Using a Hybrid ML Model on Network Intrusion Detection System

Alan James
Dept of Computer Science Engineering
Lovely Professional University
Phagwara, Punjab
alaanjames@gmail.com

Ashok Kumar
Dept of Computer Science Engineering
Lovely Professional University
Phagwara, Punjab
ashoksamota0@gmail.com

Tuttaganti Sri Harsha
Dept of Computer Science Engineering
Lovely Professional University
Phagwara, Punjab
harshatuttaganti.123@gmail.com

Bhavanam Narendra Reddy
Dept of Computer Science Engineering
Lovely Professional University
Phagwara, Punjab
bhavanamnarendra4@gmail.com

Avula Bharatasimha Reddy
Dept of Computer Science Engineering
Lovely Professional University
Phagwara, Punjab
bharatasimhareddy229@gmail.com

Nagulapalli Gunadeep Kushal
Dept of Computer Science Engineering
Lovely Professional University
Phagwara, Punjab
gunadeepkushal18@gmail.com

***Abstract*—. An Intrusion Detection System is designed to keep malicious users and unauthorized people from accessing information useful for a particular organization or a group. IDS is very important for the secure transfer of files and data across channels. Cyberworld is now prone to hackers, and attacks. Therefore, the most advanced and relevant technology Machine Learning is used here in this paper for its study and improvements. In this paper, we try to propose some improvements in the existing system and how we can tackle it using ML algorithms that cater to solving it. To improve it, we will optimize the features to be selected for training the model and building it.**

Keywords—intrusion detection, machine learning, algorithms

I. Introduction

With the growing number of cybersecurity threats and attacks, intrusion detection systems (IDSs) have become a critical component of network security. An IDS is a security mechanism that monitors network traffic and alerts the system administrators when it detects any suspicious or malicious activity. The performance of an IDS depends on its ability to accurately detect and classify intrusions while minimizing the rate of false alarms. Attempts to intrude on computer networks have gained greater prominence due to the increasingly significant role that internet systems play in our daily lives.[1] Hence, attackers may have leveraged the vulnerabilities in these technologies to exploit the data of individuals and organizations. To safeguard internet systems, various protective measures have been implemented, such as installing firewalls, anti-virus software, and intrusion detection systems (IDSs), along with the enforcement of proper security policies. Among these measures, IDSs have gained significant attention from the public for safeguarding their internet systems. IDSs are utilized to detect, categorize, and potentially respond to legitimate activities.

A perfect IDS system would have the capability to detect every attack with zero false positives, although this is a challenging task. The primary objective of an IDS is to identify unlawful behavior within the host or network. Essentially, an IDS is designed to detect unauthorized activity and can monitor a wide range of network activity. Whenever an attack is detected, the IDS will alert the appropriate managers with a warning message.

An ideal IDS system would be able to detect all attacks with 100% accuracy, but achieving this is difficult. There are two primary approaches used by HIDS and NIDS for intrusion detection: misuse detection and anomaly detection. Misuse detection is employed by IDSs that match existing signatures against network traffic to detect attacks. When a match is found, the IDS takes action as the traffic is deemed harmful. Typically, IDS actions involve alerting the network administrator and logging the intrusion event. However, IDSs that rely solely on misuse detection is unable to detect new or previously unknown attacks. Network administrators must update the stored signatures to remain protected against such attacks.

Anomaly detection employs soft computing techniques such as neural networks [10], and data mining [8] to identify attacks. Although anomaly detection can detect novel attacks, it often produces more false positives. The system's performance may degrade due to computational overload and an increase in the number of transmission packages.

One major problem with the real intrusion detection technique is that it still requires human engagement to distinguish between normal traffic and intrusions. The new challenges posed by "Big Data" and "Cloud Computing" are also very concerning. The intrusion detection engine must actively gather and evaluate a vast amount of data produced by these two pervasive technologies. Frequently, the IDS must cope with multi-dimensional data produced by these large quantities of data. It is important to keep in mind that the intrusion dataset can be extremely large. Not only has there been an increase in observations, but the number of observed attributes has also expanded greatly. As a result, there may have been a considerable number of false positive results [3].

The classification problem in intrusion detection is how to determine whether network traffic behavior is normal or belongs to one of four assault groups, such as probing, denial of service, user to root, and root to local. The main goal is to develop precise classifiers that can accurately distinguish between intrusive and typical activities. Despite several proposals for intrusion detection systems based on different soft computing paradigms, it is still believed that the use of these methods is still somewhat limited.

This research paper suggests utilizing the NSL-KDD dataset as an alternative to the commonly used KDD Cup 1999 benchmark intrusion detection dataset [4] to construct a network intrusion detection system. We implement ten machine learning methodologies and combine them with various evaluation techniques to create a network intrusion detection system to undertake a full examination of the proposed research.

Machine learning is a method that makes use of test data or previous experiences to enhance performance standards using computer software. Models are defined by parameters, and learning entails running a computer program while utilizing training data to adjust the model's parameters. The model might be descriptive, which makes it easier to learn from data, or predictive, which makes it possible to make future predictions. Classification and clustering are the two primary methods that machine learning often uses to carry out predictive or descriptive tasks. While in clustering, the classes are not predetermined during the learning process, the algorithm predicts the most likely category or label for new observations into one or more predefined classes.

The KDD dataset has been advocated for use by numerous academics in the past to identify assaults [6][7]. These suggestions, meanwhile, have not produced a good detection rate. Moreover, the current IDS evaluate every feature, which can lead to incorrect classification of intrusions and require a long time while creating the model. Despite issues with and criticisms of the KDD dataset's system evaluation, academics continue to use it to assess their models.

Researchers have used conventional machine learning approaches like Naive Bayes, Decision Trees, and Support Vector Machines to solve the shortcomings of several intrusion detection algorithms. Although these methods have increased detection accuracy, handling the significant volumes of data involved still calls for professional expertise and engagement. Since they rely on techniques that let machines train without human intervention, these shallow classifiers might not perform as well for multiclass issues with a lot of features [5].

