DOI: 10.21917/ijct.2023.0426

INTRUSION DETECTION SYSTEM TO AVOID MALICIOUS INTRUDERS IN HIGHER LAYER NETWORK SECURITY

E. Kamalanaban[1], S. Madhusudhanan[2], D. Jennifer[3] and M. Jayaprakash[4]

[1]*Department of Computer Science and Engineering, Veltech Hightech Dr.Rangarajan Dr.Sakunthala Engineering College, India*
[2]*Department of Computer Science and Engineering, Prathyusha Engineering College, India*
[3]*Department of Computer Science and Engineering, Panimalar Engineering College, India*
[4]*Department of Information Technology, RMK Engineering College, India*

Abstract

Online criminals are focusing their attention more and more on ordinary computer users, seeking to take advantage of them through a variety of social and technological exploitation techniques. Some hackers are getting more skilled and determined. The ability to conceal their identities, keep their communications secret, keep their finances separate from their activities, and make use of private infrastructure are all areas in which cybercriminals have shown a high degree of proficiency. It is of the utmost importance to safeguard computers with surveillance systems that are able to identify complex varieties of malware. In this paper, we utilized machine learning algorithm to validate the samples from different datasets. The machine learning classifier is utilized to find the efficacy of the entire model in validating the class samples. The simulation is conducted in python to test the efficacy of the model against various class of datasets. The results show that the proposed method achieves higher degree of accuracy than the other models.

Keywords:

IDS, Security, Attack, Network Security

1. INTRODUCTION

An additional critical component of internet security is the intrusion surveillance system. Intrusion Detection Systems (IDS) are responsible for analyzing the traffic on a network or a portion of the general computing environment in order to look for indications of malicious behavior.

As a result of increased research and development efforts in this field, significant progress has been made in a number of subfields of artificial intelligence (AI), including pattern identification and anomaly finding. These advancements have been made possible as a result of recent breakthroughs in AI. It is common practice to use neural networks (NN) to find solutions to challenges of this kind, and the range of situations in which they can be applied has grown substantially over the past few years. The increase in the total quantity of computing capacity that is currently available is the primary explanation for this phenomenon. Researchers were forced to make adjustments to the IDS architectures that were already in place as a result of this circumstance [1].

IDSs, also known as IDS, are organizations that monitor computer systems and networks for unwanted or illegal activity. These organizations are also known by their acronym, IDS. When it comes to classifying IDS, there are a variety of different techniques. The primary differentiating factors between the first two categories, which are divided according to where the malicious behavior was identified, are the network and the server [2]. Network-based IDSs are concerned with the security of networks and perform monitoring and analysis of data collected from networks. Additionally, these systems monitor and analyze data collected from other networks. In host-based IDSs, the software environment of a particular computer, in addition to the processes and events that take place on that machine, is monitored in order to identify any nefarious actions that may have taken place [3].

Signature-based (misuse-based), anomaly-based, and hybrid IDSs (IDSs) are three further subcategories of IDSs that can be further subcategorized based on the data analytics techniques that have been implemented. The purpose of the signature-based technique is to analyze network packets or data from a specific system (such as logs), with the goal of locating signatures or patterns that are indicative of malicious behavior. This can be done in order to protect the system from potential threats. Because it makes use of data that has been pre-annotated and retrieved from a database, this technique is significantly more effective than other approaches when it comes to defending against attacks that have been identified in the past [4]. It is a simple strategy that gets the job done, but it does have some drawbacks, such as the inability to distinguish brand-new types of attack and the requirement for regular database updates. Despite these drawbacks, the strategy is effective. However, it does get the task done [5].

An investigation of the data using an anomaly-based technique is carried out in order to identify network and system behaviors that are not typical. The goal of this endeavor is to find out what is causing the anomalous behavior. If the algorithm is trained using the data that has already been supplied, then achieving this objective might not be as difficult as it initially seems. The potential of the strategy that has been explained lies in the fact that, in contrast to other, more conventional techniques, it makes it possible to identify zero-day vulnerabilities [6]. In addition to this, it makes it much simpler to implement a more individualized and customized version of any given infrastructure. It is possible that these methods will discover situations that are anomalous but are not necessarily cyber security attacks because they are not only based on labeled data but are also taught to recognize anomalies based on data that was provided in the past. This is because these methods are taught to recognize anomalies based on data that was provided in the past. Because of this, there will be a significant increase in the number of false positive notifications [7].

