Enhancing Malware Detection in PE Files Using Hybrid Ensemble Learning Techniques

MSc Research Project

MSc Cyber Security

Jatinder Singh Saini
Student ID: 21173656

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jatinder Singh Saini
Student ID:	X21173656
Programme:	MSc Cyber Security **Year:** 2023/24
Module:	MSc Research Project
Supervisor:	Diego Lugones
Submission Due Date:	31/01/2024
Project Title:	Enhancing Malware Detection in PE Files Using Hybrid Ensemble Learning Techniques
Word Count:	8572 **Page Count:** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jatinder Singh Saini

Date: 30/01/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Malware Detection in PE Files Using Hybrid Ensemble Learning Techniques

Jatinder Singh Saini
21173656

Abstract

Malware has been a major issue in the online world for a decade, and its prevalence is expected to grow. As new technologies emerge, they inspire hackers to write harmful code and use it to steal information and do other malicious activities. Hackers mainly focus on Windows portable files as these files are carriers for malicious code. Machine learning methods have become a viable malware detection tool. However, malware developers are using these machine learning techniques to trick detection, which emphasizes the need for a more robust strategy. To enhance the detection approach, the study explores the capabilities of hybrid ensemble learning, focusing on tree-based algorithms, employing Gradient Boosting, Random Forest, AdaBoost, and a stacking classifier to strengthen the precision and as well as resilience of systems intended for malware detection. By combining the strengths of these diverse algorithms, this study intends to improve the efficacy and generalizability of malware recognition, offering a possibly promising approach to dealing with malware threats. The study findings have shown that the stacking classifier has achieved a 99% accuracy rate by combining the three algorithms and the application developed was able to make predictions on PE file samples more quickly, highlighting its potential for enhancing malware detection systems and making it an effective contribution in the cyber security domain.

Keywords: Ensemble Learning, Machine Learning, Malware Detection.

1 Introduction

Malicious Software is slowly becoming a major issue that harms IT infrastructure and the digital world as it can perform illegal things like steal data or damage computer systems (Alenezi et al., 2020). Malware can hide or connect itself to any file or device on a computer or network. Computers store sensitive data essential for an individual and an organization. As Internet access has increased, so has the number of individuals downloading and running portable executable files such as.exe has increased (Stokes et al., 2010). Since these files are the vectors for malicious code, it may lead to a serious problem for an individual or an organization that completely relies on their computer systems to perform important tasks or run their entire business. According to the 2023 Malware Bytes report [1], malware in businesses increased a lot i.e., 71% by the end of November.

[1] https://go.malwarebytes.com/

Data protection and privacy are being threatened by malware more often. Therefore, issues related to malware detection are gaining interest among researchers. Nowadays, malware authors are targeting Portable executable files, also known as PE files, which are runnable files that are used with Windows OS (Kim et al., 2022). Various approaches are used by researchers, among the various approaches, one of them is the traditional signature-based method (Cloonan, 2017). Signature-based methods in cybersecurity depend on malware patterns or signatures that have already been defined, making them effective against recognized threats. Though it may be good at detecting signature-based malware, nowadays, there is a trend among researchers to use a machine learning (ML) approach for malware identification as ML algorithms can identify malware by learning from diverse patterns maximizing their impact in capturing polymorphic behavior malware and reducing false negative recognition (Ucci, Aniello and Baldoni, 2019).

The advantage of employing ML algorithms for malware detection lies in their ability to reduce false negatives. Machine learning models, however, can generalize from diverse datasets, learning the underlying patterns of both normal and malicious behavior. This adaptability enables them to make more nuanced decisions, resulting in a lower false positive rate and decreasing the chances of incorrect prediction. (Amer and Aziz, 2019)

In the case of an ML-based classifier, a potential challenge may arise, as malware creators are using the same machine learning techniques to outsmart the defenses. This means that using just one machine learning method to catch malware isn't enough; it can be tricked. Malware developers employ ML strategies to evade malware detectors. This is undeniable that ML approaches are used on a large scale nowadays for malware detection. Malware detectors based on a single ML algorithm can be bypassed. Single models may have several limitations. They might exhibit limited generalization capabilities. Additionally, single models may not be scalable for large or high-dimensional datasets, and they may lack the flexibility to effectively handle various data types or problem domains. (Singh and Singh, 2021).

These challenges highlight the need for an innovative approach. For tasks like malware detection, it’s important to recognize these constraints and look for alternative methods, such as a hybrid ensemble approach i.e. combining the ensembles. Ensemble learning approaches are effective methods for enhancing malware detection efficiency (Yerima, Muttik, and Sezer, 2015). The ensemble method can improve reliability and PE malware detection accuracy.

In this project report, a hybrid ensemble technique is utilized to enhance the performance of malware detection technologies, incorporating both bagging and boosting techniques, employing three algorithms—AdaBoost, Random Forest (RF), and Gradient Boosting, along with stacking classifiers to take advantage of the variety available and all three are algorithms are tree-based which has been proven beneficial in the case of prediction or classification in past years (Euh et al., 2020). Different strategies are employed by each of these ensemble algorithms. One of these algorithms is RF, which combines the knowledge of multiple decision trees. Random Forest prevents overfitting by training these trees on different groups of data and traits. It also gives us a model that is strong and easy to understand. Gradient Boosting, on the other hand, works in steps, improving the results of weak models one at a time. Its main goal is to reduce the mistakes made by earlier models so that a very accurate and strong group can be made. Meanwhile, AdaBoost, short for Adaptive Boosting, employs a unique strategy by assigning varying weights to individual models. It dynamically adapts by emphasizing the importance of data points that were previously misclassified, resulting in a well-balanced and adaptive ensemble (Ijaz, Durad and Ismail, 2019). The hybrid ensemble can use individual

strengths of algorithms and combine them to make a strong detection system identifying malicious and benign files (Kuchipudi et al., 2023). The focus of this paper is to explore ensemble models for improving performance, and how well they can generalize across malware. The research aims to advance cybersecurity by highlighting these things and showing their generalizing capabilities, efficacy, and durability, of ensemble techniques for improving precision detection systems and resiliency. The findings can be used to show the more effective and adaptive defenses against malicious software.

1.1 Research Question

Is the implementation of hybrid ensemble learning approaches effective in improving the accuracy of malware detection in Portable Executable (PE) files?

2 Related Work

Several researchers have contributed their studies to secure system breaches with the detection of malware. It has been classified into different sections as follows:

2.1 Traditional Signature-Based Detection Methods

(Savenko et al., 2019) presented an approach for obtaining malware signatures via API call tracing. This approach takes use of the frequency variation and interactivity of API calls during execution between malicious and benign software. Their testing results showed that this technique works, with malware detection accuracy reaching up to 96.56 percent. (Bahador, Abadi and Tajoddin, 2019) investigate behavioral malware detection, focusing on the examination of system call traces and the large performance overheads associated with these strategies. They provided HLMD, a new technique. Their trials on a benign and malicious dataset produced outstanding results, with average accuracy of 95%, recall of 92%, and F-measure of 89%, respectively. (Garg and Yadav, 2019) explored supervised learning's potential for detecting malware, concentrating specifically on the frequency of API calls in portable executable formats.

2.2 Detection of Malware Using ML

The studies by (Saad, Briguglio and Elmiligi, 2019), (Singh and Singh, 2021), and (Rathore, Agarwal, S. K. Sahay, et al., 2018) collectively emphasized the paramount importance of addressing the growing menace of malware attacks and the vital role of ML in enhancing malware detection and prevention systems. These works share a common theme of recognizing the evolving landscape of cybersecurity, marked by an increasing volume, complexity, and sophistication of malware attacks, which necessitates innovative solutions. Other researchers, like (Liu et al., 2020), (Sethi et al., 2019), and (Wu et al., 2018), addressed the critical difficulties of malware identification in today's fast-expanding technological ecosystem. (Liu et al., 2020) largely focus on the Android environment, citing the proliferation of Android applications as a key concern, as does the simultaneous growth of Android malware. Their research provides a complete overview of malware detection approaches in Android that use ML. Similarly, (Sethi et al., 2019) recognize the growing threat of malware to computer systems, stressing the ineffectiveness of signature-based detection strategies. In another study (Kumar Ajayand Abhishek, 2020) Brazilian dataset was used, which contains malware and samples, and seven ML models were developed for classifying the malicious or benign file. A two-step approach (Thosar et al., 2021) to malware identification using Convolutional Neural Networks and Gradient Boosting Classifiers was presented in another research. Using the Virus

MNIST dataset for malware family classification and the EMBER 2018 dataset for binary classification, the suggested approach outperforms state-of-the-art methods with 93% accuracy and 96% accuracy, respectively.

