Journal of Information Systems Engineering and Management
2025, 10(15s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Applying Ensemble Machine Learning Techniques to Malware Detection

Saad Mamoun AbdelRahman Ahmed
Applied College, King Faisal University, KSA. smaahmed@kfu.edu.sa

ARTICLE INFO

Received: 28 Nov 2024

Revised: 22 Jan 2025

Accepted: 01 Feb 2025

ABSTRACT

Cybersecurity is facing serious problems with the proliferation of malware in the internet world. The ever-changing nature of malicious software makes it difficult for traditional detection technologies to keep up. In order to make malware detection systems more accurate and resilient, this study investigates how to apply ensemble machine learning techniques. Using meta-learning frameworks like stacking and boosting in conjunction with various base models like logistic regression, Gaussian Naïve Bayes, and random forest allows the suggested method to make the most of each model's strengths while reducing their shortcomings. By utilizing a large dataset that includes both malicious and benign samples, the stacking algorithm surpassed the rivals in the prediction process, with a recall, precision, and f1-score of 100 after using the encoding method to convert the dataset from a numerical to a categorical format.

Keywords: lorem ipsum.

INTRODUCTION

The digital revolution has brought immense convenience and connectivity to modern life, but it has also introduced significant security challenges. Malware, or malicious software, remains one of the most persistent threats to information systems. Malware's ability to infiltrate networks, compromise data, and disrupt operations poses serious risks to both individuals and organizations. Despite advances in cybersecurity technologies, the rapid evolution of malware techniques continues to outpace traditional defense mechanisms. Signature-based methods, which rely on known patterns of malware behavior, are often ineffective against sophisticated, previously unseen variants. This growing threat highlights the urgent need for innovative detection methods capable of adapting to the complex and dynamic nature of malware [1, 2].

One promising solution lie is using machine learning (ML) techniques, particularly ensemble methods, to malware detection. Machine learning has revolutionized cybersecurity by enabling systems to identify patterns and anomalies indicative of malicious activity. Ensemble learning, a subfield of machine learning, combines the predictive power of multiple models to achieve higher accuracy and resilience. By aggregating the outputs of algorithms, ensemble techniques address the limitations of individual models, such as overfitting and poor generalization. This makes them particularly well-suited for the multifaceted nature of malware detection, where distinguishing between legitimate and malicious behavior often requires nuanced analysis [3].

Ensemble methods like stacking, bagging, and boosting have shown exceptional promise in malware detection applications. Bagging techniques, like Random Forest, train multiple decision trees on random subsets of data, enhancing robustness and reducing variance. Boosting algorithms like XGBoost, focus on correcting the weaknesses of previous models by iteratively improving predictions. Stacking, on the other hand, combines the outputs of multiple base models using a meta-model, achieving a synergistic effect. These techniques not only improve detection accuracy but also reduce false positives, a critical factor in maintaining user trust and system performance [4]. Pretium vulputate sapien nec sagittis aliquam malesuada. Auctor neque vitae tempus quam. Aenean sed adipiscing diam donec adipiscing. Magnis dis parturient montes nascetur ridiculus mus mauris. Placerat in egestas erat imperdiet sed euismod nisi porta lorem. Vel facilisis volutpat est velit egestas dui. Ultrices gravida dictum fusce ut placerat orci nulla pellentesque dignissim. Egestas tellus rutrum tellus pellentesque eu tincidunt tortor aliquam nulla. Mattis pellentesque id nibh tortor id. Ut venenatis tellus in metus vulputate.

The success of ensemble learning in malware detection hinges on its ability to analyze large, complex datasets generated by modern computing systems. Malware detection systems often process vast amounts of data, including

network logs, file metadata, and application behavior, to identify potential threats. Ensemble models excel in handling such datasets due to their capacity to process diverse features and learn intricate patterns. Feature engineering plays a vital role in this context, enabling the extraction of relevant characteristics that differentiate between benign and malicious entities. Advanced techniques like natural language processing (NLP) for analyzing code snippets and deep feature extraction for behavioral analysis further enhance the capabilities of ensemble-based models [5, 6].