Researchers have begun focusing their attention on developing IDS by making use of techniques associated with machine learning in an effort to find a solution that will fix the issues that were discussed earlier. The use of artificial intelligence techniques, such as machine learning, enables the independent mining of large data sets for the purpose of gaining new insights [8].

Machine learning-based IDSs are able to accomplish satisfactory detection levels when there is a sufficient amount of training data available. This is due to the fact that models created through machine learning are sufficiently generalizable to recognize novel attacks in addition to attack variants [9]. Because they do not require a significant amount of subject knowledge to develop and construct, IDSs that are based on machine learning are simple to develop and create [10].

Deep learning, a subfield of machine learning, has shown remarkable progress in terms of efficiency in recent years. These improvements have been demonstrated by the subfield. Deep learning approaches are superior to more conventional machine learning methods in terms of their ability to manage enormous amounts of data. Deep learning is a subfield of machine learning. The methodologies of deep learning have the ability to automatically acquire feature representations from raw data, after which they can output the outcomes of their work. These approaches are not only effective but also easy to implement.

2. LITERATURE SURVEY

The feature engineering requires knowledge in the pertinent field, and the quality of the features themselves is frequently the limiting factor in the performance of recognition systems. The development of automated feature learning is a crucial component of detection methods that are based on deep learning. The all-encompassing nature of these methods is rapidly propelling them to the forefront of IDS research, where they are swiftly becoming the industry standard [11].

Since deep learning techniques are able to process raw data directly, they are also able to perform categorization while simultaneously accumulating features. This is made possible by the fact that deep learning techniques can process data in their native format. In [12] suggested using convolutional neural networks in the process of identifying things as a means to improve accuracy. Both the NSL-KDD and the UNSW-NB 15 databases were used in the research that was carried out by each of these organizations, respectively. Feature vectors are a specific form of data structure that is utilized by these collections. The feature vectors were originally transformed into photographs because CNNs work more effectively with data that is represented in a two-dimensional (2D) space. The total number of features increased from 41 to 464, and one-hot coding was utilized for the hypothetical features. Following that, we crafted a singular image by combining all of the individual 8-byte segments into one. Instead of the places being left blank, zeros were used to fill them in. A picture with the measurements of 8 pixels by 8 pixels was produced as a result of the application of the feature vectors to the production process. In the end, they decided to construct a CNN that had three levels in order to better coordinate the attack [13].

For the feature extraction phase of the process, an unsupervised deep learning model can be utilized, while shallow models can be utilized for the classification stage of the process. According to author [14], the process of feature extraction was carried out using a sparse autoencoder, and the identification of attacks was carried out using an XGBoost model. During the course of their investigation, they made use of the information provided by NSL-KDD. They chose to conduct SMOTE sampling after determining that the information was biased in a particular direction. In order to ensure that no one group is disproportionately represented, the SMOTE method divides the bigger groups into an extremely large number of smaller subgroups that are easier to manage. A sparsity restriction is incorporated into the sparse autoencoder as a means of improving the autoencoder capacity to identify samples with which it was not previously familiar. In the end, a classification of the data that was collected was accomplished through the use of an XGBoost algorithm.

Deep learning models generate results that are less than stellar when they are applied to datasets that are either too small or too imbalanced. Despite the fact that these models have made remarkable development in their ability to analyze large amounts of data, this is still the case. Even if there is only a limited amount of data to work with, increasing the recognition rate can be accomplished through the use of an adversarial learning strategy.

Using a GAN, it able to enhance the overall quality of the data. Because of the imbalance and the absence of novel data that is included in the dataset, machine learning models that have been trained on the KDD99 dataset suffer from poor generalizability. This is because the dataset contains neither balanced nor novel data. In order to address these issues, they utilized a GAN in order to increase the total quantity of data that was collected. The KDD-99 flow data were very similar to the findings that the GAN model generated, which had a high degree of similarity [15].

It is feasible to identify attack variants that would not have been possible without the inclusion of this generated data in the training collection. They chose eight different attacks to examine and then contrasted the accuracy rates of the initial dataset with those of the expanded dataset. The findings of the experiments demonstrated that adversarial learning was responsible for an improvement in the accuracy of seven out of the eight distinct types of attacks.