2.3 Ensemble Learning Approaches

(Gupta and Rani, 2020) and (Sayadi, Patel, D, et al., 2018) cover critical areas of the ever-evolving field of cybersecurity, with a special emphasis on improving malware detection, in their separate research. (Gupta and Rani, 2020) emphasize the importance of malware detection and categorization in computer and network security. They suggest two separate techniques to address this, using an ensemble approach and big data technology to enhance malware detection at scale. (Sayadi, Patel, D, et al., 2018) on the other hand, shifted their attention to Android smartphone security, noting the significant increase in malicious programs targeting the Android platform. They understand the difficulties faced by sophisticated obfuscation tactics, which render standard static analysis worthless. EnDroid, a dynamic analytic framework aimed at very precise Android malware detection, is proposed in response. EnDroid makes use of a plethora of dynamic behavioral elements, such as system-level behavior traces and application-level harmful behaviors. Their method employs feature selection algorithms to remove noise and identify key behavioral features. They efficiently discriminate between harmful and benign apps by using ensemble learning techniques. Both studies emphasize the need to employ modern technology and new strategies to address rising cybersecurity issues. (Feng et al., 2018) focused on Android smartphone security and the necessity for precise malware detection in their study. They presented the EnDroid framework, focusing on dynamic behavior aspects for accurate identification. Similarly, (Abikoye, Gyunka and OLUWATOBI, 2020) highlighted how widely the Android operating system is used as well as the accompanying growth in harmful applications. Their technique for malware detection in Android employs ensemble methods, which combine several base models to improve classification performance.

2.4 Hybrid Ensemble Learning Techniques

The research by (Pektas and Acarman, 2017), (Bhagwat Sakshiand Gupta, 2022), and (Bountakas and Xenakis, 2023) all have the same goal: to improve cybersecurity in the face of increasing threats. Pektaş and Acarman's research focuses on malware detection in Android, recognizing the value of feature-based learning in increasing security. They used both static and dynamic characteristics to effectively categorize Android malware into different families, answering the requirement for a complete threat detection methodology. Bhagwat and Gupta's study also dives into Android malware, highlighting the vulnerability of open-source platforms. Their unique system uses dynamic features and meta-heuristic feature selection to identify malware with a stunning 95.3 percent accuracy. Finally, Bountakas discussed phishing email attempts, highlighting how they evolve amid crucial conditions like the COVID-19 epidemic. Their methodology blends Ensemble Learning with hybrid characteristics, providing two approaches to improve phishing email detection: Stacking and Soft Voting Ensemble Learning. Their methodology beats prior approaches, with an F1-score of 0.9942. The deployment of hybrid ensemble learning approaches to address emerging threats is a common thread in these investigations. Furthermore, scalability and computing difficulties are frequent themes, highlighting the difficulty of certain cybersecurity activities. (Smmarwar Santosh K.and Gupta, 2022) had a goal to optimize the security of Android devices. They understand the crucial need for robust malware detection, particularly for Android applications. While (Martin et al., 2019) emphasize the requirement for big, labeled datasets for machine learning classifiers, (Smmarwar Santosh K.and Gupta, 2022) addressed the issue of changing malware variants that evade existing detection approaches. (Smmarwar Santosh K.and Gupta, 2022) presented a

unique framework for optimizing malware characteristics for enhanced detection by combining feature selection approaches such as wrapping feature selection, random forest, and greedy stepwise selection. Furthermore, both research makes use of ML classifiers, emphasizing the importance of this method in current cybersecurity. In another study (Rimon Saiful Islamand Haque, 2023) used RF and K-Nearest Neighbour classifier, to develop a hybrid model for malware detection as well as classification. The author gathered malware samples updated the dataset, and tested their suggested method against conventional approaches, showing that it was more accurate.

2.5 Table comparison

Author	Focus	Major Classifiers/ Algorithms	Accuracy	Challenges Addressed
(Savenko et al., 2019)	API call-based malware detection	Support Vector Machines (SVM)	Up to 96.56%	Distinguishing malware from benign applications based on API call behaviors.
(Bahador, Abadi and Tajoddin, 2019)	Hardware performance-based detection	Singular Value Decomposition	Avg. 92.50%	Focusing on independent malicious programs and reducing detection complexity while improving accuracy.
(Garg and Yadav, 2019)	API calls frequency in portable executables	Supervised Learning Algorithm	93%	Dealing with obfuscation techniques used by attackers and effectively distinguishing malware from benign files.
(Liu et al., 2020)	Malware Detection (Android)	ML Algorithms	NA	Malware Detection in Android
(Sethi et al., 2019)	Classification and Detection of malware	ML Algorithms	High but not mentioned	Signature-Based Detection
(Wu et al., 2018)	Malware Evasion with Reinforcement Learning	Random Forest	NA	Malware Evasion
(Saad, Briguglio and Elmiligi, 2019)	Challenges with applying ML to the detection of malware in the real world	Random Forest	NA	Challenges in real-world malware detection, solutions for improving detection

(Singh and Singh, 2021)	Detailed study of ML techniques for malware detection	Ensemble Learning	Malware accuracy increased from 15.75% to 93.5%.	Problems in creating malware classifiers and future directions for improved detection
(Rathore, Agarwal, S. Sahay, et al., 2018)	Detection of malware using ML and deep learning	Random Forest (RF), Deep Neural Network	NA	Challenges in enhancing malware detection and future research directions
(Kumar Ajay and Abhishek, 2020)	Improving detection	Seven ML Models	94%	Enhancing detection
(Thosar et al., 2021)	Enhancing detection using Gradient Boosting (GB) and Convolutional Neural Network (CNN)	GB and CNN	96% and 93% accuracy	Challenges in malware detection and classification
(Feng et al., 2018)	Android malware detection	Ensemble Learning	NA	Evasion tactics, dynamic analysis
(Abikoye, Gyunka and OLUWATOBI, 2020)	Android malware detection	RF, Support Vector Machine (SVM), k-nearest Neighbours (KNN)	Base models: 97.9%, Ensemble: 98.16%	Evolving malware variants, detection optimization
(Gupta and Rani, 2020)	Malware detection and big data	RF, SVM, KNN	Base models: 97.9%+, Ensemble: 98.16%	Handling big data, generalization performance
(Sayadi, Patel, P D, et al., 2018)	Phishing email detection	Ensemble Learning	Achieved Hardware Performance Counters up to 17%	Evolving phishing email threats, improved detection accuracy
(Pektas and Acarman, 2017)	Android malware classification	Machine Learning Classifiers	92%	Scalability, Data Collection
(Bhagwat Sakshiand Gupta, 2022)	Android malware detection	Adaptive Boosting, Extreme Gradient Boosting	95.3%	Feature selection, Unknown malware

(Bountakas and Xenakis, 2023)	Phishing email detection	Ensemble Learning	F1-score: 0.9942	Evolving phishing emails
(Rimon Saiful Islamand Haque, 2023)	Detection of malware	RF and KNN	98%	Improving Detection
(Smmarwar Santosh K.and Gupta, 2022)	Malware feature selection	Random Forest, Decision Tree, SVM RBF	Up to 91.80%	Feature selection, Detection robustness

2.5 Gaps in Related Work

In the existing research on using machine learning (ML) methods and ensemble methodologies, it's evident that substantial progress has been made by the researchers in detecting malware. However, there are notable gaps in the current research. One Key area or gap is the necessity for better adaptation to emerging malware techniques. Many studies focus on specific aspects of malware behavior, but they often fall short of acknowledging the ever-changing and dynamic nature of malware development. Moreover, the quality and availability of datasets used in these studies are crucial. The field of malware is expanding, and there is a clear need for more diversified and up-to-date datasets that can comprehensively cover the malware landscape. Without the use of such datasets, the effectiveness of ML models in detecting may be limited. While ensemble learning approaches reveal many advances in detecting malware, it is important to stay up-to-date with the growing strategies employed by malware authors. Moreover, to construct a comprehensive detection system that can effectively deal with the ever-changing field of malware, there is a need for extraction and selection of necessary features. By bridging these existing gaps by doing the mentioned above the detection mechanism may remain both robust and precise.

