Complex & Intelligent Systems (2023) 9:5693–5714
https://doi.org/10.1007/s40747-023-01013-7

ORIGINAL ARTICLE

A novel ensemble learning-based model for network intrusion detection

Ngamba Thockchom[1] · Moirangthem Marjit Singh[1] · Utpal Nandi[2]

Received: 2 August 2021 / Accepted: 17 February 2023 / Published online: 3 April 2023

Abstract
The growth of Internet and the services provided by it has been growing exponentially in the past few decades. With such growth, there is also an ever-increasing threat to the security of networks. Several efficient countermeasures have been placed to deal with these threats in the network, such as the intrusion detection system (IDS). This paper proposes an ensemble learning-based method for building an intrusion detection model. The model proposed in this paper has relatively better overall performance than its individual classifiers. This ensemble model is constructed using lightweight machine learning models, i.e., Gaussian naive Bayes, logistic regression and decision tree as the base classifier and stochastic gradient descent as the meta-classifier. The performance of this proposed model and the individual classifiers used to build the ensemble model is trained and evaluated using three datasets, namely, KDD Cup 1999, UNSW-NB15 and CIC-IDS2017. The performance is evaluated for binary class as well as multiclass classifications. The proposed method also incorporates the usage of a feature selection method called Chi-square test to select only the most relevant features. The empirical results definitively prove that using an ensemble classifier can be immensely helpful in the field of intrusion detection system with unbalanced datasets where misclassifications can be costly.

Keywords Network intrusion detection · Ensemble learning · Machine learning

Introduction

With the increasing number of users and their dependence on the Internet, there has been an increase of attacks in the network that disrupts its normal functioning. Many different types and scales of attacks are on the rise. Security policy, antiviruses and firewalls are no longer enough to protect the network. A system that is designed to protect against targeted attacks by constantly monitoring traffic of the network is the network intrusion detection system (NIDS). Such systems analyze the data and trys to identify any anomaly in the network such as unauthorized access, alteration, damage or intrusions. The intrusion detection system (IDS) is usually employed as a software application to monitor the network and the system activities for malicious behaviors. It detects unauthorized access to the system and then reports it to administrators of the network. Various machine learning (ML) methods have been considered for building an IDS. ML algorithms are applied in IDS to identify and classify the security threats. The ML algorithm looks for patterns in the network traffic and associates it with various predefined network traffic profiles and thus helps in identifying whether a particular network traffic instance is an intrusion or not. Such approaches that use ML to build an IDS are implemented using a single classifier or hybrid classifier system. Single classifier-based systems use single ML algorithm to identify anomalies, whereas a hybrid system utilizes a combination of more than one ML algorithm, i.e., using one classifier for pre-processing the data and other to train the pre-processed data.

✉ Moirangthem Marjit Singh
marjitm@gmail.com

Ngamba Thockchom
thockchomngamba@gmail.com

Utpal Nandi
nandi.3utpal@gmail.com

1 Department of Computer Science and Engineering, North Eastern Regional Institute of Science and Technology, Arunachal Pradesh, India

2 Department of Computer Science, Vidyasagar University, West Bengal, India

This paper proposed the adoption of an ensemble approach. An ensemble learner is defined as the combination of the outputs of several learners into a single output. There are three main reasons to use an ensemble method as highlighted by Dietterich [1]. They are: (i) statistical reason: a situation where there is limited data to find the best hypothesis in the search space; (ii) computational reason: many machine learning algorithms may enter into repeated process for finding better hypothesis at the local optima; (iii) representational reason: many machine learning algorithms do not deliver the required solution domain independently and hence ensemble methods will be of better choice. Apart from the lack of datasets that represent the diverse set of real-world attacks, there is another challenge unique to IDS, i.e., systems having low detection rate and a high false positive rate can be detrimental to the network. Ensemble learning is especially suited to address these issues. Ensemble methods are also regarded as multiclassifier system that can leverage the unique properties of different classifiers and overcomes the disabilities of a single classifier, thereby diversifying the range of detection possible [26]. The ensemble method integrates multiple ML algorithms to arrive at a final solution. It is robust and less prone to bias and variance compared to the non-ensemble approach. Hence, the ensemble approach is better suited than the non-ensemble approach.

The proposed model in this paper is a combination of well-known and standard ML techniques, namely, Gaussian naive Bayes, logistic regression and decision tree and stochastic gradient descent. The main contributions of the paper include: i. a combination of learning algorithms using stacking ensemble paradigm; ii. improving the performance of IDS compared to other existing methods; and iii. the proposed method can be effectively used with several benchmark datasets. This combination of classifiers allows the result to be aggregated, thereby increasing the predictive performance. To demonstrate the proposed model's performance and its capabilities in identifying anomalies in a wide range of network traffic, three datasets, namely, KDD99, UNSW-NB15 and CIC-IDS2017 are taken for evaluation. Table 1 describes the dataset files used. The proposed model also incorporates a feature selection technique called Chi-square test. Although there is no particular ensemble method that works better than others in all circumstances, the method proposed in the paper offers a unique method to deal with IDS.

The rest of the paper is structured as follows: the next section outlines the related work; "Proposed model" contains a detailed description of the proposed model, followed by the section that contains the experimental results and discussions; the final section presents the conclusion and the references are given at the end.

Related work

Robin Sommer et al. carried out investigations on using machine learning for detection of network intrusions and explored the differences between areas where machine learning has been successfully applied in intrusion detection [2]. They stated that anomaly detection-based systems are more suitable for detecting new attacks and finally put forward some suggestions to strengthen future research in intrusion detection. Alireza Suri et al. conducted a systematic review of the literature on the latest malware detection technology that uses the data mining technique [3]. Using the 1999 DARPA IDS data set, Ke Wang et al. [4] described a payload-based system for detecting abnormal networks. Avinash R. Sonule et al. tested various datasets including UNSW-NB15 and various methods of network intrusion detection [5]. Their study shows that no specific attack detection technique can detect all types of attacks and the selection of features and multiple classifiers affects IDS performance. A data mining technique called core vector machine (CVM) was proposed by Divyasree Th et al. [6]. This method, which was an improved version of SVM, was trained and evaluated on the KDD-Cup'99 dataset giving desirable results in terms of detection rate and false positive rate. An ensemble system to improve accuracy was created using the CVM. An ensemble method for intrusion detection based on Bayesian networks and random trees was proposed by Ying Wang et al. [7]. Their model was evaluated using the KDDCup99. Gulshan Kumar et al. [8] analyzed and compared a complete overview of existing IDS based on ensemble learning, either homogeneous or heterogeneous. It was revealed that most studies were validated using the KDD dataset only. Hariharan Rajadurai et al. proposed a method based on ensemble learning with stacking and evaluated the NSL-KDD data set [9]. The method was a combination of gradient descent and random forest (RF) algorithm. Smith Rajagopal et al. used two different datasets: UNSW NB-15 and UGR'16 in their experimentation and showed that the combination of RF, logistic regression, kNN and SVM provided better predictions [10]. Sabhnani and Serpen used k-means clustering, multilayer perceptron and Gaussian classifiers to predict the attack classes of KDD'99 [11]. An ensemble approach that used classification and regression trees (CARTs), Bayesian networks (BNs), was proposed by Chebrolu et al. [12]. An ensemble classifier that uses decision tree (DT), SVM and their hybrid system was presented by Abraham and Thomas [13]. An aggregation of different categories of classifiers, SVM, MARS, ANN and its variants, was proposed by Mukkamala et al. and it was validated on the DARPA 1998 dataset [14]. Zainal et al. [15] gave an ensemble classifier that used linear genetic programming, adaptive neural-fuzzy inference system and RF. A meta-learning-based system using naive Bayes, C4.5 decision trees, VFI-voting feature intervals, kNN clustering

Table 1 Description of datasets

Dataset	Dataset file	Label (s)	Training set	Testing set
KDD'99	'kddcup.data_10_percent'	Normal, DoS, R2L, U2R, Probe	116,468	29,118
UNSW-NB15	'UNSW_NB15_training-set' + 'UNSW_NB15_testing-set'	Normal, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms	206,138	51,535
CIC-IDS2017	Friday-WorkingHours-Afternoon-DDos.pcap_ISCX + Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX + Friday-WorkingHours-Morning.pcap_ISCX + Monday-WorkingHours.pcap_ISCX + Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX + Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX + Tuesday-WorkingHours.pcap_ISCX + Wednesday-workingHours.pcap_ISCX	BENIGN, DoS_Hulk, Bot, DDoS, PortScan, FTPPatator, DoS_GoldenEye, Web_Attack_Brute_Force, SSHPatator, DoS_slowloris, DoS_Slowhttptest, Web_Attack_XSS, Infiltration, Web_Attack_Sql_Injection, Heartbleed	1,940,581	485,146

Table 2 Summary of significant related studies in chronological order

Author	Year	Proposed model	Remarks
[4]	2004	Payload-based anomaly detector	Method relies heavily on profiling expected normal payloads
[14]	2005	Ensemble of artificial neural network, support vector machine and multivariate adaptive regression spline	Ensemble was shown to outperform the individual classifiers only. This was already expected. Performance improvement as also noted by the authors is not statistically significant
[15]	2009	An ensemble of different learning paradigms, which uses weighted voting to decide. Linear genetic programming, adaptive neural-fuzzy inference system and random forest were used	Performance was significantly influenced by the assignment of weights to the individual classifiers
[7]	2016	Voting system with Bayesian network and random tree as base classifier, random committee as meta-learning algorithm	Accuracy rate for some categories was too low to be considered for building real-world IDS
[6]	2018	An ensemble core vector machine, Chi-square test for feature selection and a weighted function for dimensionality reduction	The model took lesser training and testing time compared to support vector machine (SVM)
[10]	2020	Stacking model that utilizes random forest, logistic regression, k-nearest neighbor and support vector machine	The model tested on two datasets, UNSW NB 15 and UGR'16. It performed better on UGR'16 dataset than UNSW NB 15
[20]	2020	Focus on feature selection using genetic algorithm and logistic regression. Three decision tree-based models were used for classification	Highlighted the influence of selecting the right subset of features that can have on the final outcome

and OneR was proposed by Menahem et al. [16]. Meng and Kwok [17] used an ensemble of classifiers such as nearest neighbor, DT and SVM and the results were validated on the DARPA 1998 dataset. Haq et al. [18] used Bayesian network, naive Bayes and decision tree-based J48 classifier to produce an ensemble, a hybrid method based on BFS, genetic algorithm and ranking search methods used for feature selection of the KDD'99 dataset. Tama and Rhee [19] used RF, C4.5 and CART; they also used PSO and CFS algorithms for feature selection on the KDD'99 dataset. Khammassi and Krichen [20] used genetic algorithm and DT; the genetic algorithm was used as a search strategy and DT for classification. Hariharan Rajadurai and Usha Devi Gandhi [9] proposed an ensemble method based on stacking, which was a combined approach of gradient descent and RF algorithms and evaluated on the NSL-KDD dataset. A flow-based machine learning model that monitors the network behavior for anomaly was presented by N.Satheesh et al. [21]. A novel wide and deep transfer learning stacked GRU framework proposed by N B Singh et al. was evaluated on the KDDCup99 and UNSW-NB15 dataset with an accuracy score of 99.92% and 94.22%, respectively [22]. A crow search optimization algorithm with adaptive neuro-fuzzy inference system (CSO-ANFIS) was presented by S Manimurugan et al. and validated using the NSL-KDD dataset. A 95.80% detection rate was achieved [23]. Table 2 highlights some of the significant related works that have influenced this paper. Table 2 shows in chronological order to help visualize how the research works have evolved over the years.