3. LIGHTGBM

The GBDT-based algorithm that was developed by Microsoft Research Asia and assigned the name LightGBM. The ultimate goal is to improve computational efficiency in order to solve problems associated with making predictions using large amounts of data. This will be accomplished by improving computational efficiency. Before continuing with these procedures, the GBDT algorithm will first select and separate the signs based on a previously determined order. This method requires more effort and memory than any other approach, but it provides a split-off point that is more accurate than any other method. In order to accomplish these goals, LightGBM employs a histogram-based algorithm in association with a leaf-wise tree development strategy that is limited to a maximum depth. This combination of tools is bound to a maximum tree depth.

The DT method, which is illustrated here in Figure 4, makes use of histograms as one of its components. This provides a good illustration of the s-bin discretization technique, which is used to separate consecutive floating-point eigenvalues. As soon as that step is finished, the histogram of breadth is constructed using these divisions as the fundamental units of its construction. The histogram is used to collect the important statistics after the data has been traversed for the very first time. These statistics include the sum of the gradients and the number of samples that are contained in each bin. It is possible to use the discrete value of the histogram as a tool to assist in determining the best location at which to segment the data. The costs that are typically incurred for storing data and carrying out computations are cut down thanks to the utilization of this approach.

The leaf-based growth technique of development known as level-wise requires synchronized leaf division among the plants that are on the same layer. This is done in order to maximize crop yield. It is to one advantage to optimize using a variety of procedures and to maintain control over the degree to which the model is complicated. Despite the fact that different leaves on the same layer receive information in distinctive ways, the data from those leaves is nevertheless processed in the same manner. It is possible for us to determine the anticipated decrease in entropy that will take place of attribute-based partitioning of the nodes by making use of the formula.

$$IG(B,V) = En(B) - \sum_{v \in V} BE_n(B_V) \tag{1}$$

$$En(B) = \sum_{d=1}^{D-p} d \log_2 p_d \tag{2}$$

where

$En(B)$ - information entropy

B - collection ,

p_d - ratio of the collection B w.r.t category d,

D - categories,

v - V attribute value, and

B_v - Bth subset.

This strategy is inefficient because it requires searching and splitting a large number of leaves, which results in a negligible increase in information obtained but consumes a significant quantity of additional memory. Additionally, the information gained from searching and splitting these leaves is not particularly useful. In comparison, the efficiency of the leaf-wise growth strategy can be improved by dividing the leaf into two parts only when doing so will result in the greatest information gain on the same layer. This is the case when dividing the leaf into two parts will maximize the information gain on the same layer. Because this method may produce trees with a high depth, which can contribute to overfitting, a maximum depth limitation is implemented during the development of the tree. This is done because there is a possibility that this method will result in trees having a deep trunk.

LightGBM uses the leaf-wise tree growth algorithm, as opposed to the depth-wise tree growth algorithm that is used by a significant number of other well-known algorithms. The depth-wise tree growth algorithm is more common. Using the leaf-wise growth method as opposed to the depth-wise growth method makes it possible to achieve convergence more quickly; however, if appropriate hyperparameterization is not performed, overfitting may occur. The act of making adjustments to the values of an algorithm hyperparameters in order to enhance the performance of a model is referred to as hyperparameter optimization, which is also the name of the term for the process.

To improve the overall performance of the LightGBM models, we use the Tree-structured Parzen Estimator (TPE) algorithm that we developed in this article to optimize the hyperparameters of those models. This was done so that we could improve the overall performance of the LightGBM models. To be more particular, TPE employs a method known as iterative sequential model-based optimization (SMBO) in order to construct a reaction surface model and collect additional data. This is done in order to improve the accuracy of the model. The construction of the model in SMBO is influenced by a number of variables, including the following:

$$H, \{(x^{(1)}, f(x^{(1)})), (x^{(2)}, f(x^{(2)}))\cdots(x^{(k)}, f(x^{(k)}))\} \tag{3}$$

The hyperparameter configuration space and historical lists of observed variables, respectively.

The Kernel Density Estimation (KDE) method is utilized in the development of the substitute function that is produced by the TPE methodology. In opposition to the other SMBOs, which construct the surrogate function through the use of regression as the primary method, TPE builds the function through the application of a categorization strategy. The algorithm can be disassembled into its component components, which will be discussed further on in this section.