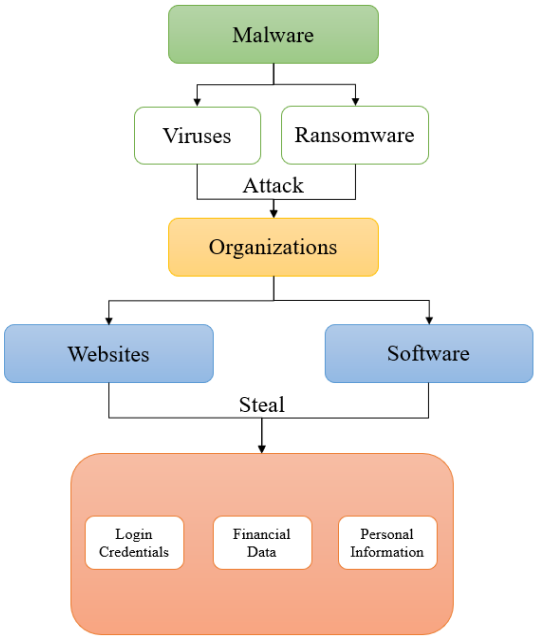
In practice, deploying ensemble learning for malware detection requires overcoming several challenges. First, the dynamic nature of malware necessitates regular model updates to maintain accuracy. Techniques such as transfer learning and online learning can be employed to adapt models to new threats. Second, computational complexity can be a barrier, especially for resource-constrained environments like mobile devices. Optimizing ensemble architectures and employing lightweight models can address this issue. Finally, interpretability remains a key concern, as ensemble models are often seen as "black boxes." Enhancing model transparency through explainable AI (XAI) techniques is essential to building trust and ensuring compliance with regulatory standards [7, 8].

This research explores the potential of ensemble techniques to redefine malware detection systems. Through rigorous experimentation on real-world datasets, it evaluates the effectiveness of various ensemble approaches in identifying known and novel malware threats. By integrating advanced feature engineering, adaptive learning strategies, and explainability tools, the study aims to develop a framework for malware detection that balances accuracy, efficiency, and transparency [9].

The findings of this research have significant implications for cybersecurity, particularly in protecting critical infrastructure and sensitive data. Ensemble-based malware detection systems can provide organizations with a proactive defense mechanism, capable of identifying and mitigating threats before they cause harm. Furthermore, the scalability of these systems makes them suitable for a wide range of applications, from enterprise networks to individual devices. As the digital landscape continues to evolve, the adoption of ensemble machine learning techniques in malware detection represents a vital step toward building resilient and secure computing environments [10].

OBJECTIVES

The objective of this research is to utilize a hybrid machine learning algorithms to enhance the accuracy of predictions. Figure 1 shows the problem statement of this paper.



The sections reminder for this paper is as follows: Section 2 presents the summary of previous papers. Section 3 describes the methodology. Section 4 illustrates the proposed ensemble algorithm. Section 5 explains the results based on performance metrics. Finally, we conclude this paper and suggest some future works.

LITERATURE REVIEW

Table 1 shows the summarization of the related papers terms of the algorithms used, the dataset, and the results based on evaluation metrics.

Gupta et al. [11] conducted two approaches to enhance malware detection performance on a wide scale, one using ensemble learning and the other utilizing big data. The first approach uses ensemble learning's weighted voting process, whereas the second selects the best set of base classifiers to stack. We test and assess the performance of the suggested approaches using a dataset including 98,150 benign and 100,200 malicious samples. Results from experiments showed that the suggested method works, as it enhances generalization performance for identifying new malware.

A hybrid deep learning model including gate recurrent unit (GRU) and deep belief network (DBN) was proposed by [12] as the basis for an algorithm for Android virus detection. Before anything else, examine the Android virus; not only are static features taken, but dynamic behavioral features with significant antiobfuscation ability are as well. create an Android malware detection model that combines deep learning with a hybrid approach. they used the DBN to process the static characteristics because they are generally independent. The GRU is employed to process the series of dynamic features because to their temporal correlation. In the end, the BP neural network receives the training data from DBN and GRU and produces the final classification results. Android malware detection model based on hybrid deep learning techniques outperforms other algorithms.

Akhtar et al. [13] employed a many machine learning algorithms to detect harmful threats and malware. The ML algorithms are Naive Byes, J48, RF, SVM, and proposed approach). The findings demonstrated that DT 99%, outperformed other classifiers in terms of detection accuracy. Malware detection results on a tiny FPR were evaluated for the DT, CNN, and SVM algorithms on a specific dataset. DT achieved 2.01%, CNN achieved 3.97%, and SVM achieved 4.63%. Since harmful software is growing in both prevalence and sophistication, these findings are noteworthy.

