Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Ensuring the Security of the Unobservable: A Novel Ensemble Framework for Enhanced Non-Executable Malware Detection

Amit Kumar Singh[1], Swapnesh Taterh[2]

[1,2]Amity Institute of Information Technology, Amity University Rajasthan, India
Email: Amit79668@gmail.com

The ubiquitous use of PDF files coupled with the increasing sophistication of malware threats necessitates robust detection mechanisms. This research investigates the potency of ensemble learning algorithms—XGBoost, AdaBoost, GradientBoosting, and Neural Networks, also the deployment of MLJAR AutoML framework or the study in fortifying the identification of malware embedded within PDF documents. The study aims to enhance detection accuracy and resilience against evolving malicious PDF-based attacks. Employing ensemble methodologies harnesses the collective strengths of multiple classifiers, amplifying the ability to discern subtle and intricate patterns indicative of malware presence within PDF files. Comprehensive experimentation and analysis across diverse datasets scrutinize the efficacy of each algorithm individually and within ensemble configurations. Results demonstrate compelling enhancements in detection accuracy, showcasing the superiority of ensemble learning over standalone classifiers. The study meticulously evaluates key performance metrics, including precision, recall, and F1-score, to validate the efficacy of the proposed ensemble models in identifying PDF-based malware variants. By shedding light on the significance of feature importance and model fusion, this research elucidates the critical attributes and integration strategies pivotal in strengthening PDF malware detection systems. The findings offer promising insights into bolstering cybersecurity measures tailored specifically for combating threats embedded within PDF files, contributing to a proactive defense against emerging malware landscapes.

Keywords: Malware, Machine Learning, Ensemble Learning, AutoML, malware detection.

1. Introduction

In recent times, there has been a significant increase in intelligent attacks that exploit documents containing malicious code, as the exchange of files becomes more prevalent. While most internet users are cautious about executable files attached to emails or websites, they often overlook the potential danger posed by documents. Consequently, documents have become a preferred channel for delivering malware. Among the various types of document-based malware, attacks utilizing PDF files are particularly prominent due to the flexibility offered by the PDF format compared to other document formats. These malicious PDF documents typically contain binary or JavaScript codes that exploit specific vulnerabilities and carry out malicious actions [1]. Numerous studies have been conducted to detect such malicious PDFs. Previous research efforts have primarily focused on extracting features from documents and applying these features to machine learning models. Commonly used features include PDF structure information, entity properties, metadata information, encoding methods, content properties, and lexicon-based features. While these hand-crafted features have demonstrated success, they require substantial effort in designing them [1].

The development of malware often revolves around goals like extracting private information from compromised machines. Alarming statistics indicate that in 2020, a ransomware incident occurred approximately every 11 seconds globally, targeting businesses [2]. Over the past decade, malware infection rates have nearly tripled, resulting in more than a billion infections[3]. The complexity and sheer size of many harmful programs make it challenging for researchers to comprehend their intricacies. It is crucial to extend the distribution of malicious content beyond web clients and educate users on countering malicious entities to effectively protect them against exploitation [4].

Given the rapid increase in both the quantity and diversity of malware, there is a pressing need for novel approaches that can identify and classify malware more efficiently [3]. Manual heuristic detection alone is neither efficient nor effective due to the rapid pace of malware propagation. Consequently, machine learning methods have gained popularity [4]. These techniques are employed to conduct static malware detection investigations, grouping malware with similar features, and revealing previously unidentified malware by associating them with their parent categories based on proximity. Although various research efforts have applied data mining and machine learning techniques, there remains a need for further exploration in this domain [32].

Malware detection is a vital approach for understanding the objectives and characteristics of various malware specimens, including viruses, spyware, and adware. This methodology plays a crucial role in the development of effective malware detection systems. There are two primary types of malware analysis techniques: static analysis and dynamic analysis. Static analysis tools aim to analyze software without executing its binary code, while dynamic analysis techniques assess the behavior of the malware by executing the binary code in a controlled environment [12].

Ensemble learning models like XGBoost, AdaBoost, and Gradient Boosting often outperform traditional classifiers such as Logistic Regression, Naive Bayes, and KNNs in malware detection due to their inherent strengths in handling complex, nonlinear relationships within data. The fundamental difference lies in their ability to combine multiple weak classifiers to create a robust, accurate prediction model. These ensemble methods excel in capturing intricate patterns and interactions among features, which are crucial in identifying sophisticated malware behaviors that might evade simpler models like Logistic Regression or Naive Bayes.

The ensemble models' effectiveness in malware detection is further accentuated by their capacity to mitigate overfitting and enhance generalization. By aggregating the outputs of various classifiers, ensemble methods reduce individual model biases and errors, resulting in more reliable predictions. Additionally, the diverse nature of ensemble models—integrating different algorithms or variations of a single algorithm—ensures a comprehensive exploration of the feature space, enabling the detection of nuanced and evolving malware threats that might escape the scope of singular classifiers like KNNs.

Moreover, ensemble learning adds a layer of complexity and adaptability to malware detection systems. By leveraging the strengths of multiple models, these ensemble methods can capture different facets of malware behavior, enhancing the overall detection accuracy. They facilitate a more nuanced understanding of malware characteristics, enabling the detection of subtle variations and previously unseen patterns. Furthermore, the flexibility of ensemble methods allows for continuous improvement and adaptation to evolving threats by easily incorporating new classifiers or adjusting existing ones, making them valuable assets in the dynamic landscape of cybersecurity.

In essence, ensemble learning models excel in malware detection by combining the strengths of multiple classifiers, mitigating weaknesses, enhancing generalization, and offering a more nuanced, adaptable approach to identifying complex and evolving malware threats.

2. Related Work

In today's era, video communication and advertising have become commonplace, but many users are unaware of the risks associated with opening malicious video files. Cybercriminals have capitalized on this and adopted video files as an attack vector. The MP4 format, being widely used, presents vulnerabilities that can be exploited across multiple platforms, leading to cyberattacks. Traditional antivirus software with signature-based techniques is limited in detecting unknown malware and zero-day attacks. Machine learning algorithms, however, have shown effectiveness in detecting known and unknown malware across different formats, domains, and platforms. For MP4 files, there is a lack of specialized feature extraction methodologies for detecting unknown malware. This paper introduces three innovative and efficient feature extraction methodologies specifically designed for unknown MP4 file malware detection. Two of the methodologies are file structure-based, while the other is knowledge-based. The methodologies are evaluated through five experiments using six machine learning algorithms and 177 datasets, representing different configurations of feature extraction, representation, and selection. The datasets consist of 6,229 files, including 5,066 benign files and 1,163 malicious files [34]. The first three experiments demonstrate the discrimination and generalization capabilities of the methodologies across multiple configurations for known and unknown MP4 file malware detection. The fourth experiment shows that applying principal component analysis (PCA) on the suggested features can improve time and space complexity and feature resilience while maintaining strong detection and generalization capabilities. In the fifth experiment, the best performing configuration of the methodologies is compared to state-of-the-art generic feature extraction methodologies such as n-grams, MinHash, and representation and transfer learning (using a CNN) for unknown MP4 file malware detection. The results reveal that the proposed configuration outperforms all other state-of-the-art methodologies, achieving an AUC, TPR, and FPR of 0.9951, 0.976, and 0.0 respectively [31].

(Esalm Amer et al. 2020) proposed a method to construct a straightforward behavioral graph for malware by utilizing word embedding to comprehend the contextual relationships between API functions in malware call sequences. They addressed the challenge of constructing a behavioral graph for viruses and presented a technique to group distinct functions with contextual characteristics. Experimental findings demonstrated a clear contrast between call sequences used by legitimate software and malware. Based on this distinction, they introduced a novel Markov chain-based technique for detecting and predicting malware. By simulating the behavior of malicious and benign software API call sequences and generating a semantic transition matrix, they illustrated the true relationship between API functions. Their models achieved an average detection accuracy of 0.990 and a false positive rate of 0.010 [1].

(Ajit Kumar et al. 2017) emphasized the need for effective malware detection techniques to protect businesses and regular users from the spread and evolution of malware. They proposed a machine learning-based method that accurately classifies samples as benign or malicious with minimal computational effort. By combining the raw value and derived values of portable executable header fields, they created an integrated feature set. Multiple machine learning methods such as Decision Tree, Random Forest, KNN, Logistic Regression, Linear Discriminant Analysis, and Naive Bayes were used for malware categorization. The suggested integrated feature set achieved a 10-fold cross-validation classification accuracy of 98.4%. On a unique test dataset, the accuracy of the integrated feature set was observed to be 89.23%, a 15% improvement over the raw feature set alone. Additionally, using only the top 15 features, they achieved 98% and 97% accuracy on the integrated and raw features, respectively. They also conducted classification accuracy tests with only the top N features (N = 5, 10, 15, 20, 25) [2].

(Eduardo de O et al. 2019) highlighted the difficulty of malware identification and the efforts made by companies and governments to mitigate these dangers. They emphasized the importance of accurate categorization algorithms, which can be achieved by utilizing large amounts of available data. However, the lack of accessible large datasets comprising both malicious and non-malicious software poses a challenge, particularly for technologies like deep learning. To overcome this constraint, the authors provided a new and sizable dataset for malware classification that was made available to the public. They then proposed a strategy for training a multiclass classification recurrent neural network (RNN), specifically an LSTM, using this dataset. The accuracy of their model was evaluated on unknown programs, which included six classes and five different forms of malware, achieving an accuracy of 67.60% for assessing unstructured malware data [3].

