in portable executable files. The first method uses bi-gram features and their statistics as coordinates and intensity of pixels. While the other method uses curves to represent one-gram features of byte sequences. The dataset that was used to build the detection model is the BIG 2015 dataset which contained 10868 samples. Deep CNN was used to train the model and it included layers of VGG19, VGG16, and ResNet50. The best accuracy was obtained when a VGG16 network with an accuracy of 99.08%.

In this paper [8], the authors combined samples from Yahoo search engine and Contagio dataset to create the malware detection model. The PDFs were traversed and for each keyword detected the frequency was incremented the K-means was used to find the objects with the highest occurrence and the final set that was created included the object and the obfuscated version of it and for each feature set there was the corresponding vector. Many models were used in creating the system such as SVM, RandomForest, J48, and Bayesian, but the most effective model was the RandomForest with obtained accuracy of 99.88%.

Authors in paper [9], Used 12470 pe samples from the Malimg dataset. Features were extracted using multi-resolution and various types of wavelets like Gabor wavelet, discrete wavelet, GIST, and many other methods. Later on, they picked 56 significant features to train the model. Two types of machine learning algorithms were used KNN and SVM. In conclusion, the achieved accuracies by the two models were 99.84% and 99.88% respectively.

In paper [10], researchers demonstrated a unique way of PDF malware detection using PDF image properties. Keypoint descriptors like SIFT and ORB, and texture feature extraction processes like local binary patterns (LBP), local entropy, and Gabor filter were applied to a grayscale version of the PDF, all of which have been described in detail in the literature. These methodologies have been done on two different grayscale visualizations of the PDF images, Markov plots, and byte plots.

Table I: Summary of literatures on PDF Malware Analysis

Reference	Dataset	Main Methodology	Features extracted	Model(s)	Performance metric	Findings
[1]	**Contagio malware dump**	2 ML classifier frameworks Hidost & Mimicus tested against white-box attacks Mimicry,	Mimicus: physical features (135) -size -character count	Random Forest	Mimicus accuracy rates dropped by 2%, 10%, and 34% on Mimicry,	Mimicry, Mimicry+, and Reverse Mimicry can go undetected in Mimicus.

	A subset of 1,800 benign and 2,198 malicious PRA Lab 300 malicious mutated using the Reverse Mimicry technique	Mimicry+ and Reverse Mimicry.	-PDF version -PDF metadata like author name Hidost: logical features -PDF logic paths		Mimicry+ and Reverse Mimicry respectively. Hidost accuracy rate decreased by 4% for Reverse Mimicry.	Hidost recognizes Mimicry and Mimicry+ but is vulnerable to Reverse Mimicry.
[2]	**viz. Evasive-PDFMal 2022**	Data pre-processing using Matlab to remove duplicates and erroneous records. Using an ensemble technique in modeling.	Size, title characters, encryption, metadata size, page number, header, image number, text, object number, font objects, number of embedded files, and the average size of all the embedded media	Adaboost	**Accuracy:** 98.84%	Low overhead of model
[3]	From **IEEE Dataport** 48K observations	Extracting features from three parts of the data set (section header, image, and import)		- ANN - CNN	**Accuracy:** 98.91%	Extracting 100 features from each data subset achieved a high accuracy
[4]	**Contagio Dataset** A subset of 10,000 benign and 10,000 malicious	Use SVM with feature selection and use evasion attacks, naive attack and gradient-descent attack, which lead to implementing countermeasures against the attacks.		-SVM	**Accuracy:** 99.7%	
[5]	Generated dataset from the extracted features. The features were extracted from 9000 samples in the Contagio dataset.	Generate an RGB image for each PDF	-children -tags -contents	VGG19 (Visual Geometry Group)	**Accuracy**: 97.3% **F1-score:** 0.795	
[6]	A dataset containing 15000	Visualization-based approach	Three different sets of features:	-SVM	**Accuracy:** 95.95%	

	malicious samples and 12000 benign	for malware detection in PE files	Intensity-based Gabor-based Wavelet-based.		When all feature sets were used.	
[7]	BIG 2015 contains 10868 samples with 9 malware families.	2 visualization methods were used: MPD and SFCM.	Features were transformed to byte plots	CNN with VGG16 architecture	**Accuracy:** 99.8%	
[8]	A generated dataset that combined samples from yahoo search engine (25%) and Contagio dataset (75%)	PDFs were characterized using the keyword embedding and their frequency.	248 features were extracted from benign and malicious files	-Bayesian - SVM -j48 -Random forest	**Accuracy:** 99.88% Obtained by using RandomForest	
[9]	9939 samples from Malimg dataset And 3131 from Melheur dataset	Used multi-resolution and wavelets to build feature vector	56 feature vectors were used	-KNN -SVM	**Accuracy:** 98.84% 98.88%	
[10]	**Contagio** Split into two partitions. (1) 6,000 benign and 6,000 malicious for testing combinations of plot type, features, and classifiers. (2) 3,000 benign and 4,980 malicious to compare approaches with popular antivirus scanners.	Image processing	Keypoint descriptors -SIFT -ORB -BRIEF Texture features -LBP -Local entropy -Gabor filter	-Random Forests -Decision Trees -KNN	**F1 score:** 0.99 using Gabor filter on a byte plot, tested on a random forest classifier.	Methodology is robust against Reverse Mimicry attacks.

1.7 Challenges

We must take into consideration all potential issues that could cause our system to fail, therefore we must list all challenges we could face and how to avoid them. For the image preprocessing and feature extraction part, we should find a good feature extraction method that could excerpt the most useful features that could distinguish benign and malicious PDFs. Furthermore, we must make sure to choose the best deep learning approach and good architecture for the selected model, GridSearch and other hyperparameter tuning techniques could prevent us from building a model that could result in underfitting

or overfitting. Paying attention to such issues and solving them as soon as possible will help us to achieve good evaluation results in the following metrics accuracy, precision, and recall.

1.8 Projection (Gantt Chart, Network Diagram)

1.8.1 Gantt Chart

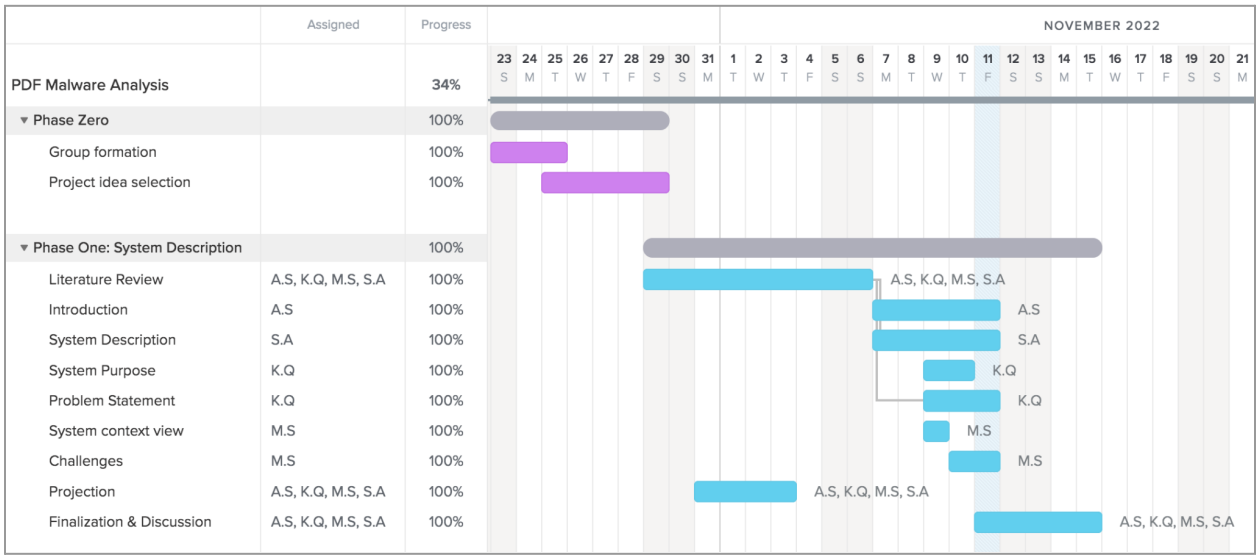

Fig [3]. Phase One Gantt chart of the proposed PDF Malware Analysis system.