Using a number of static and dynamic features, Aurangzeb et al. [14] suggested BigRC-EML as a ransomware detection and classification tool. For better ransomware detection results, they employed ensemble machine learning techniques on large datasets. In addition, a novel method for selecting features that reduces feature dimensions is introduced, which is based on Principle Component Analysis (PCA). The study used two types of datasets: one dynamic, which included 582 ransomware and 942 clean programs, and another hybrid, which included 500 applications. In this case, they were using XGBoost, Neural Network, SVM, Random Forest, and KNN as their classification models. According to their testing data, BigRC-EML attained a 98% accuracy rate, and Neural Network performed better than the other models.

TABLE 1: PREVIOUS PAPERS SUMMARIZATION

Ref	Year	Techniques	Dataset Size	Performance Metrics	Experimental Results
[11]	2020	Voting Stacking	193,530 Windows files	Precision, Recall, Accuracy, F1-score	Enhancing parameterization to improve the identification of new malware.
[12]	2020	DBN and GRU	7000 benign files and 6298 malware files	Precision, Recall, Accuracy	Accuracy = 96.82%
[13]	2022	Naive Bayes, SVM, J48, RF, Proposed Approach	17,394 files	Accuracy, TPR, FPR	Decision Tree (DT) accuracy = 99%
[14]	2022	BigRC-EML, XGBoost, Neural Network, SVM, Random Forest, KNN	582 ransomware files and 942 clean files	Accuracy	BigRC-EML accuracy = 98%

METHODS

A. Malware Dataset

In this paper, we used the Malware Detection in Network Traffic Dataset obtained from the Kaggle website that includes 23145 samples with 23 columns labels that shown in Table 2. In order to give researchers and analysts of

network malware more complete information, this dataset describes the links between flows associated with harmful or potentially malicious activity. Using malware capture analysis, the Stratosphere labs meticulously developed these labels.

TABLE. 2 DATASET FEATURES

Field Name	Description	Type
ts	The timestamp of the connection event.	string
uid	A unique identifier for the connection.	string
id.orig_h	The source IP address.	addr
id.orig_p	The source port.	port
id.resp_h	The destination IP address.	addr
id.resp_p	The destination port.	port
proto	The network protocol used (e.g., tcp).	string
service	The service associated with the connection.	string
duration	The duration of the connection in seconds.	interval
orig_bytes	The number of bytes sent from the source to the destination.	count
resp_bytes	The number of bytes sent from the destination to the source.	count
conn_state	The state of the connection.	string
local_orig	Indicates whether the source is considered local or not.	bool
local_resp	Indicates whether the destination is considered local or not.	bool
missed_bytes	The amount of missed data in the connection.	count
history	A history of data events related to the connection.	string
orig_pkts	The number of packets sent from the source to the destination.	count
orig_ip_bytes	The number of IP bytes sent from the source to the destination.	count
resp_pkts	The number of packets sent from the destination to the source.	count
resp_ip_bytes	The number of IP bytes sent from the destination to the source.	count
tunnel parents	Indicates if this connection is part of a tunnel.	set[string]
label	A label associated with the connection (e.g., “Malicious” or “Benign”).	string
detailed-label	A more detailed description or label for the connection.	string

[1] https://www.kaggle.com/datasets/agungpambudi/network-malware-detection-connection-analysis/data

Figure 2 presents a first and last five rows of the dataset before we applied any preprocessing steps.

	ts	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	proto	service	duration	orig_bytes
0	1.545404e+09	CrDn63WjJEmrWGjqf	192.168.1.195	41040	185.244.25.235	80	tcp	-	3.139211	0
1	1.545404e+09	CY9lJW3gh1Eje4usP6	192.168.1.195	41040	185.244.25.235	80	tcp	-	-	-
2	1.545404e+09	CcFXLynukEDnUlvgI	192.168.1.195	41040	185.244.25.235	80	tcp	-	-	-
3	1.545404e+09	CDrkrSobGYxHhYfth	192.168.1.195	41040	185.244.25.235	80	tcp	http	1.477656	149
4	1.545404e+09	CTWZQf2oJSvq6zmPAc	192.168.1.195	41042	185.244.25.235	80	tcp	-	3.147116	0
...	...	...	...	...	...	...	...	...	...	...
23140	1.545490e+09	C2F17zSUnGOcWzBa7	192.168.1.195	57110	185.244.25.235	6667	tcp	irc	32.840994	62
23141	1.545490e+09	C93P4z4k5IRJD1rXJg	192.168.1.195	57092	185.244.25.235	6667	tcp	irc	36.290833	62
23142	1.545490e+09	CXLZ3A2QY5E8weqpDk	192.168.1.195	123	147.251.48.140	123	udp	-	-	-
23143	1.545490e+09	CuXpFN3fWesWBXUhq1	192.168.1.195	123	82.113.53.40	123	udp	-	-	-
23144	1.545490e+09	Ct2Yhy4d33oL3yyZY9	192.168.1.195	123	89.221.210.188	123	udp	-	-	-

