Ensemble Learning

Master of Engineering
Information Technology
Hasibuzzaman
Matriculation Number: 1448140
hasibuzzaman@stud.fra-uas.de

***Abstract* — A potent machine learning technique called ensemble learning combines several weak learners to produce a strong, stable model with better prediction precision as well as stability. This study provides an extensive examination of ensemble learning algorithms and their implications in cybersecurity, with a particular focus on bagging and boosting. Sequential and parallel ensemble types are distinguished, and the latter is further subdivided into heterogeneous and homogeneous ensembles. Robustness and accuracy are the foundations of ensemble models' performance, and diversity across the merged models plays a critical role in improving predictive accuracy and resolving problems like overfitting and high-dimensional data. Ensemble learning has enormous potential to improve cybersecurity with probable usage in threat intelligence, computer security, anomaly detection, fraud detection, malware identification, intrusion detection, and phishing prevention. Organizations can enhance detection and preventive actions and create more secure systems and networks by utilizing the capabilities of several models. As ensemble learning continues to progress, real-time analytics and adaptive learning will be further strengthened, strengthening cybersecurity defenses against ever-changing threats.**

Keywords* — *Weak Learners, Strong Learner, Bagging, Boosting, Ensemble Learning, Intrusion Detection Systems, IDS, Cybersecurity, malware detection, anomaly detection, Random Forest, AdaBoost.

I. INTRODUCTION

Machine learning's ultimate goal is to analyse data and find traits in common. It entails using data sets to make predictions. Learning is a crucial strategy that builds on weak models to produce strong ones, increasing prediction accuracy and stability. This method is widely used in several fields, including computer science and cybersecurity[1][2]. Ensemble learning trains several models to address a given problem rather of creating a single one. It might also be referred to as multiple classifier systems or team learning. These techniques use the strengths of multiple models to offset the limitations of each one, increasing prediction accuracy more than single models could. Ensemble learning refers to the process of making decisions by combining multiple simple models [5]. There are two types:

a) **Sequential** ensemble model - It teaches new base learners by focusing on the mistakes made before, improving overall performance (like AdaBoost). [Figure 4.1]

b) **Parallel** ensemble model - This one teaches each base learner separately without moving the wrong predictions back into training (like Random Forest). [Figure 4.2]

There are two types of parallel ensembles: homogeneous (using the same model as weak learners) and heterogeneous (combining using various approaches).[27]

Several models combined provide better outcomes than a single model when used in ensemble learning. Accuracy and robustness are key to its success. Accuracy is making exact forecasts, whereas robustness is utilizing all the predictions through voting. When configuring these models, there are numerous variables to take into account and a plethora of algorithms to select from. Selecting the appropriate algorithm and settings requires statistical model comparison and performance estimation through testing [25].

According to research, ensembles with a variety of mixed models are more accurate. With no additional bias, this variety aids in solving regression issues like overfitting. In particular, mixed under-regularized models perform exceptionally well. When managing high-dimensional data, ensemble methods excel as well and offer a great substitute for conventional methods [5]. Through the utilization of several model capabilities, ensemble learning enhances predictive capability and efficiently handles intricate, high-dimensional information.

This article explores ensemble learning approaches, with particular emphasis on Bagging and Boosting (rather than stacking) and concludes with their own detailed examples of applying ensemble learning for malware detection, anomaly detection, and intrusion detection systems. After reading this, you'll have a thorough understanding of these techniques and their benefits. Despite their widespread use in many domains, this article focuses on how they precisely address real-world cybersecurity issues and considers potential future uses as cybersecurity advances.

II. ENSEMBLE LEARNING IN CYBERSECURITY: FUTURE POSSIBILITIES

Ensemble learning approaches have the potential to improve a variety of industries, including healthcare, renewable energy, and cybersecurity, by combining numerous models to achieve increased robustness and accuracy. This method has a number of cutting-edge uses and potential developments. The enhancement of intrusion detection systems (IDS) is one of them. When anomaly-based and signature-based detection techniques are combined, better detection rates and fewer false positives are expected. Online learning ensemble models make it possible to build real-time IDS. When it comes to malware detection, ensemble learning combines both static (code examination) and dynamic (runtime behavior) analytics to examine software behavior patterns and find unknown infections. Emails are categorized according to their content and sender information in order to weed out phishing attempts. Moreover, security is improved by examining URLs for harmful characteristics. [22][24][25]

Using a range of network traffic features improves the detection of anomalies in network traffic. While adaptive learning models that learn from fresh data can identify developing dangers, these properties aid in the detection of abnormalities. To find bogus transactions, ensemble learning looks at transaction patterns and user behavior. Finding anomalies is made easier by combining various user behavior models. By merging information from many threat

intelligence feeds, ensemble learning improves threat intelligence. This offers thorough threat notifications. Predictive analysis also gains from using threat intelligence data—both historical and present—to anticipate possible threats. Host-based Intrusion Prevention Systems (HIPS) improve endpoint security. [23] In addition to keeping an eye on endpoint activity to prevent malicious activity, these systems also have file integrity monitoring to spot unauthorized changes made to important files.[8]

Techniques for ensemble learning provide strong tools to address various cybersecurity issues. Through the utilization of various models' combined strengths, businesses can improve their detection and prevention tactics, leading to more secure networks and systems. Future advancements will strengthen cybersecurity defenses considerably by improving real-time analytics and adaptive learning capabilities [25].

III. Detailed Description of Ensemble Methods

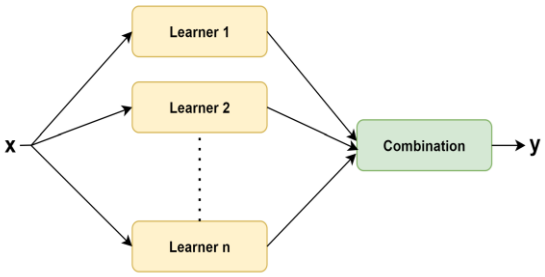


Fig. 3.1. Common Ensemble Architecture

The starting point is the training dataset, or "x" in this instance (your input data). There are a lot of weak learners in the middle part. They are referred to as "learner 1," "learner 2," and so forth. Our foundational learners are these basis learners. This procedure culminates in the integration of these foundation learners to yield our ultimate prediction, denoted as "y".

