International Journal of Basic and Applied Sciences, 14 (7) (2025) 282-295

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/s6dere11
Research paper

Designing A Multi-Agent System for Malicious PDF Detection Using Machine Learning and Generative Artificial Intelligence

Dr. Amadou Diabagate [1], Dr. Hambally Yacouba Yazid [2] *, Prof. Abdellah Azmani [3], Prof. Adama Coulibaly [1]

[1] Faculty of Mathematics and Computer Science, University Felix Houphouët-Boigny, Abidjan, Côte d'Ivoire
[2] University of Bondoukou, Bondoukou, Côte d'ivoire
[3] Faculty of Sciences and Technologies, University of Abdelmalek Essaadi, Tangier, Morocco
**Corresponding author E-mail: yazid.hambally@gmail.com*

Received: September 26, 2025, Accepted: October 11, 2025, Published: November 10, 2025

Abstract

The growing sophistication of cyber threats exploiting PDF files represents a significant challenge for modern cybersecurity systems. To address this issue, this paper introduces a modular Multi Agent System (MAS) designed for the detection of malicious PDF files. Within this architecture, one agent employs a Conditional Tabular GAN (CTGAN) to expand the training dataset and reduce class imbalance, while another agent integrates supervised machine learning models for classification. Six supervised learning models, namely Decision Tree, Random Forest, XGBoost, Support Vector Machine, Naïve Bayes, and Neural Network, are evaluated on the enriched dataset. Among them, XGBoost achieves the best performance.
The MAS coordinates autonomous agents dedicated to dataset management, data augmentation, learning, decision making, and user interaction, ensuring flexibility and scalability under both standard and adversarial conditions. To support interpretability, SHAP analysis is applied during the evaluation phase to identify the features that most strongly influence model decisions. Taken together, the proposed system demonstrates a comprehensive, explainable, and adaptable framework that contributes to strengthening PDF malware detection in sensitive digital infrastructures.

Keywords: *Malicious PDF Detection; Artificial Intelligence; CTGAN; Machine Learning; Multi-Agent System; Cybersecurity; Explainable AI.*

1. Introduction

Information systems are now indispensable to modern organizations, supporting critical activities in sectors such as higher education, finance, insurance, manufacturing, and defense [1]. They enable the storage, processing, and transfer of large volumes of data, while also facilitating communication, coordination, and decision-making. This reliance, however, creates significant vulnerabilities, as valuable data increasingly becomes the target of malicious actors exploiting weaknesses in digital infrastructures [2]. As a result, the detection and prevention of malicious activity have become central to protecting information systems [3].
The growing sophistication of cyberattacks, which are often frequent, complex, and stealthy, places constant pressure on organizations to adopt proactive defense strategies. Artificial intelligence (AI) has shown great potential to address this challenge [4]. Techniques such as machine learning, deep learning, and fuzzy logic can analyze massive data streams in near real time, detect abnormal patterns, and even anticipate intrusions before they occur [5], [6]. The convergence of AI and cybersecurity has therefore opened new avenues for safeguarding digital infrastructures. Intelligent systems can autonomously adapt to emerging threats, reduce response times, and increase resilience in data protection [7], [8].
This study focuses on applying AI techniques, specifically multi-agent systems, a generative model, and supervised machine learning, to the security of PDF files. Although PDFs remain one of the most widely used formats for document exchange, they are also a common vehicle for malware attacks [9]. To address this issue, we propose a hybrid and distributed approach built on three complementary pillars. First, a multi-agent architecture is introduced to provide modularity, scalability, and autonomous decision-making [10]. Second, a Conditional Tabular GAN (CTGAN) is integrated to address class imbalance in the dataset, which contains 32% malicious and 68% benign files, by generating synthetic examples before training [11]. Third, a set of supervised machine learning algorithms is evaluated and compared to identify the most effective model for malicious PDF detection.

The novelty of our work lies in the integration of these three components, which are rarely combined in prior research on PDF security [12]. The multi-agent design enables distributed and autonomous task management, CTGAN improves learning performance by balancing the dataset, and machine learning provides robust classification capabilities. Together, these elements form an innovative and practical contribution to document-level cybersecurity.

The remainder of this paper is organized as follows. Section 2 discusses vulnerabilities in PDF files and the motivation for automated detection. Section 3 reviews related work. Section 4 presents the methodological framework, and Section 5 describes the proposed multi-agent architecture. Sections 6 and 7 detail the Generative Augmentation Agent (GAA) and the Machine Learning Agent (MLA). Section 8 reports the simulation results, Section 9 provides a discussion and limitations, and Section 10 concludes with perspectives for future research.

2. PDF File Security

Information system security has become a cornerstone of modern digital infrastructures, playing a critical role in ensuring operational continuity and protecting sensitive data assets. Commonly referred to as cybersecurity, this domain focuses on safeguarding information systems, networks, software, and data from attacks, unauthorized access, and corruption. Its fundamental objectives are to preserve data confidentiality, integrity, and availability throughout storage, processing, and transmission phases [13].

File security forms an integral part of effective cybersecurity frameworks [5]. It encompasses the protection of digital files from unauthorized access, illegal modification, deletion, damage, and degradation. Such protective measures are especially crucial for documents containing sensitive personal data, proprietary corporate information, or classified governmental content.

PDF file security has emerged as a particularly pressing concern, given the widespread use of the Portable Document Format for disseminating confidential materials such as contracts, financial statements, and personal records. Despite their convenience, PDF files are inherently vulnerable to various forms of exploitation, including unauthorized access, malicious tampering, data exfiltration, and embedded code injection [9], [14]. Among these threats, unauthorized access remains one of the most prevalent, especially when documents lack password protection or robust encryption. In addition, embedded metadata, such as author information, creation timestamps, or reviewer comments, can inadvertently disclose sensitive content if not properly sanitized [6].

To counteract these risks, a variety of protective strategies have been developed. These include encryption, digital signatures, malicious script detection, metadata removal, and watermarking [15].

Encryption remains one of the most effective means of securing PDF files against unauthorized access. It can be configured to restrict both file opening (via an open password) and specific operations such as printing or editing (via a permission password). Digital signatures also play a pivotal role in ensuring document authenticity and integrity. By verifying that a file has not been altered post-creation and confirming the author's identity, digital signatures are particularly critical in legal and financial contexts [9].

Moreover, PDF files can contain embedded JavaScript code, which may serve as a vector for exploiting vulnerabilities in PDF readers. As such, script detection and removal are crucial [16]. Metadata cleansing is equally important, particularly when documents are disseminated publicly, to protect user anonymity and confidentiality [13]. Lastly, watermarking techniques, including visible and invisible digital watermarks, serve to deter unauthorized copying and support provenance tracking [17].

With the growing adoption of cloud-based services, securing PDF files in distributed environments has become even more challenging. To address this, advanced schemes combining encryption and key management have been proposed, enabling access control and data protection in shared, multi-user ecosystems [18].

3. Related Work

The detection of infected PDF files using artificial intelligence (AI) has become a rapidly expanding research area, motivated by the increasing sophistication of cyberattacks and the widespread exploitation of PDF documents for malware dissemination. Due to their portability and cross-platform compatibility, PDFs are widely used for sharing sensitive information but remain frequent vectors for malicious code. Recent studies have therefore explored multiple approaches to improve detection accuracy and resilience.

Early research relied on static and dynamic analysis. Laskov and Šrndić [14] analyzed embedded JavaScript and PDF structures, while Bayer [16] developed TTAnalyze for dynamic behavioral inspection in sandboxed environments. Maiorca [15] combined static and dynamic techniques to improve the detection of executable scripts. Pascanu [19] enhanced dynamic analysis with bidirectional LSTMs to prioritize suspicious sequences, reducing analysis time. Damodaran [20] highlighted hybrid detection combining static and dynamic cues, and Egele [21] focused on anomaly detection through controlled execution. Admass [13] provided a broad overview of these strategies.

Machine learning and deep learning have since played a central role. Han [22] used structural features and metadata with SVMs and decision trees, while Smutz and Stavrou [23] also demonstrated effective SVM-based models. Rieck [24] strengthened detection by integrating static and dynamic features. Dabral [25] combined layout and JavaScript features to train classifiers capable of detecting obfuscated threats. Raff [26] applied CNNs to binary files, Wang [27] employed RNNs with attention for embedded JavaScript, and Dabral's model additionally used clustering to flag anomalous behavior. Other works explored structural watermarking [17] and secure frameworks with encryption and access control [18].