Proposed model

A relatively lightweight ensemble approach is proposed here that focuses on improving intrusion detection accuracy performance. This approach is based on the use of stacking ensemble technique. The proposed model was trained and evaluated on three different datasets considering both binary class and multiclass classifications. The stacking model presented in this paper was developed using a two-level classification system: a set of base classifiers (Level 0 classifier) and a single meta-classifier (Level 1 classifier) that combines the results of these base classifiers. The base classifiers use the complete training dataset for training. The training data is input into each base classifiers and then trained using the training data, the result of which will later serve as the input for the meta-classifier. One of the many paradigms in building an ensemble model is choosing the base classifiers; instead of choosing the best performing base classifiers, three relatively weak classifiers were chosen. The base classifiers chosen for building the stacking ensemble were Gaussian naive Bayes (GNB), decision tree (DT) and logistic regression (LR). To integrate all of the outputs of the base classifiers into a single one, a meta-classifier called stochastic gradient descent (SGD) was used. The meta-classifier is responsible for determining the final prediction, i.e., it uses the output of all the base classifiers to decide the prediction after training. Firstly, the datasets are split into two sets i.e., 'training set' and 'testing set'. The proposed model is trained only with the training set and later its performance is tested and evaluated using the testing set. The splitting of the dataset helps in simulating the situation for testing the trained model using new unseen data. The dataset is divided into two sets by randomly sampling it without replacement. The training set consists of 80% of the rows and the testing set consists of 20% of the rows. The complete training dataset was then used to train the base classifiers, and the cross-validation predictions of the base classifiers were used to train the meta-classifier. This was completed to minimize the possibility of errors or misclassification so that the result will be optimal. Traditional one-class classifiers may fail in some training datasets, but the proposed model may not fail even when one classifier does not work with the dataset since another classifier might still work and hence correctly detect the intrusion.

The block diagram describing various stages of the proposed model is shown in Fig. 1. The first part is data pre-processing. Python Pandas library was used to access the dataset CSV (comma separated value) file as a data frame. The ensemble was created using modules available in the scikit-learn library.

Pre-processing

The first step was pre-processing of the data. Preliminary data analysis was performed to ensure that the file was suitable for use as input to an ML model. The dataset contains information of network traffic and had features such as IP address, protocol type and duration, depending on the dataset it consists of mostly integer (int64) and floating point (float64) values as well as some string or object values. The categorical values were transformed into numerical values. The label indicated whether the connection was malicious or normal network traffic. This was useful in training a supervised ML model to classify based on the features of the attack. Thereafter, the data was scanned for redundant data, missing values, null values or NaN values. These values can interfere in training the model. Some of the other steps completed before the data was ready for training and testing included feature scaling and label encoding. In feature scaling, the scaling of the features in the dataset was carried out to maintain a similar scale. Most of the data in the datasets have varying degrees of magnitude and range. Feature scaling process failure can cause the model to make incorrect predictions. The MinMaxScaler method was used to scale the features. The values were scaled in the range [0.1] and was determined using Eq. (1):

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}, \tag{1}$$

where x is the initial value and x' is the normalized value.

The next important step in the proposed model was to use the selection of features function. Selecting only a few features means the number of input functions will decrease. Feature selection was done to reduce the cost of training the model. The method uses Chi-Square test to select the features. Chi-square test is used in statistics to test the independence of two events. It is based on null hypothesis testing

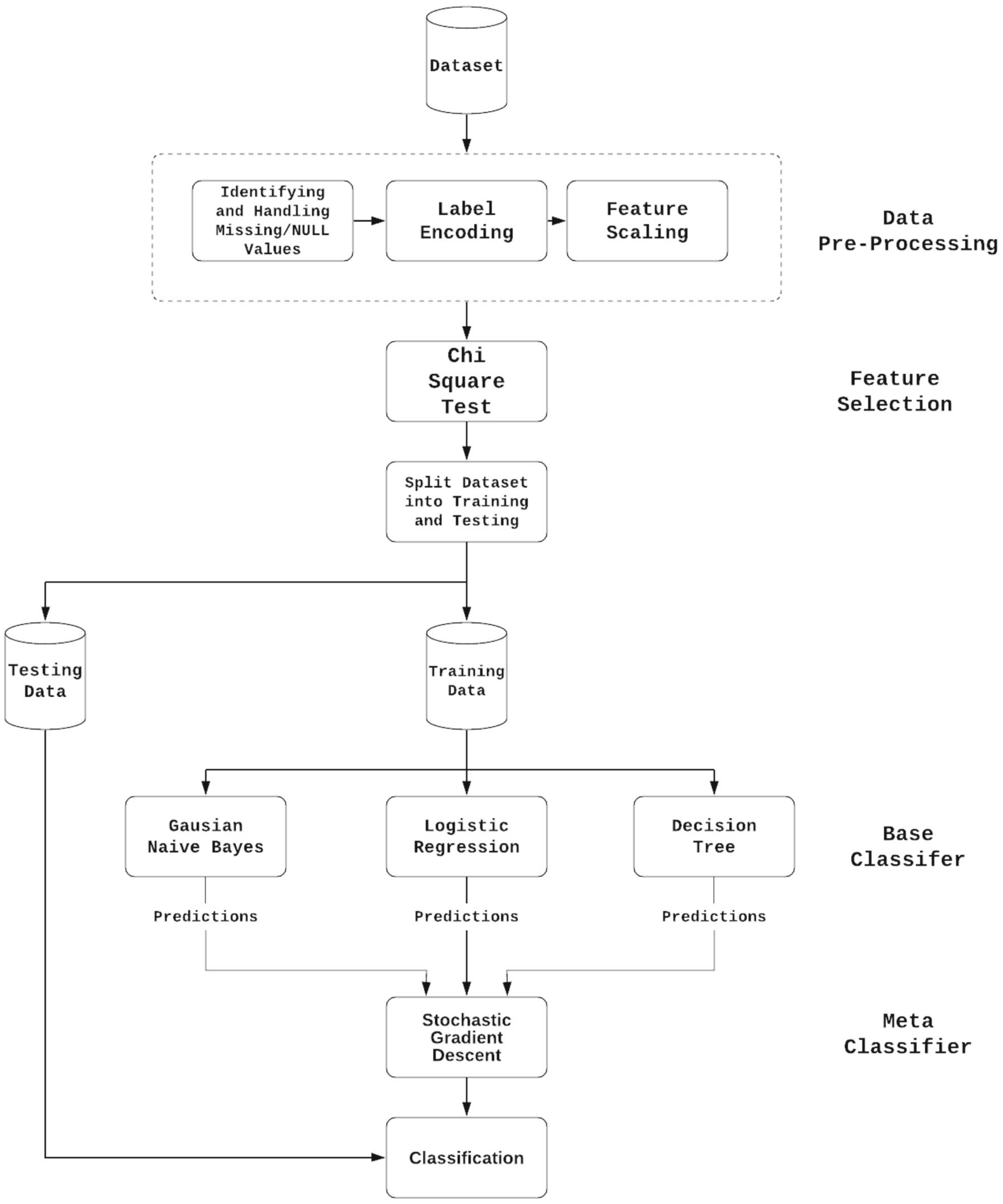


Fig. 1 Proposed model

i.e., assuming two variables are independent. Here, the Chi-square test was used to calculate the correlation coefficient for each feature of the dataset. This coefficient represents the level of influence a particular feature has on the final output label. A higher Chi-square coefficient value implies that the value of the label is more dependent on that feature and thus features that have higher chi-square coefficients are selected.

To test the correlation between a feature column and the output label using the Chi-square test, a contingency table is built first. A Chi-square score is given by Eq. (2):

$$x^2 = \sum \frac{(O_i - E_i)^2}{E_i}, \tag{2}$$

where O_i is the number of observed value(s) of a class and E_i is the number of expected value(s). The observed value is the original value of a cell in the contingency table. Its expected value is calculated by multiplying the row total (sum of all values of the row of that cell) and the column total (sum of all values of the column of that cell) and dividing it by the sum of all the observed values of the contingency table. If a feature and the output label are independent, then O_i and E_i values will be nearly same and hence results in a lower Chi-square score. Otherwise, the Chi-square score will be high.

In the proposed model, every dataset after pre-processing is processed through the feature selection process and, based on the Chi-square score, the selected features for each dataset are given in Table 3.