Step 1: Initialize the arrays of observed variables H by using the hyperparameters that were chosen at random. This is the very first thing that needs to be done.

Step 2: Define a model $p(x/y)$ for representing the data that is based on the ordered H with respect to y.

$$p(x|y) = \begin{cases} l(x) & if\ y < y* \\ g(x) & if\ y \geq y* \end{cases} \tag{4}$$

where

y^* - yth quantile,

$l(x)$ - x^i observations density

$f(x^i)$ - corresponding loss, and

$g(x)$ - residual observation density.

Step 3: The tree-like representation of l and g, choose a sizable group of candidates to evaluate with $g(x)/l(x)$. This should be done before moving on to step four. The software at the conclusion of each cycle, return the EI number x^* that is the highest.

Draw many candidates from the tree-structured form of, and evaluate them according to. On each iteration, the algorithm returns the with the greatest EI.

Step 4: Add x^* and $f(x^*)$ to the H after each repetition that is performed.

4. IDS FRAMEWORK

The architectural design of the system proposed in this research is detailed in Fig. 5. The first block is all about data processing. This process is often referred to as the data engineering. This step is critical for a successful learning process. Data processing has three steps, namely, cleaning, normalization and feature selection. The feature selection process is conducted using a filter-based method inspired by the XGBoost algorithm for generating feature importances scores. Once the required feature vector is selected; the next process involves model training using the training set. A trained model is then validated with the validation set. Finally, the test dataset is used to test the validated model. The procedure described above happens in an iterative manner till a tuned and fit model is found.

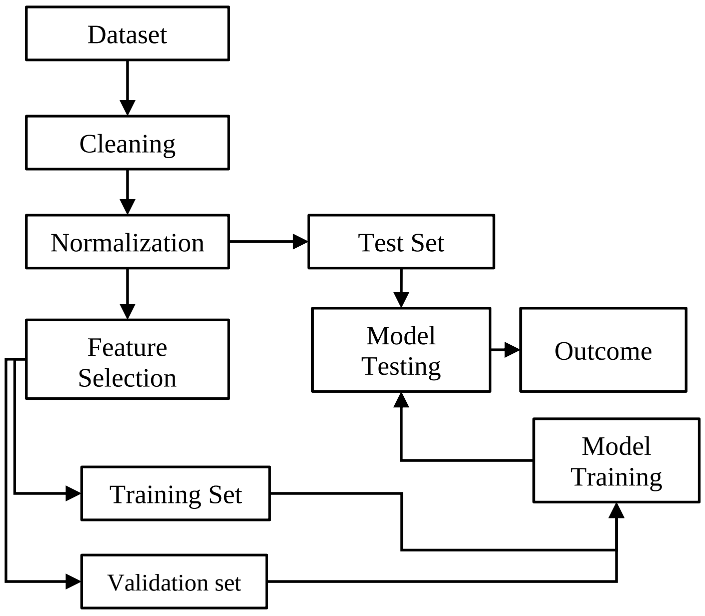

Fig.1. Proposed IDS Architecture

4.1 FEATURE NORMALIZATION

During the learning process, the high numerical value of a variety of characteristics can have an effect on a variety of machine learning algorithms, such as SVM, LR, ANN, and kNN, to name just a few of the available options. In addition to this, in order to learn about high-dimensional databases, one needs a considerable amount of computational resources. In most cases, the effects of these problems can be mitigated through the implementation of data scaling techniques. When we are dealing with the data, we scale it based on the values that are at the lowest and highest extremes. (Eq. 5).

(5)

The standardization computation will take place according to the steps specified in Algorithm 1 if a dataset is provided with an input vector (feature space) represented by $U(f_1,..., f_n)$ $1<n<N_1$, where N is the total number of instances (features) in the space. This is the case if a dataset is provided with an input vector (feature space) represented by $U(f_1,..., f_n)$.

5. DATASETS

When training a neural network, a significant amount of data is necessary because, in its absence, it is impossible to arrive at an accurate approximation of the correlation that exists between the input and the output. The process of learning through supervision is especially susceptible to having this problem occur. Unfortunately, the vast majority of IDS datasets that are openly accessible have a tendency to be quite out of date. These datasets do not provide an accurate portrayal of the traffic that occurs on contemporary networks or of the potential dangers that they pose. The solution to this problem could lay in the manual collection of data or in the modification of duplicates of previously collected datasets. Both of these options are open to consideration. It is difficult to make direct comparisons between different methods with regard to accuracy and the number of false-positive warnings generated due to the lack of a universally recognized benchmark for novel IDS implementations.