Usually, weak learners form this ensemble by simple averaging or majority voting. Which types of students are these weak ones? You can see how KNN, Decision Trees, Logistic Regression, and even Support Vector Machines (SVM) are applied here. When combined, these models provide a potent ensemble, or prediction model. The ensemble's prediction P, which is only the average of the predictions from three different models, is a continuous task in regression.

$$P = \frac{p_1 + p_2 + p_3}{3}$$

When the weights w_1, w_2 *and* w_3 add up to 1, the ensemble prediction P, which is the average of the weighted, is essentially a combination of several distinct forecasts added together.

$$w_1 + w_2 + w_3 = 1$$

$$P = w_1 . p_1 + w_2 . p_2 + w_3 . p_3$$

I employ a voting method in my attempt to forecast a class label. It's comparable to taking the mode of each guess. This makes sense with discrete functions; thus, it works rather well. Consider a system that classifies objects in binary terms, such as "True" or "False." This approach is quite dependable.

As far as we know, in 1990, Hansen and Salamon made a noteworthy observation. They stated that it is often better to use an ensemble rather than relying only on the best model that is available. I even have this simple, tidy data visualization where the x label indicates noise level, and the y axis shows error:

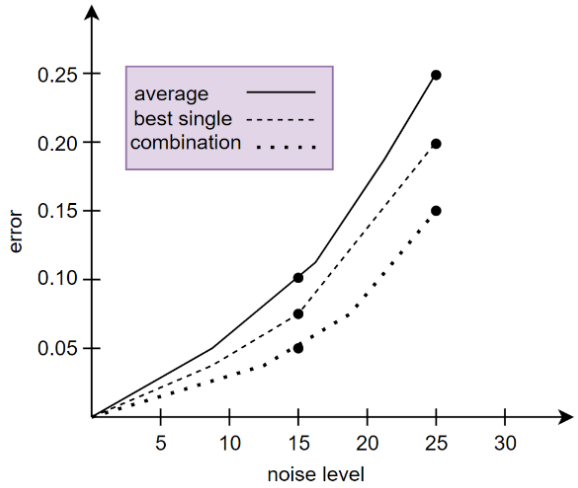


Fig. 3.2. Simplified visualization of the Hansen and Salamon [1990]'s observation.

Depending on the dataset and other variables, choosing the best model enables us to attain high accuracy. Let's now talk about the bagging method for a little while. There are two methods in which this approach can instruct weak learners. The first is called "homogeneous," where all base learners use the same model to generate an ensemble model. The other method is called "Heterogeneous," and it usually produces better test results by integrating multiple models—each of which stands for a weak learner—into one strong learner.[4]

It is simpler to set up and tune random forests and other homogeneous ensembles. Heterogeneous ensembles, on the other hand, (like stacking) combine the benefits of multiple techniques, leading to improved performance and resilience. The ability to forecast outcomes can be greatly enhanced by these methods.

IV. Diving into Ensemble Approaches: Weak Learners, Bagging, and Boosting

A. *Weak Learners*

A weak learner, also known as a weak classifier, is a model that performs marginally better than arbitrary guesswork. This indicates that, in a binary classification scenario, its accuracy is somewhat higher than 50%. Frequently, weak learners are low-variance, highly biased, simplistic models.[15]

The model's fit to the training set of data is its bias. Predictions with high bias may fall short of expectations. Thus, minimal bias helps a model make better predictions. The performance of the model on testing and validation data is the main focus of variance. Higher variance in these checks corresponds to more errors. Low variance hence contributes to improved accuracy.

Here are a few instances of poor learners: K-Nearest Neighbors (k-NN); when k is very small (like k=1), it can be considered a weak learner; Decision Stumps (a decision tree with just one split); Simple Linear Classifiers (linear

regression models used for binary classification); and Naive Bayes Classifiers (they assume independence between features and are often weak because of their simple assumptions). These typical cases of struggling students aid in problem solving but are more significant in group learning.

Many weak learners come together to produce strong learners in ensemble learning. The idea is straightforward: although while each weak learner might not perform well on its own, when combined with other weak learners' predictions (e.g., majority voting or average), they can create a very accurate & resilient model.

Why does an ensemble model need weak learners?

Reduction of Overfitting: Compared to utilizing a single complex model, ensemble approaches reduce the danger of overfitting by combining predictions.

Improved Accuracy: The aggregate forecasts of several weak learners typically outperform the predictions of any one model.

Stability: Ensemble approaches are often more stable and less susceptible to anomalies in the training data.

Let's now discuss how to turn weak learners into strong ones using a technique called bagging (Bootstrap Aggregating), in which each weak learner is taught using a random subset of input. Then, using boosting techniques like Random Forest and AdaBoost, weak learners train one after the other, averaging their predictions and focusing on the mistakes produced by the previous learner. Below is a fuller explanation of these ideas:

B. Bagging (Bootstrap Aggregating)

The idea behind bagging, or more correctly, Bootstrap Aggregation, is to combine the outputs of multiple models to provide a more complete answer than one from a single model. Because it requires interpreting several estimates, bagging is a helpful method for lowering estimate variance. Bootstrap aggregating is a homogenous parallel approach. By utilizing modified versions of a certain training dataset, the training strategy is used to create multiple base learners.

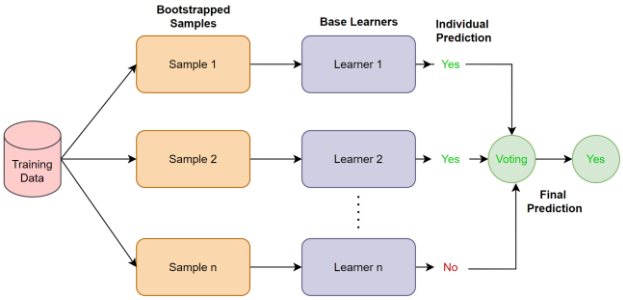


Fig.4.1 Bagging Architecture

Bootstrap Aggregating or Bagging Three primary steps are involved:

1) The bootstrapping procedure creates randomly sampled datasets (also known as subsample datasets) from the initial training data. If this procedure is carried out X times, it will have X iterations.

2) The identical learning process is used to train a new base learner each time these initial sets are repeated.[27]

3) The final prediction is ascertained by taking the average of all the guesses and calculating the total percentage of decisions.

$$f(x) = \frac{1}{N}\sum_{n=1}^{N} f_n(x)$$

- Bagging Algorithm [1]:

Input:
Data set $D = \{(x_1, y_1), (x_2, y_2), \ldots\ldots\ldots, (x_n, y_n)\}$;
Base learning algorithm;
Number of base learners N;
Process:
for $n = 1, \ldots\ldots, N$:
$h_n = \sum(D, D_{bs})$
$\%\ D_{bs}$ *is the bootstrap distribution*
end

Output:

$$H(x) = \arg\max_{y \in Y} \sum_{n=1}^{N} (h_n(x) = y)$$

Fig. 4.2. The Bagging Algorithm

C. Boosting

Boosting is a backwards-looking technique whereby successive models improve upon each other's errors or miscalculations. By teaching struggling students one after the other, each lesson builds on the one before it, reducing prejudice.