3 Research Methodology

3.1 Proposed System

Malware attacks in today's world are on the rise and are causing damage to organizations' reputations so immediate action must be taken to detect the malware. Most of the related work (Rathore, Agarwal, S. Sahay, et al., 2018), (Singh and Singh, 2021) on malware identification focuses on ML techniques and ensemble methods. In terms of detecting malware, there is a lack of focus on scalability and interpretability of ensemble methodology and selection (Bhagwat Sakshiand Gupta, 2022) of necessary features, as well as on adequately adapting ML models to developing malware techniques. In comparison to the previous study mentioned above in this section, this research focuses on enhancing the malware detection system which is done by achieving high accuracy as shown in the evaluation section of the report. This results in increasing the effectiveness and robustness in classifying or handling diverse malware instances.

Systems can be made more effective and efficient by selecting features that are important for malware identification in cybersecurity. This increases model accuracy, reduces computing burden, and makes the system more interpretable. Cyber threat identification and mitigation are both facilitated by this optimization. The research involves a thorough analysis of the suggested hybrid i.e. (combine) the ensemble model's performance for detecting malware in PE files using tree-based models, and the stacking classifier, within the hybrid ensemble

framework. It involves training and evaluating each classifier on the dataset. Understanding of the model's effectiveness and robustness in managing varied malware instances are provided by the evaluation metrics, which help classify PE files as safe or not safe. The proposed methodology involves the following stages as depicted below:

1. **Imported Libraries:** Libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Plotly were used. Also, other modules were included like SMOTE form imbalanced-learn and Stacking Classifier from Mlxtend, using these libraries extends the project's capabilities in handling the imbalanced data and implementing advanced ensemble methods.
2. **Data Loading and Data Cleaning:** During the data loading phase, the Pandas library was used to retrieve the malware dataset from a specified file path. This makes it easier to analyze later. As part of data cleaning, looking for missing values to ensure data integrity and completeness is also necessary.
3. **Data Preprocessing and Feature Selection:** Data preparation is a critical step in this research. This step includes several critical actions that must be completed to ensure the quality of data and suitability for effective model training. A key component is data preparation, which comprises translating categorical variables into numerical form, which is required by machine learning algorithms. In this project, class imbalance concerns are addressed by balancing the dataset using the SMOTE function which stands for Synthetic Minority Over-sampling Technique. Also, necessary columns or features were taken into consideration in the data frame as shown in "figure 1" to aid in achieving the overall success of the malware classification model. By prioritizing essential features and excluding irrelevant ones, the model's training process is more focused, leading to a reduction in overfitting and enhanced generalization of malware samples. This was done with the help of using the "Pefile" library. The "Pefile" library's latest update enhances its capabilities, resulting in selection the of important features In certain situations, some features may not provide valuable information, so it makes sense to remove or combine them with other similar features to reduce data dimensionality. This focused feature selection has helped improve the model's accuracy and efficiency and has also made it more interpretable. By following the above-mentioned process, the research gains clearer insights into the critical factors influencing malware classification, facilitating informed decision-making. Moreover, the dataset has been optimized ensuring a more scalable and practical solution for malware detection.

```
df = df[['e_magic', 'e_cblp', 'e_cp', 'e_crlc', 'e_cparhdr', 'e_minalloc',
       'e_maxalloc', 'e_ss', 'e_sp', 'e_csum', 'e_ip', 'e_cs', 'e_lfarlc',
       'e_ovno', 'e_oemid', 'e_oeminfo', 'e_lfanew', 'Machine',
       'NumberOfSections', 'TimeDateStamp', 'PointerToSymbolTable',
       'NumberOfSymbols', 'SizeOfOptionalHeader', 'Characteristics', 'Magic',
       'MajorLinkerVersion', 'MinorLinkerVersion', 'SizeOfCode',
       'SizeOfInitializedData', 'SizeOfUninitializedData',
       'AddressOfEntryPoint', 'BaseOfCode', 'ImageBase', 'SectionAlignment',
       'FileAlignment', 'MajorOperatingSystemVersion',
       'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion',
       'MajorSubsystemVersion', 'MinorSubsystemVersion', 'SizeOfHeaders',
       'CheckSum', 'SizeOfImage', 'Subsystem', 'DllCharacteristics',
       'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve',
       'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes', 'Malware']]
```

Figure 1: Taking necessary columns or features using 'pefile' library

4. **Data Splitting:** The function name (train_test_split) from Scikit-learn was used in this project to distribute the dataset into 2 parts: one training and another testing. The ratio of the two sets is 80:20. Because of this partition, to train ML models, a subset of the data is used and then tested on a separate set to check how well they performed. To make sure that the target variable remains distributed in both sets, the stratification parameter is used.
5. **Implementation of Model:** The research employs three separate ensemble learning methods to construct malware classification models along with one meta-classifier i.e. stacking classifier throughout the model implementation phase. Each of the three classifier algorithms AdaBoost, Gradient Boosting, and Random Forest are configured according to their individual needs. AdaBoost and Gradient Boosting use different numbers of estimators (4 and 8) respectively, while Random Forest uses different numbers of trees and splitting criteria. Using the training split created during data preparation, the preprocessed dataset was used to train the models. For the models to adapt the data's underlying patterns and relationships, they are trained by exposing them to features and their corresponding target labels and then combined with a meta-classifier (stacking) for the final prediction. This gets the models ready for evaluation and further analysis.
6. **Model Evaluation and Classification:** The model Evaluation and Classification is where the study checks the ML model's performance. In this report, the evaluation of the model is done based on the classification report and confusion matrix. The selection of appropriate metrics for model evaluation in this research is based on the related work suggested by (Gupta and Rani, 2020) and (Kumar Ajayand Abhishek, 2020) In the context of malware classification, metrics such as accuracy, precision, recall, and F1 score are chosen for this project which plays a crucial role in assessing the effectiveness of the models. Accuracy provides a view of overall correctness, while recall ensures that the model captures most actual instances. Focusing on accurately identifying malware instances, precision reduces the false positives. The F1 score offers a balanced metric which is crucial in imbalanced datasets. Altogether, these metrics provide a detailed assessment that improves the malware detection system's dependability and reduces security risk by finding the balance between false alarms and accurate malware identification.
7. **Tools selection:** To run the ML models and develop an application for the classification of PE sample files various tools were used:

- **Language and Framework Utilization:** Python was utilized, as it is easy and efficient for coding and implementation of the ML models. Also in this research, Flask (Braganca and Kho, 2023) is preferred for web applications due to its lightweight design, simplicity, and ease of integration compared to other frameworks. Its minimalistic structure allows for quick deployment and efficient handling of requests.
- **Computational Environment:** The machine learning models are implemented using Google Colab which is a free cloud-based tool for interactive coding.
- **Development environment:** Visual Studio code is used in this project as a free open-source editor for developing the application.

3.2 Data Visualization

“Figure 2” displays a bar chart that gives the visual representation of the dataset, the yellow bar representing class 1, indicates a high number of instances classified as "malware" and the blue bar indicates fewer instances of "non-malware". This suggests that there is an imbalance

in the dataset, and it is crucial to acknowledge, as it can impact the performance and interpretation of the models.

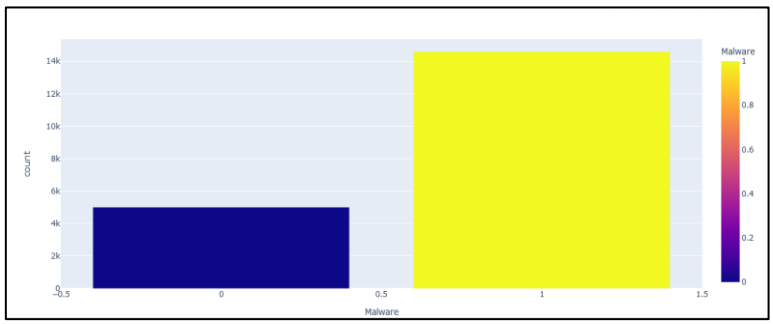

Figure 2: Bar Chart- Distribution of Malware Presence in Dataset

The impact of using SMOTE to correct dataset class imbalance is seen in “Figure 3”. The blue bar in the graph indicates 'non-malware' instances ('0'), while the orange bar represents malware instances ('1'). It's a two-bar graph. There was a significant difference in the counts between '0' and '1' in the original dataset. Nevertheless, the data was effectively balanced by using the SMOTE () function, which allowed for a fair representation of 'non-malware' and ‘malware’ observations during the training of the ML model for the research.