Fig2: Dataset Sample.

Table 3 and Figure 3 shows frequency of these labels.

TABLE 3: FREQUENCY OF THE LABELS

Label	Type	Count
Benign	Normal	1923
C&C	Malicious	6707

DDoS	Malicious	14394
PartOfAHorizontalPortScan	Malicious	122

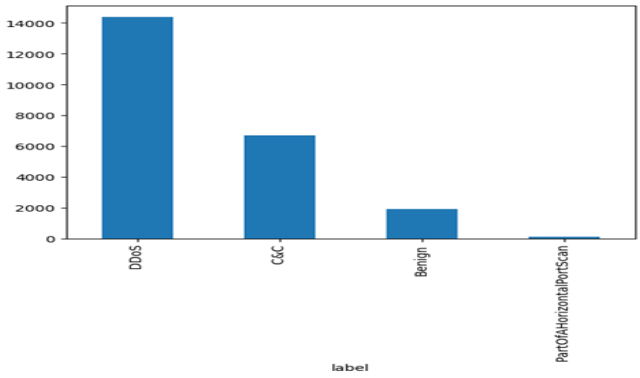


Fig. 3: Frequency of the Labels

B. Building Models

In order to determine if a file was benign or malicious, we used four different machine learning methods for classification on malware dataset. Computers may learn from data and generate inferences or predictions without human intervention through the use of machine learning algorithms, which are models for computational intelligence.

1- Random Forest (RF) Algorithm

Ensemble learning, which random forests use, combines several decision trees to produce predictions for regression and classification jobs. Among the many benefits of ensemble learning in machine learning are improved performance, resilience, and the capacity to handle complex problems. To improve their predictive power, random forests use ensemble learning techniques. While calculating the mean for the regression task, the RF calculates the average prediction for all trees in the classification job. At its heart, random forests are structured around decision trees. As illustrated in Figure 4, decision trees are hierarchical models that provide predictions by utilizing binary splits on features. In order to forecast a target variable, each division first divides the data into smaller subsets based on certain criteria.

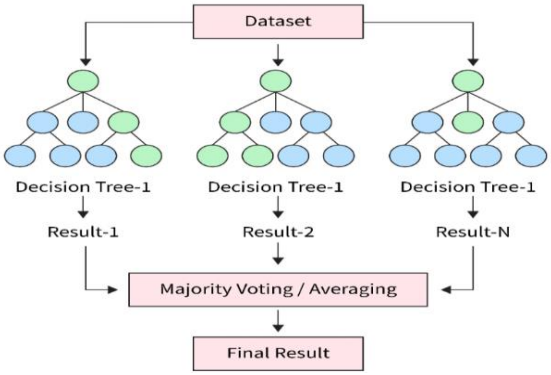


Fig. 4: RF Algorithm

Through the amalgamation of numerous decision trees' predictions, random forests are able to improve accuracy, reduce overfitting, and efficiently handle complex situations. Random forests improve prediction accuracy by incorporating many decision trees to better understand the data and provide more robust forecasts. Equation 1 shows how the RF calculates the final prediction, indicated by y, number of trees refers to N and hi(x) is the prediction for each decisions tree.

$$y\,(x) = \frac{1}{N}\sum_{i=1}^{N} h_i(x) \qquad Eq.\,(1)$$

1- Gaussian Naïve Bayes (NB) Algorithm

An adaptation of the Naïve Bayes method, Gaussian Naïve Bayes (GNB) is created with continuous data in mind. For issues where this assumption holds true, it is especially successful because it assumes that the characteristics follow a normal distribution, or Gaussian distribution. Based on Bayes' theorem, all features are independent given the class label, it is similar to all Naïve Bayes classifiers, as shown in Equation 2.