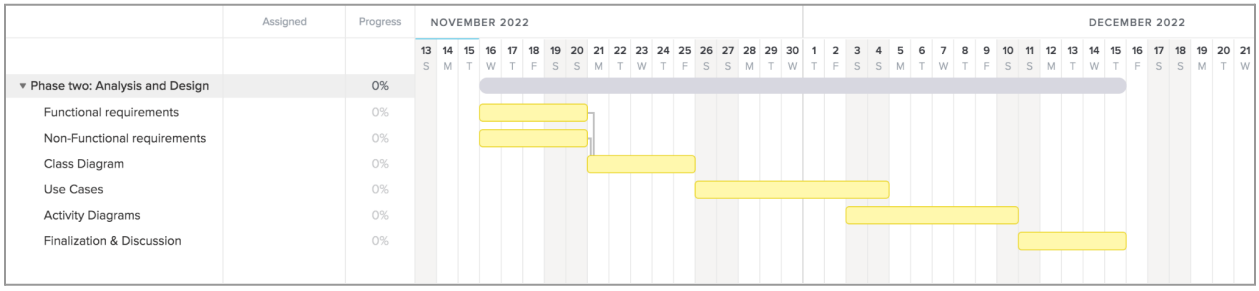

Fig [4]. Phase Two Gantt chart of the proposed PDF Malware Analysis system.

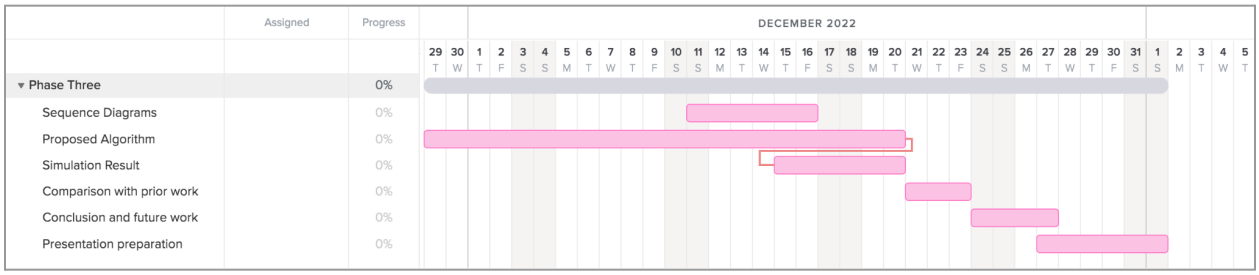

Fig [5]. Phase Three Gantt chart of the proposed PDF Malware Analysis system.

1.8.2 Network Diagrams

Table II: Project tasks and their duration.

Task	Predecessors	Duration
Introduction	-	4
System Description	Literature Review	4
System Purpose	-	1
Problem Statement	Literature Review	4
System Context view	-	1
Literature Review	-	18
Challenges	-	2
Projection	-	4
Finalization & Discussion	-	5
Functional Requirements	-	5
Non-Functional requirements	-	5
Class Diagram	Functional requirements Non-Functional requirements	5
Use Cases	-	9
Activity Diagram	-	8
Finalization & Discussion	-	5
Sequence Diagram	-	6
Proposed Algorithm	-	22
Simulation Result	Proposed Algorithm	6
Comparison with prior work	-	3
Conclusion and future works	-	4
Presentation Preparations	-	6

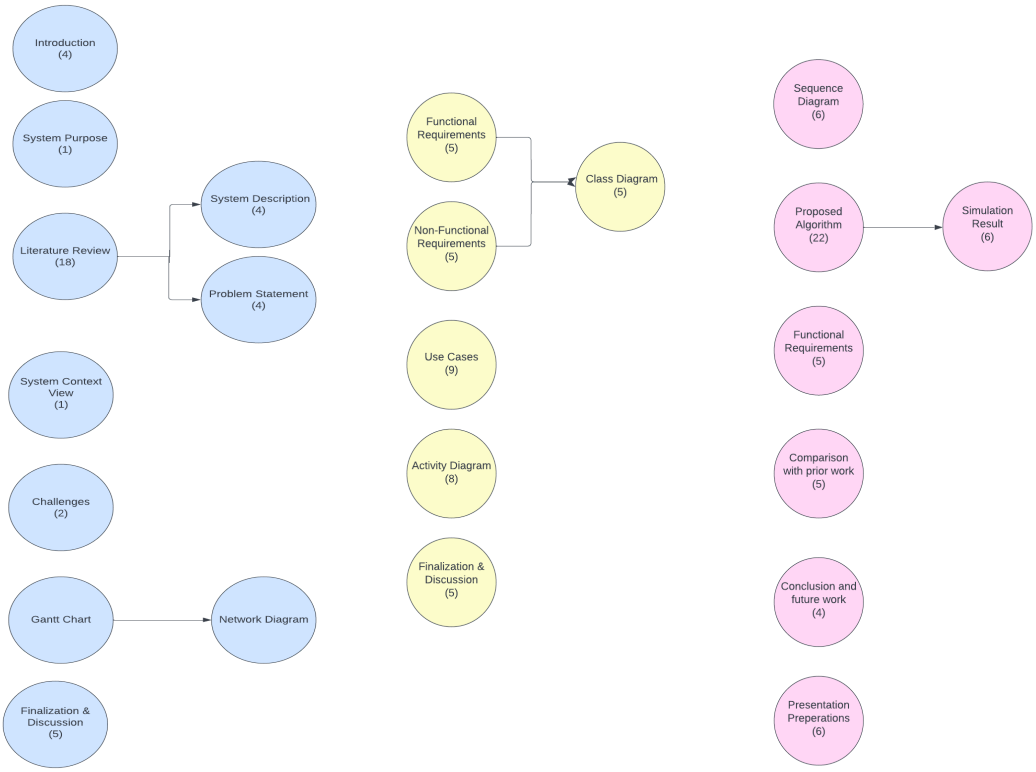

Fig [6]. Network Diagram for phases 1, 2, and 3.

2. Phase Two: Analysis and Design

2.1 Functional requirements of the System

Table III: Functional Requirements of PDF Malware Analysis System

Requirement ID	Requirement Name	Description
FR1	Sign up	Users can sign up as a standard user or a researcher with more advanced functions or a data entry person.
FR2	Login	Users verify log-in with a predefined

		username and password.
FR3	Upload document	Users upload a PDF or png file and choose whether they want to contribute their data to the system's dataset.
FR4	Feature Extraction	System extracts image properties using SIFT algorithm.
FR5	Run classification model	System inputs data into an SVM, Random Forest or a Decision tree ML algorithm.
FR6	Display classification result	System displays whether the PDF file is classified as malicious or benign
FR7	Download processed image	Users have the choice to download the processed PDF image after performing the feature extraction process.

2.2 Non-Functional properties of the System

Table IV: Non-functional Requirements of PDF Malware Analysis System

Requirement ID	Requirement Name	Type	Description
NFR1	High Accuracy	Performance	System should be able to classify PDF files as malicious or benign with a high accuracy rate.
NFR2	Low overhead	Speed	The classification process should take minimal time and

			resources to produce the result for the user.
NFR3	Data confidentiality	Security	No data breaches and unauthorized access to the PDF files contents. This is achieved by confirming users' identity with a login system and asking them whether they want to contribute their data.
NFR4	Data Backup	Reliability	Since we're working with malicious files, there should be a database backup in case these malwares crash our system or spread across the whole dataset.