Hybrid approaches have gained traction. Tzermias [28] proposed a two-tiered static-dynamic pipeline, while Jiang [29] optimized dynamic inspection with reinforcement learning. Srndic and Laskov [30] applied machine learning to structural features for static analysis. These studies underline the effectiveness of hybrid strategies, though they often suffer from data imbalance and limited adaptability.

Generative AI has introduced a new perspective in cybersecurity. Recent studies on tabular data synthesis show that modern generative models, particularly conditional tabular GANs, can produce realistic mixed-type samples while explicitly modeling cross-feature dependencies, which helps address class imbalance without distorting feature structure [31]. In security settings, GAN-based augmentation has been shown to improve generalization and reduce bias in binary threat-detection tasks, supporting more reliable classifiers under skewed distributions [32]. In parallel, deep architectures such as LSTMs and Transformers have been applied to malware and document-security analytics; however, their effectiveness often hinges on large labeled corpora and non-trivial computational budgets, which can limit deployment in resource-constrained pipelines [33]. Within this landscape, a CTGAN-plus-ensemble pipeline offers a practical balance between realism, interpretability, and efficiency for tabular, security-oriented data.

Despite these advances, no prior work has unified CTGAN-based augmentation, a modular and distributed multi-agent architecture, and supervised machine learning for malicious PDF detection. Existing studies typically apply these techniques in isolation: machine learning on imbalanced datasets, GANs without distributed coordination, or centralized models lacking adaptability.

Unlike prior approaches, our framework combines data augmentation, supervised learning, and multi-agent orchestration into a single explainable system. This integration enables dynamic rebalancing of data, robust classification, and adaptive coordination, offering a novel and pragmatic contribution to document-level cybersecurity.

4. Methodology

The methodology adopted in this study is based on a hybrid and distributed architecture structured around three core components: intelligent multi-agent systems, generative artificial intelligence (via the CTGAN model) [11], and supervised machine learning. This approach combines the modularity and coordination capabilities of software agents, the generation of balanced synthetic datasets to enhance learning performance [2], and the predictive power of supervised algorithms. The full methodological process is detailed across the following three subsections.

4.1. Multi-agent system architecture

The proposed system architecture relies on six cooperative agents interacting asynchronously:

- User Interface Agent (UIA): Serves as the user entry point, transmitting uploaded PDF files for analysis.
- Planning Agent (PLA): Coordinates the execution sequence of the various components.
- Generative Augmentation Agent (GAA): Applies the CTGAN algorithm to generate synthetic samples. It rebalances the dataset when class imbalance is detected, or enhances the training data to improve generalization.
- Machine Learning Agent (MLA): Implements a supervised learning model to predict whether a PDF file is benign or malicious.
- User Profile Management Agent (UPMA): Adjusts the system's behavior based on user history and interaction patterns.
- Data Management Agent (DMA): Oversees data storage and manages model retraining processes.
- Each agent is designed to be autonomous, responsive, and communicative. The architecture is fault-tolerant and supports incremental extension by the addition of new agents.

4.2. Generative data augmentation using CTGAN

To address the class imbalance observed in the dataset, a data augmentation phase was introduced using generative artificial intelligence, specifically the CTGAN model [11]. This section outlines the data balancing strategy followed by a brief presentation of the theoretical foundations of CTGAN.
CTGAN is particularly well-suited to our dataset, which consists of tabular attributes extracted from PDF files [31]. It enables the generation of realistic synthetic malicious examples while preserving the underlying correlation structures present in the original data. This capability increases the diversity of the training set and optimizes the performance of supervised learning algorithms. Unlike classical methods such as SMOTE or ADASYN, which often generate data without preserving the natural structure of features, CTGAN produces synthetic samples that retain both categorical and numerical dependencies. This makes it especially suitable for modeling the complex interrelations found in PDF structural attributes.

a) Dataset Description and Preprocessing

The dataset used in this study exhibits a notable imbalance: 4,315 benign PDF files versus 2,028 malicious ones, totaling 6,343 records. The dataset includes 22 static representative features, divided into two categories: general characteristics and structural characteristics. Table 1 offers a structured overview of the 22 static features. These features were selected for their potential to reveal meaningful differences between benign and malicious PDF files.

Table 1: General and Structural Characteristics of the PDF Files

Feature Name	Description
General Characteristics	
Total size of PDF	File size in bytes
Length of title text	Legitimate PDFs typically have meaningful, descriptive titles.
Encryption enabled	Indicates whether the file is password-protected
Volume of metadata	Size of the descriptive information embedded in the PDF
Number of pages	Total number of pages in the document
Header existence	Represents the PDF version parsed from the file header
Number of Embedded Images	Total number of images embedded in the document
Text Presence	Indicates whether the PDF contains readable text
Object Count	Total number of objects (text, images, streams, fonts, annotations, etc.)
Embedded Files	Number of additional files attached to the PDF
Structural Characteristics	
Stream object count	Number of binary data streams within the PDF
JavaScript presence	Number of objects containing JavaScript code
Automatic Action	Defines specific actions triggered by events (often used with malicious JS)
Launch command	Executes commands/programs (often used for data theft or malware)
Open trigger	Specifies actions upon opening, often tied to malicious JavaScript
AcroForm Tag Count	Acrobat forms potentially containing exploitable scriptable fields.
JBig2Decode Filter Presence	Indicates the use of JBig2Decode, often used to encode malicious content
Xref Length	Number of cross-reference tables managing the object structure
XFA Form Used	Indicates the presence of XML-based forms supporting scripting
Xref Entry Count	Number of entries in the Xref tables, often malformed in malicious PDFs
Rich media presence	Number of embedded multimedia or Flash objects
Trailer tag count	Number of trailer sections; abnormal counts may indicate suspicious content

Preprocessing was performed in two stages: (1) handling missing or outlier values, and (2) encoding, normalization, and final formatting for the CTGAN application. Each column was analyzed for missing values, which were imputed using the most frequent value per feature via the SimpleImputer class from sklearn. Impute.

Categorical values were encoded using Label Encoding, and normalization was applied using StandardScaler to ensure all numerical features were on the same scale, thereby improving gradient descent convergence during training. After preprocessing, we split the data into an 80/20 train–test set (5,074/1,269). To keep the evaluation unbiased and avoid any leakage, augmentation was applied exclusively to the training set. Using CTGAN, we increased the minority (malicious) class to roughly 75% of the majority (benign). This threshold was chosen to rebalance the data without overproducing synthetic samples, which can inflate the minority prior, dilute the signal from real observations, and lead models to learn generator-specific artifacts. No additional oversampling was required after augmentation. Detailed counts are provided in Section 6 (Table 5).

b) CTGAN Overview and Working Principle

CTGAN (Conditional Tabular GAN) extends traditional Generative Adversarial Networks to tabular data that combines continuous and categorical variables [31]. As in standard GANs, one network, the generator, produces synthetic samples from random noise, while a second network, the discriminator, learns to distinguish real data from generated data. This adversarial training allows the model to improve iteratively, with the generator producing samples that become progressively more realistic as the discriminator becomes more discerning. The setup is particularly suitable for cybersecurity datasets where features are heterogeneous and often weakly correlated.

Beyond this classic scheme, CTGAN introduces conditional generation so the model explicitly learns dependencies that span discrete and continuous features. Conditioning ensures that categorical values are respected and that the relationships among variables are preserved, rather than being washed out by naive sampling. In parallel, numerical attributes are represented with Gaussian mixture models, which capture multi-modal shapes and heavy tails more faithfully than simple parametric assumptions. Together, these mechanisms help produce synthetic records that mirror real-world variability and structure, not just marginal statistics.

Put simply, CTGAN models how features co-vary and then uses that knowledge to create realistic tabular samples. In class-imbalance settings such as benign versus malicious PDFs, CTGAN helps preserve cross-feature relationships and category semantics, thereby maintaining data integrity and supporting downstream generalization.

4.3. Application of machine learning

To meet security-oriented requirements on structured PDF data, we evaluate six complementary learners: an interpretable Decision Tree; Random Forest and XGBoost (eXtreme Gradient Boosting) for strong tabular performance [34], [35]; an SVM (Support Vector Machine) with a solid record on document/PDF features [21], [22]; a fast probabilistic baseline (Naive Bayes) [33]; and a shallow MLP to capture additional non-linearities. This portfolio is consistent with prior malware and PDF detection on structured features [2], [3], [9], [24]. For benchmarking, we also included a simple DummyClassifier as a baseline.

a) Description of the Selected Machine Learning Algorithms

This study incorporates six machine learning algorithms, each selected for its distinct learning paradigm and its potential relevance to the detection of malicious PDF files. The following overview outlines their underlying principles, practical strengths, and known challenges in cybersecurity contexts.