5.1 DARPA 1998 AND DARPA 1999

It is generally accepted that the availability of datasets generated by the Defense Advanced Research Projects Agency (DARPA) represents the absolute bare minimum that is necessary for public access. The construction of it made use of, among other things, data taken from the TCP/IP network as well as data taken from the operating system. The foundational security records for Solaris (root and user); Backups of the Solaris storage system.

This dataset includes information about a wide range of products, including those that are currently in working order. The period of time over which the information was collected was nine weeks, with seven weeks being devoted to the training set and two weeks being devoted to the test set. The year after DARPA 1998, which was known as DARPA 1999, came after DARPA 1998. In this particular instance, a total of 5 week worth of data was collected, with 3 week worth of data pertaining to training and 2 week worth of data pertaining to evaluation respectively. The most salient difference is the increase in the number of potential attack avenues, which is discussed further below.

Despite the fact that these datasets are frequently marketed as experimental datasets, we did not come across any innovative strategies that make use of the DARPA 1998 or 1999 datasets while conducting our investigation. This was the case despite the fact that these datasets were created in 1998 and 1999. There are a number of different potential explanations, one of which is the fact that more recent proposals are gradually replacing older datasets. This is happening because older datasets have demonstrated an inability to reproduce real network systems with sufficient precision, which is one of the reasons why more recent proposals are gradually replacing older datasets.

5.2 KDD CUP 1999

In the process of evaluating IDSs, the dataset that was displayed at the KDD Cup in 1999 is frequently used. For the objective of this endeavor, the DARPA TCP/IP dataset from 1998 is utilized. The evaluation data for the DARPA 1998 competition includes approximately 5 million submissions, while the training data for the KDD Cup 1999 includes approximately 4,900,000 connection vectors. Each vector is assigned either a normal connection status or an attack status based on the 41 individual characteristics that comprise it. This status can either be a normal connection or an attack. In addition to this, it could be any one of the four different kinds of attacks that are described below:

- *Denial of service (DOS):* The denial of service, or DOS, occurs when an adversary renders legitimate users of a service resources inaccessible by overwhelming them with an excessive number of malicious requests.
- *User to root (U2R):* The process of elevating an ordinary user on a compromised system to the root level of privileges by exploiting vulnerabilities in the system security is referred to as user to root (U2R).

- *Probe (probing):* Probing refers to the process of investigating a target or its surroundings for the purpose of acquiring information about either the target or its surroundings. Checking to see if certain ports are accessible and determining how long a connection has been established are just two instances of this.
- *Root to local (R2L):* R2L describes the process by which an adversary can acquire local capabilities on a remote computer.

The following is a list of some of the categories that can be used to categorize the 41 different characteristics:

- *Basic features*— TCP communication fundamentals
- *Content features*— Content features are features that help identify R2L and U2R attacks by characterizing invalid behaviors for single connections. These behaviors include attempts to connect to multiple hosts simultaneously. Attacks such as R2L and U2R fall under this category of behaviors.
- *Traffic features*— Characteristics of the flow of traffic that are determined by a temporal window.

Table.1. KDD Cup 1999 Features

Feature set 1	Feature set 2
Duration	Is_hot_login
Protocol_type	Is_guest_login
Service	Count
Flag	Srv_count
Src_bytes	Serror_rate
Dst_bytes	Srv_serror_rate
Land	Rerror_rate
Wrong_fragment	Srv_rerror_rate
Urgent	Same_srv_rate
Hot	Diff_srv_rate
Num_failed_logins	Srv_diff_host_rate
Logged_in	Dst_host_count
Num_compromised	Dst_host_srv_count
Root_shell	Dst_host_same_srv_rate
Su_attempted	Dst_host_diff_srv_rate
Num_root	Dst_host_same_src_port_rate
Num_file_creations	Dst_host_srv_diff_host_rate
Num_shells	Dst_host_serror_rate
Num_access_files	Dst_host_srv_serror_rate
Num_outbound_cmds	Dst_host_rerror_rate

The KDD Cup 1999 had more than its share of issues, despite the abundance of information and the widespread attention that surrounded it. Some of them were carried over from the DARPA 98 dataset, such as the fact that it was entirely synthetic and there was no investigation into the possibility of lost packets while it was being produced. Other of them were carried over from the DARPA 98 dataset. The data for the rest of them was brought over from the DARPA 98 collection. The KDD Cup 1999 has issues with repetitive records and an uneven distribution of violations, both of which are problems in and of themselves, and both of these issues are contributing factors to the overall problem.