2.3 Class Diagram

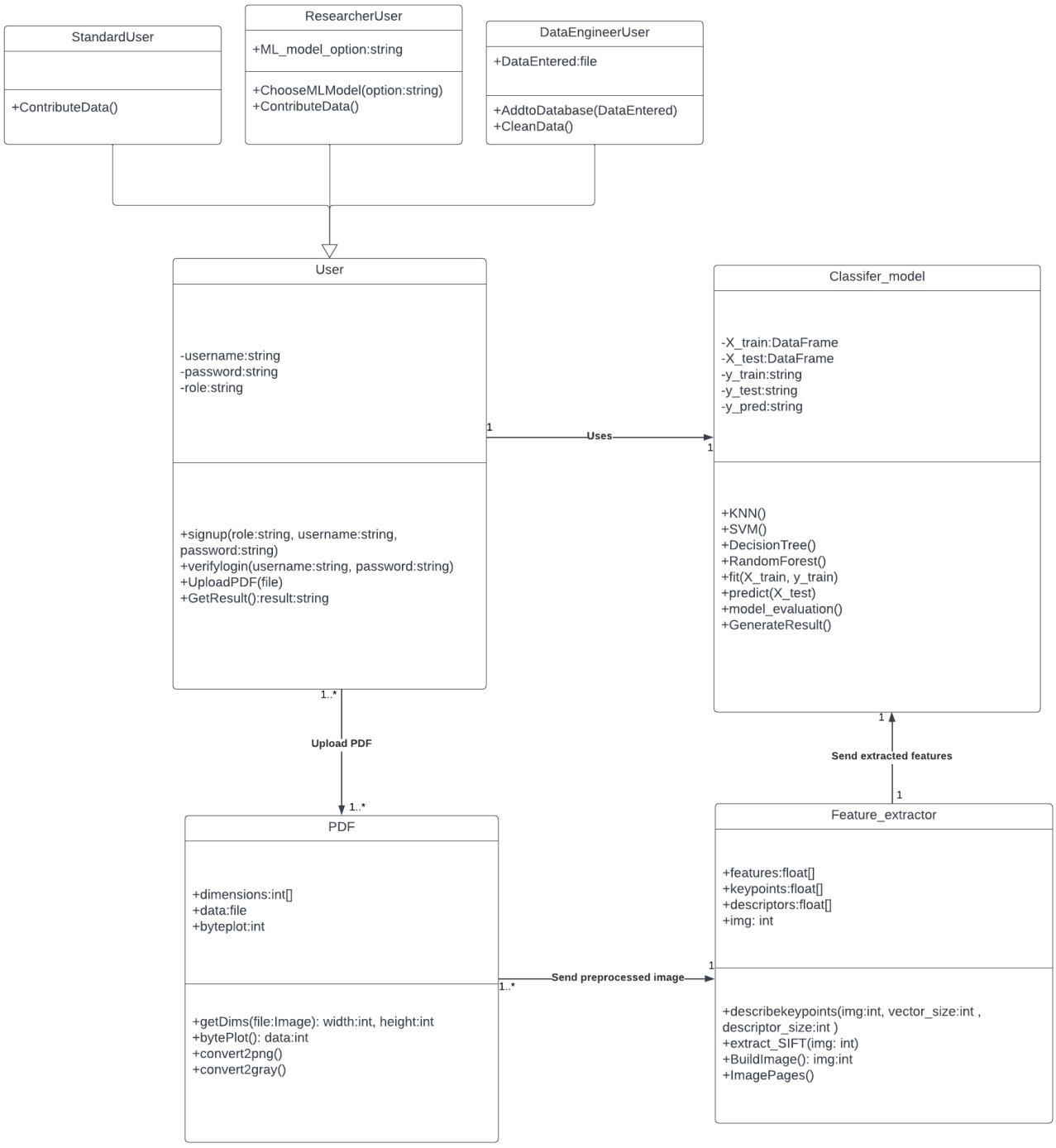

Fig [7]. Class Diagram for PDF Malware Analysis system

Figure [7] shows the relationship between User, PDF, Classifier model and Feature extractor classes, when a user logs in/signs up to the system through user class, one of three roles has to be chosen (Standard user, Researcher user, Data engineer user), each role has its own operations, a user can upload an image or a PDF, the files are sent to the feature extractor class, which will send the extracted features (key points and descriptors) to a trained machine learning model (a model out of 4 will be chosen by the user), the output then will be displayed to the user.

2.4 Use case Diagrams

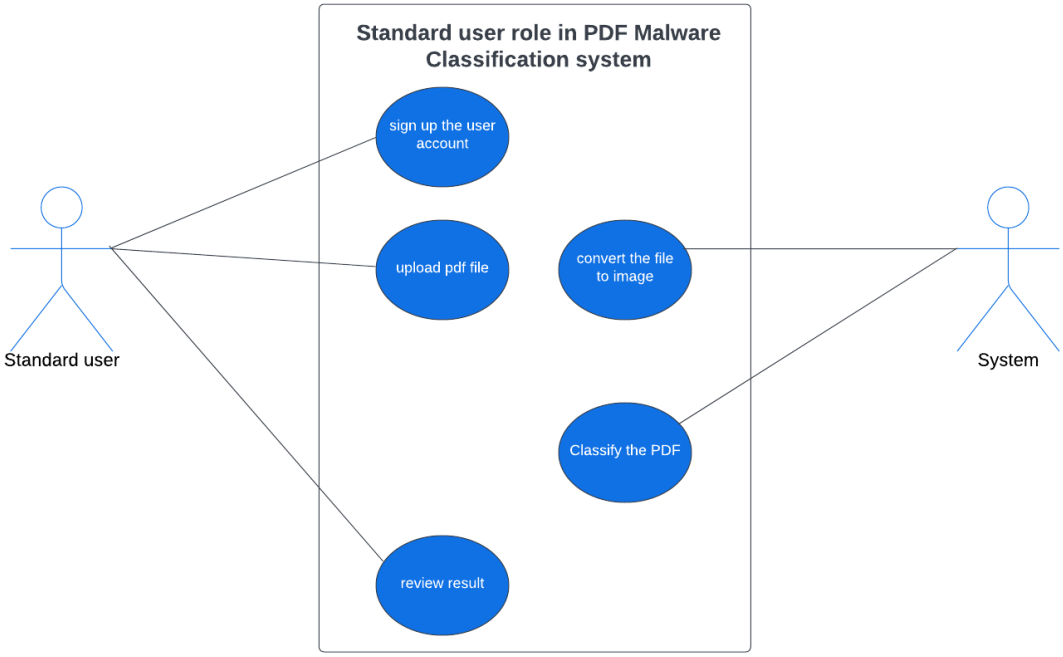

Fig [8]. Use case diagram for the standard user role

Expanded Format:

- Use case: Standard user role in the malware system
- Actors: Standard user, system.
- Purpose: Classify the PDF as malicious or benign for the user Description: The standard user enters a PDF into the system so it can detect any malware and inform the user of the result.
- Type: Primary.
- Cross-reference: FR2 FR3, FR6
- Typical course of actions:

  1- The use case begins when the standard user enters PDF into the system.

  2- The data entered by the user first undergoes preprocessing then SIFT function extracts all features.

  3- The system then is responsible to send the resulting image to the classifier

  4- Once the model is done with classifying the image a result is sent to the user.