Random Forest, introduced by Breiman [36], builds an ensemble of decision trees by training each on a randomly sampled subset of data. The ensemble's aggregated predictions reduce variance and improve generalization [37]. Moreover, Random Forests provide valuable insights into feature importance, an essential requirement in explainable security systems.

The Multilayer Perceptron (MLP) is a neural network structure comprising multiple interconnected layers [38], [39]. MLPs excel at modeling nonlinear data relationships but generally require large and balanced datasets to perform effectively. Their limited interpretability may hinder adoption in domains where decision transparency is mandatory, such as digital forensics [40].

Decision Trees are favored for their interpretability, as they recursively divide data based on metrics like Gini index or entropy, resulting in easily understandable tree structures [41]. However, their tendency to overfit, especially on small or noisy datasets, necessitates the use of depth constraints or pruning.

Support Vector Machines (SVMs) focus on identifying a decision boundary that maximizes the margin between classes [42]. While robust for structured and linearly separable data, SVMs can struggle with overlapping distributions or datasets containing noise, which are common in document classification.

XGBoost, developed by Chen and Guestrin [43], is a gradient boosting technique that integrates regularization, pruning, and parallel computation. It is known for its high performance on noisy and imbalanced datasets, making it particularly effective for malware detection applications [31], [44].

Naive Bayes is a probabilistic model that applies Bayes' theorem with the simplifying assumption of conditional independence between features. While this assumption is often violated in practice, the algorithm is fast, scalable, and often surprisingly effective in high-dimensional or sparse datasets. In malware detection, its speed is an advantage, although its accuracy may decline when features are interdependent or the class distribution is skewed [45], [46].

To complement these models, a baseline classifier was added using the most_frequent strategy from Scikit-learn's DummyClassifier. Though clearly unsuited for real-world deployment, this model predicts only the majority class and serves as a minimal reference point to highlight the added value of more advanced classifiers [47], [48].

In summary, this ensemble of models enables a balanced evaluation of accuracy, interpretability, and robustness for classifying PDF files in security-sensitive applications.

b) Tuning and Selection of Classification Models

To enhance predictive accuracy, all six classification models underwent hyperparameter optimization using the RandomizedSearchCV method. This strategy allows efficient exploration of the hyperparameter space by sampling configurations at random, offering a practical trade-off between speed and performance compared to exhaustive methods like GridSearchCV [49], [50].

The selected hyperparameters were those yielding the highest classification performance on the imbalanced training dataset. Table 2 summarizes the final configurations for each model, which were retained for evaluation and simulation in the remainder of the study.

Table 2: Summary of Machine Learning Models and their Optimized Hyperparameter Settings

Model	Optimized Hyperparameters and Values
Decision Tree	criterion = entropy; max_depth = None; min_samples_leaf = 1; min_samples_split = 4
Random Forest	bootstrap = False; max_depth = None; max_features = log2; min_samples_leaf = 2; min_samples_split = 13; n_estimators = 129
XGBoost	colsample_bytree = 0.55; gamma = 0.186; learning_rate = 0.21; max_depth = 7; n_estimators = 198; subsample = 0.79
SVM	C = 9.74755518841459; gamma = scale; kernel = rbf
Naive Bayes	var_smoothing = 0.01873817422860387

Neural Network	Activation = tanh; alpha = 0.0022; early_stopping = True; hidden_layer_sizes = (100; 50); learning_rate = adaptive; n_iter_no_change = 10; solver = adam; validation_fraction = 0.1

Hyperparameter optimization was conducted using cross-validation, a crucial procedure to enhance the models' ability to generalize beyond the training data. This phase not only contributes to result reproducibility but also establishes a solid ground for a fair and reliable comparison of model performance.
To ensure methodological rigor, we applied a Stratified K-Fold cross-validation strategy. This approach divides the dataset into K equally sized subsets (folds) while preserving the original class distribution within each fold, a key consideration in imbalanced classification contexts. At each iteration, one fold is used as the validation set, and the remaining K–1 folds form the training set. The process is repeated K times, allowing each subset to serve once as validation data. Performance scores are then averaged across folds, yielding a more stable and representative estimate of model behavior.
For this study, we adopted K = 10, a widely recommended choice in machine learning research [51], [52], as it offers a robust balance between low bias and low variance. This protocol was essential in ensuring consistency and objectivity in the evaluation of all models under consideration.

4.4. Technical pipeline description

The architecture of the proposed system follows a coherent and modular processing pipeline that transforms incoming PDF documents into informed security assessments. At the initial stage, the User Interface Agent (UIA) serves as the entry point, managing the reception and routing of each file to the internal processing chain. Once received, the Planning Agent (PLA) assumes control, orchestrating the sequence and priority of subsequent operations based on predefined rules and contextual factors.
Before initiating the learning phase, the system evaluates the dataset for potential class imbalance, a frequent challenge in cybersecurity applications. When an imbalance is detected, the Generative Augmentation Agent (GAA) activates the Conditional Tabular GAN (CTGAN) model to generate statistically consistent synthetic instances. This augmentation phase enhances the representativeness of the minority class while preserving the underlying data distribution, thereby improving the model's capacity for generalization.
The enriched dataset is then forwarded to the Machine Learning Agent (MLA), which applies a supervised learning algorithm trained on annotated data to predict whether each file poses a security threat. The outcome of this classification is systematically logged and stored by the Data Management Agent (DMA), while the User Profile Agent (UPMA) monitors the process and refines system behavior based on individual usage patterns.
The entire system operates on a distributed, event-driven architecture, where autonomous agents interact asynchronously. This decentralized structure enables real-time adaptability to varying loads, evolving threat landscapes, and user-specific profiles. It also ensures graceful degradation in the event of localized failures, thereby enhancing overall system resilience.
Development was conducted within a unified Python environment integrating the Mesa agent-based simulation framework, scikit-learn for supervised learning, and CTGAN for synthetic data generation. The infrastructure supports the reproducibility of experiments, the scalability of intelligent components, and the seamless integration of modules within a realistic cybersecurity context.

4.5. Implementation and simulation

The system was implemented using Mesa, a Python-based framework tailored for multi-agent simulation. This platform enabled a detailed modeling of inter-agent interactions and real-time coordination in a cybersecurity scenario involving the analysis of PDF documents.
The simulation was conducted on a dataset of 400 PDF files, comprising both benign and malicious samples. Each file followed a complete analytical pipeline, traversing six specialized agents: user interface, planning, data augmentation, classification, user profiling, and data management. Coordination among these agents was governed by the Planning Agent, which allocated tasks based on system load and operational priority.
The technical stack combined several supervised classifiers, including Random Forest, Support Vector Machine (SVM), and XGBoost, with synthetic data augmentation via CTGAN. This hybrid setup allowed for a robust evaluation of system performance under different operational conditions [8].
Three representative scenarios were defined to emulate real-world variability in threat levels and system usage:

- Nominal Detection Scenario – Balanced distribution of malicious and benign files, testing standard system behavior.
- Stress Scenario: Malicious Overload – Sudden surge in malicious documents, assessing system resilience and reactivity.
- Suspicious User Profile Scenario – Activation of adaptive behaviors in response to users with historically risky profiles.

System performance is evaluated through a set of carefully selected indicators, adapted to the specific objectives of each scenario. Table 3 summarizes the main metrics used, along with the agents responsible for generating or leveraging them.

Table 3: Evaluation Indicators and Agent Contributions in the Multi-Agent Detection System

Indicator	Description	Involved Agents
Accuracy/ Precision/ Recall/f1-score	Standard classification metrics	MLA, GAA
Average Processing Time	Mean time required to process one file	PLA, MLA
Data Augmentation Rate	Proportion of samples synthetically generated via CTGAN	GAA
Performance Under Load	Variation in detection metrics under stress conditions	PLA, MLA, DMA
User Profile Dynamics	Context-aware detection of risky users and adaptation of priorities	UPMA, PLA, DMA
Agent Activity Rate	Number of executed tasks per agent	All agents

Each indicator is contextualized and discussed in Section 8, according to its relevance for the given scenario. This structured evaluation allows for a nuanced assessment of the system's accuracy, adaptability, and suitability for real-world cybersecurity applications.

5. Operation and Organization of The Multi-Agent System

In light of the growing complexity and dynamic nature of cybersecurity threats, centralized detection systems often prove insufficiently responsive or scalable. To address these limitations, Multi-Agent Systems (MAS) offer a paradigm that supports decentralized, autonomous

coordination among intelligent entities. Rooted in the principles of distributed artificial intelligence, a MAS comprises multiple agents capable of interacting through mechanisms of cooperation, negotiation, or autonomous decision-making [26].
Such architectures combine the strengths of modularity, flexibility, and interoperability. They are particularly well-suited to environments requiring dynamic adaptation, intelligent task orchestration, and real-time data processing, key requirements in the field of PDF file security and threat mitigation [13], [19].