The base research paper we're studying used around at least 8 ML algorithms to conclude. In it, the researchers have tried to predict the accuracy using various performance metrics and have tried to eliminate the existing problems related to the NSL-KDD dataset's limitations.

II. Literature Review

Due to their ability to generalize, machine learning approaches are widely employed by researchers in the field of network intrusion detection to grasp technical information about intrusions that don't have any predetermined patterns.

Researchers have employed several Artificial Intelligence (AI) techniques and data mining methods to enhance Intrusion Detection Systems (IDS). Among the commonly used AI methods in network security research, the Bayesian approach has been extensively studied [9].

Previous research has utilized association rules to detect network intrusions [11]. However, the downside of association rule mining is that it tends to generate an excessive number of rules, which can increase system complexity, after applying multiple performance indicators to the entire rule set. The disadvantage of association rule mining is that it has a propensity to generate a large number of rules after applying numerous performance indicators to the entire rule set obtained, hence increasing the system's complexity. Now, the researchers' attention has switched to developing an effective classifier by applying machine learning approaches and learning from instances.

A cooperative agent architecture for an IDS was suggested in an intriguing study by [14]. The technology enables a continuous scale for intrusion detection by allowing the agents to update soft evidence and share beliefs about the likelihood of an event occurring. The suggested system uses three different types of agents: registry agent, intrusion monitoring agent, and system monitoring agent.
When necessary, the system monitoring agent interacts with the operating system and processes log data. Such agents disseminate their knowledge and opinions based on observations of one another. On the other hand, an intrusion monitoring agent updates beliefs based on BNs utilizing both observed values (hard evidence) and derived values (beliefs or soft evidence) from other agents. The system can recognize several known assaults using both hard and soft discoveries.

Earlier research [12] extensively utilized neural networks for detecting abnormality and misuse patterns. However, a potential issue with neural networks is that their training time increases when used with larger datasets such as KDD Cup 1999. Interpreting the results from neural networks can be difficult due to their black-box nature, which can make it challenging to understand the decision-making process. Despite these limitations, neural networks have shown promising results in detecting network intrusions, and researchers continue to explore ways to optimize their performance with large datasets. Some techniques include using parallel computing, reducing the input dimensionality, and selecting appropriate network architecture. Overall, neural networks remain a popular approach for network intrusion detection due to their high accuracy and ability to detect previously unseen attacks.

Bayesian Networks (BN) are commonly used to address uncertainty in decision-making and have proven effective in various research domains. The explicit inter-relationships between attributes in a dataset provided by the BN structure enable a better understanding of the studied area. Additionally, BN models offer techniques for handling missing data and avoiding overfitting. As a result of its causal and probabilistic semantics, a BN model can integrate both data and domain knowledge [13]. Humans can modify the BN to enhance the performance of the predictive model. Furthermore, probability and network learning techniques can be utilized to refine the expert-elicited network for improved prediction accuracy.

In the field of Intrusion Detection Systems (IDS), Naive Bayes [13] and decision trees [17] are among the popularly applied machine learning techniques. Decision trees are favored due to their simplicity, quick adaptation, and high categorization accuracy. On the other hand, Naive Bayes assumes that data attributes are conditionally independent rather than correlated, which may affect its performance. Nguyen and Choi [16] evaluated the top ten classifier methods for network intrusion detection and proposed a classification model.

In [15], the focus was on one-dimensional time-series modeling of network traffic events, where the aim was to predict future network traffic patterns based on historical data. To achieve this, the authors employed a combination of CNN and RNN techniques, specifically LSTM and GRU. CNNs are well-suited for extracting meaningful features from time-series data, while RNNs are ideal for modeling sequential data. The use of LSTM and GRU in this study is particularly relevant since these RNN variants are designed to address the vanishing gradient problem, which can occur in traditional RNNs when processing long sequences of data. The study's findings suggest that the combined use of CNN and RNN techniques can effectively model and predict network traffic patterns.

Furthermore, the lack of proper classification in the IDS may lead to difficulties in identifying the nature and severity of network intrusions. As a result, it becomes imperative to accurately classify network requests into their respective categories. The SVM approach proposed by Mill and Inoue [18] can be enhanced by incorporating the 5-class classification, which would provide additional insights into the accuracy of each class. This information would enable system administrators to make informed decisions and take necessary actions to prevent similar intrusions in the future. Another advantage of SVM is its ability to handle high-dimensional data, making it suitable for intrusion detection in large-scale networks. However, SVM has its limitations as it can be computationally expensive when processing large datasets, which may lead to slower response times in detecting intrusions.

Juan Wang et al. suggested a decision tree-based method for intrusion detection in their work [20]. Even though the C4.5 approach performed well in their testing, the error rate remained constant. A decision tree-based learning technique was utilized by Farid et al. [19] in 2010 to retrieve crucial features from the training dataset for intrusion detection. Their methods used the ID3 and C4.5 decision tree algorithms to find pertinent features. Each feature was given a weight value. The minimal decision tree depth at which each part is examined inside the tree is used to determine the weight, and the weights of features that do not appear in the decision tree are given a value of zero.

Abraham et al. explored ensemble and hybrid approaches utilizing Decision Trees (DT) and Support Vector Machines (SVM) for intrusion detection [23], and they concluded that the hybrid approach outperforms a direct SVM technique for all classes. The authors designed three different hybrid models by combining SVM and DT. The first model used a decision tree for feature selection and an SVM for classification, the second model employed an SVM for feature selection and a decision tree for classification, and the third model combined both approaches by using a decision tree and SVM in parallel for classification. Moreover, the third model, which utilized both SVM and DT in parallel, produced the highest classification accuracy. The study highlights the advantages of combining different machine-learning techniques to improve the accuracy of intrusion detection systems.