Ye et al. conducted research on malware identification and utilized typical ML algorithms for processes like character extraction, feature selection, and classification. However, their study did not include crucial characteristics such as file entropy, structural entropy, and certain dynamic attributes like network activity, opcode, and API traces. Additionally, they did not explore deep learning methods or multimodal approaches, which are important areas of research in recent years [9].

Panwal et al. focused their work on analyzing the malicious operations carried out by malware and Ransomware Rats. They employed specific implementations of malware analysis, including static and dynamic analysis. The stages of their research involved static property analysis, interactive behavior analysis, fully automated analysis, and manual code reversing.

They used virtual machines, hash tab tool, virus total, and cuckoo sandbox, and found that the malware was flagged as harmful by nearly 65 out of 72 antivirus engines [11].

Souri et al. provided a study on malware identification methodologies, categorizing them into signature-based and behavior-based methods. However, their study did not include an overview of relevant deep learning algorithms or a categorization of the types of characteristics used in data mining for malware detection and classification [16].

Bazrafshan et al. recognized three dominant approaches for revealing malicious software: signature-based techniques, heuristic-based methods, and behavior-based approaches. They also studied several characteristics for malware detection and analyzed malware concealing approaches. However, their development measures did not take into account dynamic or hybrid techniques [18

Ucci et al. (2019) allocated approaches based on the problem they aimed to solve, the types of characteristics collected from Portable Executable files (PEs), and the ML algorithms they employed. While their study detailed the feature taxonomy, it did not define recent exploration directions, particularly in the areas of deep learning and multimodal techniques [21].

Hashemi et al. introduced a novel method for malware detection that involved extracting unique opcodes from executable files and converting them into digital images. They utilized machine learning algorithms and achieved a 91.9% accuracy rate with their proposed detection approach [22].

Salehi et al. and Han et al. based their strategies on collected API calls. Salehi et al. gathered API calls from various binary formats and trained a classifier using API frequency. They created three different feature sets: 'API calls list,' 'API arguments list,' and 'API and arguments list,' each evaluated independently. The API arguments list performed the best with an accuracy of 98.4% and a false positive rate of approximately 3%. Han et al. used static analysis to extract APIs from the Import Address Table (IAT) database and compared the obtained API sequence to another genome to categorize malware families. Han observed a similarity rate of around 40% for malware in the same family, with a 16% false positive rate [23][24].

Cheng et al. employed WinDbg to examine native API sequences and SVM to detect shellcode malware. They achieved an overall accuracy of 94.37% but had a high false negative rate due to a small training set [25].

Chen et al. examined Wannacry characteristics using records generated by Cuckoo Sandbox. They utilized the phrase frequency-inverse document frequency (TF-IDF) to identify frequent phrases with high scores in the system logs [26].

Liang et al. proposed a behavior-based malware divergent arrangement approach that collected API calls from active malware and built a multi-layered dependencies group based on the connection of API calls' dependencies. This method enabled the determination of the level of similarity between different malware types [27].

The COVID-19 pandemic has led to a significant increase in ransomware attacks, particularly crypto-ransomware, which can cause irreversible data loss and economic harm. To combat these attacks, a new method called Xception ColSeq is proposed, which applies static analysis to detect ransomware. This method involves converting Portable Executable (PE) header files into color images in a sequential vector pattern and using the Xception Convolutional Neural Network (CNN) model for classification. This approach simplifies feature extraction, reduces processing load, and is more resilient against evasion techniques and ransomware evolution. The proposed method was evaluated using two datasets. The first dataset consisted of 1000 ransomware and 1000 benign applications, achieving an accuracy of 93.73%, precision of 92.95%, recall of 94.64%, and F-measure of 93.75%. The second dataset, created by the authors and made publicly available, included 1023 ransomware from 25 active and relevant families, along with 1134 benign applications. This dataset yielded an accuracy of 98.20%, precision of 97.50%, recall of 98.76%, and F-measure of 98.12%. These results outperformed existing methods in literature. Additionally, the authors refined the testing methodology to detect new ransomware families, including zero-day attacks, by incorporating randomly selected benign applications into the test set. This approach ensured representative evaluation performance metrics. The proposed method offers an advantageous technique for ransomware detection, which can effectively protect computer systems from cyber threats [30].

3. Artificial Intelligence in Malware Detection

Artificial intelligence has revolutionized the field of malware analysis, empowering cybersecurity professionals with advanced detection, classification, and behavioral analysis capabilities. By leveraging machine learning, automation, and threat intelligence, AI systems can detect and respond to malware threats faster and more accurately, reducing the risk of successful cyberattacks. As cybercriminals continue to develop sophisticated malware strains, AI will remain a critical tool in the fight against evolving cybersecurity threats [28].

Machine learning is a branch of artificial intelligence that uses algorithms to learn from data and make predictions or decisions without being explicitly programmed. In malware analysis, ML can be used to classify malware samples, identify new and unknown malware, and detect malware behavior.

ML algorithms can be trained on large datasets of known malware samples to learn the characteristics and behavior of malware. Once trained, the ML algorithms can be used to classify new samples as malware or benign. ML can also be used to identify new and unknown malware by detecting patterns in the data that are not present in known malware samples.

ML can also be used to detect malware behavior by analyzing system calls and other runtime data. ML algorithms can learn the normal behavior of a system and detect deviations from that behavior that may indicate the presence of malware. ML can also be used to detect specific types of malwares, such as ransomware, by analyzing their unique behavior patterns [27].

This study focuses more on Ensemble learning algorithms, as they demonstrate exceptional efficiency in the binary classification of PDF malware due to their capacity to amalgamate diverse classifiers, amplifying the accuracy and robustness of detection systems. Leveraging techniques like XGBoost, AdaBoost, and Gradient Boosting allows for the synthesis of multiple models, each contributing unique insights into identifying malicious traits within PDF files. Moreover, there is a comparative study between NaiveBayes, K Nearest Neighbors and Logistic Regression. To add onto the ML tools, the research deploys the MLJAR AutoML framework for the efficient ensemble stacking of various algorithms with the finetuned weights for optimal classification of malware samples. This ensemble approach enhances the system's ability to discern intricate patterns and anomalies specific to PDF-based malware, surpassing the limitations of singular classifiers. By combining the strengths of individual algorithms, ensemble methods create a formidable defense, improving accuracy and adaptability in swiftly detecting and categorizing PDF files harboring malicious content.

4. Methodology

4.1. Dataset description

Over the years, PDF has been the most widely used document format due to its portability and reliability. Unfortunately, PDF popularity and its advanced features have allowed attackers to exploit them in numerous ways. There are various critical PDF features that an attacker can misuse to deliver a malicious payload. Evasive-PDFMal2022 which consists of 10,025 records with 5557 malicious and 4468 benign records that tend to evade the common significant features found in each class. This makes them harder to detect by common learning algorithms.

Data collection and analysis

11,173 malicious files from Contagio, 20,000 malicious files from VirusTotal, and 9,109 benign files from Contagio. Once collected, 32 features from each were extracted, and after deduplicating the records, combined the two dataset records into one final file, which resulted in a more representative dataset of the PDF distribution. Moreover, K-means was employed, an unsupervised machine learning that clusters the resource data points into two groups by their similarity. The samples falling into the wrong cluster with the malicious label are taken as an evasive set of malicious records, with an intuition that the features of these samples were not so similar with the rest of the class so that they are not clustered with the majority of the same label samples. Thesame logic was applied for the benign records and finally combined the results with the new "Evasive-PDFMal2022".

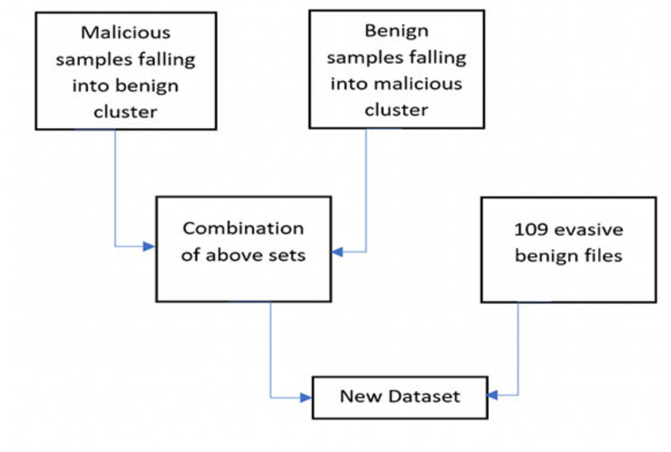

Malicious samples falling into benign cluster
Benign samples falling into malicious cluster
Combination of above sets
109 evasive benign files
New Dataset

Figure 1: Dataset Generation

Proposed features of the dataset:

37 static representative features, including 12 general features and 25 structural features extracted from each PDF file, are depicted in the table.

General features: These include the size, number of pages, content type (text or graphics), and title of the PDF file, among other details that usually characterize it. This category has 12 characteristics.

1. Character count in the title: Proper and more significant titles are often found in legitimate PDF files.

2. Metadata size: This is the part of the PDF file that contains information that may be used to include hidden content.

3. Document Encryption: This function indicates if a PDF document requires a password or not.

4. Page count: Because malicious PDF files don't care about content presentation, they often contain fewer pages—the majority of them are blank.

5. The existence of text within the PDF: Malicious PDF files may include less text because their goal is not to convey material.

6. Total document size: Because to differences in page size and content, malicious PDFs typically have a larger file size than benign ones.