5.1. System overview

The architecture proposed in this study is based on a cooperative and distributed multi-agent system composed of six autonomous agents, each responsible for a specialized function within the PDF security analysis pipeline. This modular configuration facilitates system evolution, fosters component reuse, and enhances the overall resilience of the system [1.
Agents interact asynchronously using standardized messaging protocols, allowing for efficient data exchange and coordination across layers. The system emulates a full end-to-end detection cycle within a simulated cybersecurity context, encompassing stages such as data augmentation, machine learning-based classification, planning, and user behavior profiling [27].
Simulation is conducted using the Mesa framework, enabling the modeling of dynamic interactions, workload distribution, and agent-specific intelligence in a realistic digital ecosystem.

5.2. Agents and functional roles

The proposed module represents a core component of a broader information system tasked with detecting potentially harmful PDF files. It integrates six functional agents, each fulfilling a specific role in the processing pipeline. Table 4 outlines the six core agents of the proposed system, each assigned a specific role to ensure coordinated and effective detection of malicious PDF files.

Table 4: Functional Roles and Responsibilities of Core System Agents

No.	Acronym	Agent Name	Functional Description
1	UIA	User Interface Agent	Serves as the system's entry point; handles user interactions, file intake, and output delivery of classification results.
2	PLA	Planning Agent	Coordinates all internal processes by assigning priorities and determining execution sequences based on predefined rules.
3	GAA	Generative Augmentation Agent	Detects class imbalance and activates the CTGAN model to generate synthetic samples, enhancing training data diversity.
4	MLA	Machine Learning Agent	Conducts supervised classification (e.g., Random Forest); oversees model evaluation and retraining when accuracy declines.
5	UPMA	User Profile Management Agent	Maintains risk profiles and user histories; enables behavioral adaptation of the system based on usage patterns.
6	DMA	Data Management Agent	Archives classified results, updates internal repositories, and provides data for retraining or audit purposes.

Agent communication is implemented via an asynchronous message-passing protocol, which ensures parallel processing, robustness in the face of agent failures, and extendibility [7]. The architecture is flexible enough to accommodate the integration of new agents (e.g., for behavioral analysis or network-based threat detection) without altering the core system structure.

5.3. Agent communication and coordination

Agent interactions are governed by a communication protocol inspired by FIPA's Agent Communication Language (ACL). Each message contains a performative (e.g., inform, request, confirm) and a structured content payload, ensuring unambiguous interpretation [20], [24].
The Planning Agent (PLA) plays a pivotal role in coordinating the system. It manages task sequencing, dynamically allocates processing resources, and intervenes when exceptional conditions, such as model degradation or class imbalance, are detected. This centralized coordination mechanism provides a robust foundation for adaptive system behavior, capable of responding in real time to threats or anomalous input distributions [27].

5.4. Conceptual architecture

The system architecture is organized into four distinct functional layers, each encapsulating specific responsibilities:

- Interface and Coordination Layer includes the UIA and PLA. It governs external interactions and orchestrates the processing sequence.
- Data Enrichment Layer incorporates the GAA. This layer addresses data imbalance by generating synthetic records using CTGAN to ensure a well-distributed training set.
- Decision Intelligence Layer hosts the MLA. It is responsible for classifying incoming PDF files, monitoring model performance, and triggering retraining when necessary.
- Management and Monitoring Layer includes UPMA and DMA. It handles historical user modeling, result storage, and supports downstream tasks such as audit trails and model updates.

This layered abstraction enhances clarity, promotes separation of concerns, and allows independent upgrades to each component. It ensures that decision-making, data management, and system coordination remain modular and interoperable. Figure 1 below illustrates this architectural organization.

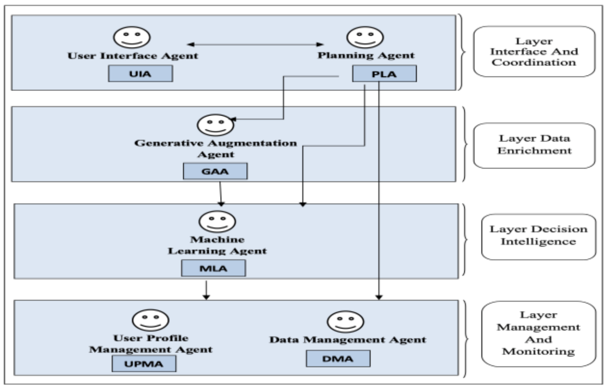
User Interface Agent
UIA
Planning Agent
PLA
Layer Interface And Coordination
Generative Augmentation Agent
GAA
Layer Data Enrichment
Machine Learning Agent
MLA
Layer Decision Intelligence
User Profile Management Agent
UPMA
Data Management Agent
DMA
Layer Management And Monitoring

Fig. 1: Layered MAS Architecture. Pipeline of Malicious-PDF Detection: CTGAN (GAA), MLA, Orchestration (PLA), Profiling (UPMA), and Data Management (DMA).

6. Data Augmentation Module: GAA Agent

In order to mitigate class imbalance and reinforce the generalization capacity of the predictive model, a generative augmentation mechanism has been integrated into the proposed multi-agent framework. This component is embodied by the Generative Augmentation Agent (GAA), which leverages Conditional Tabular GANs (CTGAN) to produce synthetic examples that are statistically aligned with the real data [19], [27].
Following the protocol in Section 4.2, CTGAN generated about 986 additional malicious samples. The training set, therefore, grew from 5,074 to 6,060 files (3,463 benign and 2,597 malicious), while the test set remained unchanged at 1,269, giving a total of 7,329. This achieves the intended minority-to-majority ratio of roughly 75%. For clarity, Table 5 summarizes the counts before and after augmentation. Before their integration into the training process, a rigorous evaluation of the synthetic data is conducted to ensure its statistical credibility and structural coherence [31].

Table 5: Dataset Counts Before and After CTGAN Augmentation

Subset	Benign	Malicious	Total	Note
Train (before)	3,463	1,611	5,074	80% split (pre-CTGAN)
Synthetic added (malicious)	—	=986	=986	CTGAN output
Train (after)	3,463	2,597	6,060	Target = 75% ratio
Test (unchanged)	—	—	1,269	Held out, no augmentation
Overall (after)	—	—	7,329	6,060 + 1,269

6.1. Univariate distribution analysis and statistical alignment

To assess the fidelity of the synthetic samples produced by CTGAN, a comparative analysis was carried out on six critical features, selected for their relevance in the classification process and their diverse typological nature, spanning structural, behavioral, and content-based dimensions:

- JavaScript presence – Number of objects containing JavaScript code
- Volume of metadata and Total size of PDF – continuous proxies of document complexity and density.
- Open trigger and Automatic actions – behavioral flags reflecting document interactivity.
- Xref length – a discrete structural metric indicative of internal referencing.

The comparative plots are presented in Figure 2, where the synthetic distributions generally track closely with their real counterparts, particularly for continuous variables like metadata volume and file size, where the density curves are almost superimposed. This visual consistency suggests that CTGAN successfully modeled the probabilistic structures underlying the original data.

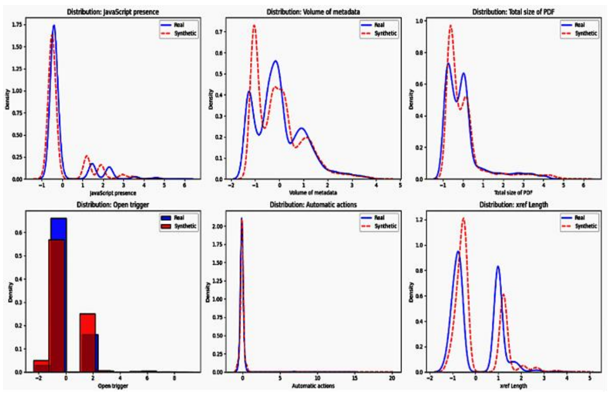

Fig. 2: Real vs. Synthetic Distributions of Six Critical Features. The Close Overlap between Real and CTGAN-Generated Samples Indicates Faithful Synthetic Data.

To strengthen these observations, the Kolmogorov–Smirnov (KS) test was applied to each variable. The results, summarized in Table 6, show that for three of the six features (JavaScript presence, Open trigger, and Automatic actions), the null hypothesis of identical distributions cannot be rejected ($p > 0.05$). Although slight discrepancies are noted for continuous and discrete features such as volume of metadata, total size of PDF, and xref length, these variations remain within tolerable limits for augmentation purposes and do not introduce systematic bias [23].