A statistical approach was employed by Geetha Ramani et al. [21] in their study from 2011 to analyze the KDD 99 dataset. By examining the inherent dependencies between features, they were able to identify the crucial features. Through their analysis, the researchers were able to identify the crucial features that contribute most to the classification of network traffic events. The study showed that the most important features for detecting network intrusions were related to the protocol used, the duration of the connection, and the number of bytes transferred in the connection. By identifying the most important and relevant features, Geetha Ramani et al.'s study provides valuable insights for developing more efficient and effective intrusion detection systems.

In their research, Panda and Patra [22] developed a hybrid clustering approach to enhance the identification of rare attack categories. The researchers integrated the COBWEB algorithm, which enables the formation of a concept hierarchy for attack type identification based on attributes, with the FFT clustering algorithm, which identifies rare attack types by selecting data points the farthest from existing cluster centers. The study aimed to address the limitation of conventional clustering approaches, which typically struggle to recognize rare attack categories due to their underrepresentation in the dataset. The hybrid clustering approach proposed in this study demonstrated a significant improvement in detecting rare attack categories. As per the study's findings, the hybrid clustering approach has the potential to be a valuable technique in enhancing the detection rate of rare attack categories

In their study, Panda and Abraham [25] developed a network intrusion detection system that utilized the NSL-KDD dataset and the Discriminative Multinomial Nave Bayes (DMNB) algorithm. The authors implemented a feature selection process to reduce the dataset's dimensionality and improve classification accuracy. To evaluate the effectiveness of DMNB, they compared its performance with other machine learning algorithms such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Decision Trees (DT). The results indicated that DMNB outperformed the other algorithms regarding accuracy, precision, recall, and F-measure. The authors concluded that their proposed approach using DMNB could serve as an effective solution for network intrusion detection in real-world scenarios.
In this paper, we are trying to go back to the basics of ML algorithms, using all the existing algorithms and the hybrid ones too for better accuracy and detection.

III. Proposed Methodologies and Evaluation

We are proposing the use of 10 machine learning algorithms, some of which are standard ones used over the years. We are also trying to implement hybrid ML models to better conclude. Hybrid models use different available algorithms and use it train one after the another arriving at a better accuracy measure. Below are the algorithms and their brief working to better understand our results.

Random Forests [24] is an ensemble learning approach that enhances the robustness and accuracy of the model by combining the predictions of various decision trees. Using a random subset of the training data and a random subset of the features for each tree, the technique builds a huge number of decision trees. Each decision tree is constructed during training using a random sample of features, and the splitting points are chosen based on the optimal split for that specific collection of data and features. Once all the trees are constructed, they each make predictions based on fresh data, and the combined forecasts of all the individual trees result in the final prediction. The study [26] offers a novel method for quickly identifying hazards in network intrusion detection systems that incorporate the RFC-RST technique. Particularly, attributes are chosen using the random forest classifier, and rules are created using the rough set theory. Rough set exploration system, R programming, and Python are all used in the model's implementation. The study's main objective is to establish the bare minimum of rules necessary to accurately reflect the knowledge contained in the dataset.

A straightforward and popular machine learning approach for classification and regression applications is K-Nearest Neighbors (KNN). It operates by figuring out how far the new data point is from every other data point in the dataset. KNN then determines the "K" nearest data points, where "K" is an integer that represents the total number of nearest neighbors that will be taken into account. The majority class among the new data point's K-nearest neighbors then determines its classification. The authors [27] used the NSL-KDD dataset for training and testing their proposed KNN-based intrusion detection system. They pre-processed the dataset by removing redundant features and balancing the classes. The authors experimented with different values of K in the range of 1 to 15 and found that the optimal value of K for the NSL-KDD dataset is 5. Overall, the research paper demonstrates the effectiveness of KNN in intrusion detection applications using the NSL-KDD dataset and shows that KNN can achieve high accuracy and low false positive rate for such applications.

The machine learning algorithm Support Vector Machines (SVM) is used for regression analysis and classification. Finding a hyperplane that maximum separates the various classes of data points in the dataset is how the algorithm operates. A hyperplane is a line that divides two groups of data points into two dimensions. The closest data points to the hyperplane are known as support vectors, and SVM operates by locating these vectors. The hyperplane's position and orientation are then optimized by the algorithm to maximize the margin or the separation between the hyperplane and the support vectors for each class. The authors [28] pre-processed the NSL-KDD dataset by removing duplicate and irrelevant features and normalizing the data. They also balanced the dataset by oversampling the minority class using the Synthetic Minority Over-sampling Technique (SMOTE). The authors experimented with different kernel functions of SVM and found that the Radial Basis Function (RBF) kernel performs better than the Linear kernel for the NSL-KDD dataset.

Naive Gaussian based on Bayes' theorem; Bayes is a probabilistic machine learning method. It is a supervised learning technique applied to classification tasks using continuous input characteristics. Each feature in the training dataset has its probability distribution first estimated using the algorithm. The term "naive" refers to the assumption that the traits are independent of one another. Each feature's estimated probability distribution typically follows the Gaussian (Normal) distribution. The program then determines the conditional probability of each class given the input features after estimating the probability distribution of each feature. This is accomplished by applying the Bayes theorem, which asserts that the conditional probability of each feature given the class and the prior probability of the class combine to produce the likelihood of a class given the input features. The algorithm determines the conditional probability of each class given the input features during prediction and chooses the class with the highest probability as the predicted class. The authors [29] conclude that the proposed system can be used in real-world scenarios for intrusion detection, and the results can be improved further by incorporating other machine learning techniques or by using more advanced oversampling methods.