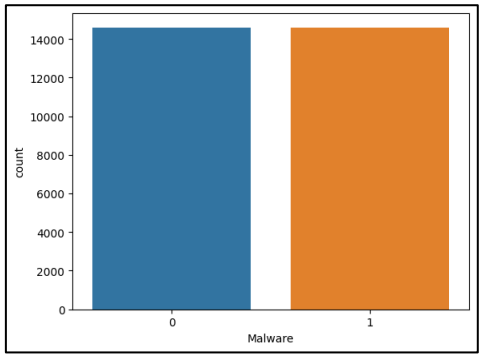

Figure 3: Effect of SMOTE Over Sampling on Data Balance

3.3 Dataset Description

The dataset presented in this study is the result of research on Malware Detection and Machine Learning. It was built using the Python package and consists of data from Portable Executable (PE) files, including both malicious and benign samples. The dataset is publicly accessible on Kaggle [2], providing a convenient platform for researchers to download and explore the data. The dataset includes properties collected from these PE files such as 'e magic,' 'SizeOfCode,' 'Subsystem,' 'ImageBase,' and many more that are crucial in characterizing these files. These properties are essential for teaching ML models to differentiate between malicious software and safe. The dataset consists of 19,612 both malicious and safe PE file samples. The diverse nature of the dataset ensures a broad spectrum of challenges encouraging collaboration and innovation in the ongoing fight against evolving malware threats. The dataset is created to help in the development of models for better malware identification.

[2] https://www.kaggle.com/datasets/amauricio/pe-files-malwares

3.4 Models Used

3.4.1 Tree-Based Algorithms:

In this research project, tree-based algorithms have been chosen because of how effective they can be in capturing complex patterns and non-linear correlations that are present in varied datasets. This provides transparency in the decision-making process. For the classification problem, the tree-based algorithm has been proven to perform better in terms of accurate predictions (Mienye and Sun, 2022). Another study shows (Jadhav and Channe, 2013) that tree-based algorithms are generally faster, especially for larger datasets, making them more efficient in handling data and capable of handling noisy data as compared to non-tree-based algorithms. Because malware data is complex, it is useful to have an algorithm that can process both numerical and categorical information. The preprocessing phase (Euh et al., 2020) is made easier by tree-based models, which are resilient to fluctuations in feature scaling, making them a powerful and practical solution for malware classification. In this study, Random Forest, Gradient Boosting, and AdaBoost were used, and all three algorithms were combined with the stacking classifier for final prediction. The reason for the choice of these algorithms is shown with the help of the study done by (Shafin et al., 2021) and (Pallippattu and Mathai, 2021).

The study (Shafin et al., 2021) shows that the Random Forest, which is known for level-wise growth, may incorporate random feature selection, ensuring a more robust and stable model, making it a preferred choice. Whereas in comparison to Random Forest the LightGBM which is known for its leaf-wise growth for fast computation, may risk overfitting due to deeper trees. Also, Gradient Boosting is preferred over Extra Trees due to its sequential learning and effective capture of complex patterns. Relying on arbitrary split decisions, Extra Trees may introduce higher variance and instability.

Another study (Pallippattu and Mathai, 2021) shows that AdaBoost has achieved higher accuracy and F1-score when compared to other single ML algorithms (C4.5 decision trees, Bayes Network, SVM, Logistic regression, JRip) used. The boosting strategy, that AdaBoost uses, merges numerous weak classifiers into a single powerful one, which is why it performs so much better. Also, the stacking that is used in this proposed study has the capability of enhancing the overall performance of the models by combining them all and providing a final output. In the sections below on research methodology, the three tree-based algorithms have been discussed along with the stacking classifier.

3.4.2 AdaBoost Classifier:

AdaBoost is a popular choice for improving the efficiency of weak learners and addressing real-world challenges. AdaBoost, short for Adaptive boosting (J. Ferreira and Figueiredo, 2012)is a sophisticated ensemble learning technique that assigns weights to data points and iterations to enhance the accuracy of weak classifiers. AdaBoost works by training a series of weak classifiers on the PE file dataset repeatedly, with each succeeding classifier focusing on the mistakes produced by its predecessors also used to improve malware detection in (PE) files. In the research, the AdaBoost classifier from Scikit-learn is used and configured using 4 base estimators, which are essentially weak learners. Each successive model learns from the examples misclassified by its predecessors as the method fits a succession of weak learners on weighted subsets of training data. The final model uses a weighted majority vote to integrate the outputs of these weak learners (Ijaz, Durad and Ismail, 2019). Following training, the AdaBoost classifier uses the aggregate decisions of weak learners to identify if a particular sample is safe or malicious.

3.4.3 Gradient Boosting Classifier:

This approach was used as one of the project's base classifiers with the Stacking Classifier. By successively refining predictions on Portable Executable (PE) files, the Gradient Boosting Classifier contributed to the Stacking Classifier. It is an ensemble learning approach that excels at constructing a robust prediction model by iteratively training a collection of weak learners, often decision trees. It works by fitting (Natekin and Knoll, 2013) new models to residual errors produced by previous models, steadily improving accuracy with each iteration. Because it excels at capturing deep patterns and relationships within data, the Gradient Boosting Classifier is well-suited for the complexity of malware detection. It constantly improves its comprehension of the dataset, allowing it to classify benign and malicious PE files. In this project, Gradient Boosting is employed for malware detection using the Scikit-learn library with eight estimators. The comparative analysis in the provided paper (Bentéjac, Csörgo and Martínez-Muñoz, 2019) suggests that the performance of gradient boosting is slightly better than other algorithms mentioned in the paper in terms of accuracy and training speed.

3.4.4 Random Forest Classifier:

Malware exhibits a diverse range of patterns and behaviors. In this research, a Scikit-Random Forest classifier with two trees has been used as an estimator to classify malware. This classifier comes under the category of bagging. The Random Forest model improves the project's classification accuracy and resilience through a network of decision trees with different subsets of features and samples, capturing various aspects of malicious behavior. Because of this diversity, the model identifies a broad spectrum of malware types and variants. Random Forest is robust since it is ensemble-based and provides robustness against overfitting (Alam and Vuong, 2013) to specific characteristics of individual malware samples. By aggregating the predictions of multiple trees, the model becomes more resilient to noise and variations within the malware dataset, enhancing its generalization. Random Forest inherently ranks the importance of features. This capability aids in determining the most crucial signs of malicious activity within PE files. It also gives us a model that is strong and easy to understand. The Random Forest classifier emerges as a compelling choice for malware detection due to its effective classification among different types of data due to its extent capability of adaptation. When compared with the multiple algorithms the authors (Alam and Vuong, 2013) showed that shown RF classifier has performed well in terms of overall accuracy for malware detection.

3.4.5 Stacking Classifier:

An ensemble approach that gives other algorithms a chance to make an impact is stacking, sometimes known as stacked generalization. This algorithm is also known as the meta-classifier. It is vital to combine different base classifiers in a way that maximizes their performance, since each may be better suited to handle certain aspects of the problem. To solve the issue, a stacking method was developed. stability and generalizability are achieved by combining predictions from a single model. Out of sample data, each base classifier makes predictions. These predictions are stored in a meta-model. The predictions include the features of its classifier's problem-solving strategy and by combining all the baseline classifiers with the meta classifier final prediction is made for improved performance. (Shafin et al., 2021)

4 Design Specification

The provided workflow diagram in “Figure 4” outlines the process for classifying whether the provided PE file is malicious or not. The dataset consists of PE file samples (benign and malicious). It shows a systematic approach for training the ML-based models, selecting the classifier, and deploying this classifier in an application that can provide immediate, actionable results.