5.3 NSL-KDD

The problems that were found in the DARPA and KDD Cup 1999 datasets prompted the creation of a new dataset known as NSL-KDD. This dataset was developed in order to resolve the problems that were found in those datasets. There are no occurrences of duplicate entries found in either the training or the testing groups. It is not necessary to generate test-specific subsets of the dataset as the number of records in the dataset can be readily managed due to this fact. The percentage of the original KDD Cup 1999 dataset that is made up of particular records is inversely proportional to the number of records that are included in each complexity level category of the competition. In spite of the fact that NSL-KDD has quite a few shortcomings, the quality of the program as a whole makes it suitable for use in IDS benchmarking.

5.4 UNSW-NB15

After making extensive use of the databases from both the KDD Cup 1999 and the NSL-KDD, the following challenges were found to exist: In some domains, our understanding is deficient, specifically in regards to the characteristics of low-footprint attacks, transportation patterns, and the dissemination of particular data sets.

The UNSW-NB15 algorithm came into existence as a direct result of these worries. UNSW-NB15 includes 49 different components. The information that has been provided to you includes two distinct attributes: label, which can either be 0 or 1, indicating that everything is normal; and attack_cat, which identifies the type of attack that was carried out; both of these attributes can be found in the information that has been provided to you.

Table.2. UNSW-NB15 features

Feature set 1	Feature set 2
srcip	sjit
sport	djit
dstip	stime
dsport	ltime
proto	sintpkt
state	dintpkt
dur	tcprtt
sbytes	synack
dbytes	ackdat
sttl	is_sm_ips_ports
dttl	ct_state_ttl
sloss	ct_flw_http_mthd
dloss	is_ftp_login
service	ct_ftp_cmd
sload	ct_srv_src
dload	ct_srv_dst

spkts	ct_dst_ltm
dpkts	ct_src_ ltm
swin	ct_src_dport_ltm
dwin	ct_dst_sport_ltm
stcpb	ct_dst_src_ltm
dtcpb	attack_cat

5.5 KYOTO2006+

The Kyoto 2006+ dataset is yet another example of benchmark data that can be readily accessed and utilized for IDS training and testing without incurring any costs whatsoever. There are twenty-four distinct characteristics of networks that are described in this article; all of these characteristics were taken from the computers at Kyoto University. 14 of the components were taken from the KDD Cup 1999, while the other ten are completely original. The collection of information took place between the years 2006 and 2009. It was designed in order to assume the role of the 1999 KDD Cup, which it did successfully. In addition, a description of the benchmark edition, which comes packed with seventeen different features, is included. (14 derived from the KDD Cup of 1999 and 3 additional).

5.6 RESULTS AND DISCUSSION

LGBM_TPE, LGBM, LogReg, GBDT and XGBDT are just some of the examples of examples of machine learning (ML) techniques that were taken into consideration during each of the two different stages that made up our experimentation plan. Other examples include LGBM_TPE, LGBM, LogReg, GBDT and XGBDT. These phases were split apart by a distinct phase break that occurred in between them. In the first phase of the experiments, we used the full feature space of the UNSW-NB15, which comprised of 42 different characteristics for both the binary and the multiclass configurations. This was done so that we could evaluate the accuracy of the classification.