7. Total number of embedded files: PDFs have the ability to attach or embed a variety of resources—such as doc files, photos, and other files—within their documents that might be exploited.

8. The average size of all embedded media: Depending on what they include, embedded files in PDFs might have different sizes. The average size may provide information about the embedded files' content.

9.Total number of objects in the PDF: Since PDFs are composed of objects, the total number of objects along with the remaining characteristics may be used to represent the PDF as a whole.

10.Number of font objects: The sorts of fonts used for the PDF text are indicated by the font objects.

11.Existence of a legitimate PDF Header: Malicious PDF files often alter the header format because PDF header obfuscation is a frequent way to avoid anti-virus scanning.

12.Document image count: PDF files have the ability to include one or more pictures.

Structural characteristics: These features give an overview of the general skeleton of the PDF and characterize the file in terms of its structure, which calls for a more thorough parsing. Regarding the PDF structure, we provide a set of twenty-five characteristics.

1. The quantity of indirect objects: This might be a sign of an effort at obfuscation.

2. Opaqueness of obfuscations: PDFs can accommodate a wide range of obfuscations,

including hex and octal string obfuscations, which are typically used for evasion efforts.

3. Count of streams: This indicates how many binary data sequences there are in the PDF.

4. Endstream count: The number of keywords that indicate the stream's end.

5. Average Stream Size: The size of the stream where the potentially harmful code is concealed

6. Count of stream objects (ObjStm): The total number of streams that hold other objects.

7. Number of Javascript keywords: As is obvious, this indicates how many objects include Javascript code, making it the most often abused aspect.

8. JS keyword count: Total number of objects with Javascript code.

9. Total number of launch keywords: The launch keyword can be used to run a programme or issue a command.

10. The quantity of URI keywords denotes the existence of a URL that the PDF file is attempting to connect to.

11. Number of Action Keywords: Indicates what to do in response to an occurrence.

12. Number of AA keywords: Indicates what to do in response to an occurrence.

13. The quantity of OpenAction keywords designates a particular action to be taken when the PDF file is opened. Most common malicious PDF files have been found to have this feature along with Javascript.

14. The number of SubmitForm tags designates the PDF button that gathers form data and forwards it to designated locations.

15. Acrobat form tags: Acrobat forms are PDF files with form fields that enable scripting technologies, which an attacker could exploit.

16. Total number of filters used: Some PDF objects have a variety of compression filters applied to them, which an attacker could potentially take advantage of.

17. The JBig2Decode filter is present: This filter is frequently used to encrypt harmful content.

18. The quantity of objects with nested filters: Since nested filters impede decoding, they may be a sign of evasion.

19. XFA: Included in some PDF 40 files, XFAs are XML Form Architectures that support scripting technologies that an attacker could exploit.

20. Colours: The PDF uses a variety of colours.

21. Trailer: The quantity of trailers packed into the PDF.

22 Xref: The quantity of Xref tables.

23. Startxref: The quantity of keywords that contain "startxref," which indicates the beginning of the Xref table.

24. Xref entries: Another frequent finding in malicious PDF files is the quantity of entries in the PDF Xref tables that are malformed.

25. RichMedia: The quantity of RichMedia keywords indicates the quantity of flash files and embedded media.

4.2 Feature Extraction Techniques

4.2.1 Correlation Matrix:

A correlation matrix is a potent tool for feature extraction in a dataset of Malware PDFs due to its ability to unveil relationships between various features. In the context of Malware PDF analysis, this matrix helps identify which features are highly correlated or influential in distinguishing between benign and malicious files. By quantifying the degree and direction of relationships among features, it pinpoints significant attributes crucial for classification, such as embedded scripts, file size, metadata details, or specific content structures indicative of malware.

This method aids in selecting the most informative and discriminative features, streamlining the feature set for subsequent analyses. For Malware PDFs, where distinguishing subtle differences is crucial, a correlation matrix assists in isolating key characteristics that serve as red flags for malicious intent. Consequently, it streamlines the feature selection process, optimizing model performance by focusing on the most relevant attributes essential for accurate malware detection in PDF files.

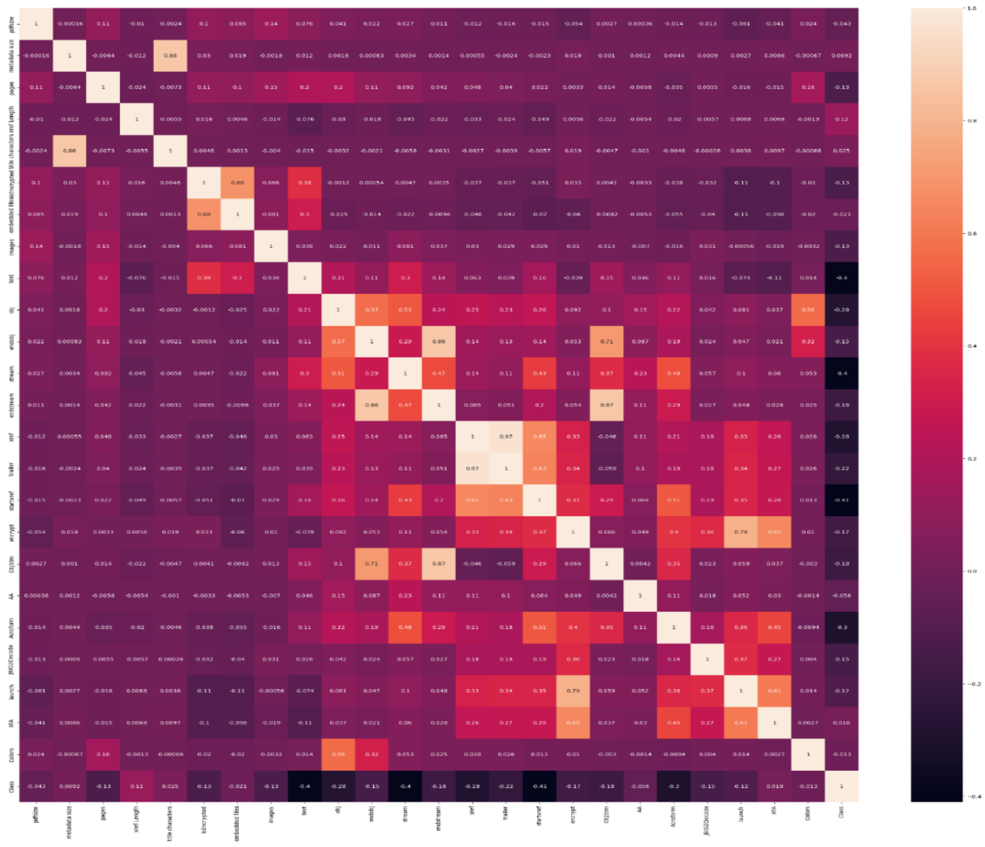


Figure 2 Correlation heatmap

4.2.2 ANOVA f-test feature selection

Performing ANOVA F-test Feature Selection proves effective for extracting features from a dataset of Malware PDFs due to its robustness in identifying pertinent attributes crucial for classification. In the realm of malware detection within PDFs, where distinguishing subtle patterns is vital, ANOVA F-test serves as a powerful tool. By evaluating the statistical significance of each feature's variance across different classes of PDFs (malware vs. benign), ANOVA F-test discerns the discriminative power of individual features. This process aids in selecting the most relevant attributes that distinctly characterize malware instances, optimizing the feature set for subsequent classification models. Moreover, in the context of PDF-based malware, where identifying nuanced markers is pivotal, ANOVA F-test excels in capturing subtle differences in features, ensuring a more precise and effective characterization of malicious PDFs. Its ability to focus on significant feature subsets enhances the accuracy and efficiency of malware classification systems tailored for PDF file analysis.

Below are the scores for the ANOVA f-test feature selection:

```
Feature 0: 1.363742
Feature 1: 94.024403
Feature 2: 118.766062
Feature 3: 4.472738
Feature 4: 4.310248
Feature 5: 83.286306
Feature 6: 71.798283
Feature 7: 411.074096
Feature 8: 80.852705
Feature 9: 1021.008699
Feature 10: 119.397852
Feature 11: 363.538904
Feature 12: 165.350587
Feature 13: 1638.658130
Feature 14: 242.578490
Feature 15: 3.603784
Feature 16: 129.506002
Feature 17: 145.307145
Feature 18: 260.949416
Feature 19: 14.714117
Feature 20: 2349.589858
Feature 21: 403.688496
Feature 22: 41.486689
Feature 23: 21.294474
Feature 24: 64.408785
Feature 25: 0.035564
Feature 26: 334.753626
Feature 27: 1.325239
```

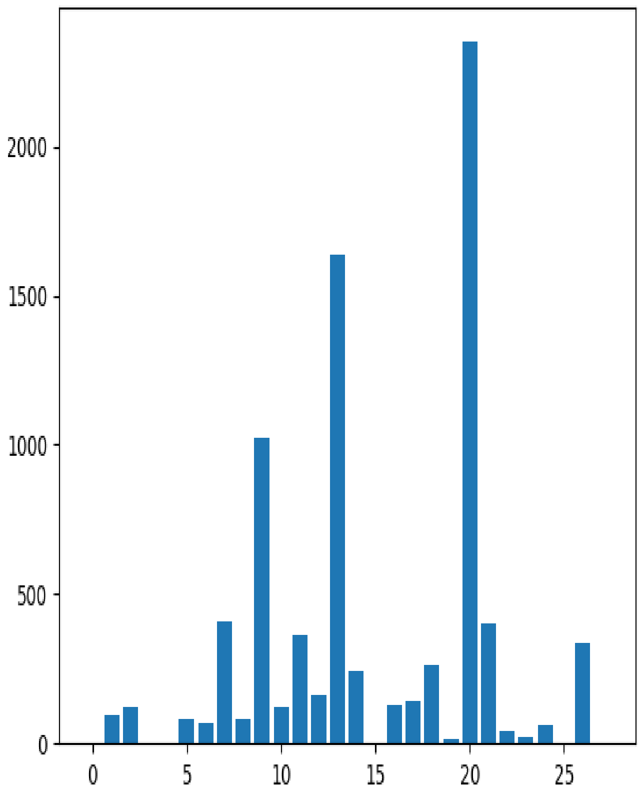


Figure 3 ANOVA f-test feature selection scores

4.3 Machine Learning models

4.3.1 Decision Tree

A decision tree partitions data by asking sequential questions, selecting the most informative features at each node to create a tree-like structure. It iteratively splits the dataset based on feature thresholds, aiming to maximize homogeneity within resulting subsets, ultimately enabling classification or regression based on learned rules.