For training the data, a two-level stacking ensemble was used. The base classifiers considered the entire input set for every individual classifier, which allows the classifiers with different hypotheses and algorithms to work independently; the meta-classifier used the cross-validated predictions of the base classifiers for training. After pre-processing, the data was fed into the proposed model. The algorithm of the proposed stacking model is given in Algorithm 1.

event has occurred. Bayesian reasoning is expressed mathematically by Eq. (3):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{3}$$

The Bayes' theorem can be applied in a dataset in the following manner as given in Eq. (4):

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}. \tag{4}$$

Here, y is the label of the class and X is its defining features. The naive assumption to the Bayes' theorem is the independence among the features. For any two independent events A and B, this can be written as Eq. (5):

$$P(A, B) = P(A)P(B). \tag{5}$$

Hence, the same can be written as Eq. (6):

Algorithm 1: Proposed Stacking Model

```
Input: Training Data
Output: Classifier $C$
1.  Step 1: Perform Chi Square test on $D$
2.  Step 2: Select 13 features with highest correlation coefficient
3.  Step 3: Train Base Classifiers
4.  for $t \leftarrow 1$ to $T$ do
5.          Train base classifiers $c_t$ namely Gaussian Naive Bayes, Decision Tree, Logistic Regression on $D$
6.  end for
7.  Step 4: Construct Training dataset for meta classifier
8.  for $i \leftarrow 1$ to $m$ do
9.          Construct a new dataset with instance $(x'_i, y_i)$, where $x'_i = \{c_1(x_i), c_2(x_i), \ldots, c_T(x_i)\}$
10. end for
11. Step 5: Train Meta classifier $c'$
12.         Train Stochastic Gradient Descent classifier $c'$ on the newly constructed data set
13. return $C(x) = c'(c_1(x), c_2(x), \ldots, c_T(x))$
```

Base classifiers

Gaussian naive Bayes (GNB) The Gaussian naive Bayes classifier and other naive Bayes classifiers are a group of classifiers that predict a class of unknown datasets based on Bayesian probability theorem. These groups of classifiers all have a common principle, that is, each pair of classification features are independent of each other. Bayes' theorem determines the probability of one event occurring when another

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}. \tag{6}$$

To construct a model for classification for all possible values of a y variable class, determine the probability of a given set of input data and then take the output with the maximum probability. Mathematically, it is expressed as Eq. (7):

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y). \tag{7}$$

Table 3 Selected dataset features

Dataset	Selected features
KDD'99	Hot, logged_in, root_shell, num_outbound_cmds, is_host_login, count, srv_count, rerror_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_same_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rat
UNSW-NB15	dur, proto, service, state, spkts, dpkts, sbytes, dbytes, rate, sttl, dttl, sload, dload, sloss, dloss, sinpkt, dinpkt, sjit, djit, swin, stcpb, dtcpb, dwin, tcprtt, synack, ackdat, smean, dmean, trans_depth, response_body_len, ct_srv_src, ct_state_ttl, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd, ct_src_ltm, ct_srv_dst, is_sm_ips_ports
CIC-IDS2017	FlowDuration, BwdPacketLengthMax, BwdPacketLengthMean, BwdPacketLengthStd, FlowIATMax, FwdIATTotal, FwdIATStd, FwdIATMax, PacketLengthStd, FINFlagCount, IdleMean, IdleMax, IdleMin

Only a small quantity of trained data is needed to evaluate the test data, which is why the training time of the Gaussian naive Bayes classifier is short. The main drawback of the naive Bayes approach is to assume independent predictive features: it is assumed that all features are unbiased of each other, but this is not always the case in real-world data.

Decision tree (DT) Decision tree is a supervised learning algorithm; it is a tree structure used to classify each input vector X; each node within the tree is a simple comparison of attributes and fields; whether the result of each comparison is true or false is used for determining whether the left child or the right child of a given specific node should be traversed. The best attributes for the root node or the child node are selected using an attribute selection measure or ASM. Using this, the best attribute for each node of the tree is selected. There are two well-defined methods for this: information gain and Gini index. For the DT classifier used in this paper, the Gini index function is used. Gini index measures the degree or probability of being misclassified when a certain variable is randomly selected. This is depicted in Eq. (8).

$$Gini = 1 - \sum_{i=1}^{n} (P_i)^2, \tag{8}$$

where P_i is the probability that the object will be assigned to a specific class.
Decision trees are considered non-parametric and thus there are no assumptions. It can handle categorical as well as numerical data and are not affected by outliers. The cost of using the tree (i.e., predictive data) is logarithmic of the number of data points used to train the tree. One disadvantage of using a decision tree is that it is biased; if some classes have a higher number of samples, then it creates biased trees. It tends to overfit the data with a large feature space.

Logistic regression (LR) This is a machine learning predictive analysis classification algorithm based on the concept of probability. It uses a cost function called the sigmoid function. This function maps each real value to a different value between 0 and 1. In classification problems, this is used to map predictions to probabilities. The hypothesis expectation of logistic regression can be defined as given in Eq. (9).

$$0 \le h_0(x) \le 1. \tag{9}$$

Logistic regression is a fast and non-resource-intensive algorithm; it can scale and perform well while dealing with large datasets. It runs on a linear model and hence performs poorly in nonlinear problems. Data scaling, normalization and checking for missing values will be a requirement.

Meta-classifier

Stochastic gradient descent (SGD) The stochastic gradient descent classifier is a variant of gradient descent. Gradient descent is a machine learning optimization technique that uses gradient (the slope of a function) to measure the response speed of one variable to changes into another variable. The algorithm runs iteratively from the initial value to find the best parameter value and find the smallest possible values of a given cost function. In a traditional gradient descent algorithm, the whole dataset is used for calculating the gradient for all iterations and has to be repeated until the minima is reached, making it computationally expensive. Stochastic gradient descent selects randomly shuffles samples from the dataset for all iterations instead of using the entire dataset. Due to the randomness in the descent, SGD takes a higher number of iterations to reach the minima, but still is less computationally intensive as compared to traditional gradient descent.

Experimental results and discussions

Three datasets, namely, KDD'99, UNSW-NB15 and CICIDS2017, were used to evaluate the proposed model. This section presents the results obtained from the proposed stacked ensemble model along with the results of the individual classifiers. The proposed model is trained in both a

Table 4 KDD'99 performance matrix result (binary classification)

Classifier	Data type	Precision	Recall	F1 score	Accuracy (%)
GNB	Normal	0.9839	0.9523	0.9678	97.47
	Intrusion	0.9690	0.9897	0.9792	
LR	Normal	0.9884	0.9415	0.9644	97.22
	Intrusion	0.9624	0.9927	0.9773	
DT	Normal	0.9884	0.9834	0.9859	98.87
	Intrusion	0.9890	0.9923	0.9907	
SGD	Normal	0.9863	0.9403	0.9627	97.09
	Intrusion	0.9616	0.9913	0.9762	
Proposed model	Normal	0.9981	0.9979	0.9980	99.84
	Intrusion	0.9986	0.9987	0.9987	

binary and multiclass classification manner and the performance results are presented for the model under different dataset environments.

Binary classification result

In this section, the experimental results for a binary classification tasks are presented. The proposed stacked ensemble model was evaluated using standard ML performance metrics, i.e., precision, recall, F1 score, accuracy and confusion matrix. The binary classification results of the classifiers on the KDD'99 dataset are given in Table 4 and the confusion matrices in Fig. 2. From Table 4 it is clear that almost all the classifiers perform well and have similar performance scores. However the confusion matrix given in Fig. 2 indicates that the misclassification rate of the proposed model is very low as compared to all the other models. The proposed model misclassified only 58 instances, whereas misclassification of GNB, LR, DT and SGD were 735, 807, 327 and 840, respectively.

Table 5 shows the performance of the classifiers on the UNSW-NB15 dataset. Due to the high class imbalanced nature of this dataset, most of the base classifiers performed poorly, but the proposed model was able to perform better than the individual classifiers. The proposed model had an accuracy score of 93.88. It can be seen from Fig. 3 that it has comparatively lower overall misclassification.

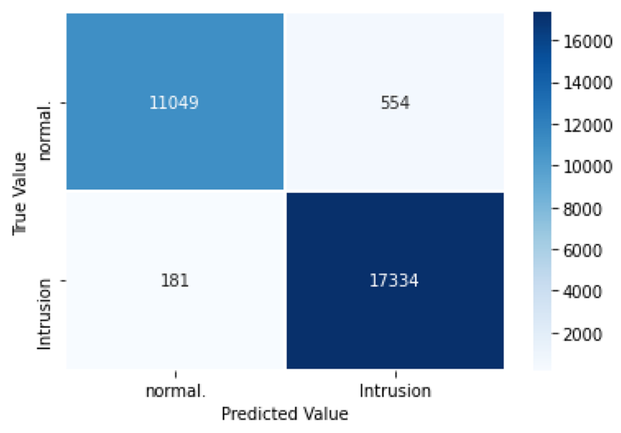
The performance results of the proposed model along with the individual classifiers for the CIC-IDS2017 dataset are given in Table 6, where it is observed that the proposed model maintains better performance. In the event of one type of classifier failing in a particular dataset, the use of an ensemble of multiple classifiers can overcome the failure and perform well. It is visible from the confusion matrix given in Fig. 4 that GNB performs very poorly with an accuracy of only 39.94%, but the proposed model delivers an accuracy of 99.80%.

Multiclass classification result

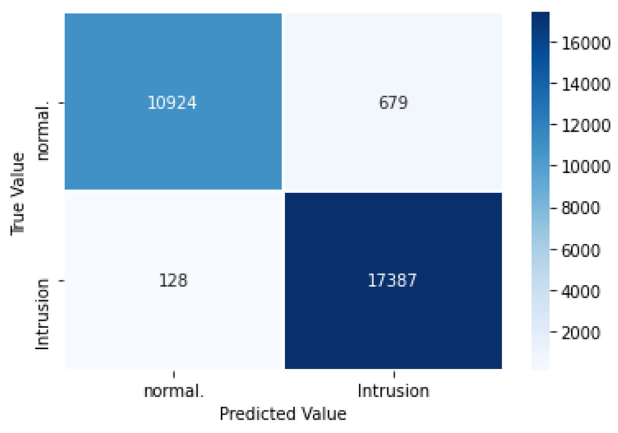
The KDD99 dataset was evaluated with 10% of data provided by the UCI KDD Archives and the results in Table 7 show that the stacked ensemble outperforms the individual classifiers in all metrics. Gaussian naive Bayes performed poorly with very high misclassifications. It misclassified the U2R attacks the most and some misclassification of R2L attacks. Logistic regression was able to overcome the issue and predicted almost all the U2R attacks correctly, but failed to classify the DoS attacks which GNB was able to classify. Decision tree had fewer false positives, but had distributed misclassifications between the DoS, U2R and R2L. Stochastic Gaussian Descent also performed similar to decision tree, but with more false positive rates. As seen in the confusion matrices given in Fig. 5, the proposed model was able to correctly identify most of the classes and had the lowest false positive rate, as it correctly identified almost all of the data labeled 'Normal'.