Table 6: Kolmogorov–Smirnov Test Comparing Distributions between Real and Synthetic Data for Six Key Features

Variable	Type	KS Statistic	p-value	Distribution Equal (p > 0.05)
Javascript presence	Binary	0.0	1.0	True
Volume of metadata	Continuous	0.1689	0.0	False
Total size of PDF	Continuous	0.1914	0.0	False
Open trigger	Binary	0.0	1.0	True
Automatic actions	Binary	0.0	1.0	True
Xref length	Discrete	0.3197	0.0	False

Although small discrepancies exist, CTGAN keeps the overall distributional structure well preserved, maintaining data representativeness and, as a result, improving classifier performance while limiting vulnerabilities linked to class imbalance.

6.2. Multivariate coherence and structural integrity

Beyond individual feature alignment, the structural validity of the synthetic data was assessed through multivariate analysis techniques to determine whether CTGAN preserved the complex interdependencies among variables.
A first level of evaluation involved Pearson correlation matrices, independently computed for both real and synthetic datasets. As illustrated in Figure 3, the matrices exhibit a high level of congruence, particularly for feature pairs involving size, volume of metadata, and xref length, which consistently show strong positive correlations. The difference in correlation coefficients between real and synthetic data remained within a ±0.05 margin, indicating that the generative model captured the essential joint relationships without distortion.

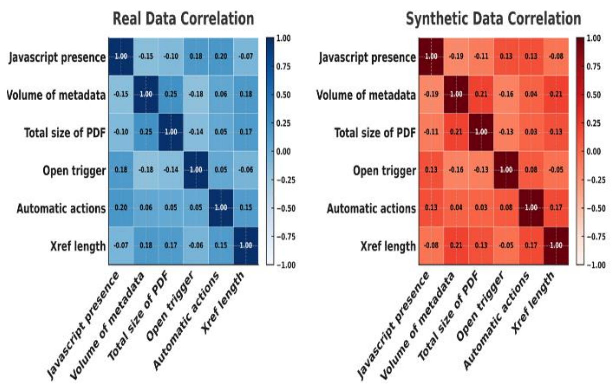

Fig. 3: Correlation Matrices (Real vs. Synthetic). Pearson Correlation Maps Computed on Real and CTGAN Data. Similar Patterns Suggest CTGAN Preserves Cross-Feature Dependencies.

To further explore the global structure, a Principal Component Analysis (PCA) was applied to the concatenated dataset (real + synthetic). As illustrated in Figure 4, the projection onto the first two principal components reveals a substantial overlap in geometric positioning, with no emergent clusters or separability between the two data types. This absence of structural divergence supports the notion that CTGAN preserved the latent topological characteristics of the original feature space.

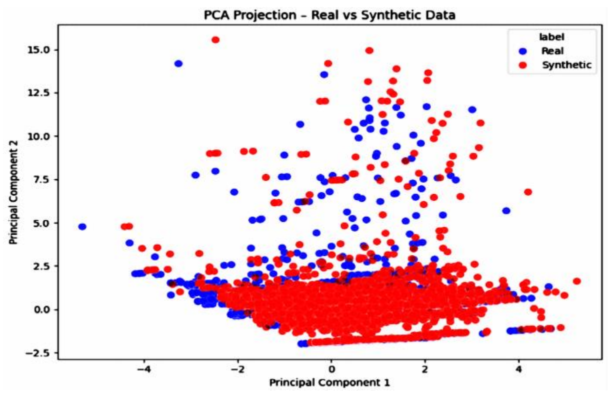

Fig. 4: 2-D PCA Projection (Real + Synthetic). Samples Projected onto the First Two Principal Components. The Strong Overlap Shows That Augmentation Does Not Introduce Spurious.

Together, these findings attest to the semantic and statistical coherence of the synthetic data. By maintaining both marginal distributions and multivariate associations, the CTGAN-powered augmentation process enhances diversity in the training set without compromising the interpretability or reliability of the downstream learning algorithms. This enables the overall system to remain performant, even in the presence of initially unbalanced datasets.

7. Classification Module: Machine Learning Agent (MLA)

The classification module, orchestrated by the Machine Learning Agent (MLA), plays a central role in the detection of potentially malicious PDF files. Designed to evaluate multiple supervised learning algorithms, this module leverages both the original and CTGAN-augmented datasets to identify the most performant and generalizable classifier. The present section presents a comparative assessment of candidate models, culminating in the selection of the optimal architecture for subsequent explainability and diagnostic stages.

7.1. Comparative evaluation of classifiers on original and augmented datasets

To ensure the robustness and stability of predictions, six supervised classifiers were benchmarked across both the real dataset and its CTGAN-augmented counterpart. The algorithms under consideration included Decision Tree, Random Forest, XGBoost, Support Vector Machine (SVM), Naive Bayes, and a shallow Neural Network. Model evaluation was conducted using stratified 10-fold cross-validation, a method known to provide reliable estimates of generalization performance, particularly in contexts characterized by class imbalance.
Performance was measured using a comprehensive set of metrics: accuracy, precision, recall, F1-score, area under the ROC curve (ROC-AUC), and Matthews Correlation Coefficient (MCC). The results, reported in Tables 7 and 8, offer an in-depth view of model behavior across both datasets.

Table 7: Cross-Validated Classification Performance on the Original Dataset (%)

Model	Accuracy	F1-score	Precision	Recall	ROC-AUC	MCC
DummyClassifier	60.16	38.14	37.86	38.42	54.40	8.76
Decision Tree	98.61	97.83	98.49	97.93	98.36	96.81
Random Forest	99.87	99.8	100.0	99.61	99.8	99.71
XGBoost	99.84	99.75	100.0	99.61	99.75	99.64
SVM Classifier	99.37	99.01	99.6	99.61	99.34	98.55
Naïve Bayes	95.47	92.73	96.21	91.11	94.09	89.53
Neural Network	98.61	97.83	98.49	97.93	98.36	96.81

Table 8: Cross-Validated Classification Performance on the CTGAN-Augmented Dataset (%)

Model	Accuracy	F1-score	Precision	Recall	ROC-AUC	MCC
DummyClassifier	55.80	46.97	46.37	47.60	54.56	9.09
Decision Tree	97,03	96,43	99,59	95,08	96,59	94,03
Random Forest	99,83	99,81	100	100	99,81	99,66
XGBoost	99,87	99,85	100	100	99,88	99,73
SVM Classifier	98,35	98,04	100	97,85	98,12	96,67
Naïve Bayes	86,73	82,35	97,07	73,42	84,92	73,88
Neural Network	97,03	96,43	99,59	95,08	96,59	94,03

The results in Tables 7 and 8 show a consistent advantage for ensemble-based models over other classifiers. On the original dataset, Random Forest performs slightly better than XGBoost. However, after CTGAN-based augmentation, XGBoost stands out with near-perfect results, reaching 99.87% accuracy, 100% recall, and a Matthews correlation coefficient of 99.73%. SVM, Neural Network, and Decision Tree also deliver strong outcomes, while Naïve Bayes records a noticeable drop in recall. The DummyClassifier serves only as a baseline, confirming that the high scores of advanced models are the result of effective learning rather than chance or bias.
These findings underscore the capacity of CTGAN-generated data to enhance model learning, particularly for ensemble methods. The synthetic data not only balanced the class distribution but also improved model sensitivity, as reflected in perfect recall values for XGBoost. Given its superior and consistent performance, XGBoost trained on the augmented dataset is retained as the reference classifier for the remainder of this study. It serves as the foundation for feature interpretability analysis (Section 7.2) and diagnostic visualization (Section 7.3).
Although these results are nearly perfect, they are still specific to the dataset of 6,343 files analyzed in this study. A logical next step would be to test the system on larger and more diverse collections.

7.2. Interpretability of the selected model: feature importance and SHAP analysis

To ensure the transparency and explainability of the classification process, we conducted an analysis of feature importance based on the XGBoost model's internal gain metrics. As illustrated in Figure 5 (A), the most influential variable is JavaScript presence, which aligns with known threat signatures in malicious PDFs. Other key predictors include metadata volume, open trigger, and xref length, highlighting a combination of behavioral and structural indicators.