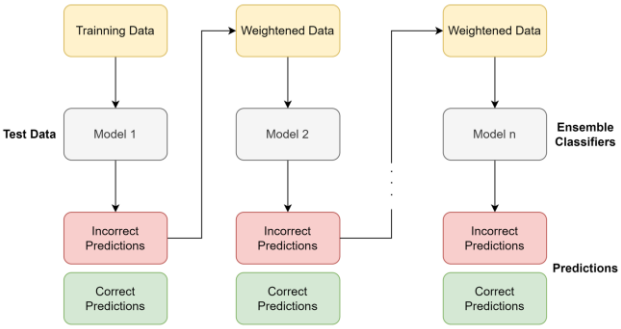


Fig. 4.3. Boosting Architecture

- Boosting Procedure [1]:

Input:
Distribution of sample D;
Data set $D = \{(x_1, y_1), (x_2, y_2), \ldots\ldots\ldots, (x_n, y_n)\}$;
Base learning algorithm;
Learning round numbers N;
Process:
$D_1 = D.$ % Initialize distribution
for $n = 1, \ldots\ldots, N$:
$h_n = \sum(D_n)$;
%Train a weak learner from distribution D_n
%Evaluate the error of h_n
$\epsilon_n = P_x \sim D_n\big(h_n(x) \neq f(x)\big)$;
$D_{n+1} = Adjust\ Distribution\ (D_n, \epsilon_n)$
end

Output:

$$H(x) = combine\ outputs \sum_{n=1}^{N} (h_1(x), \dots\dots\dots, h_n(x))$$

Fig. 4.4. Boosting Algorithm

Misclassification rate:

$$error = \frac{P_c - N}{N}$$

Where P_c denotes the Correctly predicted instances and N represents the total Number of training Instances.

V. APPLICATION OF ENSEMBLE LEARNING

Using the strengths of many models strengthens detection and prevention methods for more secure systems, making ensemble learning an excellent technique for addressing cybersecurity challenges. Upcoming developments will concentrate on improving adaptive learning and real-time analytics capabilities to fortify cybersecurity defences even further.

Because ensemble learning can increase predicted accuracy and robustness, it is useful in many domains. For example, in the banking industry, ensemble approaches are essential for tasks like stock market forecasting, fraud detection, and credit rating. These methods in medication development, patient outcome prediction, and medical diagnostics are also beneficial to healthcare. Internet-based trading? For client segmentation, churn prediction, and recommendation systems, it relies on ensemble models. In addition, sentiment analysis, natural language processing (NLP), and image identification all use ensemble learning. These techniques improve accuracy and generality by combining predictions from several models. As a result, they are indispensable when solving challenging real-world problems.

Organizations and researchers have been using ensemble approaches more often lately to boost cybersecurity initiatives and stimulate economic expansion. Important domains including virus detection, anomaly detection, intrusion detection systems (IDS), and protecting IoT settings from cyberattacks are exhibiting this tendency. Ensemble approaches are crucial tools for addressing the always changing cybersecurity threats and maintaining the resilience of digital infrastructures because of their adaptability and efficiency.

A report by Statista highlights cyberattacks' distribution among global industries in 2023.[17]

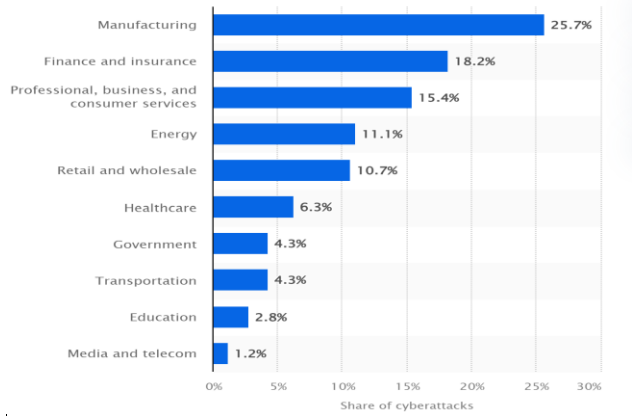


Fig. 5.1. Cyberattacks' distribution among global industries in 2023.

I'll go through great depth and provide actual test results in the parts that follow to demonstrate how bagging and boosting algorithms are used in cybersecurity settings. I hope to provide a comprehensive knowledge of how ensemble methods can effectively manage the intricacies present in cybersecurity operations through these real-world examples (as well as a few case studies). Proactive risk management techniques and well-informed decision-making are made possible by this.

VI. INTRUSION DETECTION SYSTEM (IDS) IN ENSEMBLE LEARNING

Developing a sophisticated intrusion detection system that supports real-time network security by employing ensemble learning techniques to efficiently detect and prevent intrusions and unauthorized access.

In this work, I used ensemble learning techniques to address intrusion detection in a comprehensive way. I started by using the Random Forest model, a robust bagging algorithm, to identify network intrusions. This model enhances accuracy and decreases overfitting by merging the predictions of multiple decision trees, providing it with a strong foundation for detecting illegal access and cyberthreats. To measure and improve the performance of the detection system, I use an advanced boosting method called AdaBoost. AdaBoost improves detection performance by iteratively changing the weights of erroneously categorized cases. On a given dataset, AdaBoost serves as a benchmark against the Random Forest model. The aim of this comparison is to ascertain which model performs better under certain conditions. The goal of my approach is to build a more accurate and robust intrusion detection system that can deliver the highest level of network security by combining bagging and boosting algorithms to neutralize attacks instantly.