In UNSW-NB15, due to the high class imbalance, all the machine learning models performed badly relative to other datasets. The confusion matrices in Fig. 6 show that some classes were never predicted correctly in both decision tree and stochastic gradient descent. Logistic regression performed overall slightly better. Gaussian naive Bayes performed very poorly where it was not able to correctly identify most of the cases. The stacked ensemble was able to mitigate most of the drawbacks of these individual classifiers and aggregate the performance into a model giving higher performance accuracy of 80.96%, as compared to 42.40% of GNB, 75.12% of LR, 75.85% of DT and 72.51% of SGD as seen in Table 8. The proposed model clearly was able to correctly classify most of the data.

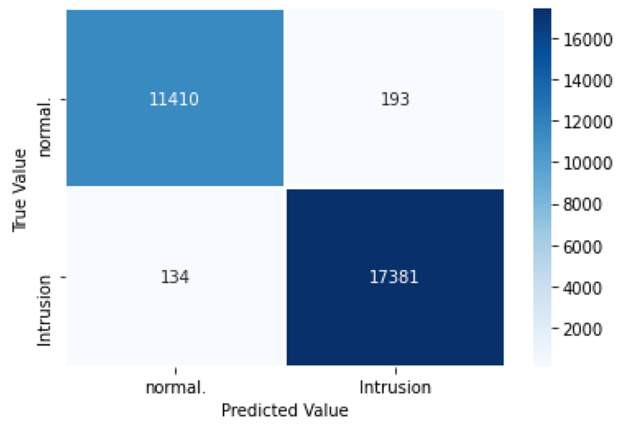
From Fig. 7 a, it can be seen that Gaussian naive Bayes again has the highest misclassification due to the high correlation between the features of the CIC IDS2017 dataset. It incorrectly classified BENIGN network traffic the most,



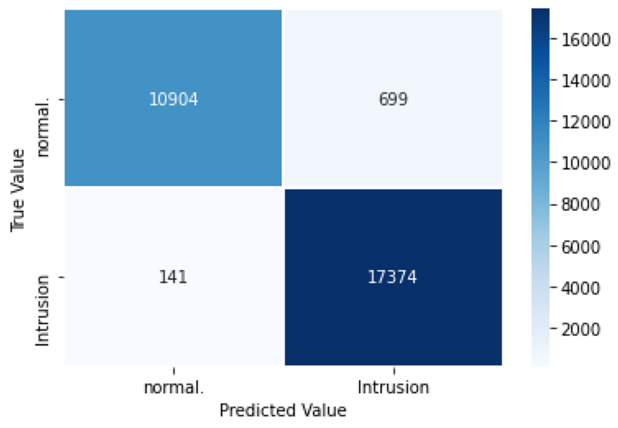
(a) Gaussian Naive Bayes



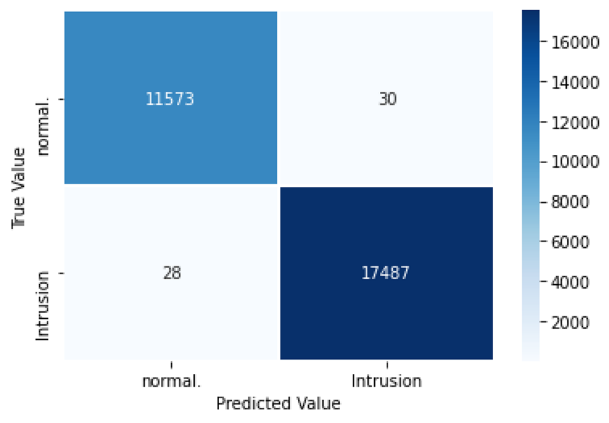
(b) Logistic Regression



(c) Decision Tree



(d) Stochastic Gradient Descent



(e) Proposed Model

Fig. 2 KDD'99 confusion matrix (binary classification)

Table 5 UNSW-NB15 performance matrix result (binary classification)

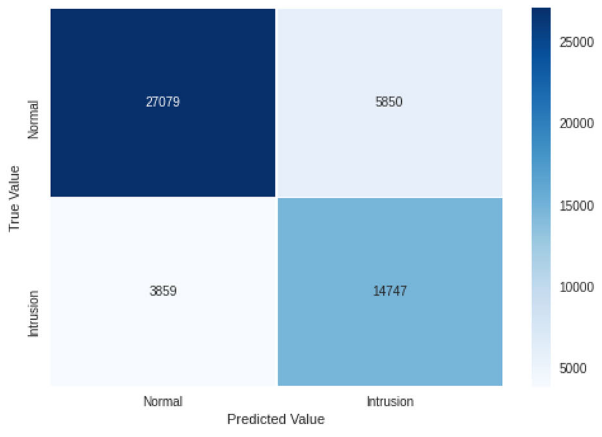
Classifier	Data type	Precision	Recall	F1 score	Accuracy (%)
GNB	Normal	0.8739	0.8318	0.8523	81.55
	Intrusion	0.7247	0.7869	0.7545	
LR	Normal	0.8683	0.9633	0.9134	88.30
	Intrusion	0.9191	0.7404	0.8201	
DT	Normal	0.9375	0.8780	0.9068	88.44
	Intrusion	0.8052	0.8960	0.8482	
SGD	Normal	0.8463	0.9992	0.9164	88.33
	Intrusion	0.9980	0.6776	0.8072	
Proposed model	Normal	0.9514	0.9531	0.9522	93.88
	Intrusion	0.9163	0.9135	0.9149	

correctly identifying only 268,677 instances out of 407,005. Logistic regression, decision tree and stochastic gradient descent, as seen in Fig. 7b–d, perform similarly, all performing better than GNB in classifying BENIGN traffic, but performed badly in cases such as DDos, DoS_Hulk, Heartbleed and DoS_Slowhttptest. Although a similar performance was measured between most of the classifier models, the proposed stacked ensemble outperforms the individual classifiers by correctly predicting most of the classes and thereby increasing the overall performance and accuracy. The proposed model was able to achieve an accuracy of 99.48% as compared to 69.97%, 96.70%, 96.27% and 91.67% of GNB, LR, DT and SGD, respectively, as shown in Table 9.

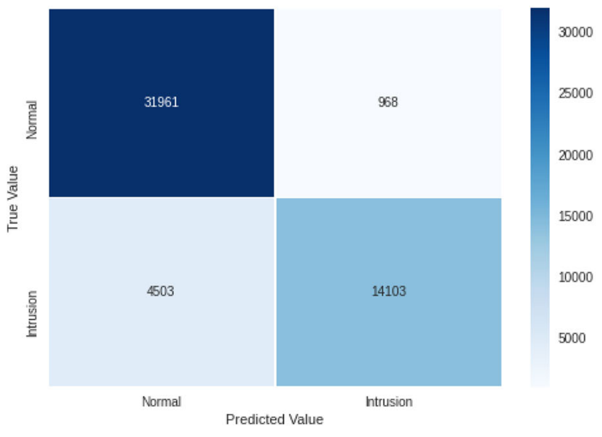
Modern network technology includes ever-increasing and complex data forms, the features and attributes of which are recorded from network traffic. Depending on the network, the numbers of functions are used for identification. There may be hundreds of these traffic instances, making it difficult for researchers and network administrators to track and understand them. This is challenging for programs that use statistical models, or even those that use functions based on machine learning to determine the nature of the network. Having only the required core features will help scale back the cost of processing sizeable data. Including feature selection in the proposed ensemble model reduces the time to process the data and the resource cost to train and test the data. One of the trade-offs of using an ensemble model for building an intrusion detection system is the increase in memory consumption and processing time. Ensemble models have to store the results of the base classifiers temporarily to serve as the input to the meta-classifier and as such large number of base classifiers or more levels of meta-classifiers can increase the level of memory consumption. An increase in the number of users on the Internet has led to the requirement of efficient algorithms that can process the data in real time. An IDS is able to detect a network traffic instance as either an attack or normal traffic. Generally, fast algorithms tend to be inaccurate and accurate algorithms tend to be slow, particularly in ML. The ensemble model presented in this paper ensures better detection performance by building an ensemble model composed of relatively lightweight learning models.

Comparisons with existing ensemble system

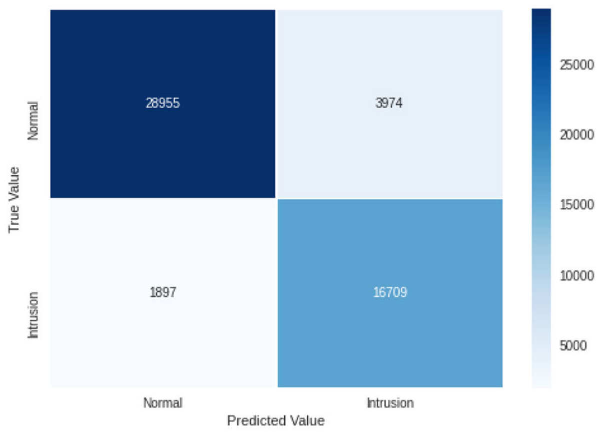
The proposed model has been compared with ensemble models that have been proposed by others in the literature. Papers that have the same evaluation dataset and parameters are compared with the proposed model using the same performance metrics. Most machine learning-based intrusion detections are carried out in the form of multiclass classification; hence, the comparisons are performed in the same approach. Table 10 shows the performance comparison with existing ensemble models with that of the proposed model using the KDD99 dataset. The performance was compared with a multiclassifier system for IDS proposed by Sabhnani and Serpen [11], a three-tier hybrid IDS designed by Hwang et al. [24] and a hybrid system that uses SVM and clustering proposed by Khan et al.[25]. Sabhnani and Serpen [11] determined that the best algorithmic approaches to tackle the specific attack categories in the KDD'99 dataset are multilayer perception for detecting 'Probe' attacks, k-means clustering for 'Denial of Service' and 'U2R' attacks and Gaussian for 'R2L' attack. Their approach relies on a specific algorithm for detecting a specific attack type, thus making it impractical in real-world scenarios when novel unrecognized attacks arise. Hwang et al. [24] employed a tier system where known attacks are filtered out from the traffic using a blacklist, i.e., misuse detector (MD) and a whitelist, i.e., anomaly detector (AD) that identifies normal traffic. Traffic that is identified as an anomaly by the AD further gets classified into the four attack categories of the KDD'99 dataset. A proposition lure learning method RIPPER was used for the blacklist and active profiling to build the whitelist. Khan et al. [25] reported the