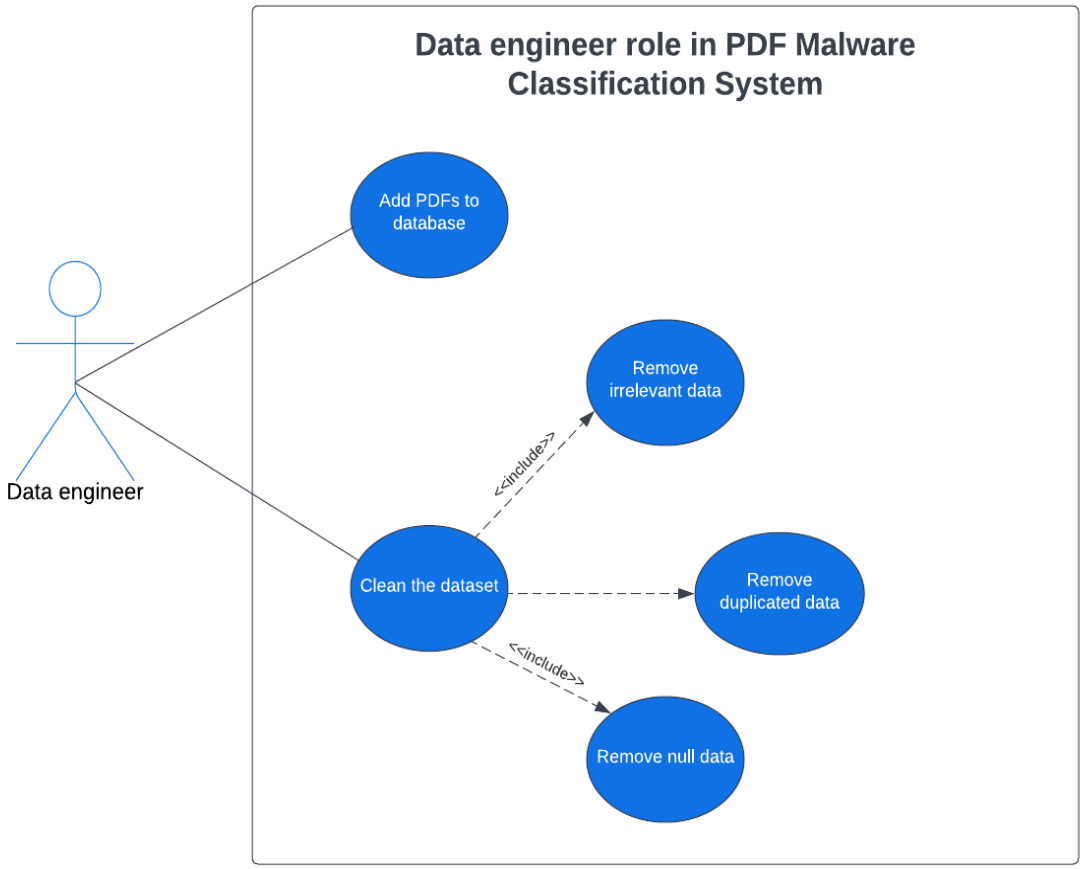

Fig [9]. Use case diagram for the data engineer user role

Expanded Format:

- Use case: Data engineer role in the malware system
- Actors: data engineer
- Purpose: Make sure the database contains useful data Description: The data engineer adds data entered by the user to the database. Processes the data using data engineering tools Type: Primary.
- Cross-reference: FR3
- Typical course of actions:

  1- The use case begins when the user contributes the data to the system.

  2- The data engineer is responsible for entering the input data into the database of the system.

  3- The Data engineer must also check that the dataset is valid by applying data preprocessing techniques such as: removing null values, irrelevant data, and duplicated data.

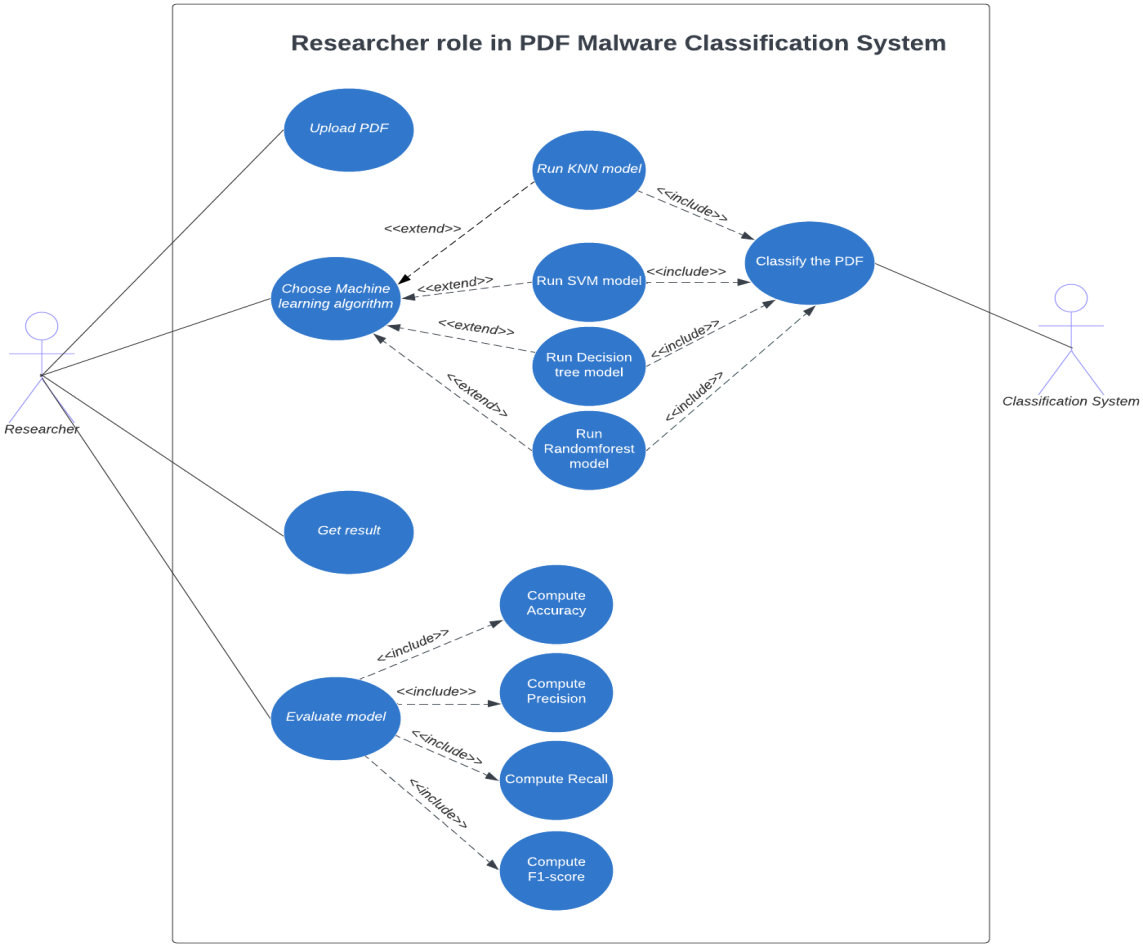

Fig [10]. Use case diagram for the research user role

Expanded Format:

- Use case: PDF malware classification for a researcher
- Actors: Research user
- Purpose: Allow the user to compare the results of different classifiers.
- Description: If the user chooses the role of a researcher they can upload the PDF and choose different machine learning algorithms
- Type: Primary
- Cross- Reference: FR3, FR4, FR5, FR6.
- Typical course of actions:

1- This use case begins when a researcher uploads a PDF and asks if it's malicious or not.

2-The machine learning model is developed by splitting data into train and test samples

3-After the processing the PDF it's transferred to the classification model and is classified.

4- After that, the model is evaluated using the following measures: Accuracy, Precision, Recall, and F1-score.

5- The generated result is shown to the user.