Regarding the Dataset, the dataset used in this investigation was taken from an online database created especially to find network breaches and made available on Kaggle [18]. This dataset, which comprises 47 columns that depict various features and summaries extracted from network traffic data (Figure 6.1, 6.2) and backed by a substantial number of rows, offers a strong foundation for conducting in-depth examinations and meticulous evaluations of ensemble learning methods within the framework of intrusion detection.[18]

One essential component of the dataset is the 'Label,' which functions to classify each observation and provide the fundamental truth required for intrusion detection. This 'Label' variable arranges instances according to different attack techniques that are seen in computer networks, such as DDoS-RSTFINFlood, DoS-SYN_Flood, and so on. Every category denotes a different kind of breach or assault, representing the wide range of dangers that might arise in network settings. Thus, in order to enable effective intrusion detection systems, this target variable plays a crucial part in recognizing and classifying instances of potentially dangerous or unauthorized network activity.

	flow_duration	Header_Length	Protocol Type	Duration	Rate	Srate	Drate	fin_flag_number	syn_flag_number	
0	0.000000	54.00	6.00	64.00	0.329807	0.329807	0.0	1	0	
1	0.000000	57.04	6.33	64.00	4.290556	4.290556	0.0	0	0	
2	0.000000	0.00	1.00	64.00	33.396799	33.396799	0.0	0	0	
3	0.328175	76175.00	17.00	64.00	4642.133010	4642.133010	0.0	0	0	
4	0.117320	101.73	6.11	65.91	6.202211	6.202211	0.0	0	1	

Fig. 6.1. First 8 columns from used Sample Dataset

	flow_duration	Header_Length	Protocol Type	Duration	Rate	Srate	Drate	fin_flag_number
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	5.763553e+00	7.759537e+04	9.062979e+00	6.635523e+01	9.146945e+03	9.146945e+03	1.807621e-06	8.655842e-02
std	2.965660e+02	4.664043e+05	8.936737e+00	1.408680e+01	9.974794e+04	9.974794e+04	8.501385e-04	2.811870e-01
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	0.000000e+00	5.400000e+01	6.000000e+00	6.400000e+01	2.087155e+00	2.087155e+00	0.000000e+00	0.000000e+00
50%	0.000000e+00	5.400000e+01	6.000000e+00	6.400000e+01	1.581371e+01	1.581371e+01	0.000000e+00	0.000000e+00
75%	1.042881e-01	2.781200e+02	1.428000e+01	6.400000e+01	1.174380e+02	1.174380e+02	0.000000e+00	0.000000e+00
max	9.943576e+04	9.815555e+06	4.700000e+01	2.550000e+02	7.340032e+06	7.340032e+06	8.484654e-01	1.000000e+00

Fig. 6.2. Statistics Summary of few columns

Data preparation is necessary in order to extract crucial information from network traffic data, such as packet size, duration, and protocol type. Use Python to train an AdaBoost and Random Forest model that can discriminate between malicious and genuine traffic using decision stumps on the labeled dataset. Examine the model's performance using metrics such as accuracy, precision, recall, and F1-score on a different test set. This experiment aims to evaluate AdaBoost's detection capabilities for network breaches.

Random Forest improves the performance of its weaker learners by detecting network intrusions with a significant degree of accuracy because it concentrates on accurately identifying tough cases. The following displays the model's output and graphical representations:

Outcome for Bagging technique:

Accuracy: 0.9818372553226998

The graphical view of training and testing outcome:

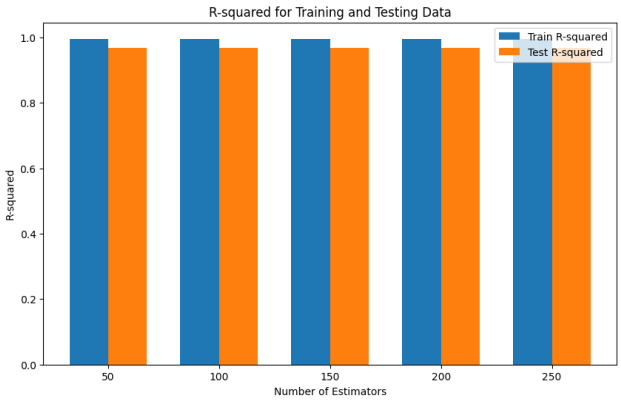


Fig. 6.3. Graph of MSE and R-Squared Data in terms of Number of Estimators.

- Classification report:

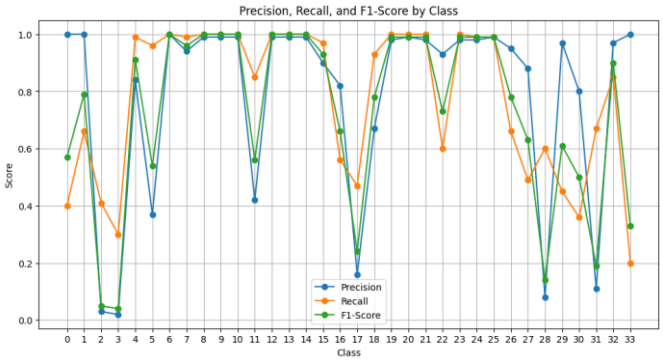


Fig. 6.4. Classification report for precision, recall and F1-score.

Outcome for Boosting method:

Accuracy: 0.7431895667930286

The graphical view of training and testing outcome:

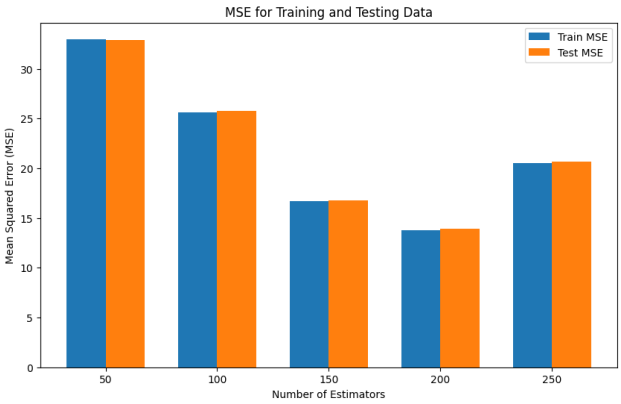


Figure 6.5: Graph of MSE and R-Squared Data in terms of Number of Estimators.

Classification report:

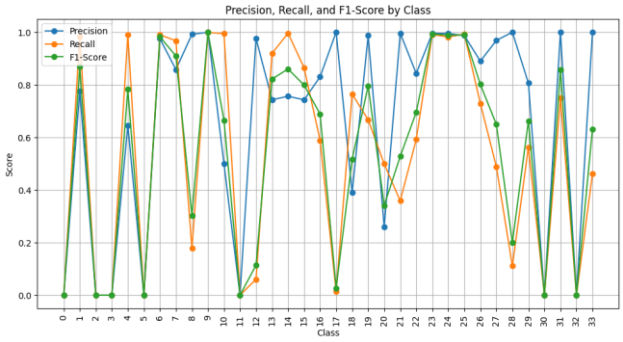


Fig. 6.6. Classification report for precision, recall and F1-score.