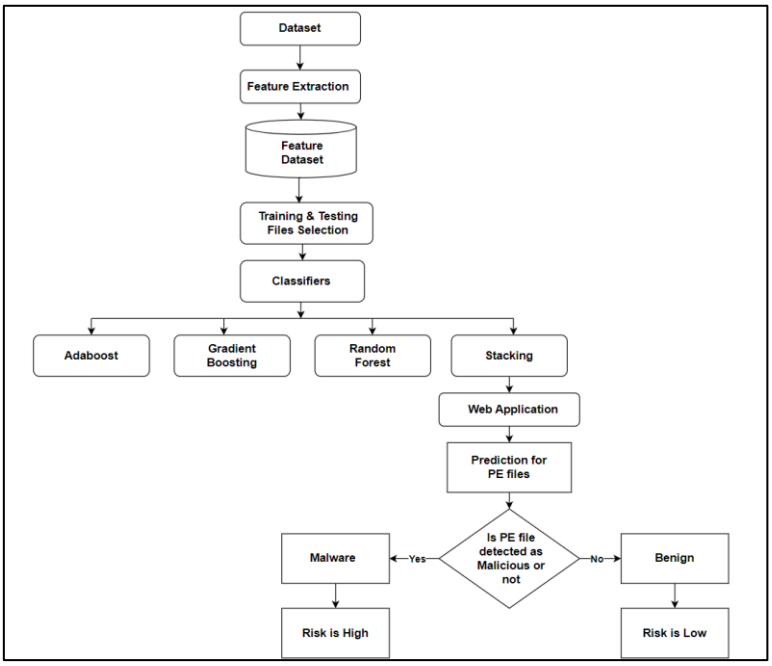

Figure 4: Architecture Design

Three tree-based classifiers and a stacking classifier have been used in the research. Python was used as the fundamental language for developing the detection model, with Google Colab and VS Code serving as the major development environments. VS Code, a free IDE, proved helpful in creating the model and giving runtime results. Initially, the dataset contains various samples of PE files. The necessary features are extracted from the dataset by exploring the ‘Pefile’ library. These features are characteristics or attributes of PE files that can be used for further analysis. The next step was to arrange these features in a feature dataset. The most crucial step in ML is separating the feature dataset in training and testing sets for malware identification. Importantly, the training set exposes the machine learning models to a wide variety of PE files, which helps them learn the data. The research follows an organized procedure with training numerous ensembles of learning algorithms (including AdaBoost, Random Forests, Gradient Boosting, and Stacking) on a training dataset, which was accomplished using Google Colab. The models can then learn to distinguish between malicious and safe files based on these patterns. However, the training model's capacity to categorize data is evaluated on the testing set. This guarantees that the models improve their dependability and performance in real-world situations by generalizing well beyond the training data. Individual models are trained and combined with the stacking classifier for the final prediction. The next step involves integrating the machine learning models into a web application and moving on to the building of a PE file classification system utilizing Flask. This simple application allows users to submit executable files for risk evaluation. When a file is submitted, the pre-trained Stacking Classifier examines its properties to estimate its risk level—high or low risk based on malware or benign classification, respectively. The Flask-based application incorporates the pre-trained Stacking Classifier. The interface provides a user-friendly application for

individuals to communicate with the malware detection system, simplifying the process of uploading the exe files and understanding the associated risk levels. When the model classifies a file as malicious, the system generates high-risk alerts, signaling a potential security threat that requires immediate attention and action. Conversely, if the file is identified as benign, the application issues low-risk alerts, providing users with confidence in the safety of the file. This approach allows users to proactively take security steps based on the severity of the risk identified by the model, contributing to a more responsive and preemptive cybersecurity strategy.

5 Implementation

Implementing a system that helps in detecting malware in PE files is a vital phase of the research project. This comprises hybrid ensemble learning approaches, carrying out a series of procedures, each of which adds to the construction of a strong and functional system. The implementation begins with the deployment of the selected ensemble learning models: AdaBoost, Gradient Boosting, RF, and the stacking classifier, each of which performs a specific function in the ensemble. The 'fit ()' method is utilized for training of model, and the 'predict ()' method is utilized to test the trained model. A dataset including both benign and malicious data from PE files is used to train the algorithm. The web application was created in Python using the Flask framework. The classifier models are integrated into the web application that is running locally, which was further used for classifying whether the file that is been input is a safe file or a malicious file. These results will be displayed on the application. The web application's implementation required the careful selection of components to provide maximum functionality and usability. Python was chosen as the language because of its versatility, vast library, and resilience in handling data processing tasks essential to malware detection from Portable Executable (PE) files. Google Colab was chosen for its cloud-based infrastructure and accessibility, allowing for seamless collaboration and using its computing capabilities. The VS code was utilized for coding the application. Flask was chosen as the framework because of its simplicity, flexibility, and quick development capabilities, making it suitable for designing a user-friendly interface.

5.1 Model Implementation

In this section, the ensemble learning model implementation is shown with the help of the code in Figures 5,6,7 and 8. The code reveals the choice of algorithm, its configuration, and its utilization for training and testing purposes. This code is essential for understanding how the models are trained and applied. The model training is done using a portion of data and evaluated using another subset of data which helps in achieving the research project's objectives.

5.1.1 AdaBoost Model:

The provided code in “Figure 5” implements AdaBoost with 4 weak learners for classification. Initializing the model, training it on labeled data, and predicting on test data showcases AdaBoost's sequential learning process.

```
adb_model = AdaBoostClassifier(n_estimators=4)
adbclf = adb_model
adbclf.fit(X_train,y_train)
y_pred = adbclf.predict(X_test)
```

Figure 5: AdaBoost Classifier Implementation Code

To show how AdaBoost works in practice, the code is essential; it highlights the sequential learning process, and how the n_estimators hyperparameter improves classification results. The hyperparameter "n_estimators" is set to 4 which shows the ensemble's depth. This code sets up the AdaBoost Model, puts it in a variable 'abdclf', trains it with X_train and y_train, which are set of labeled training data, and then uses it to predict X_test, a set of test data, saving the results in y_pred.

5.1.2 Gradient Boosting Model:

```
gbc_model = GradientBoostingClassifier(n_estimators=8)
gbc = gbc_model
gbc.fit(X_train,y_train)
y_pred = gbc.predict(X_test)
```

Figure 6: Gradient Boosting Classifier Implementation Code

"Figure 6" shows how the Gradient Boosting classifier using the scikit-learn model is created. The classifier is trained on X-train and y-train, which are labeled training data. With eight trees (n_estimators=8), it generates a classifier instance that is assigned to the variable "gbc". Lastly, it applies the trained model to the testing data (X_test), and predictions are stored in the y_pred variable. The code is important as it illustrates the practical application of the Gradient Boosting algorithm, a popular ensemble learning technique.

5.1.3 Random Forest Model:

```
#RandomForest Classifier
rf_clf = RandomForestClassifier(n_estimators=2, criterion='gini', max_depth=3,)
rf = rf_clf
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
```

Figure 7: Random Forest Classifier Implementation Code

“Figure 7” shows that the RF model is constructed with specified settings of defining the number of estimators set to 2, the criterion for splitting nodes based on 'gini' impurity, and a maximum depth of up to 3 for each tree in the forest. The classifier is trained by utilizing the training data ('X train' and 'y train') by using 'rf.fit(X train, y train)'. The model can learn and provide predictions via this process based on the specified settings and training data. Then it was used to predict the test dataset.

5.1.4 Stacking Model:

```
#Stacking classifier
clf = StackingClassifier(classifiers = [RandomForestClassifier(),GradientBoostingClassifier()], meta_classifier = AdaBoostClassifier())
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
```

Figure 8: Stacking Classifier Implementation Code

In this research, the Stacking Classifier is configured, incorporating 3 base classifiers as shown in “Figure 8”. These base classifiers make individual predictions, and their outputs serve as

input to the meta-classifier, which is useful in making final predictions. Following the initialization, the Stacking Classifier is trained using the training data, allowing it to learn and optimize its performance based on the diverse input from individual classifiers. This approach aims to harness the strength of multiple models, enhancing the overall predictive capability of the classifier.

5.2 Application for PE File Classification

"Figure 9" displays the web application for malware detection. It features a PE file classification system using hybrid ensemble learning. PE files can be effortlessly uploaded and instant, classification results from the integrated models are displayed on the application interface. The user-friendly interface allows informed decisions efficiently. "Figure 10" shows that the application prompts to select an EXE file and initiate the prediction process by selecting the file and clicking on the "Predict uploaded EXE file" button.

Figure 9: Web application interface

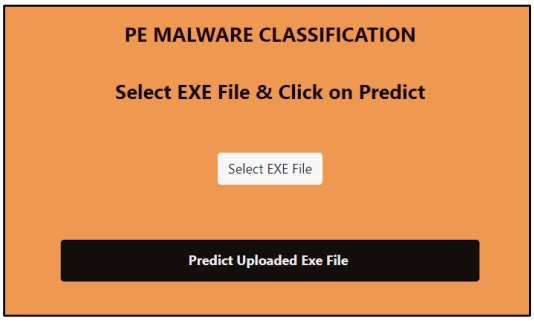

Figure 10: Uploading the EXE file

“Figure 11” shows that the application successfully detected and classified a given input executable file as malware based on the analysis. Utilizing classification algorithms, the system analyzed the file's characteristics and behavior to predict its malicious nature.