Decision trees serve as an effective model for PDF malware detection in machine learning due to their interpretability, scalability, and ability to discern complex relationships within features. Specifically, in PDF analysis, decision trees excel in capturing hierarchical feature interactions that signal malware presence. Their transparent structure allows insights into feature importance, aiding in understanding how certain attributes contribute to classification. Additionally, decision trees adapt well to diverse feature types common in PDF analysis, making them versatile for identifying intricate patterns within PDF files, thus rendering them suitable for robust and accurate malware detection in this domain.

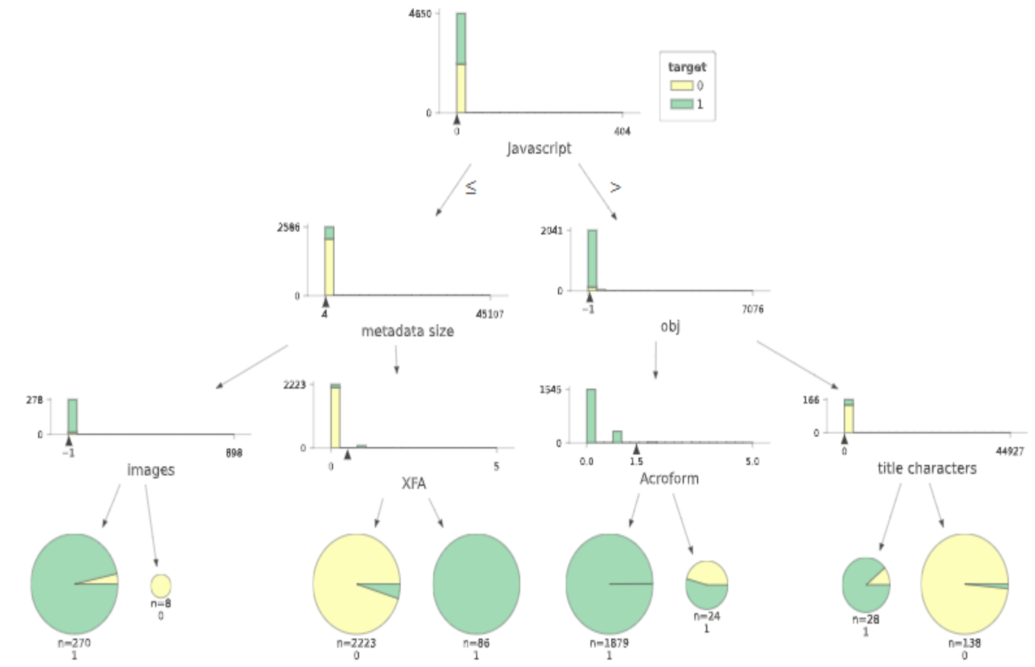


Figure 4: Decision Tree 1

Parameters set for the Decision Tree:

classification_default_params = {"criterion": "gini", "max_depth": 3}

The Shap important features were also found in the training process of Decision Trees as below

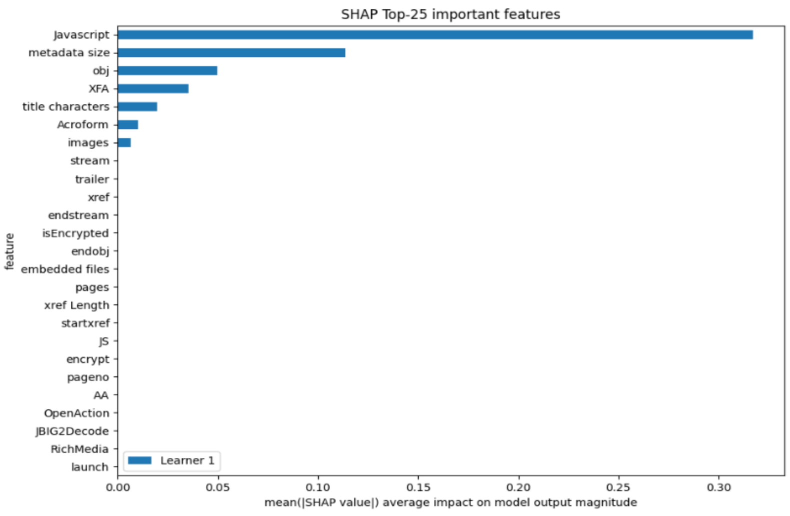


Figure 5: Shap important features - Decision Tree

4.3.2 Light Gradient Boosting

Light Gradient Boosting Machine (LightGBM) operates by building an ensemble of decision trees in a gradient boosting framework. It employs a leaf-wise strategy for tree growth, selecting the leaf with the maximum loss reduction. LightGBM optimizes based on the gradient of the loss function, uses histograms to speed up computations, and employs a novel technique called Gradient-Based One-Side Sampling (GOSS) for efficient training, resulting in fast, accurate predictions. Light Gradient Boosting Machines (LightGBM) excel in classifying Malware PDFs due to their efficiency in handling large datasets, accommodating varied feature types common in PDF analysis. Their leaf-wise growth strategy optimizes performance, swiftly capturing intricate patterns indicative of malware while reducing overfitting. LightGBM's ability to handle high-dimensional data and its inherent speed make it adept at discerning nuanced features, enhancing accuracy in distinguishing Malware PDFs from non-malicious ones.

The Parameters for the LightGradientBoosting:

```
 classification_bin_default_params = {
"objective": "binary", "metric": "binary_logloss", "num_leaves": 31,  "learning_rate": 0.1,
"feature_fraction": 0.9, "bagging_fraction": 0.9, "min_data_in_leaf": 10 }
```

The Learning curve for the LightGradientBoosting is as follows:

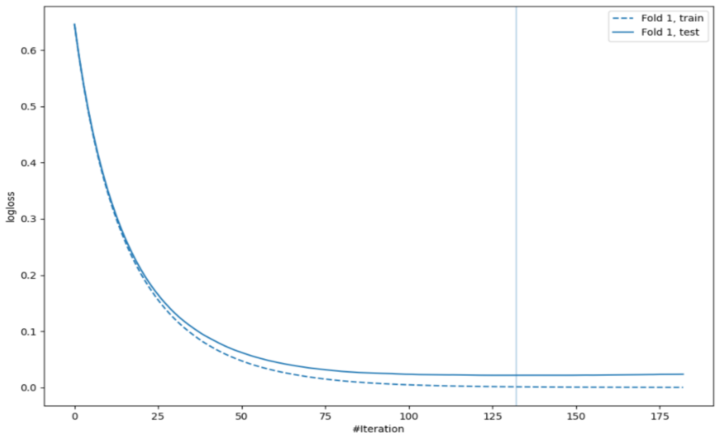


Figure 6: Learning curve for LightGradientBoosting

The Shap Important features used for the training of LightGradientBoosting:

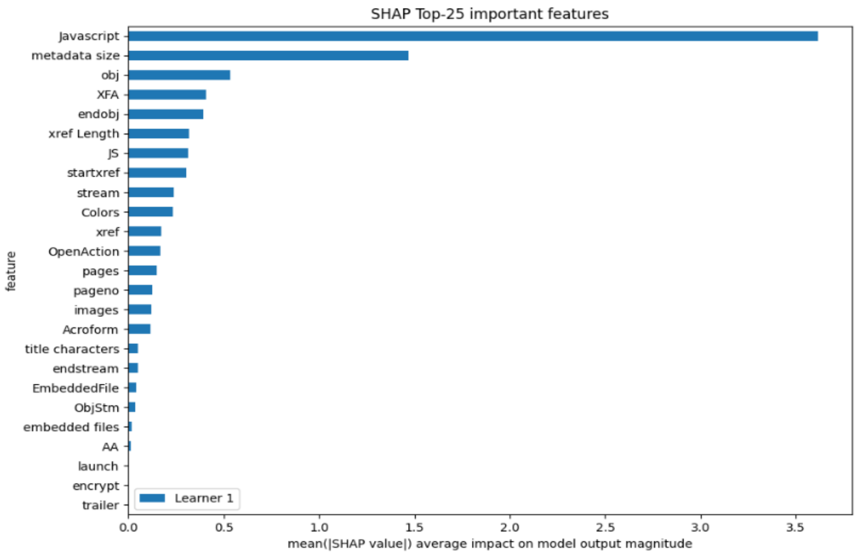


Figure 7: Shap Important Features LightGradientBoosting

4.3.3 XtremeGradientBoost

XtremeGraientBoost operates by iteratively constructing multiple decision trees, minimizing prediction errors. It employs a gradient boosting framework, sequentially adding trees to correct previous models' mistakes. Each tree compensates for the residuals of the preceding ones, enhancing the overall predictive power. XGBoost employs regularized learning to control model complexity, using gradient descent optimization to optimize an objective function, ensuring the creation of a robust ensemble of trees, resulting in highly accurate predictions across diverse datasets. XGBoost, with its feature importance analysis, efficiently identifies crucial attributes within Malware PDFs. By discerning the most relevant features, it constructs decision trees to capture intricate patterns indicative of malware. Through iterative refinement and focusing on these vital traits, XGBoost optimizes classification accuracy, effectively distinguishing Malware PDFs from benign ones by leveraging the right set of discriminative features.