VII. Malware Detection in Ensemble Learning

Creating an innovative approach to identify malicious software, or malware, in a sizable dataset of data on system activity. In order to strengthen cybersecurity measures by swiftly and precisely identifying and removing potential dangers from malicious software, this project will make use of state-of-the-art machine learning techniques, in particular ensemble learning algorithms. This fortifies networks and guards against cyberattacks that compromise data integrity.

In this study, I use ensemble learning techniques to implement a thorough approach for identifying dangerous software in system behavior data. In order to find possible malware occurrences, I first used the Random Forest model, which is renowned for being robust in handling complex datasets. Random Forest reduces overfitting and increases accuracy by combining predictions from several decision trees, providing a strong base for malware detection and bolstering security against online attacks. Subsequently, I provide AdaBoost, an advanced boosting method, to enhance the detecting system's performance even further. AdaBoost improves detection skills and offers a comparison reference against the Random Forest model on the same dataset by iteratively adjusting the weights of misclassified cases. This comparison analysis seeks to identify the model that perfoms better under various circumstances. This method aims to create a malware detection system that is more accurate and resilient, capable of mitigating threats in real time and guaranteeing the highest level of system security, by utilizing the advantages of both bagging and boosting algorithms.

Regarding Dataset, consisting of 100,000 rows and 35 columns, the dataset records various attributes associated with system measurements and network activities [19]. Each row represents a single observation; the columns cover properties such as hash, millisecond, classification, policy, lock, and status. The 'Classification' column serves as an intrusion detection target label by categorizing observations into distinct classes. As shown in Figures 7.1, 7.2, preprocessing procedures are performed in order to handle missing data and standardize numerical attributes prior to training the model. So that intrusion classifications may be accurately predicted, the dataset is used to train models through ensemble learning approaches.

hash	millisecond	classification	state	usage_counter	prio
42fb5e2ec00	0	malware	0	0	3069378560
42fb5e2ec00	1	malware	0	0	3069378560
42fb5e2ec00	2	malware	0	0	3069378560
42fb5e2ec00	3	malware	0	0	3069378560
42fb5e2ec00	4	malware	0	0	3069378560
42fb5e2ec00	5	malware	0	0	3069378560
42fb5e2ec00	6	malware	0	0	3069378560
42fb5e2ec00	7	malware	0	0	3069378560
42fb5e2ec00	8	malware	0	0	3069378560
42fb5e2ec00	9	malware	0	0	3069378560
42fb5e2ec00	10	malware	0	0	3069378560

Fig 7.1. First 6 columns from used Sample Dataset

	hash	millisecond	classification	state	usage_counter	prio	static_prio	normal_prio
count	1.000000e+05	100000.000000	100000.000000	1.000000e+05	100000.0	1.000000e+05	100000.000000	100000.0
mean	4.775183e+07	499.500000	0.500000	1.577683e+05	0.0	3.069706e+09	18183.900070	0.0
std	2.742118e+07	288.676434	0.500003	9.361726e+05	0.0	2.963061e+05	4609.792765	0.0
min	7.358730e+05	0.000000	0.000000	0.000000e+00	0.0	3.069190e+09	13988.000000	0.0
25%	2.505040e+07	249.750000	0.000000	0.000000e+00	0.0	3.069446e+09	14352.000000	0.0
50%	5.130835e+07	499.500000	0.500000	0.000000e+00	0.0	3.069698e+09	16159.000000	0.0
75%	6.562869e+07	749.250000	1.000000	4.096000e+03	0.0	3.069957e+09	22182.000000	0.0
max	9.965388e+07	999.000000	1.000000	4.326605e+07	0.0	3.070222e+09	31855.000000	0.0

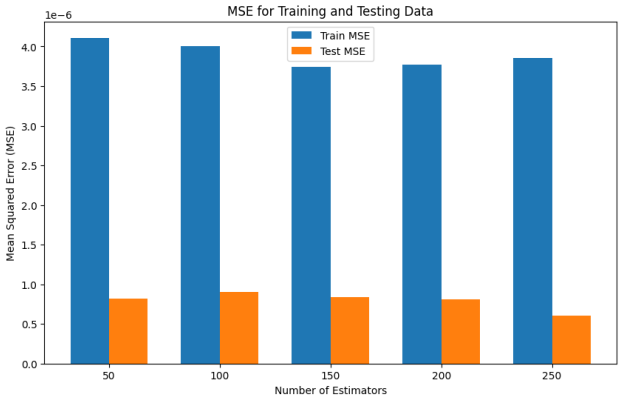
Fig. 7.2. Statistics Summary of few columns

To assess the model's effectiveness, evaluations of accuracy, precision, recall, F1-score, mean squared error (MSE), and R-squared are required for both training and testing datasets. The analysis reveals that both models are capable of good generalization. Furthermore, Random Forests often perform well in malware detection because they are resistant to overfitting and adept at managing big feature spaces. Conversely, AdaBoost is notable for its remarkable efficacy at detecting malware. The graphical representations of the outcomes for both models are displayed below:

Outcome for Bagging method:

Accuracy: 0.99945

The graphical view of training and testing outcome:



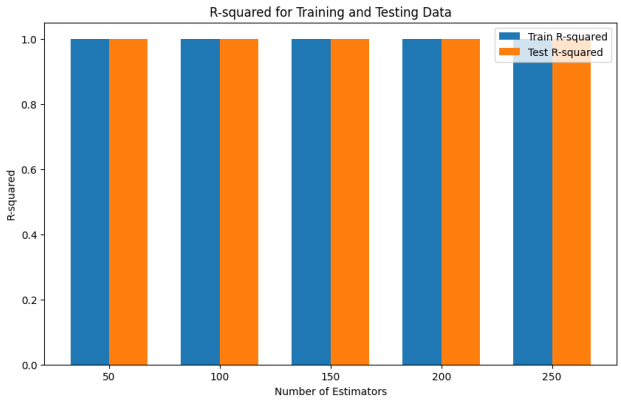


Fig. 7.3. Graph of MSE and R-Squared Data in terms of Number of Estimators.

Classification report:

Fig. 7.4. Classification report for precision, recall and F1-score.