Decision Trees are a widely used machine learning method for regression and classification tasks. They create a tree-like structure that maps out decision pathways and their corresponding outcomes. Decision nodes in the tree represent a choice based on a particular feature or attribute, while each branch represents a potential value or outcome for that feature. During the training phase, the algorithm identifies the most useful feature to split the data at each node. This process continues recursively until a specified stopping criterion, such as a minimum number of samples or maximum tree depth, is reached. By partitioning the data into subsets that are as distinct as possible, Decision Trees can make accurate predictions for new inputs. Decision trees, however, can be vulnerable to overfitting, particularly if the tree is allowed to go too deep or if there is noisy or irrelevant input. This problem can be solved by using ensemble methods to join numerous Decision Trees and enhance their performance, such as Random Forest or Gradient Boosting. The authors [30] implemented a Decision Tree algorithm to detect network intrusions and achieved an accuracy of 98.27% on the NSL-KDD dataset. The authors also noted that feature selection played an important role in achieving high accuracy, and they employed a feature selection algorithm to improve the performance of their system.

When there are only two possible values for the output variable, such as 0 or 1, the statistical procedure known as logistic regression is performed. It calculates the likelihood that the input belongs to a certain class by fitting the data to a logistic function, often known as a sigmoid function. Any input value is translated by the logistic function into a value between 0 and 1, which represents the likelihood that the input belongs to one of the two classes. By minimizing a cost function, such as the binary cross-entropy loss, the method learns how to estimate the logistic function's ideal coefficients. The algorithm iteratively changes the coefficients until the cost function is reduced. The cost function assesses the difference between the anticipated probabilities and the actual labels. The logistic function is used by the model to forecast the likelihood that a new input will belong to a specific class during the testing phase. The final anticipated class label can subsequently be obtained by thresholding the predicted probability at a specific value, such as 0.5. After conducting experiments on the NSL-KDD dataset, the logistic regression model [31] exhibited promising results in identifying network intrusions. By performing a feature selection process, the model's performance was significantly enhanced. This process helped to identify features that had a strong correlation with network intrusions. As a result, the proposed model demonstrated high accuracy and a low rate of false positives in detecting network intrusions. Furthermore, the study found that logistic regression was computationally efficient, making it a viable choice for real-time intrusion detection systems.

The widely used machine learning method known as Gradient Boosting Classifier (GBC) is utilized for classification problems. It is a kind of ensemble learning technique that combines several weak learners to produce a model that is more reliable and accurate. The GBC algorithm constructs decision trees iteratively using their mistakes as a training set. The method begins the training process with a basic decision tree, often known as a weak learner. Following training on the training set of data, the weak learner's predictions are assessed. The following decision tree in the ensemble is trained using the mistakes made by the weak learner. This process continues until the desired precision is attained or the specified number of trees has been planted. The GBC approach modifies the weights of the training data after each iteration to give priority to the samples that the previous tree incorrectly identified. This is done to force successive trees to concentrate on the data that are more difficult to categorize. To get the best weights for the decision trees, the approach additionally uses gradient descent optimization. The final forecast is created by combining the predictions from each decision tree. A weighted average of each decision tree's predictions, with the weights based on the accuracy of the individual trees, is used by the GBC method. The proposed [32] GBC algorithm outperforms other traditional classification algorithms such as Logistic Regression, K-Nearest Neighbor, and Naive Bayes on the NSL-KDD dataset. The adaptive learning rate technique used in the GBC algorithm helps to improve the accuracy of the model by dynamically adjusting the learning rate for each iteration, which helps to avoid overfitting. The feature selection technique used in the study helps to identify the most relevant features for intrusion detection, which reduces the complexity of the model and improves its accuracy.

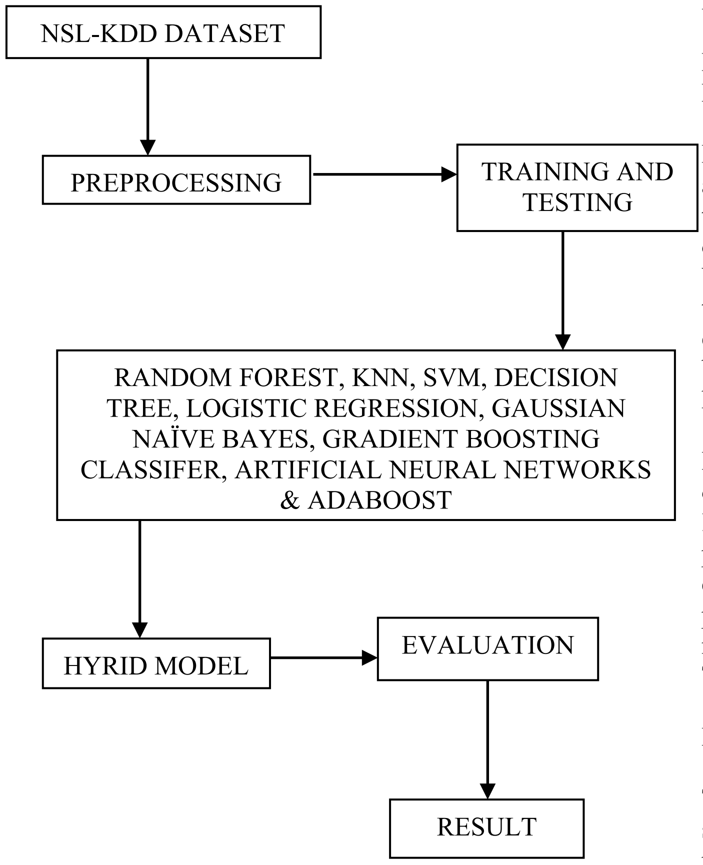
Machine learning algorithms known as Artificial Neural Networks (ANN) are modeled after the structure and operation of the human brain. An artificial neural network (ANN) is made up of interconnected nodes (neurons) arranged in layers. Each neuron conducts a straightforward mathematical calculation on its inputs and outputs the result to the layer below it. Each neuron in the input layer of an ANN corresponds to one input feature, and the input layer receives data in the form of a vector. The input data is transformed in several ways by the hidden layers, which are sandwiched between the input and output layers. Each neuron in the hidden layer receives input from every neuron in the layer below, and every neuron in the layer above receives its output. The ultimate result of an ANN is generated by the output layer, which can be either a single value (in the case of regression) or a probability distribution over several categories (in the case of classification). The authors [33] used different ANN models with varying numbers of hidden layers, neurons, and activation functions to perform the classification task. The key findings and observations from the study are that ANNs outperformed traditional machine learning algorithms such as decision trees, k-nearest neighbor, and Naive Bayes in terms of accuracy, precision, recall, and F-measure. Increasing the number of hidden layers and neurons in the ANN models improved the performance of the classifier up to a certain point, beyond which it started to overfit the training data.