The Parameters for the XtremeGradientBoost Algorithm:

classification_bin_default_params = { "objective": "binary:logistic", "eval_metric": "logloss", "eta": 0.1, "max_depth": 6, "min_child_weight": 1, "subsample": 1.0, "colsample_bytree": 1.0 }

The Learning curve for the XtremeGradientBoost is as follows:

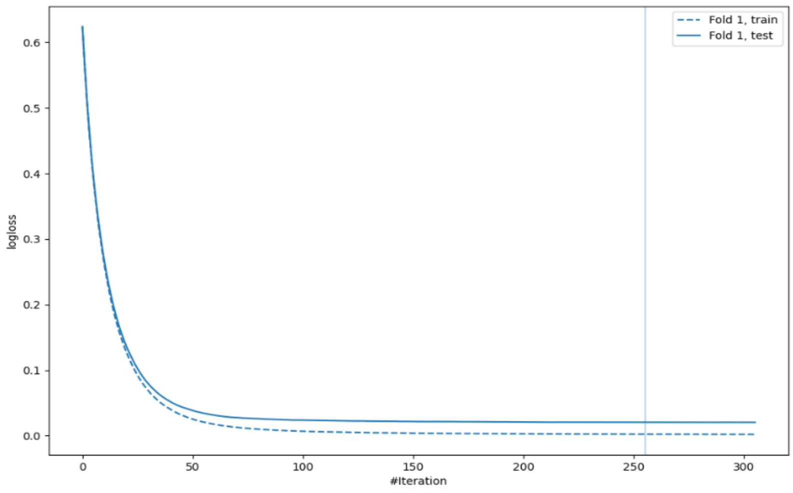


Figure 8: Learning curve for XtremeGradientBoost

The Shap Important features used for the training of XtremeGradientBoost:

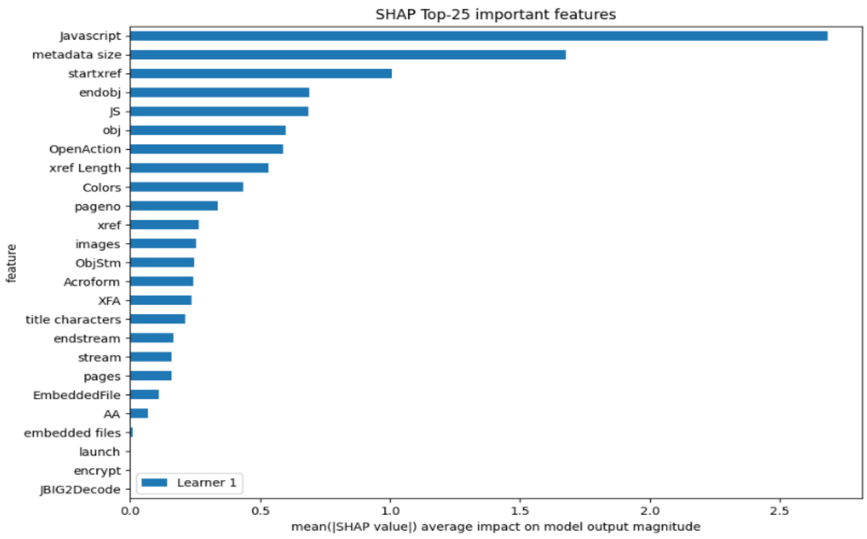


Figure 9: Shap Important features of XtremeGradientBoost

4.3.4 Random Forest

Random Forest algorithm operates by constructing multiple decision trees during training. Each tree is built using a random subset of the dataset and a random subset of features. Through a process of voting or averaging among these trees, Random Forest makes predictions. It aggregates the outputs of individual trees to determine the final classification or regression result, reducing overfitting and enhancing accuracy by leveraging the collective wisdom of diverse trees in the forest. Random Forest proves highly beneficial in binary classification of Malware PDFs by leveraging ensemble learning. It aggregates multiple decision trees, each trained on varied subsets of features and data. This approach fosters robustness against overfitting and noise, enhancing accuracy in discerning complex patterns indicative of malware traits within PDF files. By collectively voting on classifications, Random Forest effectively harnesses feature importance, enabling precise identification of distinguishing characteristics, ultimately facilitating reliable differentiation between Malware PDFs and benign ones.

The parameters for the Random Forest Algorithm:

classification_default_params = {"criterion": "gini","max_features": 0.6, "min_samples_split": 30, "max_depth": 6 }

The learning curve of Random Forest during training:

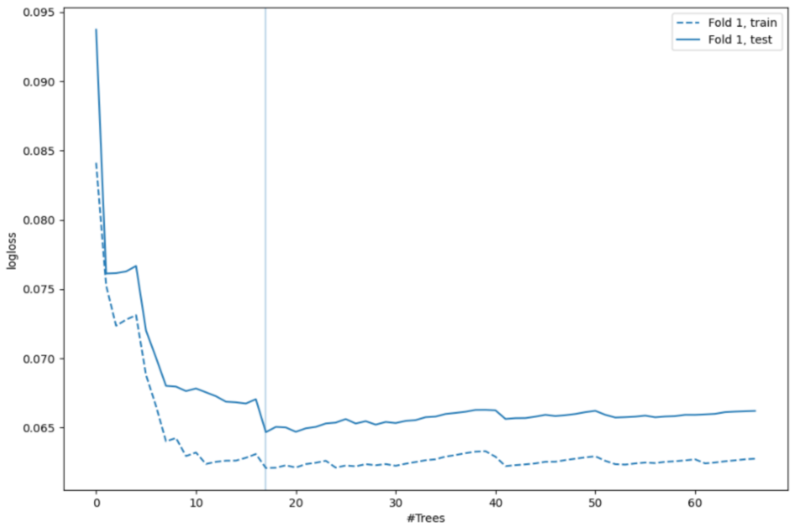


Figure 10: The learning curve of Random Forest

4.3.5 K Nearest Neighbors

K Nearest Neighbors (KNN) classifies data based on similarity measures. When tasked with a new sample, it identifies the K closest instances in the training set, determining the predominant class among its neighbors to assign the sample to that class. The algorithm's simplicity lies in its reliance on proximity-based decision-making, where the class of a point is determined by the consensus of its neighboring data points in the feature space, making it a straightforward yet effective classification method.

K Nearest Neighbors (KNN) employs feature similarity to classify PDFs. By measuring distances between feature vectors of known benign and malicious PDFs, KNN identifies the K closest neighbors to an unlabeled PDF. The algorithm then assigns a class based on the majority label among these neighbors. In the context of PDF malware detection, KNN utilizes feature similarity to categorize new PDFs by comparing their feature patterns to those of known malware and benign PDFs, determining their classification based on the closest feature matches within the dataset.

The parameters for the training of KNN:

default_params = { "n_neighbors": 5, "weights": "uniform" }

4.3.6 Neural Networks

Neural Networks for binary classification employ interconnected layers of nodes to process data. Input features are fed into the network, each node processing information and passing it to subsequent layers through weighted connections. Through a process of forward propagation, these weights are adjusted iteratively using training data to minimize error. With activation functions aiding in non-linear transformations, the network learns to discern patterns in the data. Finally, for binary classification, the output layer typically employs a sigmoid function, assigning probabilities for the data belonging to each class.

Neural Networks classify Malware PDFs from benign ones by leveraging features extracted from the PDF files. Initially, feature extraction from PDFs, such as metadata, structural components, or content-based features, is crucial. These features form the input data fed into the Neural Network. Through training, the network learns patterns distinguishing Malware PDFs from benign ones by adjusting weights in interconnected layers. It discerns complex relationships within the features to identify malicious traits. With a binary classification output layer, the network assigns probabilities, determining if a given PDF exhibits characteristics indicative of malware, enabling effective differentiation between benign and malicious PDFs.