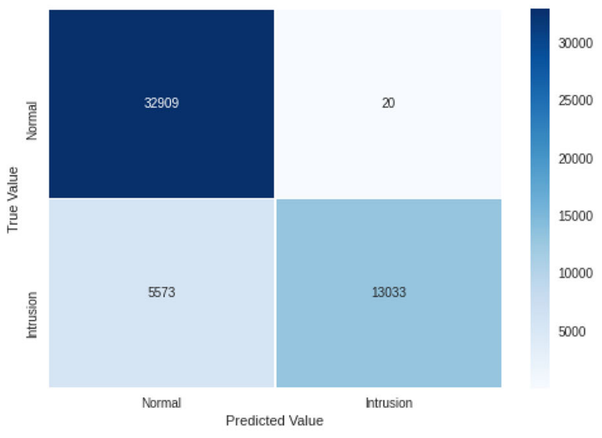
(a) Gaussian Naive Bayes



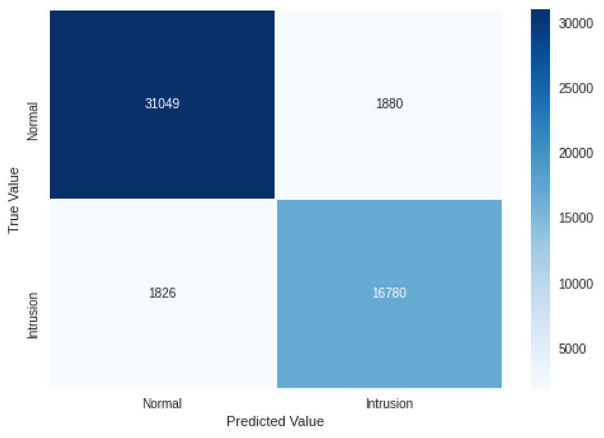
(b) Logistic Regression



(c) Decision Tree



(d) Stochastic Gradient Descent



(e) Proposed Model

Fig. 3 UNSW-NB15 confusion matrix (binary classification)

Table 6 CIC-IDS2017 performance matrix result (binary classification)

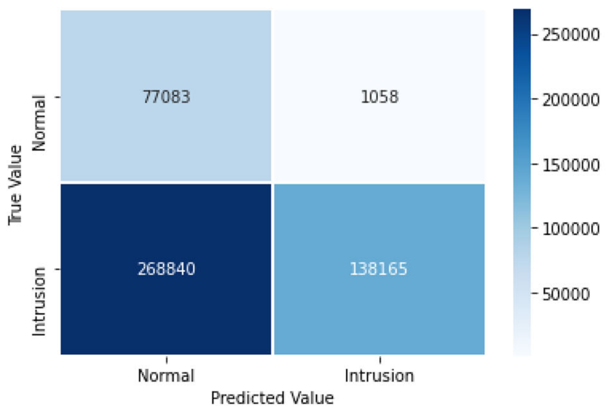
Classifier	Data type	Precision	Recall	F1 score	Accuracy (%)
GNB	Normal	0.2090	0.9871	0.3450	39.94
	Intrusion	0.9915	0.2873	0.4455	
LR	Normal	0.9545	0.6647	0.7837	94.12
	Intrusion	0.9395	0.9940	0.9660	
DT	Normal	0.9742	0.9122	0.9422	98.20
	Intrusion	0.9835	0.9954	0.9894	
SGD	Normal	0.9890	0.6258	0.7666	93.89
	Intrusion	0.9333	0.9987	0.9649	
Proposed model	Normal	0.9941	0.9936	0.9938	99.80
	Intrusion	0.9988	0.9989	0.9988	

limitation of SVM for classification of IDS dataset and identified and presented a method to enhance the training time of SVM using hierarchical clustering analysis. As evident in Table 10, the proposed model has higher overall accuracy of detection in almost all categories of attacks. For classifying R2L attacks, only the model proposed by Khan et al. [25] was able to achieve 91% accuracy, whereas the proposed model delivered only 60% accuracy. Table 11 compares the performance of the models based on the performance metrics used in Refs. [10] and [20]. The proposed model is compared with the ensemble model proposed by Rajagopal et al.[10]. They used a two-level stacking. Random forest (RF), logistic regression (LR) and k- nearest neighbor (kNN) were base classifiers and SVM was the meta-classifier. No feature selection method was used, instead the whole feature set was used for training and testing purposes. Khammassi and Krichen [20] argued that some machine learning algorithms applied to intrusion detection would be time consuming and instead proposed a unique approach for feature selection. A wrapper approach to genetic algorithm and logistic regression was used for feature selection. Classification was performed using three decision tree (DT)-based classifiers, namely, C4.5, RF and naive Bayes tree. As shown in Table 11, the ensemble methods proposed by Rajagopal et al. [10] and Khammassi and Krichen [20] had higher prediction in some types of attacks. However, the proposed model is able to deliver better prediction performance in attack types where Rajagopal et al. [10] and Khammassi and Krichen [20] were unable to predict attacks accurately.

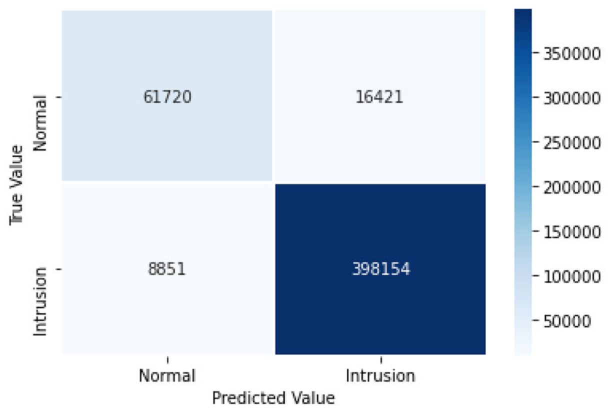
The proposed method performs well on a diverse range of intrusion detection datasets.

Conclusion

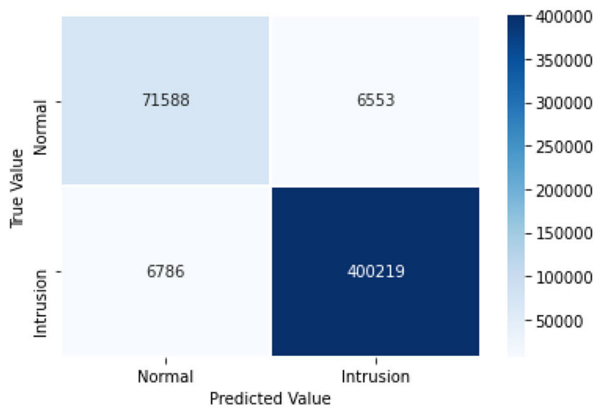
In this paper, an ensemble learning-based model is presented with the aim of overcoming the weakness of individual machine learning classifiers. The weakness of a classifier is overcome by the strength of another classifier. Three separate datasets, namely, KDD99, UNSW-NB15 and CIC-IDS2017, were used to evaluate system performance under various network environments. The purpose of developing this ensemble intrusion detection technique is to create a model that can detect attacks using a combination of learning models. The results show that the proposed stacked ensemble-based model exploits the benefit of using several classifiers that perform better in different cases. The proposed model has been shown to improve the overall performance significantly. Reduced false positive rate and increased accuracy were achieved. It is concluded that ensemble classifiers are suitable for classifying data within the field of intrusion detection system as its datasets are of high class imbalance. Although the overall performance of the model is better, there are some aspects where scope for further improvements such as detecting attacks like U2R and R2L in KDD'99, reconnaissance and backdoor attacks in UNSW-NB15 and DoS, web attacks, etc., in the CIC-IDS2017 dataset exists. The ensemble method proposed in this paper performs well in network intrusion detection datasets where there is high class imbalance. Further improvement can be made to solve the class imbalance problems with data augmentation technique, i.e., synthetic oversampling. Cost-sensitive learning models can be pursued to improve the detection of classes with very few instances.