A result is produced by merging the predictions of various models using the machine learning process known as ensemble learning. On the other hand, hybrid models combine different kinds of machine learning methods, including decision trees and neural networks. Ensemble hybrid models use many hybrid models to get a final forecast, using both approaches. Several hybrid models with somewhat varying configurations are trained on the same dataset in ensemble hybrid machine learning. The final prediction is then created by combining each of these unique models using a particular technique, such as voting or averaging. The concept behind ensemble hybrid machine learning is that by integrating the benefits of various models, a model can be created that is more accurate and reliable than any single model. Yet it's vital to remember that ensemble hybrid ML can be computationally expensive and might need a lot of resources to train and test the models. The paper [34] proposes a hybrid ensemble model for network intrusion detection using the NSL-KDD dataset. The model combines multiple ML algorithms, including decision trees, random forests, and gradient-boosting classifiers, to improve the accuracy and robustness of intrusion detection. The research concludes that the proposed hybrid ensemble model can effectively detect network intrusion and has the potential to be applied in real-world scenarios.

A machine learning approach called AdaBoost (Adaptive Boosting) combines several weak learners (simple, inaccurate classifiers) to produce a strong learner (a more accurate classifier). AdaBoost's core principle is to iteratively train a series of weak classifiers on various iterations of the training data while giving each training example weights based on how well the previous classifiers did on that example. The weighted average of each weak classifier is then used to create the final strong classifier. AdaBoost concentrates more on the challenging cases in the training data by increasing the weight of incorrectly classified examples while decreasing the weight of well-categorized examples. With the help of this adaptive

weighting system, the final classifier will be better equipped to deal with samples that are challenging to categorize.
A well-liked technique for handling binary classification issues is AdaBoost (where the goal is to assign each input to one of two classes). It has been utilized in a wide range of applications, including image recognition, object detection, and text classification. The suggested method [35] may accurately identify previously unknown attacks and is resistant to many forms of attacks. The study offers a thorough assessment of the suggested approach using several performance criteria, such as accuracy, precision, recall, F1-score, and AUC, and demonstrates that the strategy performs well on all of them. In comparison to employing a single classifier, the suggested ensemble learning strategy, which integrates multiple classifiers using the AdaBoost algorithm, can greatly increase the accuracy of intrusion detection.

IV. Diagram of Implementation


NSL-KDD DATASET
PREPROCESSING
TRAINING AND TESTING
RANDOM FOREST, KNN, SVM, DECISION TREE, LOGISTIC REGRESSION, GAUSSIAN NAÏVE BAYES, GRADIENT BOOSTING CLASSIFER, ARTIFICIAL NEURAL NETWORKS & ADABOOST
HYRID MODEL
EVALUATION
RESULT

V. Evaluation Metrics Used

We are using four metrics to define the ability of the machine learning algorithms to evaluate the results. Accuracy Rate, Precision Score, Recall Value, and F1 Score are the metrics for our discussion and evaluation.
Accuracy Rate: The accuracy of a model is the proportion of accurate predictions it makes for a given collection of data. For comparing how well several models perform, choosing the best model for a given problem, and optimizing model parameters, accuracy is helpful. It is also employed to assess a model's performance during its training and testing phases.

The accuracy rate is calculated as follows:

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)
Yet, accuracy might not always be the optimal metric to employ. For instance, accuracy may be deceptive when the dataset is unbalanced and there are disproportionately more samples of one type than another. Other metrics like precision, recall, F1-score, or area under the curve (AUC) may be more appropriate in these circumstances.

Precision Score: A statistic called precision is used in machine learning to assess how well a categorization model is performing. Out of all the positive predictions made by the model, it calculates the percentage of true positives (i.e., accurate positive predictions). In other words, it assesses the accuracy of the model's optimistic forecasts.
Precision is calculated as:

Precision = True Positives / (True Positives + False Positives)

Precision is helpful when there is a significant cost associated with false positives (predicting a positive when the class is negative), and we wish to reduce the frequency of false positives.
Yet, accuracy might not always be the appropriate statistic to employ. When there are disproportionately more examples of one class than another in the collection, it can be deceptive. In these circumstances, a model could attain high precision by consistently predicting the majority class.

Recall Rate: Recall, which is often referred to as sensitivity or the true positive rate, is a statistic used in machine learning to assess how well a classification model is performing. Out of all real positives in the dataset, it calculates the percentage of true positives (i.e., accurate positive forecasts). In other words, it assesses how well the model can recognize success stories.
The recall is calculated as:

Recall = True Positives / (True Positives + False Negatives)

The recall is helpful when the cost of false negatives is significant and we want to reduce the number of missed positive cases. False negatives occur when we forecast a negative when the true class is positive. Recall has the benefit of measuring a model's ability to recognize good examples, independent of how many false positives it may also produce. Therefore, the recall may not provide an accurate view of a model's performance on its own. For instance, a model with high recall but low precision (the percentage of positive predictions that are accurate) may produce a lot of false positives, which in some applications may be expensive or harmful.

F1 Score: A machine learning metric called the F1 score is used to assess how well a categorization model is performing. It measures a model's ability to accurately

identify positive examples while avoiding false positives and false negatives and is the harmonic mean of precision and recall.
The F1 score is calculated as:

F1 score = 2 * (precision * recall) / (precision + recall)

A score of 1 implies flawless precision and recall, whereas a score of 0 indicates that the model is unable to accurately recognize any positive examples. The F1 score ranges from 0 to 1. The F1 score has the benefit of providing a fair assessment of a model's performance, accounting for both precision and recall. It is especially helpful when we want to strike a balance between finding good instances and avoiding mistakes and the cost of false positives and false negatives is nearly equal. The F1 score, however, might not necessarily be the most useful metric.