The Parameters for Neural Networks during training:

default_nn_params = {"dense_layers": 2,"dense_1_size": 32,"dense_2_size": 16,"dropout": 0,"learning_rate": 0.05,"momentum": 0.9,"decay": 0.001}

The Learning curve of Neural Networks during the training:

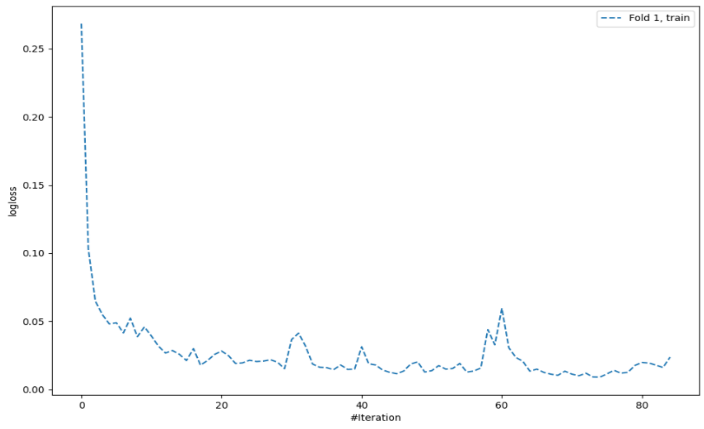


Figure 11: Learning curve for neural networks

4.3.7 Ensemble Stacking Model

Ensemble Stacking combines multiple diverse models in a two-layered approach for improved predictive performance. In this method, base models generate predictions on the dataset, which become input for a meta-model or second-layer model. The meta-model learns to combine these diverse predictions to produce the final output, leveraging the collective wisdom of the base models. This approach harnesses different modeling techniques, capturing varied aspects of the data's complexity. Stacking mitigates individual model weaknesses and enhances overall predictive power by enabling the meta-model to learn from the strengths and weaknesses of the diverse base models, fostering more accurate and robust predictions in machine learning.

The Stacked process is shown below:

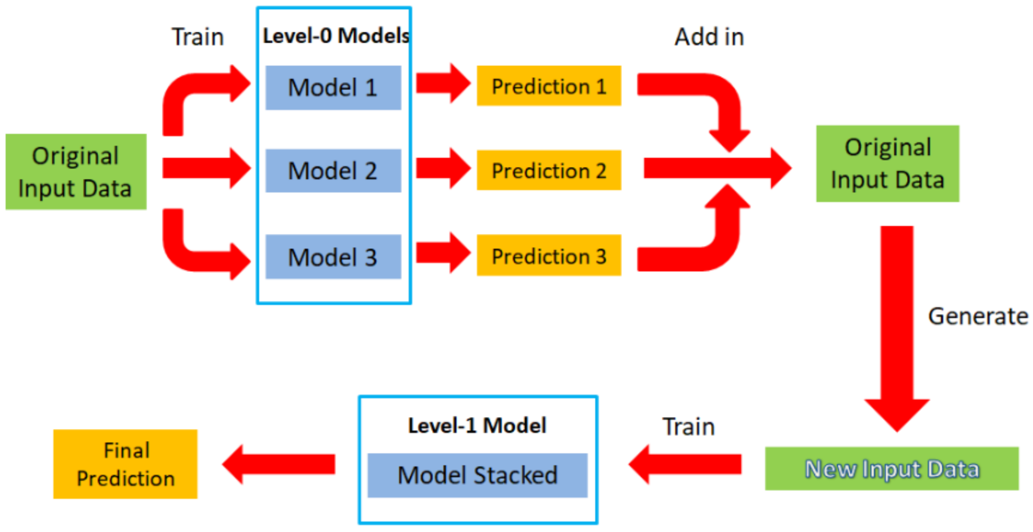


Figure 12: Ensemble Stacking using MLJAR AUTOML

In the Ensemble stacked model, there is a dual layer stacking, the first layer comprises of the 4 models chosen as mentioned below with the assigned weights and their predictions are accounted as per the assigned weights. The weights for each of these models are assigned after finetuning multiple combinations of the best performing models. The second layer takes the input from the first layer's predictions and gives the final prediction.

The parameters for the Ensemble Stacking:

Model	Weight
LightGradientBoosting	2
XtremeGradientBoosting	1
RandomForest	1
NeuralNetwork	3

The learning curve for the Ensemble Model:

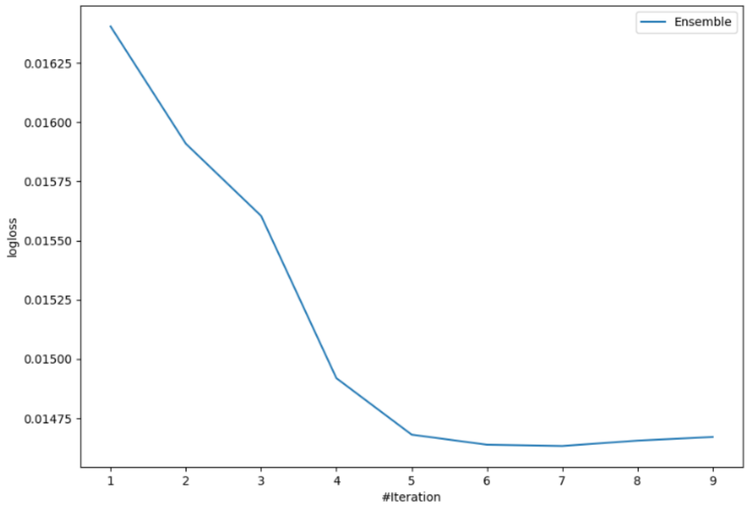


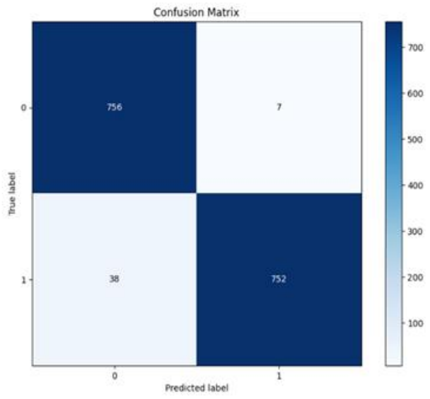
Figure 13: Learning curve for the Ensemble Model

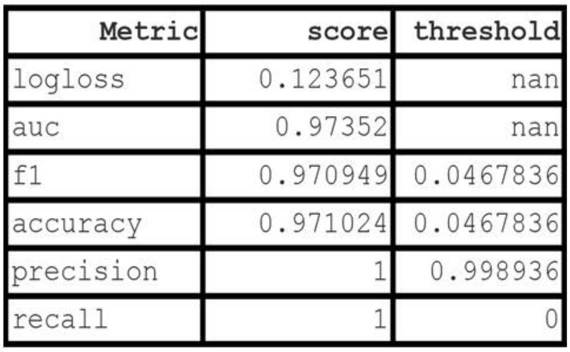
5. Results and Analysis

The performance of various ML models, including XGBoost, Gradient Boosting, Neural Networks, Random Forest, KNNs, Decision Tree, and Ensemble Stacking, is meticulously evaluated for PDF malware detection. The evaluation encompasses diverse metrics—Accuracy, F1 score, Recall, Precision—across multiple datasets. Each model's efficacy in correctly classifying malware versus benign PDFs is scrutinized using confusion matrices to visualize classification performance. Comparative analyses highlight the strengths and weaknesses of individual models, showcasing their ability to capture subtle malware traits and generalization across different datasets. The section elucidates how ensemble techniques like stacking harness collective model insights to potentially outperform individual classifiers, offering a comprehensive assessment of their effectiveness in combating PDF-based malware threats.

A confusion matrix is a fundamental tool in evaluating the performance of classification models in machine learning. It is a tabular representation that illustrates the performance of a model by comparing predicted and actual values across different classes. The matrix consists of four sections: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP signifies instances where the model correctly predicts the positive class, while TN represents correct predictions of the negative class. FP indicates instances where the model incorrectly predicts the positive class, and FN represents instances incorrectly classified as the negative class. From this matrix, various performance metrics are derived, including accuracy, precision, recall (sensitivity), specificity, and F1 score. These metrics provide insights into the model's ability to correctly identify instances of each class and its overall effectiveness in classification. The confusion matrix aids in understanding model strengths and weaknesses, facilitating adjustments or improvements to enhance predictive accuracy and reliability.

5.1 Decision Tree Results:



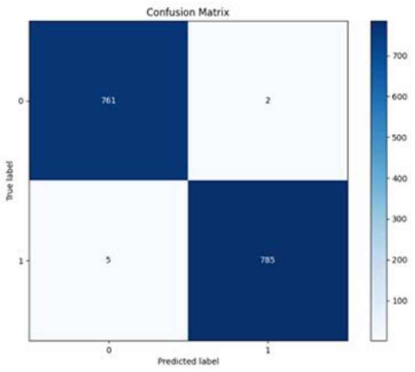


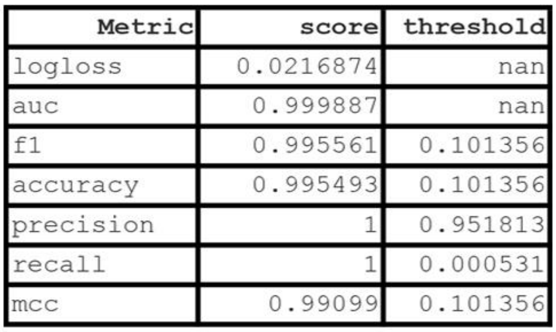
Metric	score	threshold
logloss	0.123651	nan
auc	0.97352	nan
f1	0.970949	0.0467836
accuracy	0.971024	0.0467836
precision	1	0.998936
recall	1	0

Figure 14: Decision Tree Confusion Matrix and Performance

This model has high interpretability, scalability and a strong ability to discern complex relationships within features, therefore has high precision, Decision Tree performs well with an accuracy of 97%.