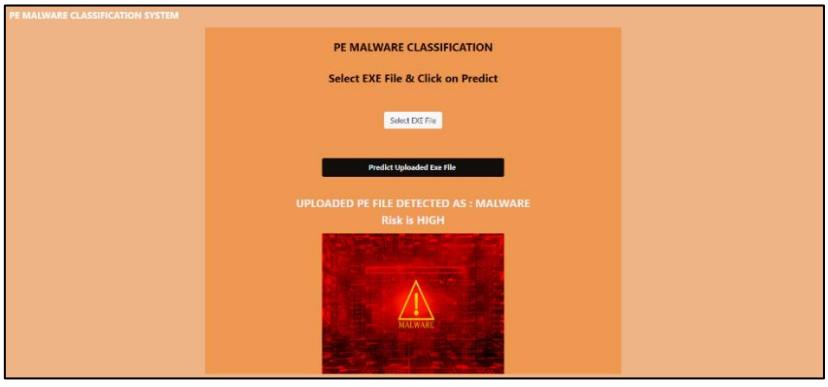

Figure 11: The file is detected as malware

“Figure 12” shows that the application has detected the executable file as safe. The application's ability to label the file as safe highlights its effectiveness. For this, the application shows the alert as "Uploaded PE file Detected as Benign".

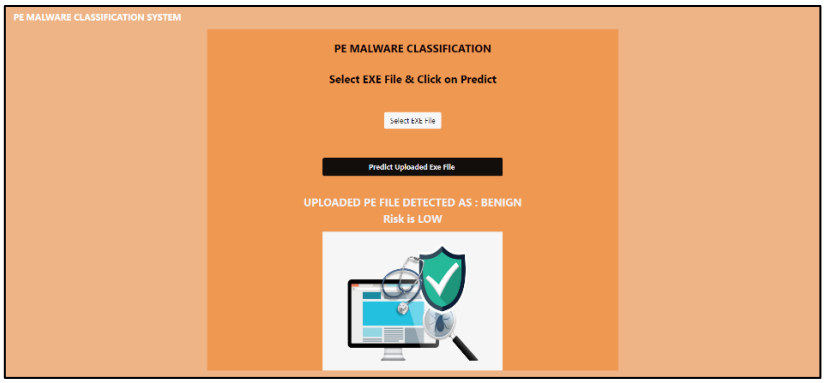

Figure 12: The file is detected as benign

6 Evaluation

The results of the suggested hybrid learning method are detailed here for malware or benign classification, with extensive experimental results. For the experiment, the dataset was used from Kaggle, which contains both malicious and benign files. Results were produced using the Scikit-Learn library with detailed accuracy and the confusion matrix. For the detection of the sample file, two sets of data were created: 80 % data for training, and the testing set contained 20%. Four distinct metrics were employed to evaluate the suggested approach i.e. Precision, Accuracy, F1 score, and Recall. The metrics performance is described using True Positive (TP), False Negative (FN), True Negative (TN), and False Positive (FP). Ensemble models used for the experiments were Gradient Boosting, AdaBoost, RF, and the stacking model. In this section, the proposed study is compared with the previous research in the same field. The results of experiments for the evaluation are provided in the below Table 1 and Table 2. Before evaluating the efficiency of the models, the metrics are defined to get a clear understanding of the results. The following metrics are defined:

- **Accuracy:** The accuracy of the model's prediction is provided by the ratio of correctly predicted examples to total occurrences.
- **Precision:** The % of correct predictions out of all the predictions that turned out to be true.
- **Recall:** Total % of correct predictions that were positives to the actual total positives.
- **F1 Score:** Precision and recall measures are combined and weighted to provide a harmonic mean. The model's effectiveness improves when the value is closer to 1.

6.1 Evaluation of Experimental Results (Proposed Work)

Classification Models	Precision	Recall	F1 score	Accuracy
AdaBoost (AB)	0.95	0.95	0.95	95%
Gradient Boosting (GB)	0.96	0.96	0.96	96%
Random Forest (RF)	0.95	0.95	0.95	95%
Stacking Classifier (AB+GB+RF)	**0.99**	**0.99**	**0.99**	**99%**

Table 1: Overall PE file Classification Result (Malware or benign) of the Proposed Work based on the Metrics

Table 1 displays the results for classification using various classifiers. The AdaBoost and the RF classifier both were able to reach precision, recall, an F1 score of 0.95, and an accuracy of 95%. The models were able to learn patterns from both classes i.e. (malware and benign). The outcome suggests that, for the given experiment both the ensemble learning methods have

learned to make predictions with similar effectiveness, maybe because of a well-balanced dataset, comparable decision boundaries, suitable hyperparameter settings, or the distinctive characteristics of the classification job. In the case of Gradient Boosting, the model was able to reach precision, F1 score, and recall of 0.96, performing better than the AdaBoost and RF classifier with an accuracy rate of 96%. With its sequential error-correcting process, Gradient Boosting can adapt better to relationships in complicated data and potentially able to capture more complex patterns than AdaBoost and Random Forest. The models were able to perform well, but the stacking classifier resulted in outperforming all three models and gave the results combining the three ensemble models with a 99 % accuracy rate and the precision, recall, and F1 score of 0.99. With this the stacking classifier stands out as the most effective and precise model in comparison to the individual ensemble models that were used, showcasing its potential for the classification of PE files as malware or benign.

6.2 Experimental Results (Related Work- Accuracy Comparison)

A comparison of work is shown in Table 2 between the proposed research and the relevant works in the field. Compared to other works, the proposed method gives slightly better performance. In the provided Table 2 comparing the results it's evident that the proposed stacking model combining three ensemble Algorithms outperforms individual models in terms of accuracy. Though Gradient Boosting in the proposed study has shown the same accuracy as in the related work, the AdaBoost and Random Forest were able to perform well compared to the related work. The work done by (Rimon Saiful Islamand Haque, 2023) shows the hybrid approach, achieving an accuracy of 98%. The proposed stacking classifier outperforms it with an accuracy rate of 99%.

Work	**Classification Models**	**Classification Accuracy (Related Work)**	**Classification Accuracy (Proposed work)**
(Kumar Ajayand Abhishek, 2020)	AdaBoost	94%	95%
(Thosar et al., 2021)	Gradient Boosting	96%	96%
(Sethi et al., 2019)	Random Forest	88.23%	95%
(Rimon Saiful Islamand Haque, 2023)	Hybrid: RF + KNN	98%	NA
Proposed study	**Stacking: AB+ RF+GB**	NA	**99%**