Boosting method outcome:

Accuracy: 1.0

The graphical view of training and testing outcome:

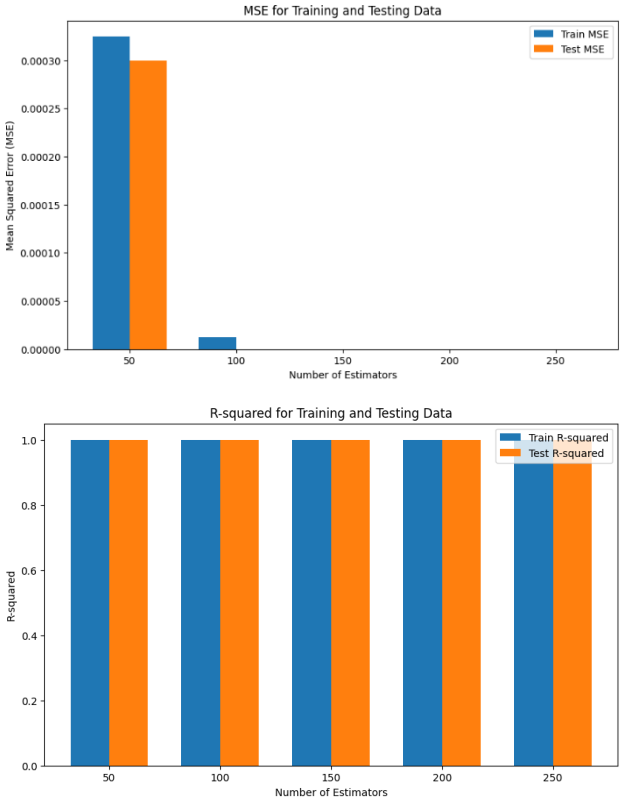


Fig. 7.5. Graph of MSE and R-Squared Data in terms of Number of Estimators.

Classification report:

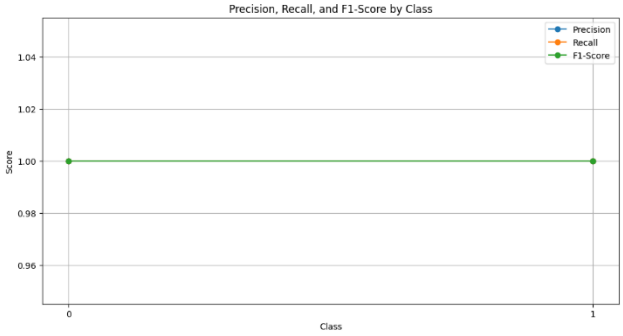


Fig. 7.6. Classification report for precision, recall and F1-score.

VIII. Anomaly Detection in Ensemble Learning

In order to protect financial and organizational assets against attacks and unauthorized access, a system that can accurately and quickly detect anomalies in real-time network traffic must be developed.

I use an all-encompassing strategy that integrates cutting-edge methods to accomplish real-time anomaly identification in network traffic. Initially, I employ sophisticated ensemble learning models, with the Random Forest model serving as the leader due of its reputation for adeptly managing intricate datasets. Random Forest improves the accuracy of the system and lessens overfitting by mixing predictions from several decision trees, giving the system a solid defense against cyberattacks. I then introduce AdaBoost, a sophisticated boosting algorithm that has been fine-tuned to improve the efficiency of the system. By iteratively improving misclassifications, AdaBoost enhances detection abilities and offers a framework for comparison with the Random Forest model on the same dataset. The goal of this meticulous side-by-side comparison is to determine which model performs better under certain conditions. By combining bagging and boosting algorithms, this approach aims to offer an extremely efficient and refined anomaly detection system that is prepared for both real-time threat mitigation and network integrity protection.

Regarding Dataset, the data utilized in this study came from an online database designed specifically to spot anomalies in network traffic. Each of the 5000 rows and 35 columns in this sizable dataset for research [20] has a range of attributes extracted from network traffic data. Giving distinct class labels to observations, the target column 'class' is crucial in assisting in the identification of irregularities such as brute force assaults, slow post attacks, slow loris attacks, HTTP floods, ICMP floods, and more. Different sorts of dangers that are present in network settings can be identified and classified thanks to anomaly detection, which is based on this classification. These attributes, when combined with target labels, give the dataset a solid foundation for a thorough evaluation of anomaly detection methods, ensuring the development of precise and effective detection systems for real-time threat prevention.

	ifInOctets11	ifOutOctets11	ifoutDiscards11	ifInUcastPkts11	ifInNUcastPkts11	ifInDiscards11	ifOutUcastPkts11
0	1867925250	902237363	0	52007310	16978	0	7197292
1	1994338334	903845459	0	52098054	16986	0	7227073
2	2116573334	905396546	0	52185853	16994	0	7255792
3	2257767832	907308930	0	52287097	17015	0	7291152
4	2342047724	908534112	0	52347521	17043	0	7313830

Fig. 7.1. First 6 columns from used Sample Dataset

	ifInOctets11	ifOutOctets11	ifoutDiscards11	ifInUcastPkts11	ifInNUcastPkts11	ifInDiscards11	ifOutUcastPkts11
count	4.998000e+03	4.998000e+03	4998.000000	4.998000e+03	4998.000000	4998.000000	4.998000e+03
mean	2.161260e+09	1.281552e+09	36377.437975	9.150696e+07	20425.803922	36377.561825	2.680787e+07
std	1.233852e+09	1.153395e+09	74228.977904	5.800683e+07	7936.895813	74228.989718	2.328438e+07
min	1.426588e+06	1.618430e+05	0.000000	7.013690e+05	2735.000000	0.000000	2.230760e+05
25%	1.094045e+09	3.724623e+08	0.000000	5.020809e+07	15081.000000	0.000000	8.049108e+06
50%	2.180395e+09	8.935678e+08	0.000000	7.906267e+07	20987.000000	0.000000	1.745086e+07
75%	3.221204e+09	2.448435e+09	4216.000000	1.313896e+08	26931.000000	4216.000000	4.038847e+07
max	4.294416e+09	4.294061e+09	196630.000000	2.439828e+08	35238.000000	196630.000000	9.369831e+07

Fig. 7.2. Statistics Summary of few columns

It is necessary to extract important features like protocol type, packet size, and duration from the network traffic data in order to perform data preprocessing on it. Python-based AdaBoost and Random Forest models are trained on the labeled dataset to distinguish between attack and normal

traffic using decision stumps. A different test set is used to evaluate the model's performance indicators, which include accuracy, precision, recall, and F1-score. This experiment aims to assess AdaBoost's network breach detection performance.