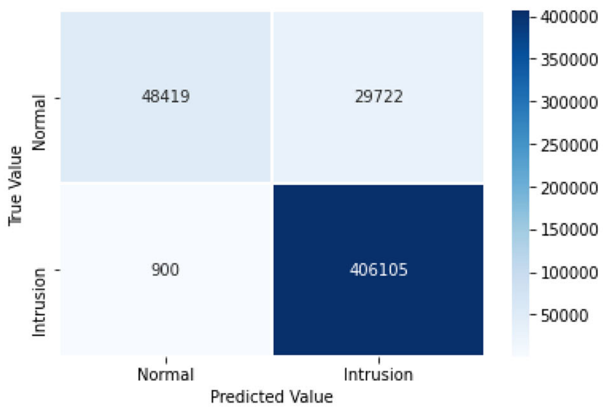
(a) Gaussian Naive Bayes



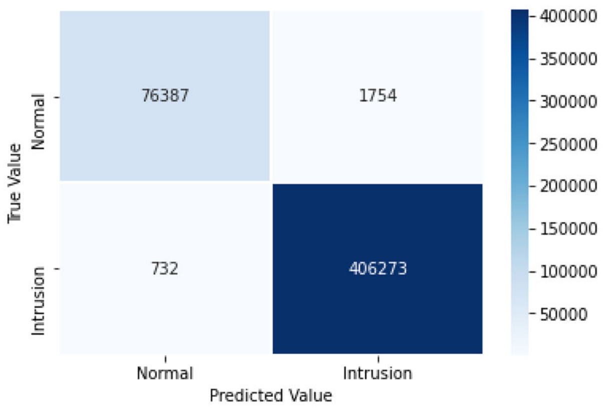
(b) Logistic Regression



(c) Decision Tree



(d) Stochastic Gradient Descent

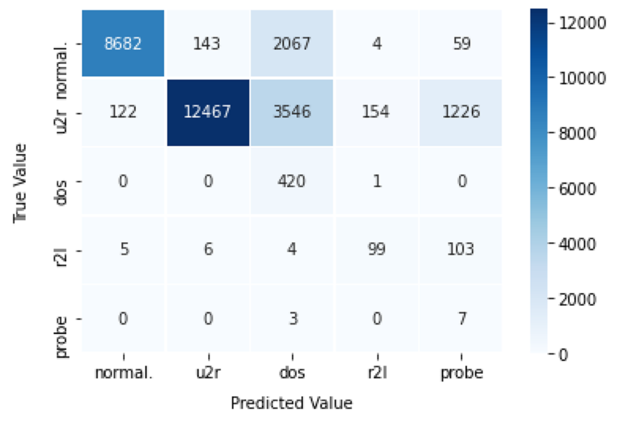


(e) Proposed Model

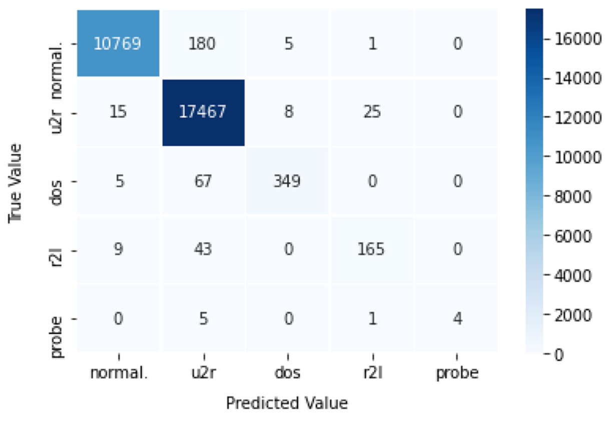
Fig. 4 CIC-IDS2017 confusion matrix (binary classification)

Table 7 KDD'99 performance matrix result (multiclass classification)

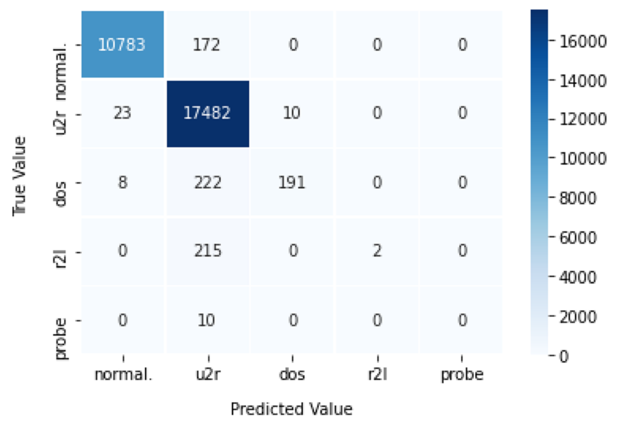
Classifier	Data type	Precision	Recall	F1-score	Accuracy (%)
GNB	Normal	0.9855	0.7925	0.8785	74.43
	DoS	0.9881	0.7117	0.8275	
	U2R	0.0695	0.9976	0.1300	
	R2l	0.3837	0.4562	0.4168	
	Probe	0.0050	0.7000	0.0099	
LR	Normal	0.9973	0.9830	0.9901	98.74
	DoS	0.9833	0.9972	0.9902	
	U2R	0.9640	0.8289	0.8914	
	R2l	0.8593	0.7603	0.8068	
	Probe	1.000	0.4000	0.5714	
DT	Normal	0.9971	0.9842	0.9906	97.73
	DoS	0.9658	0.9981	0.9816	
	U2R	0.9502	0.4536	0.6141	
	R2l	1.000	0.0092	0.0182	
	Probe	0.000	0.000	0.000	
SGD	Normal	0.9967	0.9975	0.9971	98.56
	DoS	0.9920	0.9954	0.9937	
	U2R	0.8855	0.7719	0.8248	
	R2l	0.9043	0.8709	0.8873	
	Probe	1.000	0.3000	0.4615	
Proposed model	Normal	0.9974	0.9976	0.9975	99.19
	DoS	0.9919	0.9955	0.9937	
	U2R	0.8817	0.7790	0.8272	
	R2l	0.9038	0.8663	0.8847	
	Probe	1.000	0.3000	0.4615	



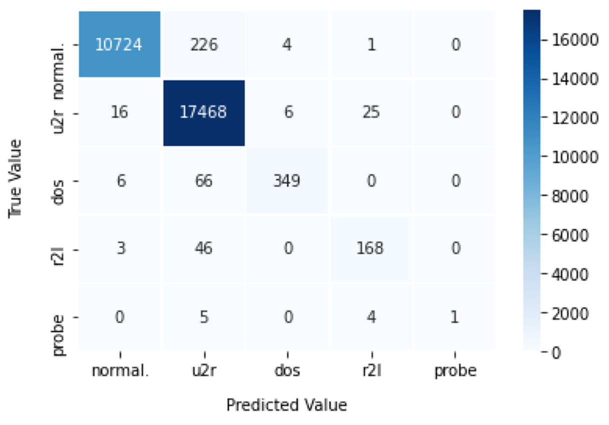
(a) Gaussian Naive Bayes



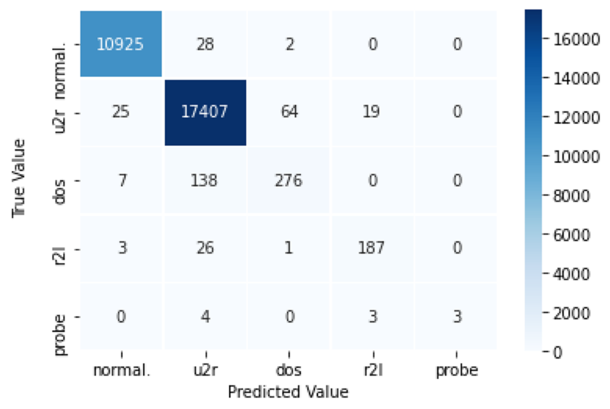
(b) Logistic Regression



(c) Decision Tree

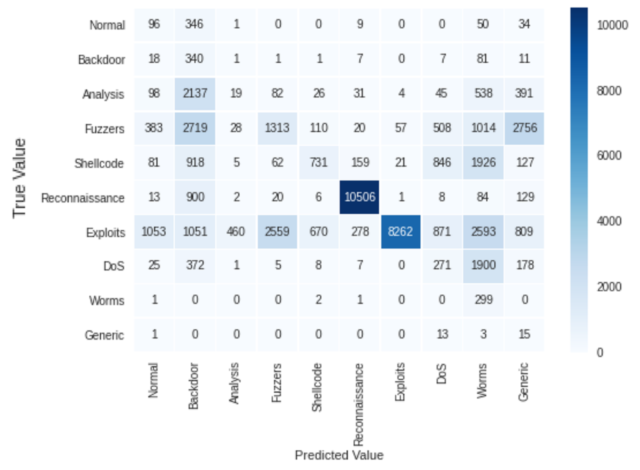


(d) Stochastic Gradient Descent

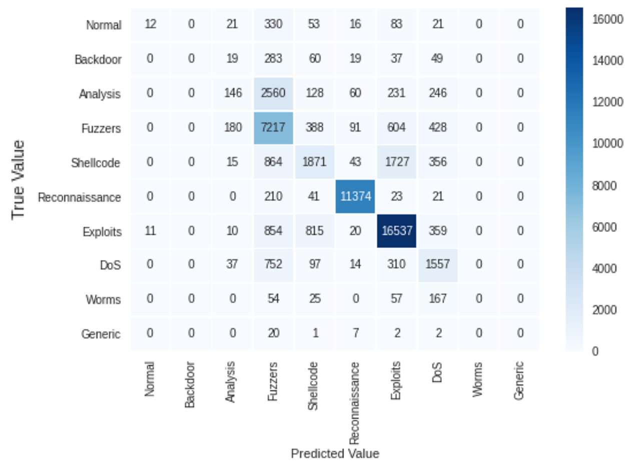


(e) Proposed Model

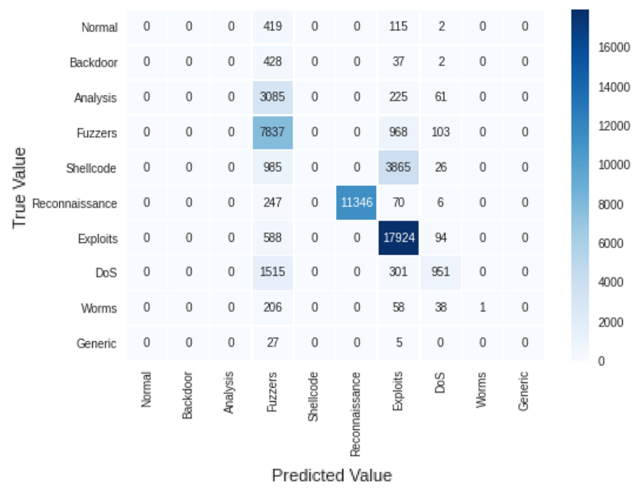
Fig. 5 KDD'99 confusion matrix (multiclass classification)