2.5 Activity Diagram

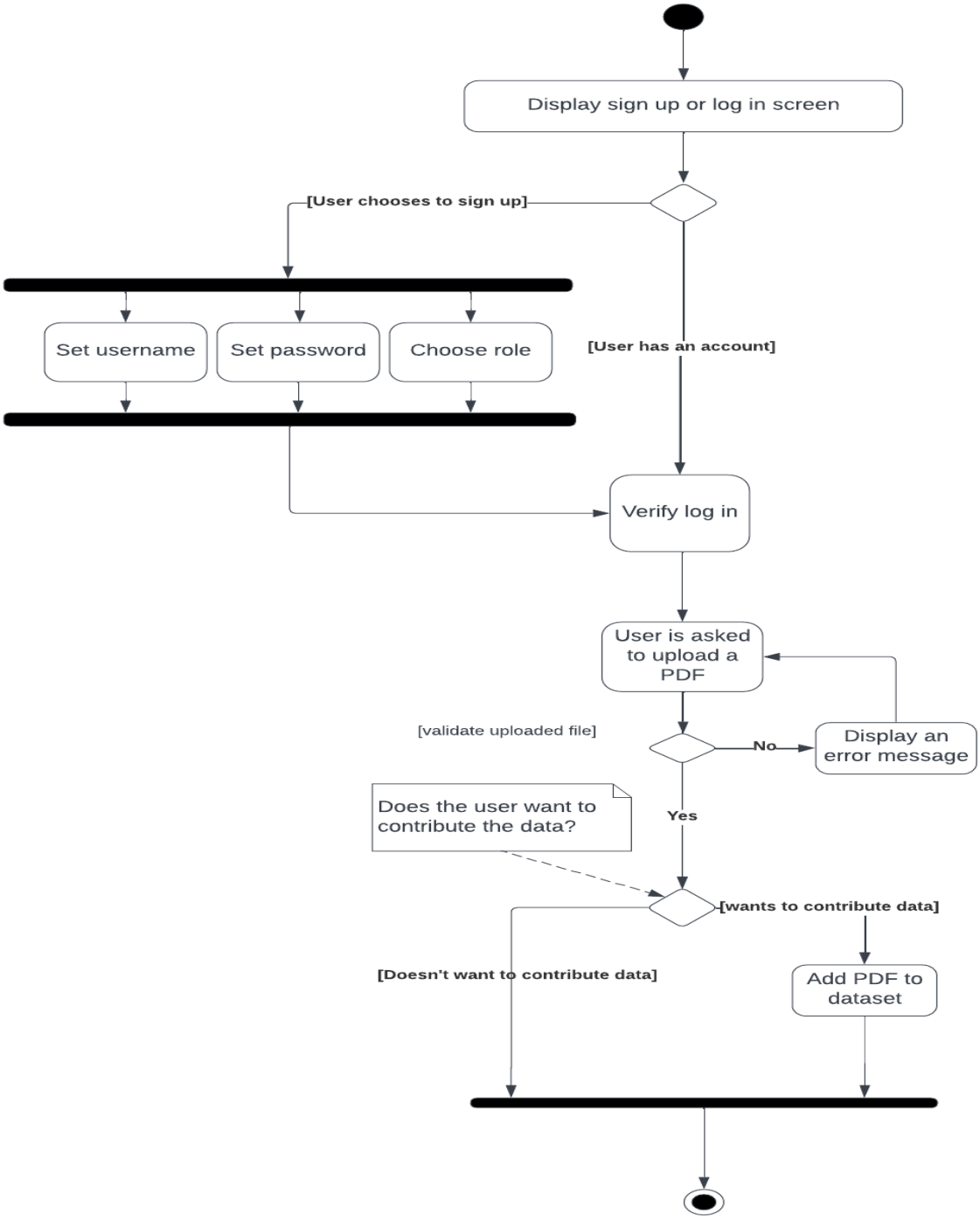

Fig [11]. Activity Diagram for sign up/login page in the user interface.

Figure [11] describes the signup and login of the user to the system. If the user doesn't have an account, they are asked to fill in their information by setting a username, password and choosing their role

as either a standard user, researcher, or data engineer. If the user has registered in the system already they are asked to verify their login details. The user then uploads a PDF as either a .PDF or .png file extension. If the file type uploaded is not supported, an error message is displayed to the user, and they're asked to upload the document again. The user is also given the choice to contribute their data to re-train the model again or not.

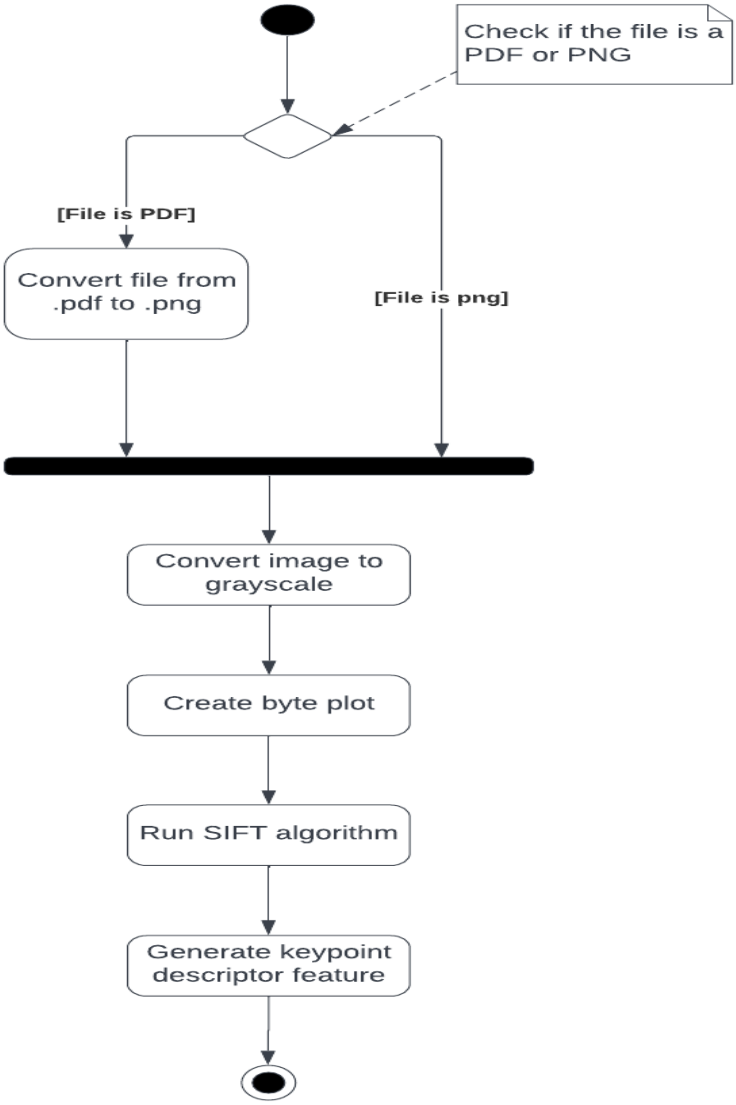

Fig [12]. Activity Diagram for image processing of a PDF file.

Figure [12] explains how the PDF files will be converted to images, once all files have a unified form they will be sent to the image preprocessing stage. The image preprocessing stage includes transforming the image to grayscale and then creating a byte plot. Lastly, SIFT Algorithm uses the byte plot generated to generate the keypoint descriptor for the features.