There are three Important Features about GNB:

- Using the input features, GNB determines the posterior probability of each class and then chooses the class that has the highest probability. This makes GNB a probabilistic classifier.
- Modeling the feature likelihood using a Gaussian distribution, defined by the mean and variance of the data, is the Gaussian distribution assumption.
- Quick and Scalable: GNB's processing efficiency and ability to handle big datasets are both contributed by its simplicity.

$$P(C_k \backslash \mathrm{X}) = \frac{P(X \backslash C_k)\, P(C_k)}{\mathrm{P(X)}} \qquad Eq.(2)$$

Where:

- P(Ck\X): X is the feature vector, and Ck is the posterior class.
- P(Ck): Prior probability.
- P(X\Ck): Likelihood of the feature vector X given class Ck.
- P(X): Marginal likelihood (constant for all classes).

2- Logistic Regression (LR) Algorithm

When it comes to binary classification tasks, one popular supervised learning approach is logistic regression. To be clear, it's not a regression model; rather, it's a categorization model. By mapping predictions to a range between 0 and 1, logistic regression commonly employs the logistic (sigmoid) function. It then uses this range to estimate the likelihood that an input belongs to a certain class. The following is an example of how logistic regression models the relationship between input attributes (X) and the likelihood of a binary target variable (Y):

$$\mathrm{P(Y = 1|X)} = \sigma(\mathrm{z}) = \frac{1}{1 + \mathrm{e}^{-\mathrm{z}}}$$

C. Performance Metrics

F1-score, accuracy, recall, and precision are the four metrics utilized to evaluate a classification task. TP: True Positive, FN: False Negative, TN: True Negative, and FP: False Positive are acronyms used in the following formula and simplified explanation:

Accuracy: is the percentage of correctly predicted relative to all samples.

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \qquad (3)$$

Precision: is the proportion of anticipated positive samples that really.

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad (4)$$

The recall is the proportion of correctly predicted positive samples to the total number of expected positive samples.

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad (5)$$

F1-score: is calculated by computing the weighted average of Precision and Recall.

$$\text{F1} - \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

PROPOSED ENSEMBLE LEARNING APPROACH

Proposed Technique Overview

The proposed Technique used is a Stacking classifier that included three algorithms: RF, GNB and LR.

Preprocessing

We applied a label encoding method as a preprocessing step. this preprocessing method employed to convert the categorical data to numerical data a following: A unique integer is assigned to each distinct category (label) in the data. It is frequently employed in situations where the categorical data lacks an intrinsic order.

Training Machine Learning Classifiers

There were labels in the training data that showed whether a certain output was predicted to be in a certain class. By comparing it to the reference data, the main goal is to teach the learning model to correctly pinpoint the position of unknown data. However, we found that in some cases, the best results or the fewest mistakes could have been achieved with just one learning model. In order to correctly determine the sample's location, we used an ensemble learning strategy, which comprised developing numerous hypotheses from the training data and then combining them. By combining the results of multiple models, this method greatly enhanced the model's overall efficiency, which in turn increased the outputs' accuracy. More so than with individual models, this approach produced a robust and stable model. We train each machine learning classifier in our ensemble in a systematic way so that we can build our ensemble model. The LR Classifier, Random Forest, and GNB are all part of the classifiers discussed in previous Section. Each classifier's unique structure, hyperparameters, and capabilities are vital to a thorough learning process. After training, these classifiers form the basis of the ensemble method. By utilizing the Stacking method, this strategy improves the ensemble's capability.

Ensemble Stacking Classifier

Stacking, also known as Stacked Generalization, is a method of ensemble learning that constructs a more robust meta-model by integrating the predictions of numerous base models, which are considered weak learners individually. By learning to integrate the outputs of multiple models through a meta-model, stacking takes advantage of the strengths of each model, unlike other ensemble methods such as bagging or boosting, as shown in Figure 5.