Table.3. Average Values of all the datasets

Model	Accuracy	Recall	Precision	f1-Measure
LGBM_TPE	0.9091	0.9014	0.9156	0.9085
LGBM	0.8881	0.8605	0.9116	0.8853
LogReg	0.8193	0.8198	0.8189	0.8194
GBDT	0.8438	0.8280	0.8553	0.8414
XGB	0.9630	0.8196	0.9011	0.8584

Table.4. Performance of DARPA 1998 AND DARPA 1999

ML method	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
LGBM_TPE	95.46	95.17	87.60	82.37	99.06	89.95
LGBM	94.17	93.82	80.40	74.07	99.95	85.08
LogReg	97.75	94.56	84.03	79.96	95.26	86.94
GBDT	71.71	71.35	63.06	61.53	89.49	71.91
XGB	94.61	94.33	89.03	84.77	97.46	90.92

Table.5. Performance of KDD CUP 1999

ML method	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
LGBM_TPE	95.55	95.27	87.68	82.46	99.16	90.04
LGBM	94.27	93.91	80.48	74.14	99.95	85.17
LogReg	97.85	94.65	84.11	80.04	95.36	87.03
GBDT	71.78	71.42	63.12	61.59	89.57	71.98
XGB	94.70	94.42	89.12	84.85	97.55	91.01

Table.6. Performance of NSL-KDD

ML Method	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
LGBM_TPE	96.87	96.51	91.67	96.89	91.66	92.84
LGBM	91.54	89.63	79.44	93.24	79.45	80.76
LogReg	99.11	93.14	84.97	91.88	85.12	87.32
GBDT	64.77	63.85	74.06	57.55	75.16	65.19
XGB	94.18	93.81	80.05	96.77	80.06	61.97

Table.7. Performance of UNSW-NB15

ML Method	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
LGBM_TPE	96.33	95.66	93.97	96.38	93.99	93.69
LGBM	87.93	87.06	79.15	85.93	79.15	79.96
LogReg	95.21	96.83	87.65	93.64	87.65	89.48
GBDT	64.98	64.22	74.59	65.40	74.58	62.20
XGB	95.47	95.08	81.92	96.57	81.90	83.96

Table.8. Performance of KYOTO2006+

ML Method	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
LGBM_TPE	97.18	96.51	94.79	97.23	94.82	94.51
LGBM	88.70	87.82	79.85	86.69	79.85	80.67
LogReg	95.09	97.68	88.42	94.46	88.42	90.27
GBDT	65.55	64.78	75.25	65.98	75.24	62.75
XGB	96.31	95.92	82.64	97.42	82.63	84.70

The XGBoost technique for selecting features was incorporated into the process during the second iteration. Following this, a reduced feature vector was constructed, which includes 19 distinct characteristics, as shown in Table.1 and Table.2. We ran experiments using the binary classification stage as well as the multiclass classification stage while utilizing the optimal feature vector.

In Table.3-Table.8, we can see the results obtained by the ML algorithms when the binary classification method was applied to the decreased feature vector that was obtained from the previous table. This was done in order to ensure that the findings are as accurate as possible. The outcomes of the ML algorithm attempt at multiclass categorization are presented in the Table.3-Table.7 that can be found further down the page.

These tables show the results that were obtained when the entire feature space was used in comparison to the reduced feature vector. AC appears in each picture to denote the accuracy of the training data. The precision obtained through the process of validation is denoted.

The proposed IDS experiments all made use of the Adam solver, which is a stochastic gradient-based technique that functions exceptionally well with massive datasets. This was done in place of the more conventional stochastic gradient descent.

The speed at which the instruction was given could be changed as necessary. According to Table.4, the proposed IDS network that had the best overall performance used 150 neurons and an adaptive learning rate of 0.02 in order to achieve an accuracy rate of 87% over the test set. This was accomplished by using a neural network that had the greatest overall performance.

In spite of the fact that the LR technique was capable of carrying out a maximum of one thousand iterations, the number ten was selected for the random state. The results of conducting binary classification on the test set using either the entire feature space or just a subset of that space are presented in Table.4 and Table.5, respectively. In terms of accuracy, the full feature space generated rates of 80%, while the subset produced rates of 78%.

According to the findings, a kNN classifier with three neighbors that utilized the entirety of the feature space was able to achieve a test score of 84% in a multiclass classification scenario. This score was achieved by successfully classifying the data. In the case of the reduced feature dimension, the kNN technique achieved an accuracy of 85% when it was given 9 neighbors and was not permitted to overfit the data. This was the situation when it was not allowed to overfit the data.

We put the DT classifier through its tests by employing a variety of models, some of which were based on the height of the branches at their broadest point, in order to evaluate its performance. The range that is provided by the maximum_depth_values parameter indicates the stages that could be used in the game. According to the findings, the DT was able to achieve a test score of 85% when it was used with 19 features, but it was only able to achieve an accuracy of 88% when it was used with 42 features in a binary categorization scenario. When the binary classification was performed using both the full feature dimension and the reduced one, the DT achieved a higher test accuracy score than the other ML techniques did. This was the case when both the complete feature dimension and the reduced one were used.