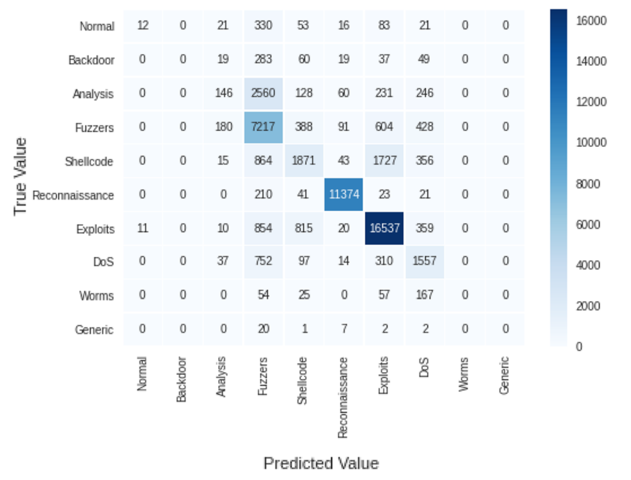
(a) Gaussian Naive Bayes



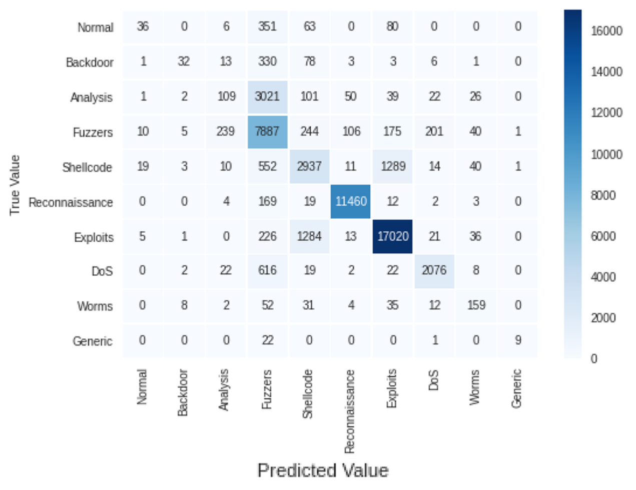
(b) Logistic Regression



(c) Decision Tree



(d) Stochastic Gradient Descent

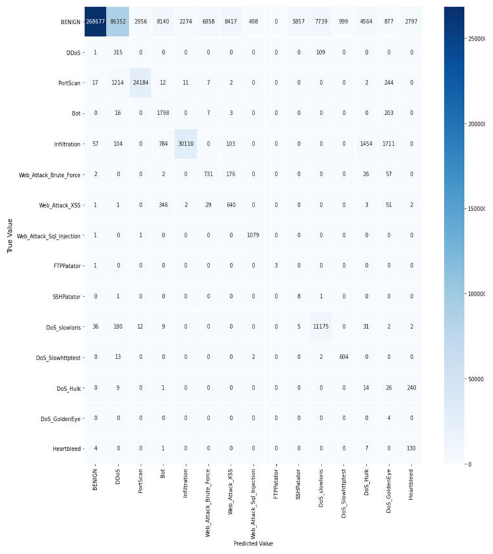


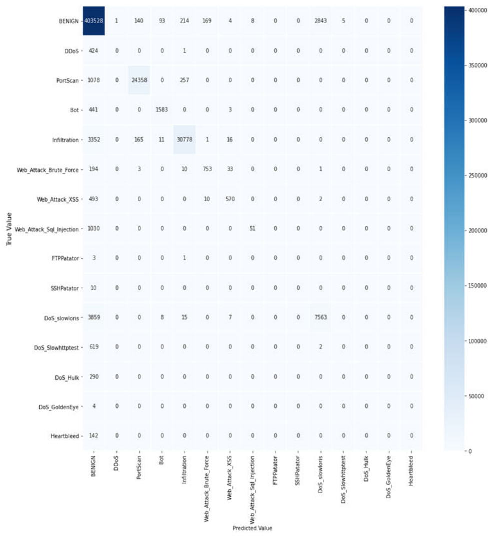
(e) Proposed Model

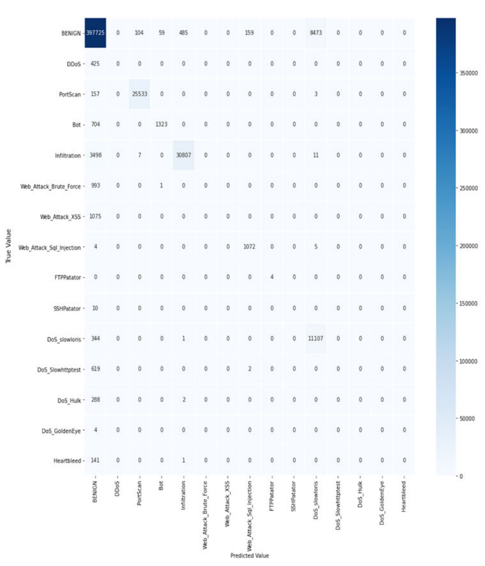
Fig. 6 UNSW-NB15 confusion matrix (multiclass classification)

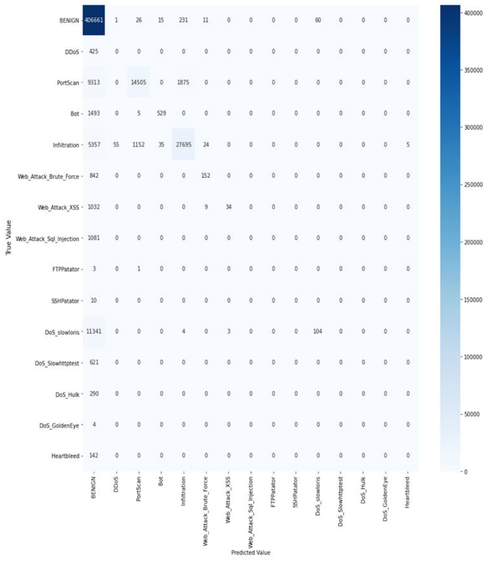
Table 8 UNSW-NB15 performance matrix result (multiclass classification)

Classifier	Data type	Precision	Recall	F1 score	Accuracy (%)
GNB	Normal	0.0542	0.1791	0.0832	42.40
	Backdoor	0.0387	0.7280	0.0735	
	Analysis	0.0367	0.0056	0.0097	
	Fuzzers	0.3248	0.1473	0.2027	
	Shellcode	0.4703	0.1499	0.2273	
	Reconnaissance	0.9535	0.9003	0.9261	
	Exploits	0.9900	0.4440	0.6131	
	DoS	0.1054	0.0979	0.1015	
	Worms	0.0352	0.9867	0.0680	
	Generic	0.0033	0.4687	0.0066	
LR	Normal	0.5217	0.0223	0.0429	75.12
	Backdoor	0.000	0.000	0.000	
	Analysis	0.3411	0.0433	0.0768	
	Fuzzers	0.5490	0.8101	0.6545	
	Shellcode	0.5377	0.3837	0.4478	
	Reconnaissance	0.9768	0.9747	0.9757	
	Exploits	0.8432	0.8887	0.8654	
	DoS	0.4856	0.5627	0.5213	
	Worms	0.000	0.000	0.000	
	Generic	0.000	0.000	0.000	
T	Normal	0.000	0.000	0.000	73.85
	Backdoor	0.000	0.000	0.000	
	Analysis	0.000	0.000	0.000	
	Fuzzers	0.5109	0.8797	0.6464	
	Shellcode	0.000	0.000	0.000	
	Reconnaissance	1.000	0.9723	0.9859	
	Exploits	0.7605	0.9633	0.8500	
	DoS	0.7412	0.3436	0.4696	
	Worms	1.000	0.0033	0.0065	
	Generic	0.000	0.000	0.000	
SGD	Normal	0.000	0.000	0.000	72.51
	Backdoor	0.000	0.000	0.000	
	Analysis	0.3044	0.5921	0.4021	
	Fuzzers	0.6440	0.5589	0.5984	
	Shellcode	0.5286	0.2305	0.3210	
	Reconnaissance	0.9164	0.9756	0.9451	
	Exploits	0.8272	0.8861	0.8557	
	DoS	0.5048	0.5048	0.5048	
	Worms	0.0000	0.000	0.000	
	Generic	0.0000	0.000	0.000	
Proposed model	Normal	0.5000	0.0671	0.1184	80.96
	Backdoor	0.6037	0.0685	0.1230	
	Analysis	0.26913	0.0323	0.0577	
	Fuzzers	0.5963	0.8853	0.7126	
	Shellcode	0.6149	0.6023	0.6085	
	Reconnaissance	0.9837	0.9820	0.9829	
	Exploits	0.9113	0.9147	0.9130	
	DoS	0.8815	0.7502	0.8106	
	Worms	0.5079	0.5247	0.5162	
	Generic	0.8181	0.28125	0.4186	


(a) Gaussian Naive Bayes


(b) Logistic Regression


(c) Decision Tree


(d) Stochastic Gradient Descent

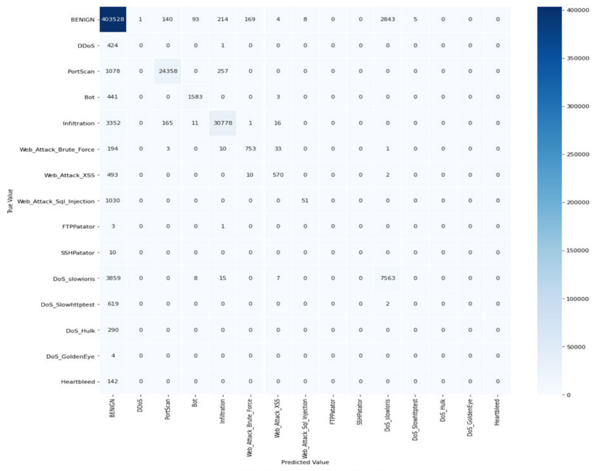

(e) Proposed Model

Fig. 7 CIC-IDS2017 confusion matrix (multiclass classification)

Table 9 CIC-IDS2017 performance matrix result (multiclass classification)