The Process of Stacking:

- *Level-0 Base Models: A number of models are trained separately using the same dataset. The dataset is used to create predictions for each base model.*
- *A meta-model is a distinct model that is trained using the base models' predictions. The meta-model figures out how to merge these forecasts for maximum efficiency. Although more complicated models (such as Gradient Boosting) are also possible, the meta-model is typically a simpler one, such Logistic Regression.*
- *The basic models use additional data to construct predictions during inference, which leads to the final prediction. The meta-model takes these forecasts and produces a final forecast.*

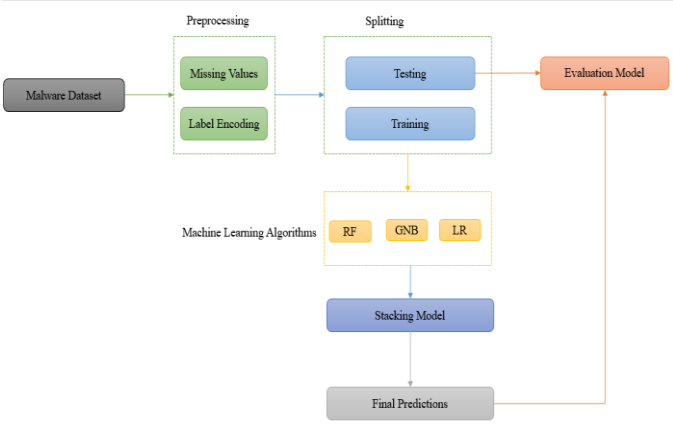


Fig. 5: Flow Chart of Stacking Algorithm

RESULTS

Here we detailed the experimental outcomes of using four ML methods on malware dataset and evaluating them using four metrics: f1-score, accuracy, recall, and precision. After that, we have a look at the experimental setups for each algorithm, which contain the values of the parameters.

Experimental Setup

This section presents the parameters used for each algorithm, as shown in Table 4.

TABLE 4: PARAMETERS OF MACHINE LEARNING ALGORITHMS

Algorithm	Parameters
RF	n_estimators is 100 criterion is gini max_depth is 2 random_state is 42.
GNB	Priors = None var_smoothing= 1e-9
LR	Penalty = l2 Solver = lbfgs max_iter = 100
Stacking classifier	estimators = RF, GNB final_estimator = LR

The dataset needs to be split into two halves, called training and testing, before it can be fed into machine learning algorithms. It is possible to construct models using these techniques on the training dataset and then evaluate their efficacy on the testing dataset. The following is the training-to-testing ratio used in this paper: During training, 80% of the dataset is utilized, while the remaining portion is utilized for testing.

Experimental Results

Here we show the outcomes of applying the three machine learning algorithms discussed in this article—RF, GNB, and LR—to the malware dataset. Following that, we contrasted the outcomes with a stacking classifier using four assessment metrics: accuracy, f1-score, recall, and precision.

After applying the four machine learning algorithms to the dataset, the performance results are shown in Figure 6 and Table 5. In comparison to other machine learning algorithms, Stacking outperforms them in the prediction process, with scores of 99.9 for accuracy, 100 for precision, 100 for recall, and 100 for f1-score.

Table 5:Performance Results

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
GNB	99.24	99	99	99
RF	99.15	99	99	99
LR	97.66	98	98	98
Stacking	99.9	100	100	100

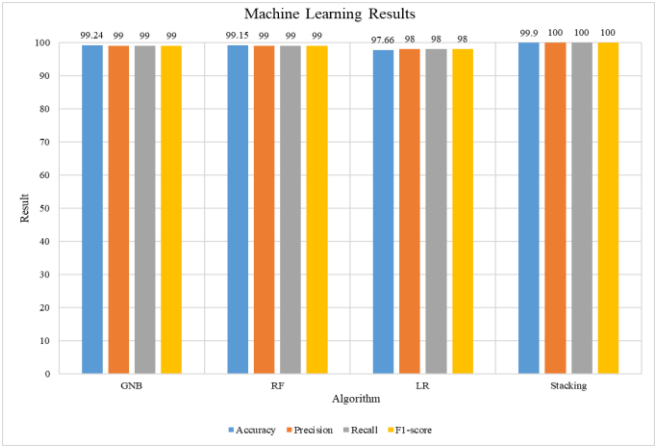


Fig 6:Performance Results

DISCUSSION

Here we go over the results of our experiment to identify malware assaults using the dataset that was described earlier. These results are derived from a stacking classifier, RF, GNB, LR. The stacking classifier achieved better. The authors of [19] used deep learning algorithms on the malware dataset, and the results reveal an accuracy of 96.82, as shown in Table 6. Using 7 classifiers, other articles achieved 98% accuracy, such as [14]. Using the same dataset, our results showed an improvement of 99.9 percent accuracy.