5.2 LightGradientBoosting Results:



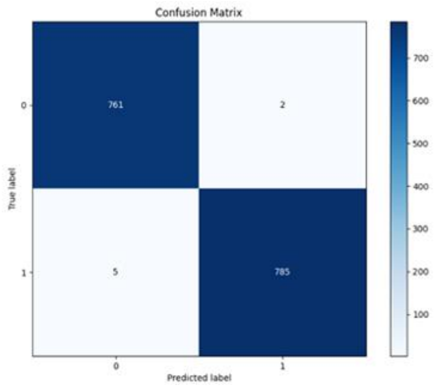


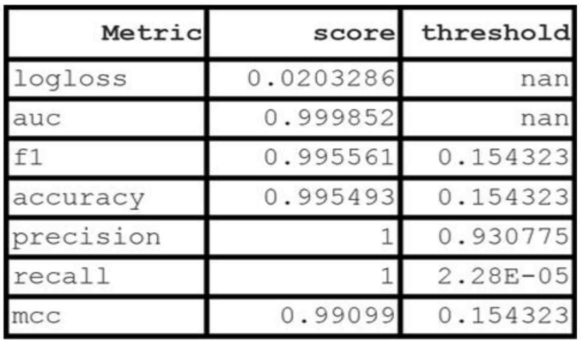
Metric	score	threshold
logloss	0.0216874	nan
auc	0.999887	nan
f1	0.995561	0.101356
accuracy	0.995493	0.101356
precision	1	0.951813
recall	1	0.000531
mcc	0.99099	0.101356

Figure 15: LightGradientBoosting Confusion Matrix and Performance

Light Gradient Boosting performs ensemble learning, the model has a leaf-wise growth strategy along with Gradient-Based One Side Sampling (GOSS) and hence is more optimized, performs exceptionally well with an accuracy of 99.5%

5.3 XGBoost Results:



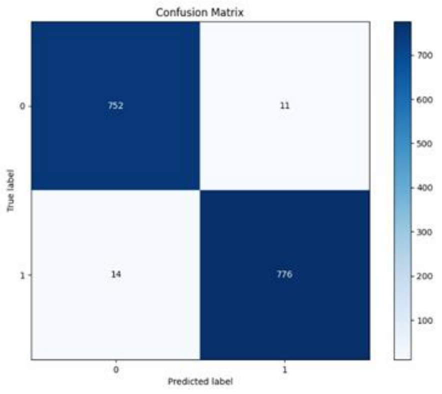


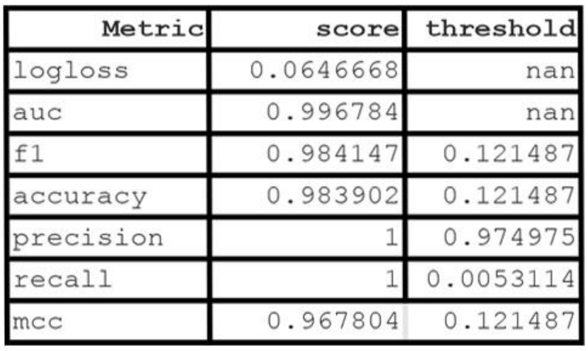
Metric	score	threshold
logloss	0.0203286	nan
auc	0.999852	nan
f1	0.995561	0.154323
accuracy	0.995493	0.154323
precision	1	0.930775
recall	1	2.28E-05
mcc	0.99099	0.154323

Figure 16: XGBoost Confusion Matrix and Performance

XG Boost, also being an ensemble learning algorithm, deploys sequential addition of trees to correct the errors of the previous models. XG Boost performance is similar to that of LightGradientBoosting, with and accuracy of 99.5% but is more efficient than LightGradientBoosting because of the lower log loss.

5.4 Random Forest Results:



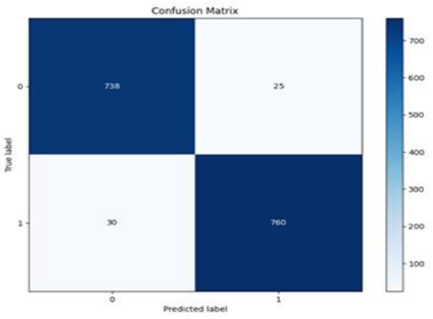


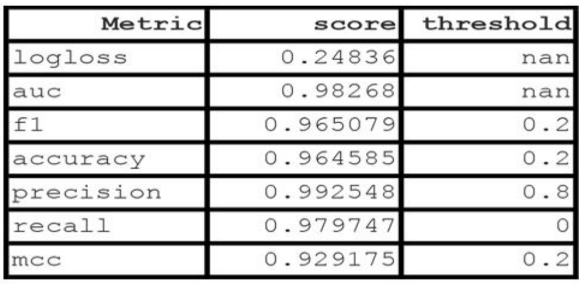
Metric	score	threshold
logloss	0.0646668	nan
auc	0.996784	nan
f1	0.984147	0.121487
accuracy	0.983902	0.121487
precision	1	0.974975
recall	1	0.0053114
mcc	0.967804	0.121487

Figure 17: Random Forest Confusion Matrix and Performance

Random Forest performs well and gives a good accuracy of 98.3%. Although the logloss is significantly higher than that of the previous models and the higher logloss. A lower log loss value means better predictions of the malware samples.

5.5 K Nearest Neighbors Results:



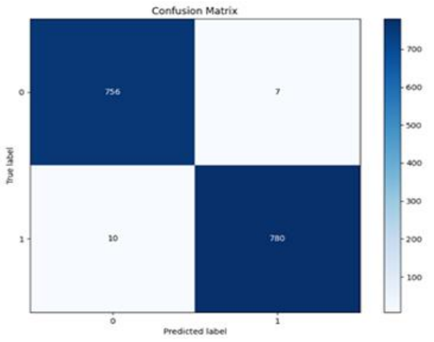


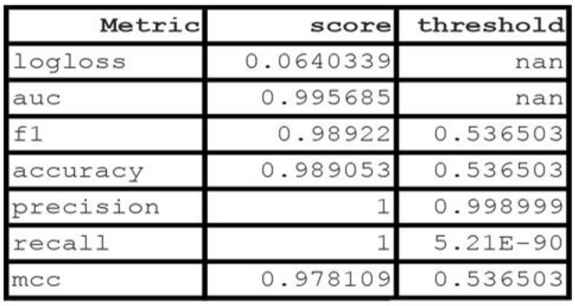
Metric	score	threshold
logloss	0.24836	nan
auc	0.98268	nan
f1	0.965079	0.2
accuracy	0.964585	0.2
precision	0.992548	0.8
recall	0.979747	0
mcc	0.929175	0.2

Figure 18: KNearestNeighbors Confusion Matrix and Performance

KNN has an accuracy of 96,4% which is less than that of Random Forest, there is a significant increase in log loss metric for the KNN model. KNN proves to the least efficient model of the chosen models for this problem statement.

5.6 Neural Networks Results:



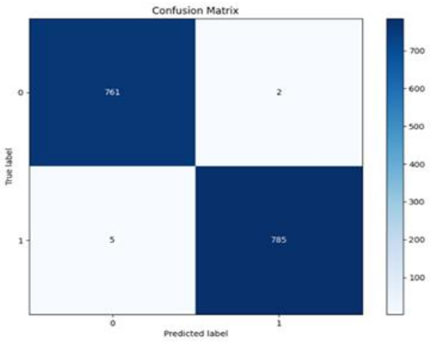


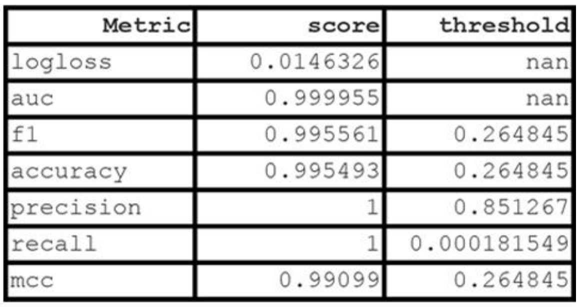
Metric	score	threshold
logloss	0.0640339	nan
auc	0.995685	nan
f1	0.98922	0.536503
accuracy	0.989053	0.536503
precision	1	0.998999
recall	1	5.21E-90
mcc	0.978109	0.536503

Figure 19: Neural Networks Confusion Matrix and Performance

The Neural Networks performance is similar to that of Random Forest, this model is precise, has lower logloss and a good accuracy of 98.9% which is higher than that of Random Forest.

5.7 Ensemble Stacked Model Results:





Metric	score	threshold
logloss	0.0146326	nan
auc	0.999955	nan
f1	0.995561	0.264845
accuracy	0.995493	0.264845
precision	1	0.851267
recall	1	0.000181549
mcc	0.99099	0.264845

Figure 20: Ensemble Stacked model Confusion Matrix and Performance

The ensemble stacked model, comprising Layer-0 models—XGBoost, Neural Networks, Light Gradient Boosting, and Random Forest—displayed superior performance. Employing varying weights, it achieved outstanding results: 99.54% accuracy and 100% precision. This amalgamation of diverse models within the ensemble stacking framework showcased exceptional predictive power, leveraging the strengths of individual classifiers to achieve remarkably high accuracy and precision in classification tasks.