Classifier	Data type	Precision	Recall	F1 score	Accuracy (%)
GNB	BENIGN	0.9995	0.6601	0.7951	69.97
	DDoS	0.0035	0.7411	0.0071	
	PortScan	0.8906	0.9412	0.9152	
	Bot	0.1620	0.8870	0.2740	
	Infiltration	0.9294	0.8772	0.9025	
	Web_Attack_Brute_Force	0.0957	0.7354	0.1694	
	Web_Attack_XSS	0.0685	0.5953	0.1228	
	Web_Attack_Sql_Injection	0.6833	0.9981	0.8112	
	FTPPatator	1.0000	0.75	0.8571	
	SSHPatator	0.0013	0.8	0.0027	
	DoS_slowloris	0.5873	0.9758	0.7333	
	DoS_Slowhttptest	0.3767	0.9726	0.5431	
	DoS_Hulk	0.0022	0.0482	0.0043	
	DoS_GoldenEye	0.0012	1.000	0.0025	
	Heartbleed	0.0409	0.9154	0.0784	
LR	BENIGN	0.9712	0.9914	0.9812	96.70
	DDoS	0.0000	0.0000	0.0000	
	PortScan	0.9875	0.9480	0.9673	
	Bot	0.9339	0.7809	0.8506	
	Infiltration	0.9840	0.8967	0.9383	
	Web_Attack_Brute_Force	0.8070	0.7575	0.7815	
	Web_Attack_XSS	0.9004	0.5302	0.6674	
	Web_Attack_Sql_Injection	0.8644	0.0471	0.0894	
	FTPPatator	0.0000	0.0000	0.0000	
	SSHPatator	0.0000	0.0000	0.0000	
	DoS_slowloris	0.7264	0.6604	0.6918	
	DoS_Slowhttptest	0.0000	0.0000	0.0000	
	DoS_Hulk	0.0000	0.0000	0.0000	
	DoS_GoldenEye	0.0000	0.0000	0.0000	
	Heartbleed	0.0000	0.0000	0.0000	
DT	BENIGN	0.9796	0.9771	0.9784	96.27
	DDoS	0.0000	0.0000	0.0000	
	PortScan	0.9956	0.9937	0.9947	
	Bot	0.9566	0.6526	0.7759	
	Infiltration	0.9843	0.8975	0.9389	
	Web_Attack_Brute_Force	0.0000	0.0000	0.0000	
	Web_Attack_XSS	0.0000	0.0000	0.0000	
	Web_Attack_Sql_Injection	0.8694	0.9916	0.9265	
	FTPPatator	1.0000	1.0000	1.0000	
	SSHPatator	0.0000	0.0000	0.0000	
	DoS_slowloris	0.5667	0.9698	0.7154	
	DoS_Slowhttptest	0.0000	0.0000	0.0000	
	DoS_Hulk	0.0000	0.0000	0.0000	

Table 9 (continued)

Classifier	Data type	Precision	Recall	F1 score	Accuracy (%)
	DoS_GoldenEye	0.0000	0.0000	0.0000	
	Heartbleed	0.0000	0.0000	0.0000	
SGD	BENIGN	0.9269	0.9992	0.9617	91.67
	DDoS	0	0	0	
	PortScan	0.9245	0.5645	0.7010	
	Bot	0.9117	0.2599	0.4046	
	Infiltration	0.9292	0.8050	0.8627	
	Web_Attack_Brute_Force	0.8735	0.1529	0.2602	
	Web_Attack_XSS	0.9189	0.0316	0.0611	
	Web_Attack_Sql_Injection	0.000	0.000	0.000	
	FTPPatator	0.000	0.000	0.000	
	SSHPatator	0.000	0.000	0.000	
	DoS_slowloris	0.6964	0.0034	0.0067	
	DoS_Slowhttptest	0.000	0.000	0.000	
	DoS_Hulk	0.000	0.000	0.000	
	DoS_GoldenEye	0.000	0.000	0.000	
	Heartbleed	0.000	0.000	0.000	
Proposed model	BENIGN	0.9957	0.9985	0.9971	99.48
	DDoS	0.7892	0.4847	0.6005	
	PortScan	0.9959	0.9952	0.9956	
	Bot	0.9787	0.9787	0.9787	
	Infiltration	0.9971	0.9653	0.9809	
	Web_Attack_Brute_Force	0.9474	0.9436	0.9455	
	Web_Attack_XSS	0.9944	0.9916	0.9930	
	Web_Attack_Sql_Injection	0.9981	0.9981	0.9981	
	FTPPatator	0.000	0.000	0.000	
	SSHPatator	0.000	0.000	0.000	
	DoS_slowloris	0.9742	0.9922	0.9831	
	DoS_Slowhttptest	0.9983	0.9758	0.9869	
	DoS_Hulk	0.6441	0.8862	0.7460	
	DoS_GoldenEye	0.000	0.000	0.000	
	Heartbleed	0.000	0.000	0.000	

Table 10 Performance comparison with existing ensemble methods (KDD99)

Author	Sabhnani and Serpen [11]	Hwang et al. [24]	Khan et al. [25]	Proposed model
Classifier(s)	ANN, Gaussian classifier, k-means clustering	MD, AD, SVM	SVM, clustering	DT, LR, GNB, SGD
Attack type	Accuracy (%)			
Probe	88.7	99.16	97	99.98
DoS	97.3	97.65	23	99.89
U2R	29.8	76.32	43	98.81
R2L	09.6	46.53	91	60

Table 11 Performance comparison with existing ensemble methods (UNSW NB15)

Author	Rajagopal et al. [10]		Khammassi and Krichen [20]		Proposed model	
Classifier(s)	RF, LR, kNN, SVM		GA, DT		DT, LR, GNB, SGD	
Attack type	Precision	Recall	Precision	Recall	Precision	Recall
Analysis	67.44	11	44.44	9.92	98.80	98.11
Backdoor	70	10.79	51.51	6.92	31.40	08.41
DOS	41.6	25	36.0	4.11	61.14	86.20
Exploits	63.41	85	60.2	92.31	60.14	61.30
Fuzzers	64.42	60.97	70.3	69.11	91.71	91.00
Generic	99.42	98.32	99.7	97.93	54.55	19.35
Normal	91.67	91.82	92.3	90.72	55.81	13.82
Reconnaissance	90.65	74.8	90.88	76.15	47.56	07.98
Shellcode	68.65	58.22	53.57	47.46	49.82	51.48
Worms	57.69	37.5	46.8	38.46	86.01	75.55

Data availability All Datasets are freely available in the internet. 1. KDD Cup 1999 (https://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data) 2. UNSW-NB15 (https://research.unsw.edu.au/projects/unsw-nb15-dataset) 3. CIC-IDS2017 (https://www.unb.ca/cic/datasets/ids-2017.html)

Declarations

Conflict of interest There are no competing interests.

References

1. Dietterich TG (2000) Ensemble methods in machine learning multiple classifier systems MCS 2000. Lect Note Comput Sci. 1:1857
2. Sommer R, Paxson V (2010) Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA
3. Souri A, Hosseini R (2018) A state-of-the-art survey of malware detection approaches using data mining techniques. Hum Cent Comput Inf Sci 8:3. https://doi.org/10.1186/s13673-018-0125-x
4. Wang K, Stolfo SJ (2004) Anomalous Payload-Based Network Intrusion Detection. In: Jonsson E (ed) Recent Advances in Intrusion Detection. Lecture Notes in Computer Science, Heidelberg
5. Sonule AR, Kalla M (2020) Unsw-Nb15 Dataset and Machine Learning Based Intrusion Detection Systems. Inter J Eng Adv Technol. 9:2249–8958
6. Divyasree TH, Sherly KK (2018) A Network Intrusion Detection System Based On Ensemble CVM Using Efficient Feature Selection Approach. Procedia Comp Sci. 143:442–449
7. Wang Y, Shen Y, Zhang G (2016) Research on intrusion detection model using ensemble learning methods. Inter Software Eng Ser Sci. 22:422–425
8. Kumar G, Thakur K, Ayyagari MR (2020) MLEsIDSs: machine learningbased ensembles for intrusion detection systems—a review. J Supercomput 76:8938–8971. https://doi.org/10.1007/s11227-020-03196-z
9. Rajadurai H, Gandhi U (2020) A stacked ensemble learning model for intrusion detection in wireless network. Neural Comput & Applic. https://doi.org/10.1007/s00521-020-04986-5
10. Rajagopal S, Kundapur PP, Hareesha KS (2020) A Stacking ensemble for network intrusion detection using heterogeneous datasets". Sec Commun Net 20:4586875
11. Sabhnani M, Serpen G (2003) Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context. Inter Mach Learn Model. 1:2009–2215
12. Chebrolu S, Abraham A, Thomas J (2005) Feature deduction and ensemble design of intrusion detection systems. Comput Secur 24(4):295–307
13. Abraham A, Thomas J (2005) Distributed intrusion detection systems: a computational intelligence approach. In: Abbass HA, Essam D (eds) Applications of information systems to homeland security and defense. Idea Group Inc., Publishers, Hershey, pp 105–135
14. Mukkamala S, Sung A, Abraham A (2005) Intrusion detection using an ensemble of intelligent paradigms. J Netw Comput Appl 28(2):167–182
15. Zainal A, Maarof M, Shamsuddin S et al (2009) Ensemble classifers for network intrusion detection system. J Inf Assur Secur 4:217–225
16. Menahem E, Rokach L, Elovici Y (2009) Troika—an improved stacking schema for classifcation tasks. Inf Sci 179(24):4097–4122
17. Meng Y, Kwok LF (2013) Enhancing false alarm reduction using voted ensemble selection in intrusion detection. Int J Computl Intell Syst 6(4):626–638

18. Haq NF, Onik AR, Shah FM (2015) An ensemble framework of anomaly detection using hybridized feature selection approach (hfsa). SAI Intel Syst Confer (IntelliSys). 22:989–995
19. Tama BA, Rhee KH (2015) A combination of PSO-based feature selection and tree-based classifers ensemble for intrusion detection systems. In: Rhee KH (ed) Advances in computer science and ubiquitous computing. Springer, Newyork
20. Khammassi C, Krichen S (2017) A GA-LR wrapper approach for feature selection in network intrusion detection. Comp Sec 70:0167–4048. https://doi.org/10.1016/j.cose.2017.06.005
21. Satheesh N, Rathnamma MV, Rajeshkumar G, Vidya Sagar P, Pankaj Dadheech SR, Dogiwal PV, Sengan S (2020) Flow-based anomaly intrusion detection using machine learning model with software defined networking for OpenFlow network. Microproc Microsyst. 79:103285
22. Singh NB, Singh MM, Sarkar A, Mandal JK (2021) A novel wide & deep transfer learning stacked GRU framework for network intrusion detection. J Inform Security Appl. 61:102899
23. Manimurugan S, Majdi A-q, Mustaffa Mohmmed C, Narmatha RV (2020) Intrusion detection in networks using crow search optimization algorithm with adaptive neuro-fuzzy inference system. Microproc Microsysts. 79:103261
24. Hwang T, Lee T, Lee Y (2007) A three-tier ids via data mining approach. Proc ACM Workshop Min Net. 55:1–6
25. Khan L, Awad M, Thuraisingham B (2007) A new intrusion detection system using support vector machines and hierarchical clustering. VLDB J Int J Very Large Data Bases 16(4):507–521
26. Polikar R (2006) Ensemble based systems in decision making. IEEE Circuits Syst Mag 6(3):21–45