Table 2: Comparison of Related Work

6.3 Evaluation of the proposed method based on the Confusion Matrix:

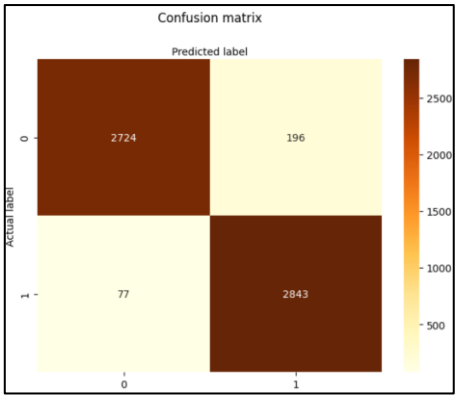

Figure 13: Confusion Matrix (CM) For AdaBoost

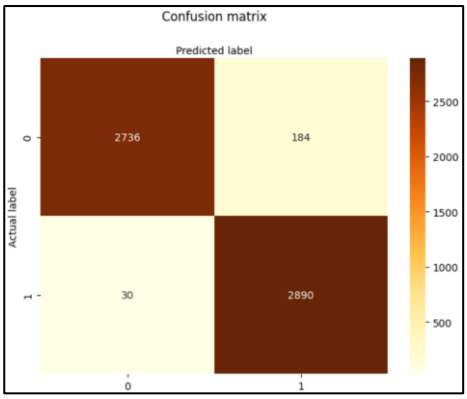

Figure 14: CM for AdaBoost Gradient Boosting

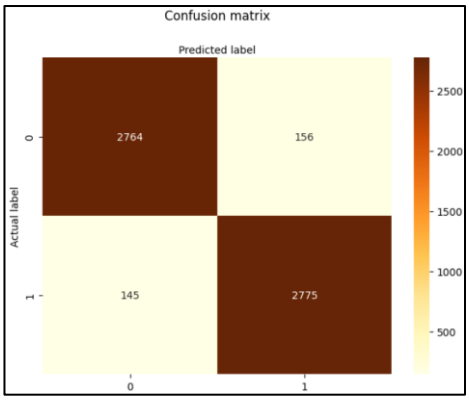

Figure 15: Random Forest CM

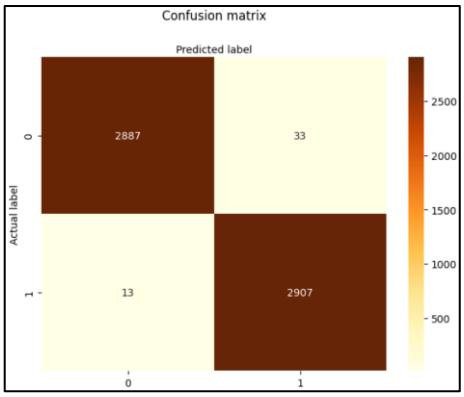

Figure 16: CM for Stacking Classifier

An insightful confusion matrix shows how well the model performs with the data. Below are the results explained using the confusion matrix in Figures 13,14,15,16.

1. **AdaBoost Classifier:** In “Figure 13”, " the confusion matrix reveals the model's classification performance. 2,724 samples were identified correctly as benign i.e. (TN) and the model accurately predicted 2,843 instances as malware, demonstrating a (TP). However, there were 196 (FP) and 77 (FN) classifications, indicating areas for improvement in handling certain data complexities.
2. **Gradient Boosting Classifier:** As per “Figure 14” this classifier was able to achieve 2736 true negatives and 2,890 positive results. There were 184 (FP) and 30 false negatives. However, there is an opportunity for improvement, specifically in dealing with and minimizing the issues related to false positives and false negatives.
3. **Random Forest Classifier:** “Figure 15” shows that the classifier identifies 2,775 instances as benign and 2,764 instances as malware. However, the model faced challenges with 156 (FP) and 145 (FN). Random Forest was the highest in terms of showing misclassified results.
4. **Stacking Classifier:** In “Figure 16” the matrix shows that 2887 instances were predicted as benign correctly i.e. non-malware (TN=2887) and 2907 instances were correctly predicted as malware (TP=2907). For FP, the model has predicted 33 instances as malware when they were benign, and for FN the model has predicted 13 instances as benign but they were malware. The matrix as per "Figure 16" shows the stacking classifier's ability to capture diverse aspects of data and mitigate the limitations of individual models making it a stable choice for PE file categorization in cybersecurity applications.

6.4 Discussion

The project implementation goal was to enhance the accuracy of the detection of malware in PE files. The experiment's results provided a quantitative evaluation of the combined ensemble models in malware or benign classification of PE files. The dataset used in this proposed study was diversified i.e. contains a variety of safe or malware file samples than the related work, in 'Table 2'. The results achieved by the stacking classifier in this study show that performed better when compared to the work done in the previous study by the hybrid model, shown in 'Table 2'. After the implementation of the model in this paper, the accuracy achieved for Gradient Boosting (GB) was 96%, and the accuracy rate for both the AdaBoost and the Random Forest was 95% which is less in the case of the (GB) classifier. The Gradient Boosting algorithms achieved the same accuracy as the accuracy of the (GB) model shown in the related work in Table 2. Though Gradient Boosting performed the same as previous work, the reason for not

outperforming the previous work done as per (Table 2) may be the large set that was used in the previous work by the researchers. In this study, the dataset that is used is not larger compared to the dataset in previous work. Also, when it comes to the individual model in this project the RF model as per the matrix in “Figure 15” displays high false positives and negatives in comparison to other models that are implemented. However, it achieved accuracy higher than the related work, as shown in Table 2. The misclassification results as per “Figure 15” show the need for improvement in the model reliability and the resilience to malware detection in the proposed work. The potential reason may be the model becomes more prone to misclassification when various trees are aggregated because a more complex decision boundary may result. The expectation was that the individual models may perform slightly better than the existing work, as shown in Table 2, and this expectation was achieved with the help of a stacking classifier, combining the three models. The results of this paper show the overall reliability and efficiency in the cyber field for PE file classification.

7 Conclusion and Future Work

The project’s purpose was to improve malware detection in PE files through a hybrid ensemble method. The dataset from Kaggle was used which contains Portable executable Samples. As the dataset was complex only important features were extracted from the dataset. In the further part of the research, three tree-based models and the stacking classifier were used for the detection of the malware. After the implementation of the models, the RF classifier as well as the AdaBoost classifier got an accuracy of 95% percent and the Gradient Boosting Classifier got 96%, but the overall accuracy when aggregated to the stacking classifier was achieved by 99% percent. Also, a user-friendly application was developed that helped in classifying the given file as not safe or benign, building a robust and accurate system to detect malware. The measurements achieved through the evaluation of the hybrid ensemble method for malware detection show significant success in achieving high accuracy, F1 score, precision, and recall. The stacking classifier stands out with a remarkable 99% accuracy rate. This outperforms the accuracy rates reported in existing literature, showcasing the effectiveness of the proposed methodology. The comparative analysis with related work reveals that the hybrid ensemble approach incorporating tree-based and stacking classifiers provides superior results. Enhancing accuracy for malware detection using a hybrid ensemble learning strategy is the main emphasis of the research question. Focusing on the accomplishment of algorithms, the obtained results affirmatively answer the research question by demonstrating that the hybrid ensemble approach significantly enhances the efficiency and accuracy of malware detection. By presenting a very precise hybrid ensemble learning method for real-time malware detection, the study's results have social and industrial implications. Cybersecurity experts can achieve improved decision-making and flexibility to evolving threats by utilizing this strategy.

More research is required to fully comprehend the potential of this ensemble technique. To begin, a larger and more diverse dataset is necessary to accurately depict the expanding malware ecology. Incorporating more recent and complex malware samples into the algorithms can increase their detection of upcoming threats. Furthermore, research might concentrate on optimizing ensemble learning techniques. Experimenting with various combinations of basic and meta-classifiers can lead to more durable and adaptive systems. The integration of advanced feature extraction methods, such as deep learning and dynamic analysis, can also help to enhance malware detection. These methods have the potential to capture intricate patterns and behaviors in malware that traditional features may miss.

References

Abikoye, O., Gyunka, B. and OLUWATOBI, A. (2020) 'Optimizing Android Malware Detection Via Ensemble Learning', *International Journal of Interactive Mobile Technologies (iJIM)*, 14, pp. 61–78. Available at: https://doi.org/10.3991/ijim.v14i09.11548.

Alam, M.S. and Vuong, S.T. (2013) 'Random Forest Classification for Detecting Android Malware', in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pp. 663–669. Available at: https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.122.

Alenezi, M.N. *et al.* (2020) 'Evolution of Malware Threats and Techniques: A Review', *Article in International Journal of Communication Networks and Information Security*, 12(3). Available at: https://doi.org/10.17762/ijcnis.v12i3.4723.

Amer, A. and Aziz, N.A. (2019) *Malware Detection through Machine Learning techniques*.
Bahador, M.B., Abadi, M. and Tajoddin, A. (2019) 'HLMD: a signature-based approach to hardware-level behavioral malware detection and classification', *The Journal of Supercomputing*, 75. Available at: https://doi.org/10.1007/s11227-019-02810-z.

Bentéjac, C., Csörgo, A. and Martínez-Muñoz, G. (2019) 'A comparative analysis of gradient boosting algorithms', *Artificial Intelligence Review*, 54, pp. 1937–1967. Available at: https://api.semanticscholar.org/CorpusID:221283893.

Bhagwat Sakshi and Gupta, G.P. (2022) 'Android Malware Detection Using Hybrid Meta-heuristic Feature Selection and Ensemble Learning Techniques', in V. and G.P.K. and F.J. and Ö.T. Singh Mayank and Tyagi (ed.) *Advances in Computing and Data Sciences*. Cham: Springer International Publishing, pp. 145–156.

Bountakas, P. and Xenakis, C. (2023) 'HELPHED: Hybrid Ensemble Learning PHishing Email Detection', *Journal of Network and Computer Applications*, 210, p. 103545. Available at: https://doi.org/https://doi.org/10.1016/j.jnca.2022.103545.