TABLE 6:COMPARISON BETWEEN PREVIOUS AND OUR WORK

Ref	Year	Algorithms	Results
[12]	2020	DBN and GRU	Accuracy = 96.82
[14]	2022	BigRC-EML, XGBoost, Neural Network, SVM, Random Forest, and KNN	BigRC-EML accuracy = 98%
Our Work	2024	RF, GNB, LR, Stacking	Accuracy of stacking = 99.9

The following objectives were met in light of our findings:

- In order to achieve the highest detection accuracy in the dataset when compared to other algorithms in this study or earlier work, we constructed a strong ensemble learning method using three algorithms: RF, GNB, and LR.
- Even though we used the same computer settings, our detection method was faster than prior studies.
- The robust technique embedded into the dataset enables a greater accuracy value.

Here are the main points that highlight the contributions of this paper:

- To distinguish benign from malicious files in a bigger malware dataset, we need to create an ensemble learning approach that makes use of resilient machine learning techniques.
- Minimizing false positives and missed detections by attaining a high degree of accuracy in differentiating between benign and malicious files. Our top priority is to minimize the possibility of falsely tagging reputable files while also making sure that the detection system can properly identify malware assaults.
- Greater Detection Precision: By sifting through mountains of data, machine learning systems can spot the telltale signs of malware attempts. As a result, the detection accuracy is higher than with more conventional approaches.

CONCLUSION

In order to determine the file type and normalcy, this research applies four machine learning algorithms to malware datasets. Afterwards, we transformed the dataset from a categorical to a numerical format by applying the encoding technique. With a precision of 100, a recall of 100, and a f1-score of 100, the results showed that the stacking algorithm outperformed the competitors in the prediction process. The aforementioned dataset, along with another machine learning dataset, will be subject to the deep learning algorithms in future study.

REFRENCES

[1] Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-Rimy, B. A. S., Eisa, T. A. E., & Elnour, A. A. H. (2022). Malware detection issues, challenges, and future directions: A survey. Applied Sciences, 12(17), 8482.

[2] Aslan, Ö. A., & Samet, R. (2020). A comprehensive review on malware detection approaches. IEEE access, 8, 6249-6271.

[3] Rincy, T. N., & Gupta, R. (2020, February). Ensemble learning techniques and its efficiency in machine learning: A survey. In 2nd international conference on data, engineering and applications (IDEA) (pp. 1-6). IEEE.

[4] Zounemat-Kermani, M., Batelaan, O., Fadaee, M., & Hinkelmann, R. (2021). Ensemble machine learning paradigms in hydrology: A review. Journal of Hydrology, 598, 126266.

[5] Damaševičius, R., Venčkauskas, A., Toldinas, J., & Grigaliūnas, Š. (2021). Ensemble-based classification using neural networks and machine learning models for windows pe malware detection. Electronics, 10(4), 485.

[6] Azeez, N. A., Odufuwa, O. E., Misra, S., Oluranti, J., & Damaševičius, R. (2021, February). Windows PE malware detection using ensemble learning. In Informatics (Vol. 8, No. 1, p. 10). MDPI.

[7] 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.

[8] Islam, R., Sayed, M. I., Saha, S., Hossain, M. J., & Masud, M. A. (2023). Android malware classification using optimum feature selection and ensemble machine learning. Internet of Things and Cyber-Physical Systems, 3, 100-111.

[9] Alamro, H., Mtouaa, W., Aljameel, S., Salama, A. S., Hamza, M. A., & Othman, A. Y. (2023). Automated android malware detection using optimal ensemble learning approach for cybersecurity. IEEE Access.

[10] Al-Andoli, M. N., Sim, K. S., Tan, S. C., Goh, P. Y., & Lim, C. P. (2023). An ensemble-based parallel deep learning classifier with PSO-BP optimization for malware detection. IEEE Access.

[11] Gupta, D., & Rani, R. (2020). Improving malware detection using big data and ensemble learning. Computers & Electrical Engineering, 86, 106729.

[12] Lu, T., Du, Y., Ouyang, L., Chen, Q., & Wang, X. (2020). Android malware detection based on a hybrid deep learning model. Security and Communication Networks, 2020(1), 8863617.

[13] Akhtar, M. S., & Feng, T. (2022). Malware analysis and detection using machine learning algorithms. Symmetry, 14(11), 2304.

[14] Aurangzeb, S., Anwar, H., Naeem, M. A., & Aleem, M. (2022). BigRC-EML: big-data based ransomware classification using ensemble machine learning. Cluster Computing, 25(5), 3405-3422.