6. CONCLUSION

Research into this area has become one of the top priorities of the growing importance of using deep learning algorithms in practical applications. Utilizing numerous deep networks, which are part of the larger category of methods known as deep learning, is one way to boost the efficiency of IDSs. Deep learning models have substantially better fitting and generalization abilities when compared to more surface-level machine learning models. Deep learning techniques, on the other hand, don't need any feature engineering or domain knowledge, in contrast to shallow machine learning models. This is in stark contrast to the requirement for such knowledge in shallow machine learning models. Nevertheless, in order for deep learning models to fulfill the real-time requirements of IDSs, the execution of these models typically takes an inordinate amount of time.

REFERENCES

[1] Jiankun Hu, Xinghuo Yu, D. Qiu and Hsiao-Hwa Chen, "A Simple and Efficient Hidden Markov Model Scheme for Host-Based Anomaly Intrusion Detection", *IEEE Network*, Vol. 23, No. 1, pp. 42-47, 2009.

[2] K.K. Gupta, and R. Kotagiri, "Layered Approach Using Conditional Random Fields for Intrusion Detection", *IEEE Transactions on Dependable and Secure Computing*, Vol. 7, No. 1, pp. 35-49, 2010.

[3] S. Devaraju and S. Ramakrishnan, "Performance Analysis of Intrusion Detection System using Various Neural Network Classifiers", *Proceedings of International Conference on International Conference on Recent Trends in Information Technology*, pp. 1033-1038, 2011.

[4] Mendonça, R. V., Teodoro, A. A., Rosa, R. L., Saadi, M., Melgarejo, D. C., Nardelli, P. H., & Rodríguez, D. Z. (2021). IDS based on fast hierarchical deep convolutional neural network. *IEEE Access*, *9*, 61024-61034.

[5] Neveen I. Ghali, "Feature Selection for Effective AnomalyBased Intrusion Detection", *International Journal of Computer Science and Network Security*, Vol. 9, No. 3, pp. 285-289, 2009.

[6] R. Plutchik, "*Emotion: Theory, Research, and Experience*", Academic Press, 1980.

[7] P.R. Kanna and P. Santhi, "Unified Deep Learning Approach for Efficient IDS using Integrated Spatial-Temporal Features", *Knowledge-Based Systems*, Vol. 226, pp. 107132-107143, 2021.

[8] H. Hindy, E. Bayne and M. Bures, "Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study", *Proceedings of International Conference on Network*, pp.1-14, 2020.

[9] M. Zhou, L. Han, H. Lu and C. Fu, "Intrusion Detection System for IoT Heterogeneous Perceptual Network", *Mobile Networks and Applications*, Vol. 33, No. 1, pp. 1-14, 2020.

[10] L. Xiao, X. Wan, X. Lu and Y. Zhang, "IoT Security Techniques based on Machine Learning: How do IoT Devices use AI to Enhance Security?", *IEEE Signal Processing Magazine*, Vol. 35, No. 5, pp. 41-49, 2018.

[11] B. Gobinathan and V.P. Sundramurthy, "A Novel Method to Solve Real Time Security Issues in Software Industry using Advanced Cryptographic Techniques", *Scientific Programming*, Vol. 2021, pp. 1-9, 2021.

[12] Z.K. Maseer, "Benchmarking of Machine Learning for Anomaly Based IDSs in the CICIDS2017 Dataset", *IEEE Access*, Vol. 9, pp. 22351-22370, 2021.

[13] X. Li and L. Wu, "Building Auto-Encoder IDS based on Random Forest Feature Selection", *Computers and Security*, Vol. 95, pp. 101851-101865, 2020.

[14] T. Saba and S.A. Bahaj, "Anomaly-based IDS for IoT Networks through Deep Learning Model", *Computers and Electrical Engineering*, Vol. 99, pp. 107810-107818, 2022.

[15] R. Ferdiana, "A Systematic Literature Review of IDS for Network Security: Research Trends, Datasets and Methods", *Proceedings of International Conference on Informatics and Computational Sciences*, pp. 1-6, 2020.