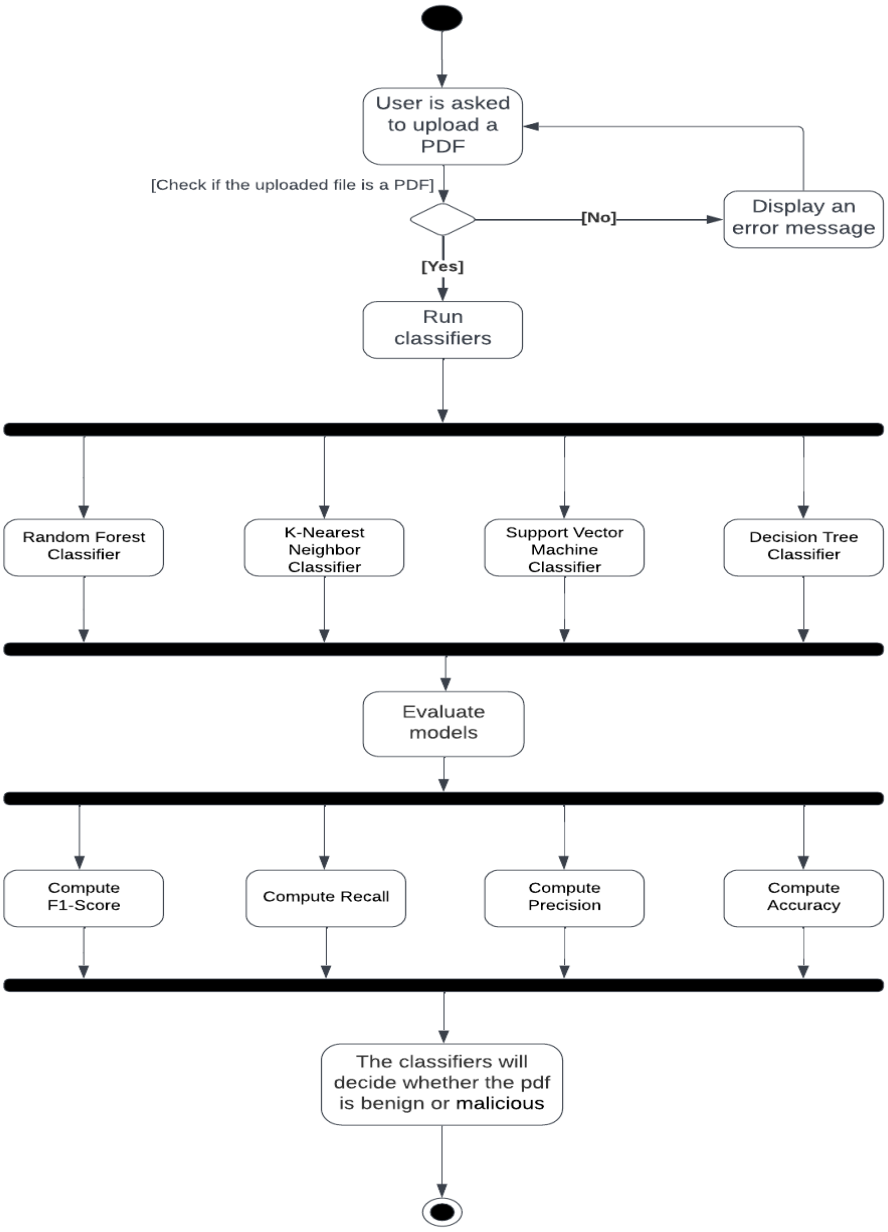

Fig [13]. Activity Diagram for ML classifiers when a standard user uploads a PDF.

Figure [13] shows how the image descriptors will be sent as input to 4 different classifiers which are: Random Forest, KNN, SVM, and Decision Tree. Later on, all models are evaluated using the following metrics: f1-score, recall, precision, and accuracy.

3. Phase Three

3.1 Sequence Diagrams

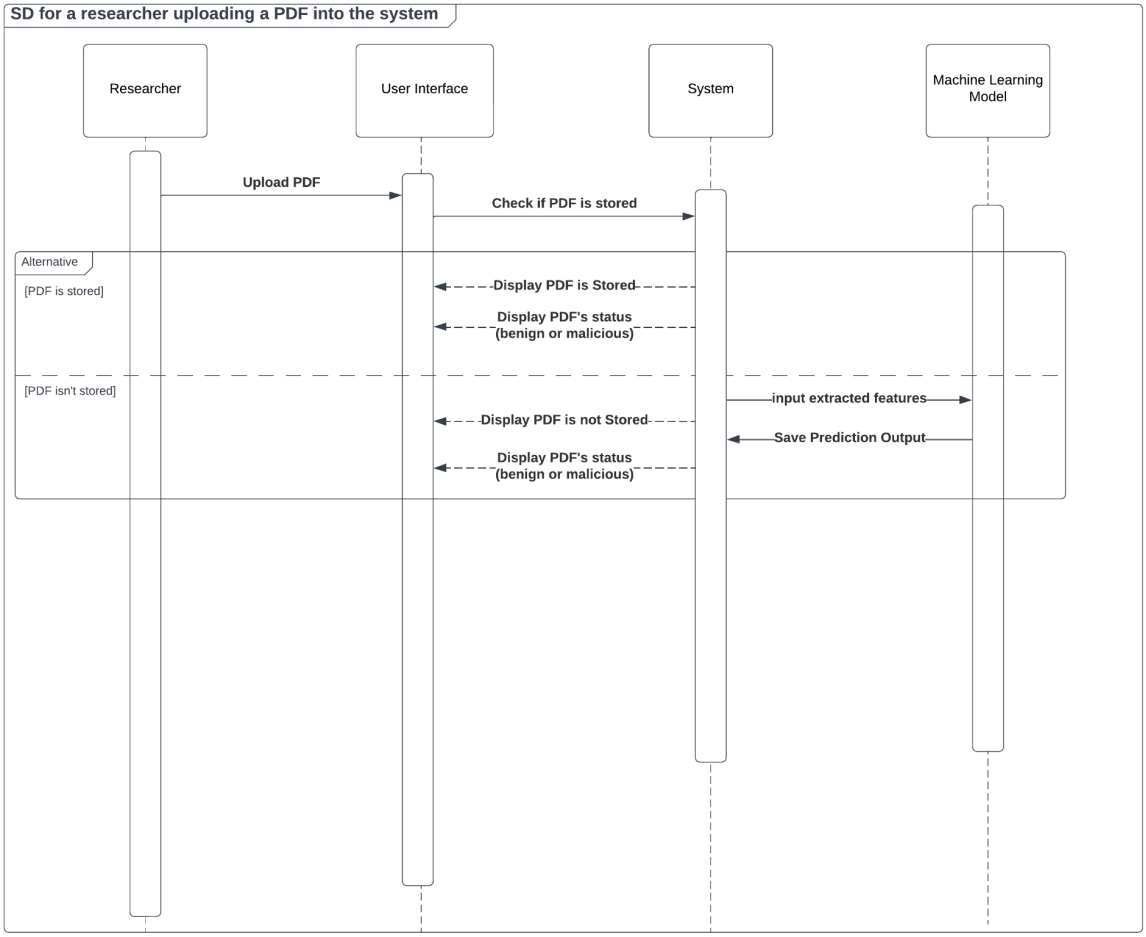

Fig [14]. Sequence Diagram for a researcher uploading a PDF into the system.

Figure [14] shows the researcher uploading a PDF into the system, our system then checks whether the uploaded PDF is stored in the system or not, if the file is stored in system's database a message will be displayed telling the researcher that the PDF is stored and it's status (benign or malicious), otherwise the system will display the PDF isn't stored, the system will then extract the PDF features and input it into one of 4 trained machine learning models , the prediction output for the PDF will be stored in the system now, and the prediction output will be displayed for the researcher.