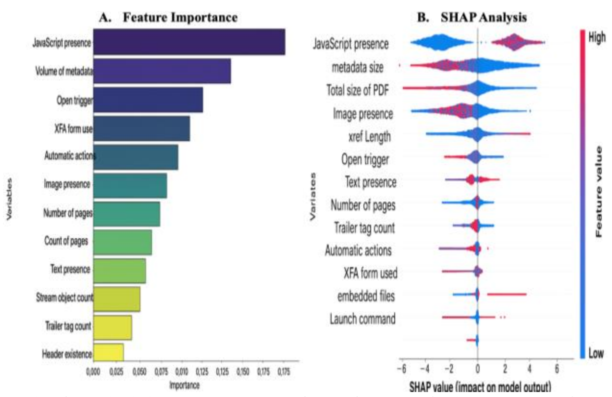

Fig. 5: Model Feature Importance and SHAP (Xgboost). (A) Top 10 Features by Xgboost Gain (E.G., Javascript, Metadata Volume, Openaction). (B) SHAP Summary: Color Encodes Feature Value (Red = High, Blue = Low); The Horizontal Axis Shows Mean Absolute Impact on the Prediction. Both Views Highlight the Same Drivers of Maliciousness.

In Figure 5 (B), a SHAP (Shapley Additive exPlanations) analysis was carried out to provide a local and global interpretation of the model's predictions. The SHAP summary plot reveals that high values of JavaScript presence, volume of metadata, and total size of PDF

consistently push predictions toward the malicious class, while low values mitigate that likelihood. Conversely, variables such as header existence and text presence exhibit a more nuanced impact, suggesting interaction effects.
Collectively, the interpretability analysis confirms that the model's decision-making aligns with domain-relevant indicators. It also enhances user trust by revealing the logical structure behind individual predictions.

7.3. Visual diagnostics and error analysis

The performance of the selected XGBoost model is further validated through visual inspection of its classification behavior. The Receiver Operating Characteristic (ROC) curve, shown in Figure 6 (A), demonstrates near-optimal separation, with an area under the curve of 0.9997, a hallmark of a highly discriminative classifier.
Complementing this, the confusion matrix presented in Figure 6 (B) reports only 14 false positives and 22 false negatives out of 7,329 total instances. This remarkably low error rate reinforces the classifier's precision, sensitivity, and overall reliability.
Taken together, these diagnostic tools provide compelling evidence that the model not only performs well in terms of global metrics but also maintains operational reliability in fine-grained scenarios. The augmentation strategy via CTGAN, combined with ensemble learning, yields a classification pipeline that is both intelligent and interpretable, meeting the key demands of modern cybersecurity systems.

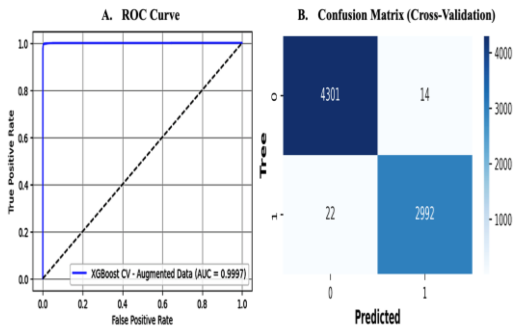

Fig. 6: (A) ROC Curve Demonstrating Near-Perfect Discrimination for Xgboost. (B) Confusion Matrix with Very Few Errors Overall, Confirming Both Sensitivity and Precision.

8. Simulation Results Analysis: Global and Scenario-Based Evaluation

This section presents a comprehensive examination of the simulation results derived from the implementation of the proposed Multi-Agent System (MAS) for malicious PDF detection. Conducted over a set of 400 files within a simulated cyber-environment, the evaluation focuses on two complementary dimensions: a global assessment of system behavior and a detailed scenario-specific analysis designed to probe the system's resilience and responsiveness under varying operational contexts.
Three representative scenarios were explored: (1) detection under standard conditions, (2) simulated overload due to a malicious file surge, and (3) detection adaptation based on suspicious user behavior.
Each scenario was assessed using specific methods tailored to its operational context and evaluation goals.

- Scenario 1 – Standard Detection: assesses baseline performance using accuracy, precision, recall, F1-score, and processing time under normal operating conditions.
- Scenario 2 – Simulated Attack: evaluates system resilience under stress, focusing on latency, performance degradation, queuing behavior, and overload management.
- Scenario 3 – Suspicious User Profile: tests adaptability to abnormal user behavior based on profile-specific metrics and UPMA activity logs.

8.1. Global system overview

The global simulation provides a macro-level view of the MAS's internal dynamics. Each of the 400 PDF files submitted triggered interactions across six specialized agents: the User Interface Agent (UIA), Planning Agent (PLA), Machine Learning Agent (MLA), User Profile Management Agent (UPMA), Data Management Agent (DMA), and Generative Augmentation Agent (GAA).
The frequency of agent activations was tracked to analyze task distribution and the architectural equilibrium of the system. As shown in Figure 7, the core agents UIA, PLA, MLA, UPMA, and DMA were uniformly activated for each file, reflecting their fundamental role in the classification pipeline. In contrast, the GAA was triggered only five times throughout the simulation, underscoring its targeted function in correcting class imbalance.

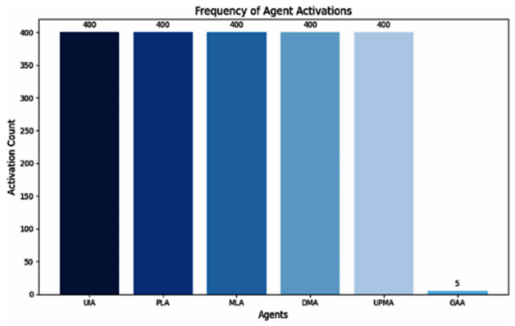

Fig. 7: Agent Activation Frequencies During the Simulation. Core Agents (UIA, PLA, MLA, UPMA, DMA) Are Triggered for Every File, Whereas the GAA Activates Only When Class Imbalance Is Detected, Reflecting Targeted Use of Augmentation.

This sharp contrast highlights a rational design logic in which essential agents remain continuously engaged while specialized modules intervene selectively and only when warranted, ensuring both efficiency and scalability.

8.2. Detection under standard conditions scenario

This baseline scenario replicates typical operational conditions with a balanced distribution of benign and malicious PDF files. Its purpose is to validate the system's core functionality and classification accuracy under controlled and non-adversarial inputs.
The resulting confusion matrix (Figure 8) demonstrates the classifier's effectiveness: 292 benign files were correctly identified, 107 malicious files were accurately flagged, with only one benign instance misclassified and no malicious files undetected. These figures translate to exceptionally high-performance metrics and, notably, a false negative rate of zero, crucial in cybersecurity applications where undetected threats can lead to severe consequences.

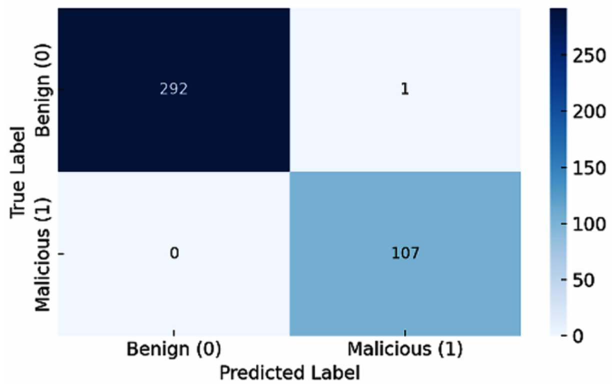

Fig. 8: Confusion Matrix (Standard Scenario). Error Rates Are Very Low, Especially the False-Negative Rate, Consistent with the ROC Results and Supportive of Reliable Detection.

8.3. Scenario 2 – simulated attack with malicious overload

This stress-test scenario examines the MAS's robustness under intense adversarial conditions by varying the proportion of malicious files to 50%, 80%, and 90%. The goal is to observe the system's ability to preserve accuracy and response coherence under escalating threat intensity.
The results, summarized in Table 9, reveal a controlled decline in performance metrics as malicious density increases. Nevertheless, even at the extreme threshold of 90% malicious files, the system maintains an accuracy of 92% and an F1-score of 0.89, evidencing a commendable level of resilience.

Table 9: Detection Performance Under Malicious Overload Conditions

Proportion of Malicious Files	Accuracy	Recall	F1-score
50%	0.99	0.98	0.99
80%	0.96	0.93	0.94
90%	0.92	0.88	0.89

These results underscore the system's capacity to operate effectively under hostile data distributions, with minimal degradation in critical detection metrics.

8.4. Scenario 3 – suspicious user profile adaptation

This scenario explores the behavioral intelligence layer of the MAS by analyzing user-submitted file histories to detect abnormal patterns indicative of elevated risk. Each simulated user profile was tracked for the proportion of submitted files deemed malicious, forming the basis for adaptive security responses.
As illustrated in Figure 9, while most users maintained a moderate submission profile, certain individuals consistently uploaded malicious content at rates exceeding 30%, triggering elevated scrutiny by the UPMA and PLA.