VI. Experimental Results And Discussion

The NSL-KDD dataset was already divided into a training and testing dataset for easier access and evaluation purposes. The training dataset had the dimensions of (125973, 42) while the testing dataset had dimensions of (22544, 42).
To accurately capture the results of the models, the dataset was further divided according to the attack types. The dataset consists of four attack types:
DoS Attacks: A DoS (Denial of Service) attack is a sort of cyberattack in which the attacker tries to stop a website or online service from being accessible by flooding it with traffic and impairing its capacity to respond to valid requests. The attacker accomplishes this by either flooding the target system with a significant amount of traffic from numerous sources or by taking advantage of weaknesses in the infrastructure or software of the system. Attacks like the Smurf, Neptune, and Teardrop are examples of this.

Probe Attack: A cyber-attack known as a probing attack includes searching a network or computer system for vulnerabilities and information. The attacker sends queries or requests to the target system or network using a variety of tools and techniques to elicit answers that reveal system information or flaws. Attacks known as probing can be automated or manual and leverage a variety of protocols, including ICMP, TCP, UDP, and HTTP. Examples include Satan, ipsweep, and nmap attacks.

User to Root (U2R) Attack: In a User to Root (U2R) attack, the attacker gains illegal access to a system or network to escalate their privileges from that of a regular user to that of a privileged system administrator, also referred to as the root user. In a U2R assault, the attacker first takes advantage of a system flaw or vulnerability, like a buffer overflow, SQL injection, or an unsecured configuration, to log in as a regular user. After gaining entry, the attacker employs several strategies to analyze their privileges to root, which gives them total control of the system. Eject, load module and Perl assaults are a few examples.
Root to Local (R2L) Attack: An attempt to access lower-privileged user accounts is made by an attacker with root-level access to a system or network during a Root to Local (R2L) attack. In an R2L attack, the attacker attempts to use flaws in the system or network to access a user account with lesser privileges, giving them the ability to perform unlawful acts or steal confidential information. Ftp write, guess passwords, and imap attacks are a few examples.

Here are the findings for better understanding:

Accuracy Rate Comparison
Table 1

ML Algorithm	DoS	Probe	U2R	R2L
Random Forest	0.99785	0.99662	0.99775	0.98079
KNN	0.99715	0.99077	0.99703	0.96705
SVM	0.99371	0.98450	0.99632	0.96793
Gaussian Naïve Bayes	0.86733	0.97898	0.93562	0.97259
Decision Tree	0.99167	0.97824	0.93300	0.99632
Logistic Regression	0.99400	0.98384	0.96570	0.99683
Gradient Boosting Classifier	0.99808	0.99646	0.98118	0.99775
Artificial Neural Network	0.99668	0.99242	0.97420	0.99806
AdaBoost	0.99819	0.99670	0.99857	0.97944
Hybrid Model (RF, KNN & SVM)	0.99814	0.99283	0.99744	0.97198
Hybrid Model (RF, ANN & AdaBoost)	0.99825	0.99679	0.99816	0.98095
Hybrid Model (GBC, SVM & KNN)	0.99790	0.99283	0.99734	0.97325
Hybrid Model (GBC, ANN & AdaBoost)	0.99837	0.99687	0.99888	0.98071

From the above-given table 1, we can see that Random Forest, ANN, GBC, and AdaBoost have given a high score on testing individually on each of the attacks mentioned. On using a hybrid model the accuracy has increased in the final used model which is a combination of GBC, ANN, and AdaBoost which individually have given a good score. In the context of the NSL-KDD dataset, the accuracy rate can be a helpful indicator of how well a machine learning model performs in general at differentiating between legitimate traffic and attack traffic.

Precision Rate Comparison
Table 2

ML Algorithm	DoS	Probe	U2R	R2L
Random Forest	0.99705	0.99277	0.87956	0.96954
KNN	0.99709	0.98508	0.85073	0.95439
SVM	0.99380	0.98365	0.82909	0.96264
Gaussian Naïve Bayes	0.84830	0.96051	0.97911	0.95508
Decision Tree	0.99225	0.95509	0.81372	0.95559
Logistic Regression	0.99414	0.97967	0.83763	0.96071
Gradient Boosting Classifier	0.99801	0.99344	0.90549	0.97367
Artificial Neural Network	0.99666	0.98844	0.90445	0.97401
AdaBoost	0.99806	0.99510	0.94022	0.97020
Hybrid Model (RF, KNN & SVM)	0.99784	0.98994	0.88054	0.96291
Hybrid Model (RF, ANN & AdaBoost)	0.99830	0.99417	0.90337	0.97315
Hybrid Model (GBC, SVM & KNN)	0.99780	0.98978	0.87335	0.96706
Hybrid Model (GBC, ANN & AdaBoost)	0.99831	0.99355	0.93432	0.97616

From the given Table 2, we can observe that Ada Boost performs the best in getting a higher precision score followed by Random Forest and ANN. The proposed Hybrid Model of GBC, ANN, and AdaBoost respectively produce a better precision score than being individually given by the models. A model with high precision can accurately identify attack traffic while minimizing false positives, and therefore, is more useful in practice. Furthermore, the improvements which can be done to improve the precision score can be sampling techniques, such as oversampling the minority class or undersampling the majority class. This helps to balance the dataset.