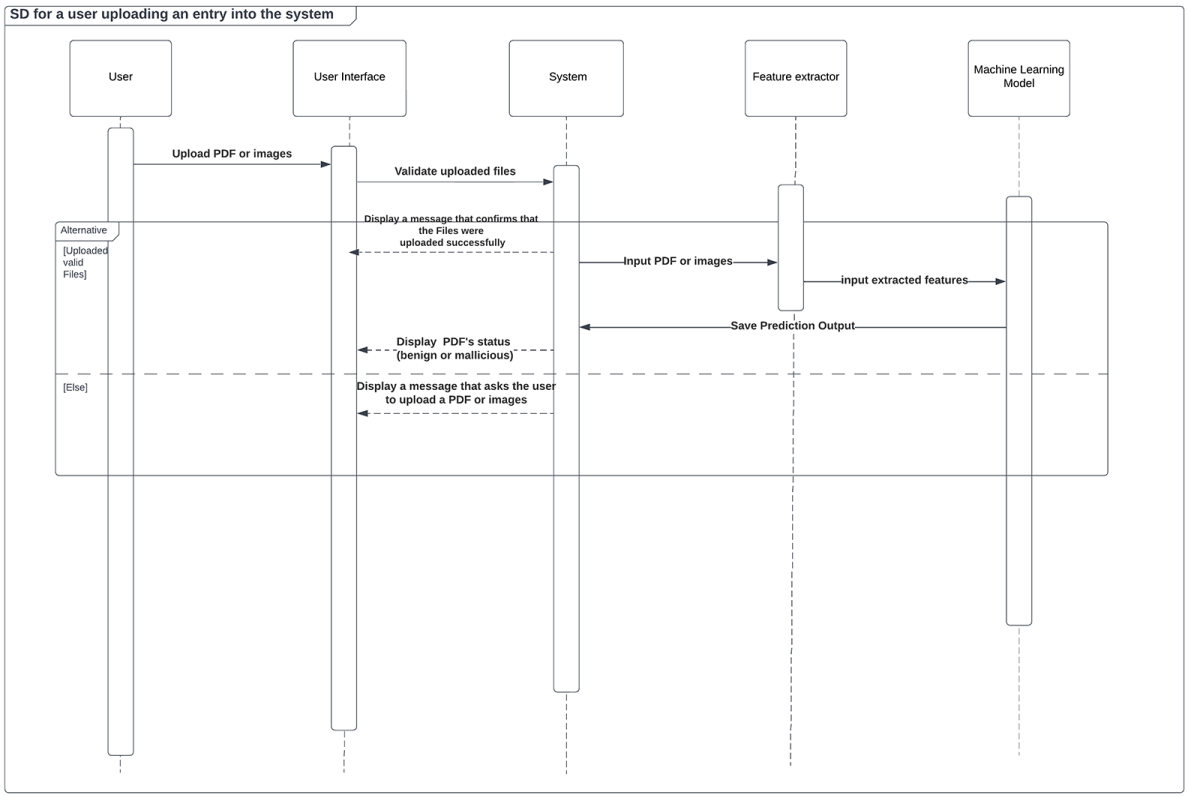

Fig [15]. Sequence diagram for a user uploading an entry into the system.

Figure [15] shows a sequence diagram for a standard user uploading a PDF or an image into the system, our system then validates the uploaded files, a message confirming that the file was successfully uploaded will be displayed for the user if the uploaded file is valid, otherwise, the system will display a message asking the user to upload a PDF or an image file only, the system will send the uploaded files to the feature extractor, in which the features extracted will then be sent to the trained machine learning model, the predicted output will be stored in the system, and then displayed to the user.

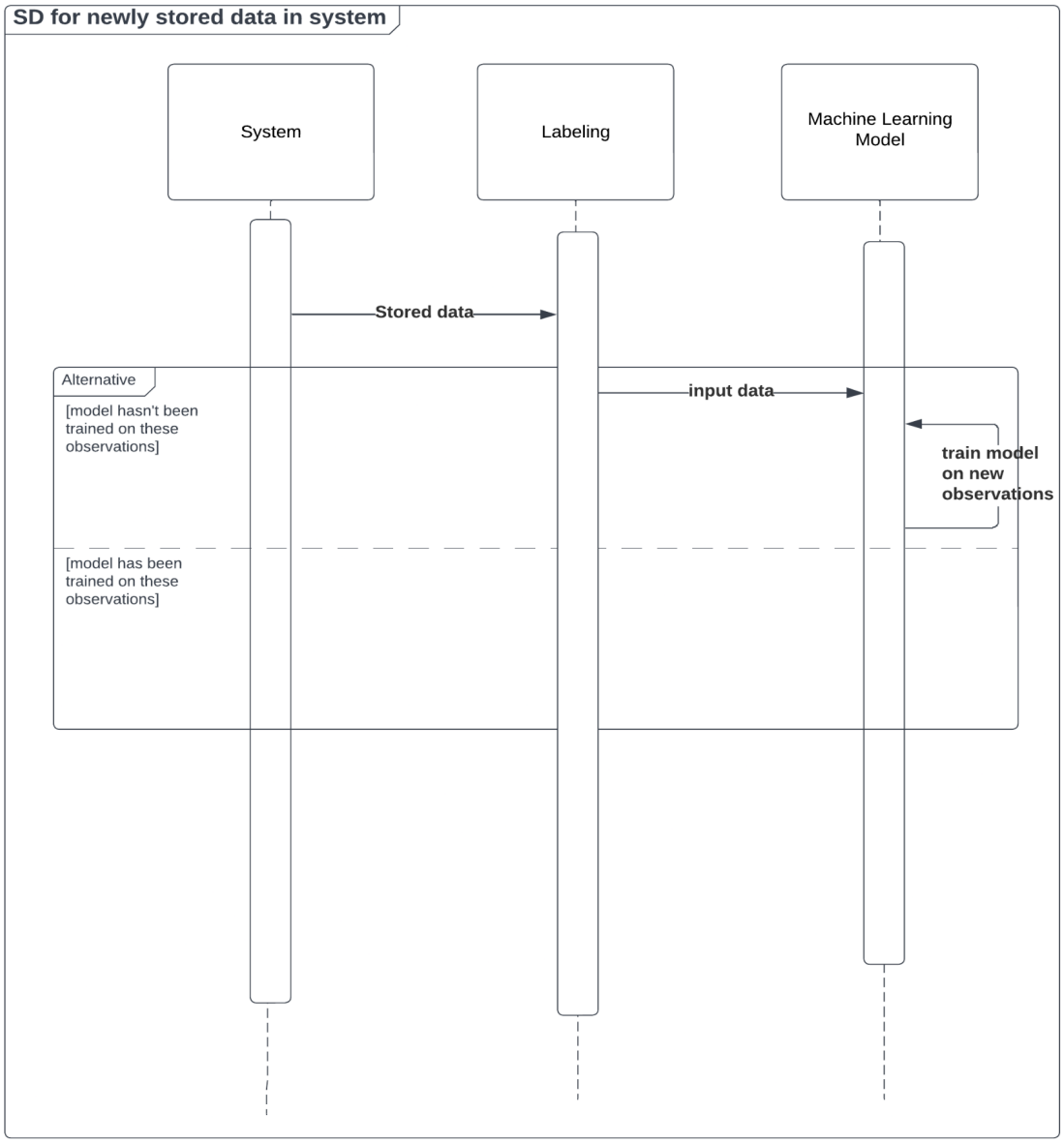

Fig [16]. Sequence diagram for newly stored data.

Figure [16] shows a sequence diagram for retraining the machine learning model on new data points, newly stored data are annotated and labeled by 4 annotators, then the machine learning model will be trained on these labeled data points.

3.2 Proposed Algorithm

The system's proposed algorithm can be accessed from the github repository [14]. The general pipeline architecture is made up of 3 main components (1) Dataset preparation (2) Model training (3) Model evaluation as shown in Figure [17] below. Dataset preparation includes changing the PDF files from .pdf format to byte plot images as .png format. It also includes the dataset directory organization and feature extraction. The data preparation module outputs a train test split of the data into the model training module. This module trains the data on 4 different classifiers as selected by the user. The model is then sent to an evaluation module to calculate performance metrics and give the classification output to the user.

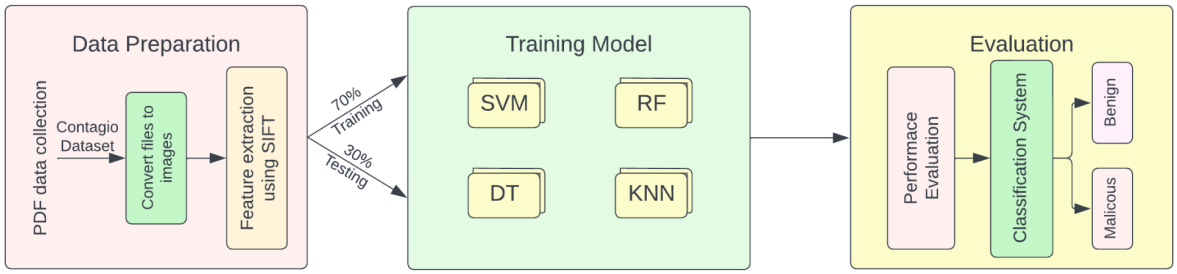

Fig [17]. The architecture of the proposed algorithm.