It is observed that the bagging-based Random Forest model, which may generalize well to a variety of situations, has good MSE and R-squared values and outstanding accuracy. Below are the outcomes of both models along with their corresponding graphical representations:

Outcome for bagging method:

Accuracy: 0.9826666666666667

The graphical view of training and testing outcome:

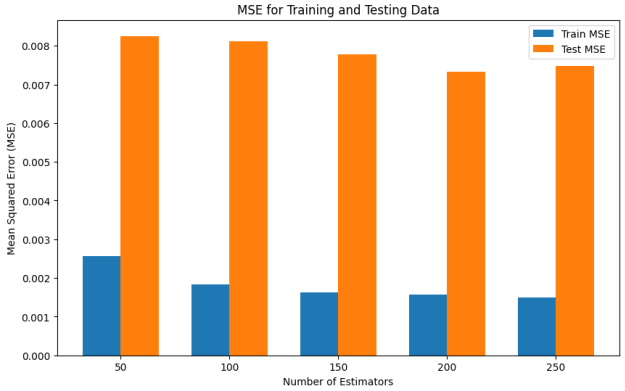


Fig. 8.3. Graph of MSE and R-Squared Data in terms of Number of Estimators.

Classification report:

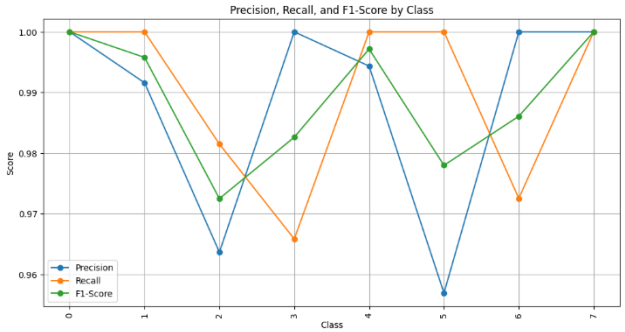


Fig. 8.4. Classification report for precision, recall and F1-score.

Outcome for Boosting method:

Accuracy: 0.626

The graphical view of training and testing outcome:

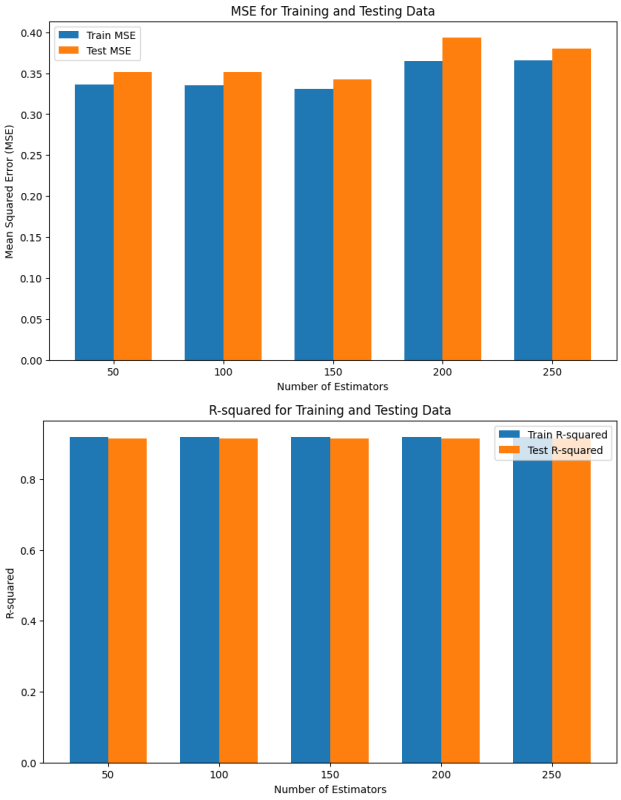


Fig. 8.5. Graph of MSE and R-Squared Data in terms of Number of Estimators.

Classification report:

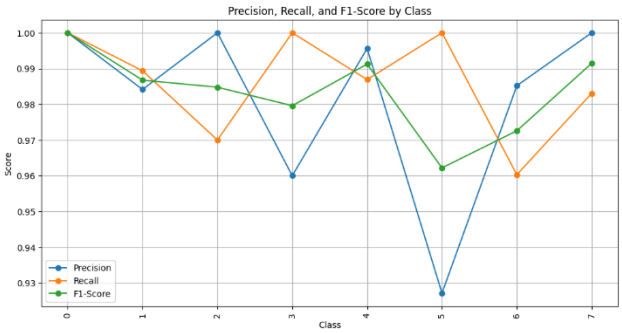


Fig. 8.6. Classification report for precision, recall and F1-score.

IX. Result and Discussion

The following table [Figure 9.1] summarizes the author's three examples. It compares the accuracy of the final prediction outcome for the bagging and boosting techniques in ensemble learning using the random forest model for bagging and the AdaBoost model for boosting.

TABLE I. ACCURACY COMPARISON

Accuracy Comparison		
Applications	*Bagging (Random Forest)*	*Boosting (AdaBoost)*
IDS	98%	74.32%
Malware Detection	99%	100%
Network Anomaly Detection	98.26%	62.6%

Fig. 9.1. Accuracy Comparison between Bagging method and Boosting method for different datasets.

According to what I discovered; a number of important variables affect how accurate machine learning models are. To begin with, it is imperative to have a high-quality dataset. Make that the preprocessed data is accurate, devoid of null values, and clean. Selecting a model architecture that makes sense given the nature of the problem and other relevant variables is another crucial step. Third, fine-tuning hyperparameters is essential. This includes parameters such as random_state, learning_rate (whose default value is 1), which affects the weights of weak learners, and n_estimators (the number of weak learners to train iteratively), which have a substantial impact on model performance. To combine several models for increased accuracy, lastly take into account ensemble approaches.

The precision, recall, and F1-score classification report that I computed from the TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) was displayed in Figures [4, 5, and 6]. When I use these phrases, what I mean by "TP" is the moment at which the model accurately predicts the positive class. For example, a true positive happens when a patient with cancer is correctly identified by a cancer detection algorithm. When the model predicts the negative class accurately, TN happens. It's a true negative, for example, if a spam filter accurately identifies an email as not spam. In terms of FP, the model here predicts the positive class incorrectly. For example, a false positive occurs when the spam filter incorrectly classifies a valid email as spam; this is known as a false negative (FN) and occurs when the model mis predict the negative class. It is a false negative, for instance, if the cancer detection model is unable to identify a patient who has the disease. locates the following mathematical representation:

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

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

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

$$F1score = \frac{2TP}{2TP + FP + FN}$$

Understanding these linkages facilitates the construction of models that perform well on training data as well as efficiently generalize to unknown data in terms of model generalization.