Recall Rate Comparison
Table 3

ML Algorithm	DoS	Probe	U2R	R2L
Random Forest	0.99852	0.99555	0.94754	0.97425
KNN	0.99342	0.98606	0.93143	0.95265
SVM	0.99371	0.96907	0.91056	0.94854
Gaussian Naïve Bayes	0.90081	0.97323	0.60157	0.89097
Decision Tree	0.99107	0.97620	0.91824	0.88727
Logistic Regression	0.99367	0.97049	0.93066	0.94470
Gradient Boosting Classifier	0.99809	0.99545	0.93356	0.97315
Artificial Neural Network	0.99659	0.98793	0.95707	0.95542
AdaBoost	0.99820	0.99695	0.96552	0.97192
Hybrid Model (RF, KNN & SVM)	0.99771	0.98769	0.94865	0.95829
Hybrid Model (RF, ANN & AdaBoost)	0.99846	0.99711	0.96188	0.97385
Hybrid Model (GBC, SVM & KNN)	0.99793	0.98786	0.93833	0.95821
Hybrid Model (GBC, ANN & AdaBoost)	0.99837	0.99687	0.98406	0.97003

In the above-given table 3, Recall Rate calculation gives a better output when using SVM, GBC, and Ada Boost. Our proposed hybrid model performs efficiently considering all the algorithm's performance and evaluation. Therefore, recall rate provides a more nuanced measure of a model's performance, as it focuses specifically on its ability to identify the positive class (i.e., attack traffic) while minimizing false negatives. The recall is particularly important for reducing false negatives, such as incorrectly classifying an attack. A model with a high recall can correctly identify a larger proportion of the attacks in the dataset and is therefore more effective in detecting malicious activity.

F1 Score Comparison
Table 4

ML Algorithm	DoS	Probe	U2R	R2L
Random Forest	0.99758	0.99482	0.90277	0.97286
KNN	0.99710	0.98553	0.87831	0.95344
SVM	0.99360	0.97613	0.84869	0.95529
Gaussian Naïve Bayes	0.85795	0.96654	0.66091	0.91620
Decision Tree	0.99171	0.96502	0.83802	0.91331
Logistic Regression	0.99390	0.97497	0.86486	0.95232
Gradient Boosting Classifier	0.99804	0.99444	0.91415	0.97337
Artificial Neural Network	0.99662	0.98814	0.92435	0.96423
AdaBoost	0.99781	0.99612	0.94055	0.97096
Hybrid Model (RF, KNN & SVM)	0.99804	0.98790	0.88594	0.96066
Hybrid Model (RF, ANN & AdaBoost)	0.99804	0.99495	0.93696	0.97405
Hybrid Model (GBC, SVM & KNN)	0.99787	0.98880	0.89419	0.96249
Hybrid Model (GBC, ANN & AdaBoost)	0.99840	0.99496	0.95146	0.97255

From the above-given table 4, F1 Score has shown a fluctuating observation for various algorithms. Random Forest, ANN, and AdaBoost prove to be the better algorithm individually. On putting the best of them for a hybrid calculation, we see that it improves on the parent algorithms and gives a better score. As we know the more the score is closer to 1, the higher its efficiency.

With the above tables, we can easily say that the hybrid model using ANN, AdaBoost, and GBC has proven metrically that its detection rate is better than the normal algorithms. It also has given a higher precision and F1 score too. Therefore, through the above-given calculations and tabular observations, we can identify the hybrid model's ability to get the intrusion detection system working.

VII. Conclusion and Future Scope

The overall conclusion from this research suggests that we have come a long way from detecting the accuracy, and usability of different algorithms for classification and prediction. Earlier research was limited to a few numbers of ML models. Over the years, different algorithms and methods have been introduced to improve upon the earlier given results and conclusions. Our research has found that experimenting hybrid model using different individual models' results has provided a way to give the output that we desire.

Our research has also given a high score for accuracy, precision, recall rate, and F1 score too. These four-evaluation metrics are the most important. Artificial Neural Networks have proven to be the best algorithm which has consistently given results matching almost to perfection. The capacity of ANNs to learn intricate non-linear correlations between input characteristics and output classes is one of their advantages. Large data sets can be handled by ANNs, and they generalize well to new data. Moreover, ANNs are flexible tools for evaluating the NSL-KDD dataset since they may be used for both supervised and unsupervised learning.

Finally, it should be mentioned that no system can be made completely secure using the best algorithms while still defending our resources from network threats. Because of this, the field of computer security research is always evolving and complex.

References

[1] MyCert, 2007. Network Security Incidents in Malaysia. http://www.mycert.org.my.

[2] Panda, Mrutyunjaya & Abraham, Ajith & Das, Swagatam & Patra, Manas. (2011). Network intrusion detection system: A machine learning approach. Intelligent Decision Technologies (IDT) Journal, IOS Press. 5. 347-356. 10.3233/IDT-2011-0117.

[3] S. Suthaharan and T. Panchagnula, "Relevance feature selection with data cleaning for an intrusion detection system," 2012 Proc. IEEE Southeastcon, pp. 1–6, 2012.

[4] Ali A. Ghorbani, Wei Lu and M. Tavallaee, "Network intrusion detection and prevention: Concepts and Techniques, Advances in Information security", Springer, 2010.

[5] [5] L. Lipton, Zachary C., John Berkowitz, and Charles Elkan. (2015) "A critical review of recurrent neural networks for sequence learning." arXiv preprint arXiv:1506.00019.

[6] I. Levin, "KDD-99 Classifier Learning Contest: LLSoft s Results Overview," SIGKDD explorations, vol. 1, pp. 67-75, 2000.

[7] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory," ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, 2000.

[8] L. C. Wuu, C. H. Hung and S. F. Chen, "Building Intrusion Pattern Miner for Snort Network Intrusion Detection System. Journal of Systems and Software", vol. 80, Issue 10, pp. 1699-1715, 2007.

[9] Chebrolu, S., A. Abraham and J.P. Thomas, "Feature deduction and ensemble design of intrusion detection system". Comput. Secur., 24: 295-307. DOI: 10.1016/j.cose.2004.09.008

[10] N. Toosi and M. Kahani, "A New Approach to Intrusion detection Based on An Evolutionary Soft Computing Model Using Neuro-Fuzzy Classifiers. Computer Communications", vol. 30, Issue 10, pp. 2201-2212, 2007.