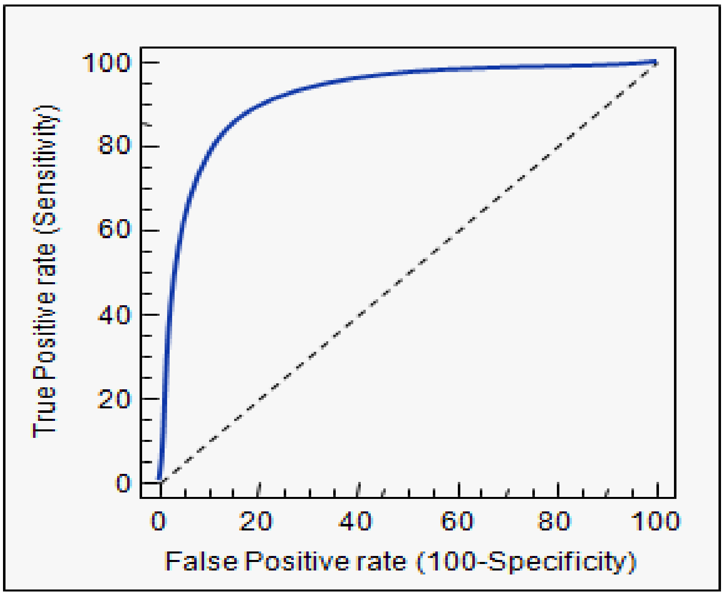


Figure 21: ROC curve of the ensemble stacked model

A Receiver Operating Characteristic (ROC) curve for 99 indicates the model's strong ability to distinguish between classes. With a high Area Under the Curve (AUC) close to 1, it suggests excellent performance, achieving minimal false positives while maximizing true positives. This signifies robust predictive power and a reliable model for classification tasks.

6. Conclusion and Future Work

In the quest to bolster cybersecurity against PDF malware threats, this study explored diverse machine learning techniques, specifically ensemble learning models, alongside traditional classifiers. The investigation focused on identifying and classifying malicious PDF files by harnessing the collective power of multiple classifiers. Among the array of models examined, the ensemble stacked approach emerged as the most robust and effective in this domain.

The ensemble stacked model, featuring a Layer-0 composition of XGBoost, Neural Networks, Light Gradient Boosting, and Random Forest, showcased unparalleled performance. With varying weight distributions, this amalgamation demonstrated exceptional accuracy, recording an impressive 99.6%, coupled with perfect precision at 100%. Such stellar results underscore the strength of harnessing diverse classifiers within an ensemble framework, leveraging their collective intelligence to discern intricate patterns indicative of PDF-based malware.

This research illuminates the significance of ensemble learning methodologies in enhancing PDF malware detection systems. The ensemble stacked model's ability to capitalize on the strengths of individual classifiers while mitigating their limitations highlights its superiority over singular classifiers and even other ensemble configurations explored in this study. Moreover, it emphasizes the potential of ensemble learning to offer proactive defense mechanisms against evolving malware landscapes.

For future work, delving deeper into feature engineering and selection methodologies to identify more informative features specific to PDF malware could fortify the ensemble stacked model further. Additionally, exploring novel ensemble combinations, integrating newer classifiers, or incorporating deep learning architectures within the ensemble could augment the model's efficacy in tackling emerging and sophisticated PDF-based malware threats. Furthermore, addressing scalability concerns and real-time implementation feasibility of ensemble approaches remains an avenue for future research to make these systems more practical and widely deployable in real-world cybersecurity settings.

Ethics Declaration

Conflict of Interest- The authors declare that they have no conflicts of interest regarding the publication of this study.

Ethical approval- This article does not contain any studies with human participants or animals performed by any of the authors.

Dataset Availability- All the dataset that used in this studied is available on https://www.unb.ca/cic/datasets/pdfmal-2022.html.

References

1. Amer, E., & Zelinka, I. (2020). A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence. Computers & Security, 92, 101760.
2. Kumar, A., Kuppusamy, K. S., & Aghila, G. (2019). A learning model to detect maliciousness of portable executable using integrated feature set. Journal of King Saud University-Computer and Information Sciences, 31(2), 252-265.
3. Andrade, E. D. O., Viterbo, J., Vasconcelos, C. N., Guérin, J., & Bernardini, F. C. (2019). A model based on LSTM neural networks to identify five different types of malware. Procedia Computer Science, 159, 182-191.
4. Zhong, W., & Gu, F. (2019). A multi-level deep learning system for malware detection. Expert Systems with Applications, 133, 151-162.
5. Jerlin, M. A., & Marimuthu, K. (2018). A new malware detection system using machine learning techniques for API call sequences. Journal of Applied Security Research, 13(1), 45-62
6. Rezaei, T., Manavi, F., & Hamzeh, A. (2021). A PE header-based method for malware detection using clustering and deep embedding techniques. Journal of Information Security and Applications, 60, 102876
7. Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. Journal of Systems Architecture, 112, 101861.
8. Al Sarah, N., Rifat, F. Y., Hossain, M. S., & Narman, H. S. (2021). An efficient android malware prediction using Ensemble machine learning algorithms. Procedia Computer Science, 191, 184-191.
9. Bawazeer, O., Helmy, T., & Al-hadhrami, S. (2021, July). Malware Detection Using Machine

Learning Algorithms Based on Hardware Performance Counters: Analysis and Simulation. In Journal of Physics: Conference Series (Vol. 1962, No. 1, p. 012010). IOP Publishing

10. Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, 101663.
11. Han, W., Xue, J., Wang, Y., Huang, L., Kong, Z., & Mao, L. (2019). MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. computers & security, 83, 208-233.
12. Bala, N., Ahmar, A., Li, W., Tovar, F., Battu, A., & Bambarkar, P. (2021). DroidEnemy: battling adversarial example attacks for Android malware detection. Digital Communications and Networks.
13. Révay, L. (2019). From malware testing to virtualization. Procedia Computer Science, 150, 751-756.
14. Karbab, E. B., & Debbabi, M. (2019). MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports. Digital Investigation, 28, S77-S87.
15. Sung, Y., Jang, S., Jeong, Y. S., & Hyuk, J. (2020). Malware classification algorithm using advanced Word2vec-based Bi-LSTM for ground control stations. Computer Communications, 153, 342-348.
16. O'Shaughnessy, S., & Breitinger, F. (2021). Malware family classification via efficient Huffman features. Forensic Science International: Digital Investigation, 37, 301192.
17. Hsiao, S. C., Kao, D. Y., Liu, Z. Y., & Tso, R. (2019). Malware image classification using one-shot learning with siamese networks. Procedia Computer Science, 159, 1863-1871.
18. Al Sarah, N., Rifat, F. Y., Hossain, M. S., & Narman, H. S. (2021). An efficient android malware prediction using Ensemble machine learning algorithms. Procedia Computer Science, 191, 184-191.
19. Mat, S. R. T., Ab Razak, M. F., Kahar, M. N. M., Arif, J. M., & Firdaus, A. (2022). A Bayesian probability model for Android malware detection. ICT Express, 8(3), 424-431.
20. Ucci, D., Aniello, L., & Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. Computers & Security, 81, 123-147.
21. Sadek, I., Chong, P., Rehman, S. U., Elovici, Y., & Binder, A. (2019). Memory snapshot dataset of a compromised host with malware using obfuscation evasion techniques. Data in brief, 26, 104437.
22. Milosevic, N., Dehghantanha, A., & Choo, K. K. R. (2017). Machine learning aided Android malware classification. Computers & Electrical Engineering, 61, 266-274.
23. Bhat, P., & Dutta, K. (2021). A multi-tiered feature selection model for android malware detection based on Feature discrimination and Information Gain. Journal of King Saud University-Computer and Information Sciences.
24. Zakeya, N., Ségla, K., Chamseddine, T., & Alvine, B. B. (2022). Probing AndroVul dataset for studies on Android malware classification. Journal of King Saud University-Computer and Information Sciences, 34(9), 6883-6894.
25. Rathore, H., Samavedhi, A., Sahay, S. K., & Sewak, M. (2021). Robust malware detection models: learning from adversarial attacks and defenses. Forensic Science International: Digital Investigation, 37, 301183.
26. Menéndez, H. D., Bhattacharya, S., Clark, D., & Barr, E. T. (2019). The arms race: Adversarial search defeats entropy used to detect malware. Expert Systems with Applications, 118, 246-260.
27. Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Computer Applications, 153, 102526.
28. Shhadat, I., Hayajneh, A., & Al-Sharif, Z. A. (2020). The use of machine learning techniques to advance the detection and classification of unknown malware. Procedia Computer Science, 170,

917-922.

29. Abhishek Chopra, Nikhill Vombatkere, Arash Habibi Lashkari,”AuthAttLyzer: A Robust defensive distillation-based Authorship Attribution framework”, The 12th International Conference on Communication and Network Security (ICCNS), 2022, China.
30. Moreira, C. C., Moreira, D. C., & de Sales Jr, C. D. S. (2023). Improving ransomware detection based on portable executable headers using xception convolutional neural network. Computers & Security, 130, 103265.
31. Tsafrir, T., Cohen, A., Nir, E., & Nissim, N. (2023). Efficient feature extraction methodologies for unknown MP4-Malware detection using Machine learning algorithms. Expert Systems with Applications, 219, 119615.
32. Singh, A. K., Taterh, S., & Mitra, U. (2023). An Efficient Tactic for Analysis and Evaluation of Malware Dump File Using the Volatility Tool. SN Computer Science, 4(5), 457.
33. Singh, A. K., Gandhi, V. C., Subramanyam, M. M., Kumar, S., Aggarwal, S., & Tiwari, S. (2021, April). A Vigorous Chaotic Function Based Image Authentication Structure. In Journal of Physics: Conference Series (Vol. 1854, No. 1, p. 012039). IOP Publishing.
34. Maryam Issakhani, Princy Victor, Ali Tekeoglu, and Arash Habibi Lashkari1, “PDF Malware Detection Based on Stacking Learning”, The International Conference on Information Systems Security and Privacy, February 2022