A **well-built model** should aim for reasonably good performance and maintain comparable R-squared values and Mean Squared Error (MSE) values with low residuals.

when the **MSE Train data < MSE Test data** and **R-squared Train data > R-squared Test data** it suggests that the model is overfitting.

If **both the training and test MSE** are high, and the **R-squared** values are low, it suggests underfitting. A model that is underfitted performs poorly because it is overly basic to identify underlying patterns in the data.

As per shown the graph of Figures [6.3, 6.5, 7.3, 7.5, 8.3, 8.5]

For IDS application: The model appears to be underfitting while the bagging methodology works well for this test, as demonstrated by the high MSE values and reasonably low R-squared values in the boosting method. The test and R-squared values are near and high, but the train value's MSE is lower than the test MSE data, suggesting some overfitting. This is further demonstrated in Figure [6.3], where multiple classes have low f1-scored values, indicating a model that needs to be improved. Even if the accuracy looks sufficient right now, there's always space to improve and deal with overfitting problems. Still, the bagging ensemble model does well in identifying the more precise intrusion in our dataset.

For malware detection application: The bagging and boosting models both generalize well with no overfitting or underfitting because the MSE values of the train and test data are extremely close to 0. In addition, the R-squared values of the train and test data are high and closed with each other. Even though the accuracy of the boosting model is superior.

For Anomaly detection application: The model has overfitting in the boosting algorithm, as indicated by the MSE values of the test and train data, which are very close, and the MSE Train < MSE Test, which is also a bit high. The R-squared train is slightly higher than the test values, indicating that the bagging method's MSE and s-squared values are good to perform, with room to improve efficiency and minimize the overfitting issues for well-generalization. As a result, the bagging model is also working effectively to identify the precise anomalies.

X. ACKNOWLEDGMENT

I would like to thank Prof. Dr. Andreas Pech for giving me the opportunity to work on this topic which enriches my understanding and knowledge on this topic and for providing me with the proper guidance to write this paper.

REFERENCES

[1] Zhi-Hua Zhou, Ensemble Methods: Foundations and Algorithms, CRC Press, 2012.

[2] Polikar, R., 2012. Ensemble learning. Ensemble machine learning: Methods and applications, pp.1-34.

[3] Pandit, P.V., Bhushan, S. and Waje, P.V., 2023, May. Implementation of Intrusion Detection System Using Various Machine Learning Approaches with Ensemble learning. In 2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT) (pp. 468-472). IEEE.

[4] Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 11, 169-198.

[5] I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," in IEEE Access, vol. 10, pp. 99129-99149, 2022, doi: 10.1109/ACCESS.2022.3207287.

[6] Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning, 51(2), 181-207.

[7] Peter Sollich and Anders Krogh, "Learning with ensembles: How overfitting can be useful," Advances in Neural Information Processing Systems, Vol. 8, 1995, https://papers.nips.cc/paper_files/paper/1995/hash/1019c8091693ef5c5f55970346633f92-Abstract.html (link resides outside of ibm.com).

[8] Gautam Kunapuli, Ensemble Methods for Machine Learning, Manning Publications, 2023.

[9] Buczak, A. L., & Guven, E. (2016). "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection." IEEE Communications Surveys & Tutorials, 18(2), 1153-1176

[10] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 42, No. 4, 2012, pp. 463-484, https://ieeexplore.ieee.org/document/5978225 (link resides outside of ibm.com).

[11] Saxe, J., & Berlin, K. (2015). "Deep neural network based malware detection using two-dimensional binary program features." In 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), 11-20.

[12] Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning, 51(2), 181-207.

[13] Dietterich, T. G. (2000). Ensemble methods in machine learning. In International workshop on multiple classifier systems (pp. 1-15). Springer, Berlin, Heidelberg.

[14] Meidan,Yair, Bohadana,Michael, Mathov,Yael, Mirsky,Yisroel, Breitenbacher,Dominik, ,Asaf, and Shabtai,Asaf. (2018). detection_of_IoT_botnet_attacks_N_BaIoT. UCI Machine Learning Repository. https://doi.org/10.24432/C5RC8J..

[15] Rokach, L. (2010). Ensemble-based classifiers. Artificial Intelligence Review, 33(1), 1-39.

[16] Muhammad Usman, S., Khalid, S. and Bashir, S. (2021) "A deep learning based ensemble learning method for epileptic seizure prediction," Computers in biology and medicine, 136(104710), p. 104710. doi: 10.1016/j.compbiomed.2021.104710.

[17] Global cyberattacks in industries 2023 (no date) Statista. Available at: https://www.statista.com/statistics/1315805/cyber-attacks-top-industries-worldwide/ (Accessed: June 3, 2024).

[18] Chakraborty, S. (2023) "IoT Intrusion Detection." Kaggle.

[19] Saravana, N. (2018) "Malware Detection."

[20] Kasassbeh, M. (2020) "Network anomaly detection dataset."

[21] Ahmad, S., Javed, A. and Khan, M. J. (no date) Applications of Machine Learning in Cyber Security.

[22] Biggio, B., Fumera, G. and Roli, F. (2017) "Security evaluation of pattern classifiers under attack," arXiv [cs.LG]. Available at: http://arxiv.org/abs/1709.00609.

[23] Moustafa, N. and Slay, J. (2015) "The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems," in 2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS). IEEE.

[24] Sarasamma, S. T., Zhu, Q. A. and Huff, J. (2005) "Hierarchical Kohonenen net for anomaly detection in network security," IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society, 35(2), pp. 302–312. doi: 10.1109/tsmcb.2005.843274.

[25] Chatterjee S, Byun YC. Voting Ensemble Approach for Enhancing Alzheimer's Disease Classification. Sensors (Basel). 2022 Oct 9;22(19):7661. doi: 10.3390/s22197661. PMID: 36236757; PMCID: PMC9571155.

[26] Singh, A. (2018) A comprehensive guide to ensemble learning (with python codes), Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/ (Accessed: June 6, 2024).

[27] What is ensemble learning? (2024) Ibm.com. Available at: https://www.ibm.com/topics/ensemble-learning (Accessed: June 6, 2024).

[28] Cyberpro Magazine (2024) Mastering ensemble learning: A comprehensive guide to boosting model performance, Medium. Available at: https://medium.com/@cyberprosocial/mastering-ensemble-learning-a-comprehensive-guide-to-boosting-model-performance-434bdacc6f2a (Accessed: June 6, 2024).