[11] M. Panda and M.R.Patra, "Mining association rules for constructing network intrusion detection model", International Journal of applied engineering and research, 4(3) (2009),381-398.

[12] S. Mukkamala, G. Janoski, and A. H. Sung, “Intrusion detection using Neural network and support vector machines”, in Proceedings of IEEE International conf. on Neural networks, 2002, 1702-1707, IEEE Computer Society Press, USA

[13] Heckerman, D., D. Geiger and D.M. Chickering, 1995, “Learning Bayesian networks: The combination of knowledge and statistical data”. Machi. Learn., 20: 197-243. DOI: 10.1007/BF00994016

[14] Gowadia, V., C. Farkas and M. Valtorta, 2005. PAID: “A probabilistic agent-based intrusion detection system”. Comput. Secur., 24: 529-545. DOI: 10.1016/j.cose.2005.06.008

[15] Vinayakumar, R., K. P. Soman, and Prabaharan Poornachandran. (2017) "Applying convolutional neural network for network intrusion detection." 2017 International Conference on Advances in Computing, Communications, and Informatics (ICACCI), 1222-1228. IEEE.

[16] H. A. Nguyen and D. Choi, “Application of data mining to network intrusion detection: classifier selection model”, in: Proceedings of Challenges for Next Generation Network Operations and Service Management (APNMOS 2008), Y. Ma, D. Choi, and S. Ata eds, Lecture notes in computer science, Vol. 5297 (2008), 399-408, Springer.

[17] N. B. Amor, S. Benferhat and Z. Elouedi, “Naïve Bayes Vs Decision Trees in Intrusion detection system”, in: Proceedings of ACM symposium on applied computing, 2004, 420-424.

[18] John Mill and A. Inoue, “Support vector classifiers and network intrusion detection”, in: Proceedings of 2009 IEEE Intl. conf. on the fuzzy system, 2004, WA, USA, 1, 407-410.

[19] Dewan Md. Farid, Nouria Harbi, and Mohammad Zahidur Rahman, "Combining Nave Bayes and Decision Tree for Adaptive Intrusion Detection," International Journal of Network Security & Its Applications, Vol. 2, No. 2, April 2010, pp. 12-25.

[20] Juan Wang, Qiren Yang, Dasen Ren, “An intrusion detection algorithm based on decision tree technology,” In the Proc. of IEEE Asia-Pacific Conference on Information Processing, 2009.

[21] Geetha Ramani R, S.SivaSathya, SivaselviK, "Discriminant Analysis based Feature Selection in KDD Intrusion Dataset,", International Journal of Computer Application VoI.31, No.ll, 2011

[22] M. Panda, and M. R. Patra, A Hybrid Clustering approach for network intrusion detection using COBWEB and FFT, Journal of Intelligent system, 18(3) 2009), 229-245

[23] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, “Modelling intrusion detection system using hybrid intelligent systems”, Journal of network and computer applications. 30 (2007), 114-132.

[24] L. Breiman, Random Forests, *Machine Learning*, 45 (2001), 5-32.

[25] M. Panda, A. Abraham, and M.R.Patra, Discriminative multinomial Naïve Bayes for network intrusion detection, in Proceedings of 6th Intl. conf. on information assurance and security (IAS-2010), Aug. 2010, USA, 5-10, IEEE Press.

[26] Nanda, N.B., Parikh, A. (2019). Hybrid Approach for Network Intrusion Detection System Using Random Forest Classifier and Rough Set Theory for Rules Generation. In: Luhach, A., Jat, D., Hawari, K., Gao, XZ., Lingras, P. (eds) Advanced Informatics for Computing Research. ICAICR 2019. Communications in Computer and Information Science, vol 1076. Springer, Singapore.

[27] R. K. Patel and K. V. Patel. "Application of KNN Algorithm for Intrusion Detection on NSL-KDD Dataset." International Journal of Computer Applications 168, no. 10 (2017): 24-28.

[28] A. K. Pandey, S. B. Gupta, V. K. Singh, and R. K. Tiwari. "Intrusion Detection in Computer Networks using Support Vector Machines and NSL-KDD Dataset." International Journal of Recent Technology and Engineering 9, no. 2S2 (2021): 2494-2498.

[29] N. B. Waghmare and V. G. Gadicha. "Intrusion Detection using Naive Bayes Classifier with Oversampling on NSL-KDD Dataset." International Journal of Computer Applications 180, no. 38 (2018): 34-39.

[30] M. K. Shah, M. T. Ahir, and H. M. Shah. "Intrusion Detection System Using Decision Tree Algorithm Based on NSL-KDD Dataset." International Journal of Engineering and Technology (IJET) 9, no. 1 (2017): 14-19.

[31] H. A. Khan, M. U. Khan, and M. A. Khan. "Intrusion Detection System for Network Security using Logistic Regression on NSL-KDD Dataset." International Journal of Computer Science and Information Security (IJCSIS) 17, no. 6 (2019): 45-50.

[32] S. B. Utpat, S. S. Limaye, and S. A. Rane. "Intrusion detection using gradient boosting with adaptive learning rate on NSL-KDD dataset." In 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), pp. 1-6. IEEE, 2019.

[33] S. S. Muthukrishnan and K. Thanushkodi. "Intrusion detection in computer networks using neural networks with NSL-KDD dataset." In 2015 IEEE Technological Innovation in ICT for Agriculture and Rural Development (TIAR), pp. 123-128. IEEE, 2015.

[34] S. Sultana, S. U. Khan, and K. Tariq. "An Efficient Hybrid Ensemble Model for Network Intrusion Detection using NSL-KDD Dataset." In 2019 15th International Conference on Emerging Technologies (ICET), pp. 1-6. IEEE, 2019.

[35] X. Zhang and H. Zhang, "An Ensemble Learning Approach for Network Intrusion Detection based on NSL-KDD Dataset," 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA), Zhangjiajie, China, 2017.