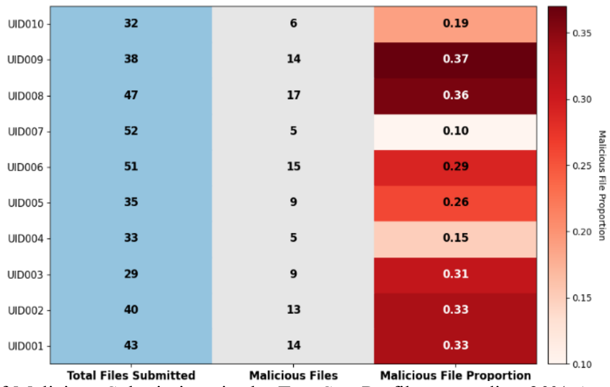

Fig. 9: Distribution of Users by Share of Malicious Submissions in the Test Set. Profiles exceeding 30% Are Flagged as High-Risk and Prioritized for Scrutiny by UPMA and PLA.

User risk was categorized adaptively based on the proportion of malicious files submitted. Users with 10% or fewer malicious files were considered low-risk, triggering normal system behavior. Those with a proportion between 10% and 30% were classified as moderate-risk and placed under monitored status with activity logging. Finally, users exceeding 30% of malicious submissions were deemed high-risk, prompting prioritized scrutiny and alert generation by the system.
This scenario highlights the MAS's ability to integrate contextual behavioral insights into its detection pipeline, enhancing both precision and adaptability in dynamic cybersecurity environments.

9. Discussion

This section offers a reflective analysis of the system's capabilities, interpretability, architectural contributions, and existing limitations, with perspectives for future enhancement.

9.1. Strengths and interpretability of the proposed framework

The proposed hybrid system, combining generative data augmentation, explainable machine learning, and multi-agent coordination, achieves both high predictive performance and strong interpretability. The use of SHAP (Shapley Additive exPlanations) enables detailed local explanations of individual predictions, fostering transparency and trust in high-stakes cybersecurity contexts [53–55].
In particular, SHAP analyses consistently identified features such as JavaScript count, OpenAction triggers, and volume of metadata as dominant predictors of malicious behavior. These findings are consistent with established heuristics in the domain of PDF-based threat detection [56], reinforcing the model's alignment with real-world security insights.

9.2. Added value of CTGAN and agent-based orchestration

An important contribution of the proposed system is the use of CTGAN to address class imbalance, a persistent challenge in cybersecurity datasets where benign files vastly outnumber malicious ones [26]. Unlike classical resampling techniques such as SMOTE, CTGAN learns complex feature dependencies and produces coherent synthetic samples, thus improving both balance and realism.
Complementing this, the multi-agent architecture, coordinated by agents such as the Planning Agent (PLA), Generative Augmentation Agent (GAA), and Machine Learning Agent (MLA), provides modularity and flexibility. This decentralized design allows the system to adapt dynamically to workload variations, an important property for scalable and evolving cyber defense [57].

9.3. Comparative performance and model robustness

The evaluation highlights the clear advantage of ensemble-based approaches. On the original dataset, Random Forest obtained slightly better results, but after CTGAN augmentation, XGBoost consistently outperformed all classifiers, reaching near-perfect scores. This outcome illustrates its effectiveness for structured and feature-rich cybersecurity data [58] and is consistent with prior research emphasizing the resilience of ensemble models, particularly XGBoost, in managing complex decision boundaries, mitigating overfitting, and resisting adversarial noise [54]. The inclusion of a baseline DummyClassifier further confirmed that these strong performances are the result of effective learning rather than chance or dataset bias.

9.4. Limitations and research perspectives

Despite these promising outcomes, several limitations and practical constraints merit attention. The system was evaluated offline on a dataset of 6,343 files and further tested with 400 additional PDFs outside the training set. While the results are close to perfect, confirming them on larger and more diverse collections would further strengthen the system's generalizability. Beyond scale, external validation on heterogeneous corpora and across additional document formats (e.g., Word, Excel) is needed to assess transferability and production readiness. Such an evaluation will clarify how well the approach scales across domains and document types.
In particular, the system's robustness against adversarially crafted PDFs, i.e., files deliberately designed to evade machine-learning detectors, has not yet been assessed. Moreover, CTGAN generation and inter-agent coordination introduce computational overhead that may affect latency under heavy workload. These factors require careful optimization to support large-scale and real-time deployment.
In practical deployments, variability may also arise from noise, evolving attack strategies, and operational constraints. Moreover, the absence of real-time processing in the current implementation reduces responsiveness to fast-moving threats. Nevertheless, thanks to its agent-based design, the system already supports iterative retraining, as shown during the simulation, where the model was updated multiple times, providing a solid foundation for future integration into real-time applications.
Far from undermining the contributions of this study, these considerations highlight the practical challenges inherent to cybersecurity applications and point to promising directions for scaling the system toward real-world use.

10. Conclusion

This study has introduced a modular and intelligent framework for detecting malicious PDF files, built on the combined strengths of generative data augmentation, supervised machine learning, and a multi-agent system (MAS). Unlike previous approaches that typically apply these methods in isolation, our work demonstrates their effective integration within a single, explainable, and scalable architecture.
The use of CTGAN successfully addressed class imbalance, enriching the training data and improving classifier generalization. Among the six models tested, XGBoost consistently delivered near-perfect performance, confirming the robustness of ensemble methods in cybersecurity contexts. At the same time, the MAS architecture provided clear modularity, with agents dedicated to classification, orchestration, and data management, which enhances both scalability and maintainability. To ensure transparency, SHAP analysis highlighted features such as JavaScript presence, metadata volume, and OpenAction triggers as key indicators of malicious activity, providing interpretable insights for analysts.
Overall, this work shows that a MAS-based design can achieve both high accuracy and interpretability, offering a practical solution for strengthening PDF security in government, academic, and enterprise environments. Looking ahead, we will test the pipeline against adversarial PDFs to quantify evasion resistance, integrate the MAS with a SIEM stack for streaming ingestion and alerting, and optimize agent

coordination to reduce latency and support time-critical operations. These steps move the system from offline evaluation toward robust, real-time deployment. In parallel, we will broaden external validation on larger public datasets and additional formats such as Word, Excel, and image-based inputs, and explore continuous adaptation through active learning and specialized security agents for threats like phishing and ransomware.

Taken together, these results suggest that integrating MAS, CTGAN, and ML can meaningfully enhance the robustness of PDF document protection.

Data and Code Availability

The Python code for the multi-agent simulation (Mesa framework), data augmentation (CTGAN), and classification models (scikit-learn) is openly available at: https://github.com/amadoudiabagate/smart-mas-PDF-security.

For illustration purposes, the repository also includes a synthetic dataset that mirrors the feature schema used in this study. The full dataset employed for training and evaluation is not publicly released due to security and confidentiality constraints; however, aggregated or anonymized data may be provided by the corresponding author upon reasonable request.

References

[1] Brown, L. V. (2008). Computer security: Principles and practice. Pearson Prentice Hall.

[2] He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

[3] Kolter, J. Z., & Maloof, M. A. (2006). Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 7(1), 2721-2744. https://doi.org/10.1145/1014052.1014105

[4] Sarker, I.H., Furhad, M.H. & Nowrozy, R. AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions. SN COMPUT. SCI. 2, 173 (2021). https://doi.org/10.1007/s42979-021-00557-0

[5] Ofusori, L., Bokaba, T., & Mhlongo, S. (2024). Artificial intelligence in cybersecurity: a comprehensive review and future direction. *Applied Artificial Intelligence*, *38*(1), 2439609. https://doi.org/10.1080/08839514.2024.2439609.

[6] Abu Al-Haija, Q., Odeh, A., & Qattous, H. (2022). PDF malware detection based on optimizable decision trees. *Electronics*, 11(19), 3142. https://doi.org/10.3390/electronics11193142.

[7] Dehghantanha, A., Yazdinejad, A., & Parizi, R. M. (2023, November). Autonomous cybersecurity: Evolving challenges, emerging opportunities, and future research trajectories. In *Proceedings of the Workshop on Autonomous Cybersecurity* (pp. 1-10). https://doi.org/10.1145/3689933.3690832

[8] Hariharan, B., Siva, R., Sadagopan, S., Mishra, V., & Raghav, Y. (2023, July). Malware detection using XGBoost-based machine learning models: a review. In *2023 2nd International Conference on Edge Computing and Applications (ICECAA)* (pp. 964-970). IEEE. https://doi.org/10.1109/ICECAA58104.2023.10212327.