Braganca, W. and Kho, I.E. (2023) 'Comparative Analysis of Python Microframeworks: Flask, Dash, and CherryPy A Guide for Newly Graduated College Students'. Available at: https://doi.org/10.13140/RG.2.2.17101.82402.

Cloonan, J. (2017) *Advanced Malware Detection - Signatures vs. Behavior Analysis - Infosecurity Magazine, Infosecurity Magazine*. Available at: https://www.infosecurity-magazine.com/opinions/malware-detection-signatures/.

Euh, S. *et al.* (2020) 'Comparative Analysis of Low-Dimensional Features and Tree-Based Ensembles for Malware Detection Systems', *IEEE Access*, 8, pp. 76796–76808. Available at: https://doi.org/10.1109/ACCESS.2020.2986014.

Feng, P. *et al.* (2018) 'A Novel Dynamic Android Malware Detection System With Ensemble Learning', *IEEE Access*, PP, p. 1. Available at: https://doi.org/10.1109/ACCESS.2018.2844349.

Garg, V. and Yadav, R.K. (2019) 'Malware Detection based on API Calls Frequency', in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pp. 400–404. Available at: https://doi.org/10.1109/ISCON47742.2019.9036219.

Gupta, D. and Rani, R. (2020) 'Improving malware detection using big data and ensemble learning', *Computers & Electrical Engineering*, 86, p. 106729. Available at: https://doi.org/10.1016/j.compeleceng.2020.106729.

Ijaz, M., Durad, M.H. and Ismail, M. (2019) 'Static and Dynamic Malware Analysis Using Machine Learning', in *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 687–691. Available at: https://doi.org/10.1109/IBCAST.2019.8667136.

J. Ferreira, A. and Figueiredo, M.A.T. (2012) 'Boosting algorithms: a review of methods, theory, and applications', *Ensemble Machine Learning: Methods and Applications*, pp. 35–85. Available at: https://doi.org/10.1007/978-1-4419-9326-7_2.

Jadhav, S.D. and Channe, H.P. (2013) 'Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques', *International Journal of Science and Research (IJSR) ISSN*, 5. Available at: www.ijsr.net.

Kim, T.N. *et al.* (2022) 'DETECTING MALWARE IN PORTABLE EXECUTABLE FILES USING MACHINE LEARNING APPROACH', *International Journal of Network Security & Its Applications (IJNSA)*, 14(3). Available at: https://doi.org/10.5121/ijnsa.2022.14302.

Kuchipudi, R. *et al.* (2023) 'Android Malware Detection using Ensemble Learning', in *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, pp. 297–302. Available at: https://doi.org/10.1109/ICSCSS57650.2023.10169578.

Kumar Ajay and Abhishek, K. and S.K. and P.D. and J.Y. and C.H. and N.P. (2020) 'Malware Detection Using Machine Learning', in F. and T.S.M. and S.S.K. Villazón-Terrazas Boris and Ortiz-Rodríguez (ed.) *Knowledge Graphs and Semantic Web*. Cham: Springer International Publishing, pp. 61–71.

Liu, K. *et al.* (2020) 'A Review of Android Malware Detection Approaches Based on Machine Learning', *IEEE Access*, PP, p. 1. Available at: https://doi.org/10.1109/ACCESS.2020.3006143.

Mienye, I.D. and Sun, Y. (2022) 'A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects', *IEEE Access*, 10, pp. 99129–99149. Available at: https://doi.org/10.1109/ACCESS.2022.3207287.

Natekin, A. and Knoll, A. (2013) 'Gradient Boosting Machines, A Tutorial', *Frontiers in neurorobotics*, 7, p. 21. Available at: https://doi.org/10.3389/fnbot.2013.00021.

Pallippattu, L. and Mathai (no date) 'Malware detection on android using Adaboost algorithm', in. Available at: https://api.semanticscholar.org/CorpusID:266178751.

Pektas, A. and Acarman, T. (2017) 'Ensemble Machine Learning Approach for Android Malware Classification Using Hybrid Features', in *International Conference on Computer Recognition Systems*. Available at: https://api.semanticscholar.org/CorpusID:13967725.

Rathore, H., Agarwal, S., Sahay, S.K., *et al.* (2018) 'Malware Detection Using Machine Learning and Deep Learning', in *Journées Bases de Données Avancées*. Available at: https://api.semanticscholar.org/CorpusID:54467901.

Rathore, H., Agarwal, S., Sahay, S., *et al.* (2018) 'Malware Detection Using Machine Learning and Deep Learning: 6th International Conference, BDA 2018, Warangal, India, December 18–21, 2018, Proceedings', in, pp. 402–411. Available at: https://doi.org/10.1007/978-3-030-04780-1_28.

Rimon Saiful Islam and Haque, Md.M. (2023) 'Malware Detection and Classification Using Hybrid Machine Learning Algorithm', in G.-W. and M.-S.J.A. and M.E. and T.J.J. Vasant Pandian and Weber (ed.) *Intelligent Computing & Optimization*. Cham: Springer International Publishing, pp. 419–428.

Saad, S., Briguglio, W. and Elmiligi, H. (2019) 'The Curious Case of Machine Learning In Malware Detection', in *International Conference on Information Systems Security and Privacy*. Available at: https://api.semanticscholar.org/CorpusID:88467864.

Savenko, O. *et al.* (2019) 'Dynamic Signature-based Malware Detection Technique Based on API Call Tracing', in *ICTERI Workshops*. Available at: https://api.semanticscholar.org/CorpusID:198184557.

Sayadi, H., Patel, N., D, S.M.P., *et al.* (2018) 'Ensemble Learning for Effective Run-Time Hardware-Based Malware Detection: A Comprehensive Analysis and Classification', *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6. Available at: https://api.semanticscholar.org/CorpusID:49291657.

Sayadi, H., Patel, N., P D, S.M., *et al.* (2018) 'Ensemble Learning for Effective Run-Time Hardware-Based Malware Detection: A Comprehensive Analysis and Classification', in. Available at: https://doi.org/10.1145/3195970.3196047.

Sethi, K. *et al.* (2019) 'A Novel Machine Learning Based Malware Detection and Classification Framework', *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–4. Available at: https://api.semanticscholar.org/CorpusID:207831218.

Shafin, S.S. *et al.* (2021) 'Detection of Android Malware using Tree-based Ensemble Stacking Model', in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pp. 1–6. Available at: https://doi.org/10.1109/CSDE53843.2021.9718396.

Singh, Jagsir and Singh, Jaswinder (2021) 'A survey on machine learning-based malware detection in executable files', *Journal of Systems Architecture*, 112, p. 101861. Available at: https://doi.org/https://doi.org/10.1016/j.sysarc.2020.101861.

Smmarwar Santosh K. and Gupta, G.P. and K.S. (2022) 'A Hybrid Feature Selection Approach-Based Android Malware Detection Framework Using Machine Learning Techniques', in N. and G.B.B. and M.P.G. Agrawal Dharma P. and Nedjah (ed.) *Cyber Security, Privacy and Networking*. Singapore: Springer Nature Singapore, pp. 347–356.

Stokes, J.W. *et al.* (2010) 'WebCop: locating neighborhoods of malware on the web', in *Proceedings of the 3rd USENIX Conference on Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*. USA: USENIX Association (LEET'10), p. 5.

Thosar, K. *et al.* (2021) 'Effective Malware Detection using Gradient Boosting and Convolutional Neural Network', in *2021 IEEE Bombay Section Signature Conference (IBSSC)*, pp. 1–4. Available at: https://doi.org/10.1109/IBSSC53889.2021.9673266.

Ucci, D., Aniello, L. and Baldoni, R. (2019) 'Survey of machine learning techniques for malware analysis', *Computers & Security*, 81, pp. 123–147. Available at: https://doi.org/https://doi.org/10.1016/j.cose.2018.11.001.

Wu, C. *et al.* (2018) 'Enhancing Machine Learning Based Malware Detection Model by Reinforcement Learning', in *ICCNS 2018: Proceedings of the 8th International Conference on Communication and Network Security*, pp. 74–78. Available at: https://doi.org/10.1145/3290480.3290494.

Yerima, S., Muttik, I. and Sezer, S. (2015) 'High Accuracy Android Malware Detection Using Ensemble Learning', *IET Information Security* [Preprint]. Available at: https://doi.org/10.1049/iet-ifs.2014.0099.