3.2.1 Dataset directories

A crucial step taken to organize the data for building and testing our model was to create the image representations of the PDFs and upload them into directories on Google drive. We then performed a train and test split into two different directories. The train folder contained 10,500 image files while the test folder contained 4,500 image files. 50% of each folder contained benign files and the other half was malicious.

3.2.2 Data Pre-processing

In this stage we used byte plots to transform each PDF file into an image representation. Byte plots provide a visualization of binary data, they're a very handy tool for analyzing and interpreting binary data. Furthermore, they are used to detect patterns and trends, and they're also helpful in detecting anomalies which will help us to distinguish malicious from benign files [11]. An example of a resulting byte plot from a PDF file should be shown in Figure [18] below.

3.2.3 Feature Extraction

In our proposed algorithm we have implemented SIFT algorithm which is known to be a feature extraction technique. The idea of SIFT is to identify distinctive features such as edges or corners and

describe them as feature descriptors. Feature Descriptors are 128-bit vectors that represent the local appearance of the image. Another reason that makes SIFT algorithm useful and efficient is that it's invariant to rotation, scaling, or affine distortion.[12]

3.2.4 Machine Learning Model

For this application we have used supervised deep learning algorithms since the purpose of our system is to classify PDF into two classes which are benign and malicious. We have implemented the following classification models: KNN, Decision Tree, SVM, and Random Forest. In KNN classification method the prediction for a data point is found by selecting the K nearest neighbors in the training set and taking a majority vote among their class labels. The Decision Tree classifier constructs the tree by starting at the root node and then partitioning the data into subsets based on the values of the features. The tree is grown by repeatedly partitioning the data and adding new nodes until some stopping criterion is reached. SVM uses a hyperplane in the feature space that maximally separates the different classes, unseen data are then classified based on which side of the hyperplane they fall on. The Random Forest classifier falls under the ensemble algorithms, Random Forest consists of different decision trees it trains the model on a random data sample, and then when predicting new data points it takes the average of all the decision trees predictions. [13].

3.2.5 UI Prompt

When the user wants to use the PDF malware detection system, they’re required to either upload an image of the PDF or ask the system to create an image from the entered structured PDF. Later on, They’re asked if they want to build and train a classification model or cross-validation(?). If the user chooses to train the classification model they’re supposed to choose a classifier from the following options: KNN, Decision Tree, SVM, or Random Forest. For each model, the evaluation results will be displayed which are: Accuracy, Precision, Recall, and F1-score.

3.3 Simulation Result

We have used the following metrics to evaluate the efficiency of our model: Accuracy, Precision, Recall, and F1-score. Figure [19] below shows the equations that calculate the mentioned performance metrics.

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

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

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

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Fig [20]. The mathematical equations for the performance metrics

Table V: Performance metrics of the four proposed models

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
KNN	60.4%	46.7%	95.2%	62.7%
Decision Tree	77.9%	67%	72.2%	69.4%
SVM	95.2%	97%	88.9%	92.8%
Random Forest	89.7%	96.0%	73.7%	83.4%

Based on the results in the table above, the SVM machine learning algorithm achieved the highest scores in the performance evaluation which were: 95.2%, 97%, 88.9%, and 92.8% for accuracy, precision, recall, and f1-score respectively.

3.4 Comparison with Prior work

Testing all the models that were proposed by our system, the highest yielding model was the SVM model performing a 95.16% accuracy on the 15,000 PDF image file records.
Comparing our methodology of using the SIFT algorithm for feature extraction with other traditional PDF malware analysis methodologies, we think we have an added value of using a new technique of image analysis to classify PDF files.

Table VI: Accuracy comparison of the proposed algorithm with prior work

Reference	Sample size	Model(s)	Accuracy
[5]	9,000	VGG19	97.3%
[6]	27,000	SVM	95.95%
[7]	10,868	CNN+VGG16	99.08%
[8]	21,146	Random Forest	99.88%
[9]	13,070	SVM	98.88%
Proposed Algorithm	15,000	SVM	95.16%

3.5 Conclusion and Future work

As has been demonstrated, this paper proposed the use of image analysis in PDF malware detection; four models (SVM, KNN, Decision Tree, and Random Forest) were developed and tested. The suggested models were tested on the Contagio dataset after converting files into images and performing feature extraction. The models' goal was to binary classify the given PDF into malicious or benign. Our highest yielding proposed model performed worse compared to prior work, considering we used SIFT algorithm for feature extraction, which isn't a common methodology in PDF malware analysis, and did not use hyperparameter tuning to upgrade our algorithm. Our top priority going forward is to obtain higher performance metrics and decrease the false negative rate. This can be done by optimizing the models parameter values, performing cross-validation, doing an ensemble method, or trying new classifiers. We also aim to train the models on a larger dataset from different sources. We would like to build a maintenance system where data from contributive users can be used to retrain our model, to improve the generalizability of our data. We would also like to build a simple user interface to improve the usability of our system. This paper aimed to help not only standard users but researchers too, as it allows the user to choose four different models to compare between.

4. References

[1] K. Y. Tay, S. Chua, M. Chua, and V. Balachandran, "Towards robust detection of PDF-based malware," *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy*, 2022.

[2] Q. Abu Al-Haija, A. Odeh, and H. Qattous, "PDF malware detection based on optimizable decision trees," MDPI, 30-Sep-2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/19/3142/htm.

[3] H. Singh, J. Singh, and A. Tewari, "Static malware analysis using machine and Deep Learning," *Research Gate*, Jul-2022. [Online]. Available: https://www.researchgate.net/publication/361877301_Static_Malware_Analysis_Using_Machine_and_Deep_Learning.

[4] B. Cuan, A. Damien, C. Delaplace, and M. Valois, "Malware detection in PDF files using Machine Learning," HAL Open Science, 20-Aug-2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01704766.

[5] C.-Y. Liu, M.-Y. Chiu, Q.-X. Huang, and H.-M. Sun, "PDF malware detection using visualization and machine learning," *SpringerLink*, 14-July-2021. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-81242-3_12.

[6] K. Kancherla and S. Mukkamala, "Image visualization based malware detection | IEEE conference," *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/document/6597204/.

[7] Z. Ren, G. Chen, and W. Lu, "Malware visualization methods based on deep convolutional neural networks - multimedia tools and applications," *SpringerLink*, 16-Dec-2019. [Online]. Available: https://link.springer.com/article/10.1007/s11042-019-08310-9.

[8] G. Giacinto and I. Corona, "A pattern recognition system for malicious PDF files detection," *SpringerLink*, 01-Jan-2012. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-31537-4_40.

[9] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," *IEEE Xplore*, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8073489/.

[10] A. Corum, D. Jenkins, and J. Zheng, "Robust PDF malware detection with image visualization and processing techniques" *IEEE Xplore*, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8855273/.

[11] K. Kancherla, "Image Processing Techniques for Malware Detection," *ProQuest*, 2015. [Online]. Available: https://www.proquest.com/docview/1752256351?pq-origsite=gscholar&fromopenview=true.

[12] T. Lindeberg, “Scale invariant feature transform,” *DIVA*, 24-May-2012. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A480321&dswid=9445.

[13] B. Mahesh, “Machine learning algorithms - a review ,” *ResearchGate*, 2018. [Online]. Available: https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf.

[14] Sarabbas, “Sarabbas/pdfimgmalwareanalysis: Software engineering project,” *GitHub*. [Online]. Available: https://github.com/Sarabbas/PDFimgMalwareAnalysis.git. [Accessed: 07-Jan-2023].