[9] Torres, J., & Santos, S. D. L. (2018). Malicious PDF document detection using machine learning techniques. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)* (pp. 337-344). https://doi.org/10.5220/0006609503370344.

[10] Wooldridge, M. (2009). *An introduction to multi-agent systems*. John Wiley & Sons.

[11] Alabsi, B. A., Anbar, M., & Rihan, S. D. A. (2023). Conditional tabular generative adversarial based intrusion detection system for detecting ddos and dos attacks on the internet of things networks. *Sensors*, *23*(12), 5644. https://doi.org/10.3390/s23125644.

[12] Singh, P., Tapaswi, S., & Gupta, S. (2020). Malware detection in pdf and office documents: A survey. Information Security Journal: A Global Perspective, 29(3), 134-153. https://doi.org/10.1080/19393555.2020.1723747.

[13] Admass, W. S., Munaye, Y. Y., & Diro, A. A. (2024). Cyber security: State of the art, challenges and future directions. *Cyber Security and Applications*, *2*, 100031. https://doi.org/10.1016/j.csa.2023.100031.

[14] Laskov, P., & Šrndić, N. (2011, December). Static detection of malicious JavaScript-bearing PDF documents. In *Proceedings of the 27th Annual Computer Security Applications Conference* (pp. 373-382). https://doi.org/10.1145/2076732.2076785.

[15] Maiorca, D., & Giacinto, G. (2015). Clustering-based PDF malware detection through dynamic analysis. *Computer Fraud & Security*, 2015(5), 8–16.

[16] Bayer, U., Moser, A., Kruegel, C., & Kirda, E. (2006). Dynamic analysis of malicious code. *Journal in Computer Virology*, 2(1), 67–77. https://doi.org/10.1007/s11416-006-0012-2

[17] Khadim, U., Iqbal, M. M., & Azam, M. A. (2022). A secure digital text watermarking algorithm for Portable Document Format (PDF). *Mehran University Research Journal of Engineering & Technology*, 41(1), 100–110. https://doi.org/10.22581/muet1982.2201.10.

[18] Premarathne, U., Abuadbba, A., Alabdulatif, A., Khalil, I., Tari, Z., Zomaya, A., & Buyya, R. (2016). Hybrid cryptographic access control for cloud-based EHR systems. *IEEE Cloud Computing*, 3(4), 58-64. https://doi.org/10.1109/MCC.2016.76

[19] Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., & Thomas, A. (2015, April). Malware classification with recurrent networks. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 1916-1920). IEEE. https://doi.org/10.1109/ICASSP.2015.7178304.

[20] Damodaran, A., Troia, F. D., Visaggio, C. A., Austin, T. H., & Stamp, M. (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13, 1–12. https://doi.org/10.1007/s11416-015-0261-z.

[21] Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2012). A survey on automated dynamic malware analysis techniques and tools. *ACM Computing Surveys*, 44(2), 1–42. https://doi.org/10.1145/2089125.2089126

[22] Han, H., Giles, C. L., Manavoglu, E., Zha, H., Zhang, Z., & Fox, E. A. (2003, May). Automatic document metadata extraction using support vector machines. In *2003 Joint Conference on Digital Libraries, 2003. Proceedings.* (pp. 37-48). IEEE. https://doi.org/10.1109/JCDL.2003.1204842.

[23] Smutz, C., & Stavrou, A. (2012, December). Malicious PDF detection using metadata and structural features. In *Proceedings of the 28th Annual Computer Security Applications Conference* (pp. 239-248). https://doi.org/10.1145/2420950.2420987

[24] Rieck, K., Trinius, P., Willems, C., & Holz, T. (2011). Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4), 639–668. https://doi.org/10.3233/JCS-2010-0410.

[25] Dabral, S., Agarwal, A., Mahajan, M., & Kumar, S. (2017). Malicious PDF files detection using structural and JavaScript based features. In S. Kaushik, D. Gupta, L. Kharb, & D. Chahal (Eds.), *Information, communication and computing technology* (pp. 149-159). Springer. https://doi.org/10.1007/978-981-10-6544-6_14

[26] Raff, E., Sylvester, J., & Nicholas, C. (2017). Learning the PE header: malware detection with minimal domain knowledge. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec '17)*, 121–132. https://doi.org/10.1145/3128572.3140442

[27] Wang, Y., Cai, W. D., & Wei, P. C. (2016). A deep learning approach for detecting malicious JavaScript code. *Security and Communication Networks*, 9(11), 1520-1534. https://doi.org/10.1002/sec.1441.

[28] Tzermias, Z., Sykiotakis, G., Polychronakis, M., & Markatos, E. P. (2011, April). Combining static and dynamic analysis for the detection of malicious documents. In Proceedings of the Fourth European Workshop on System Security (pp. 1-6). https://doi.org/10.1145/1972551.1972555

[29] Jiang, T., Liu, Y., Wu, X., Xu, M., & Cui, X. (2023). Application of deep reinforcement learning in attacking and protecting structural features-based malicious PDF detector. *Future Generation Computer Systems*, 141, 325-338. https://doi.org/10.1016/j.future.2022.11.015.

[30] Srndic, N., & Laskov, P. (2014). Practical evasion of a learning-based classifier: A case study. In *2014 IEEE Symposium on Security and Privacy* (pp. 197-211). IEEE. https://doi.org/10.1109/SP.2014.20

[31] Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular data using conditional GAN. *Advances in Neural Information Processing Systems*, 32.

[32] Rahman, S., Pal, S., Mittal, S., Chawla, T., & Karmakar, C. (2024). SYN-GAN: A robust intrusion detection system using GAN-based synthetic data for IoT security. *Internet of Things*, 26, 101-212. https://doi.org/10.1016/j.iot.2024.101212.

[33] Natsos, D., & Symeonidis, A. L. (2025). Transformer-based malware detection using process resource-utilization metrics. *Results in Engineering*, 25, 104-250. https://doi.org/10.1016/j.rineng.2025.104250

[34] Ni, M., Li, T., Li, Q., Zhang, H., & Ye, Y. (2016). FindMal: A file-to-file social network based malware detection framework. *Knowledge-Based Systems*, *112*, 142-151. https://doi.org/10.1016/j.knosys.2016.09.004.

[35] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78-87. https://doi.org/10.1145/2347736.2347755.

[36] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324.

[37] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140. https://doi.org/10.1007/BF00058655

[38] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press. https://doi.org/10.1093/oso/9780198538493.001.0001.

[39] Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Prentice Hall.

[40] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges. Information Fusion, 58, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012.

[41] Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106. https://doi.org/10.1007/BF00116251

[42] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297. https://doi.org/10.1007/BF00994018.

[43] Chen, T., & Guestrin, C. (2016, August). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). https://doi.org/10.1145/2939672.2939785.

[44] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). *Classification and regression trees*. Routledge. https://doi.org/10.1201/9781315139470.

[45] Wickramasinghe, I., & Kalutarage, H. (2021). Naive Bayes: Applications, variations and vulnerabilities: A review of literature with code snippets for implementation. *Soft Computing*, 25(3), 2277-2293. https://doi.org/10.1007/s00500-020-05297-6.

[46] Ramadhan, B., Purwanto, Y., & Ruriawan, M. F. (2020, October). Forensic malware identification using Naive Bayes method. In *2020 International Conference on Information Technology Systems and Innovation (ICITSI)* (pp. 1-7). IEEE. https://doi.org/10.1109/ICITSI50517.2020.9264959.

[47] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[48] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

[49] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1), 281-305.

[50] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185), 1-52.

[51] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 1137-1143).

[52] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. Springer. https://doi.org/10.1007/978-1-4614-7138-7.

[53] Chen, J., Song, Y., Wainwright, M. J., & Jordan, M. I. (2018). Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning* (pp. 883-892). PMLR.

[54] Biggio, B., & Roli, F. (2018, October). Wild patterns: Ten years after the rise of adversarial machine learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 2154-2156). https://doi.org/10.1145/3243734.3264418.

[55] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30.

[56] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

[57] Falah, A., Pokhrel, S. R., Pan, L., & de Souza-Daw, A. (2022). Towards enhanced PDF maldocs detection with feature engineering: Design challenges. *Multimedia Tools and Applications*, 81, 41103–41130. https://doi.org/10.1007/s11042-022-11960-x

[58] Das, S., Saha, S., Priyoti, A. T., Roy, E. K., Sheldon, F. T., Haque, A., & Shiva, S. (2021). Network intrusion detection and comparative analysis using ensemble machine learning and feature selection. *IEEE transactions on network and service management*, *19*(4), 4821-4833. https://doi.org/10.1109/TNSM.2021.3138457.