Network intrusion detection system: A robust study of machine and deep learning approaches

1st Pavas Goyal
Computer Science and Engineering
Indian Institute of Technology
New Delhi, India
pavasgdb@gmail.com

***Abstract*—The rapid development and advancement in the field of internet and communication have brought about immense expansion in the size as well as traffic of networks. This has brought about the increase in the generation of novel attacks, which in turn has presented difficulties for network security to precisely identify intrusions. In addition to this it has also advanced the ascent in number of intruders having the aim to dispatch attacks within the network. In such situations Intrusion Detection Systems (IDS) come to our rescue. These frameworks help us track attacks by keeping an eye on the behaviour of network traffic. Since the intruders keep on altering their tools and techniques, fabricating a robust Intrusion Detection System becomes a challenging task.**

The recent decades have led to the increase in use of ML-based IDS frameworks. The paper describes the scientific categorization dependent on the eminent ML and DL strategies received in planning network-based IDS (NIDS) frameworks. It also gives the late patterns and progressions of ML and DL-based NIDS, regarding the proposed strategy, assessment measurements, and dataset choice. Utilizing the weaknesses of the proposed strategies, different examination challenges are featured and future extension to the exploration in improving ML and DL-based NIDS is also given.

I. Introduction

With the advent in the field of internet and communication over the most recent couple of many years, network security has gotten one of the significant spaces of center and examination. An Intrusion Detection System being one of the principle instruments for network security has subsequently become an indispensable subject of conversation. It is a framework intended to help associations screen their cloud, on-premise and hybrid conditions for dubious occasions that could show a trade off. This incorporates strategy infringement and port checking, in addition to unknown source/destination traffic.

To detect threats, network-based intrusion detection systems gather information about incoming and outgoing internet traffic. To maximise visibility, NIDS sensors are strategically placed across a network, for example on a LAN and DMZ.NIDS systems utilise a combination of signature and anomaly-based detection methods. Signature-based detection involves comparing the characteristics of collected data packets against signature files that are known to be malicious. Anomaly-based detection uses behavioural analysis to monitor events against a baseline of 'typical' network activity. When malicious or anomalous activity arises on a network, such as a sudden increase in network traffic, NIDS technologies detect the activity and generate alerts for investigation.

In the course of the most recent couple of many years, numerous ML and DL tecnhiques have been utilized to enhance these frameworks. But, because of the great expansion in measure of traffic and consistent alteration in tools and techinques used for these attacks, it has gotten exceptionally hard to adjust among accuracy and false positives. This paper plans to give a wide outline of the advancements till now in this field utilizing the ML methods. The primary idea is to give a standard to new researchers for having a high level idea in this research domain.

The remainder of the paper is organized as follows: In section-II we will mention the source of the data used for all experiments and the various proprocessing steps involved. In Section-III we will discuss the various ML and DL approaches used along with inferences obtained for each of them. In Section-IV we have discussed some of the previous research work done in this domain. in Section-V we have given the concluding remarks.

II. Data and Pre-processing

A. Data Source

The data used for this research is taken from the competition organized by the IEEE BigData conference in 2019. This data is from the Security Operations Center of a company, and has been labelled by their analysts. The data set consist of alerts investigated by a SOC team at SoD (called 'investigated alerts'). Each record is described by various statistics selected based on experts' knowledge, and a hierarchy of associated IP addresses (anonymized), called assets. For each alert in the 'investigated alerts' data tables, there is a history of related log events (a detailed set of network operations acquired by SoD, anonymized to ensure the safety of SoD clients).

B. Encoding of Categorical Data

In the data large numbers of attributes comprise of categorical data. Since many machine learning algorithms can't work with categorical data straightforwardly, the data should be changed over into numbers. For which we had two alternatives, First - to allocate number to each categorical value, Second - utilize one hot encoding. Out of these two the later is better in light of the fact that it keeps the qualities autonomous.

Relegating numbers to categorical values makes a dependeny between them, as the number which are close will in general be more related. [3].

C. Feature Selection

The dataset contains numerous features, so it gets important to eliminate the futile for the effective and quicker execution of our models. For feature determination we have utilized a random forest classifier to determine importances of every one of the attributes. After this we can channel the topmost features as per the assets available. The Fig 1 shows the importances of various attributes calculated by our algorithm. In our experiments we have taken out the 4 of the most unsignificant features.

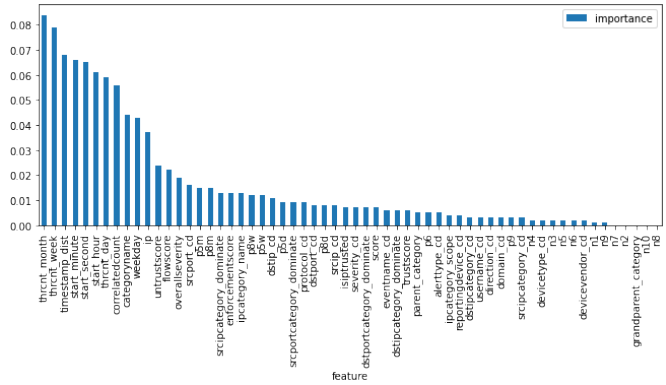

Fig. 1. Feature Importances

D. New Feature Addition

Other of the preprocessing steps incorporate the change of ip address to a usuable trait. For this we have edited the prefix out of the ip address, since full ip address don't show anything valuable and furthermore they happen in high assortment. After this cropping too, the classes of ip prefixes were many, and to lessen it further we made groups so that ip prefixes which happen in more than 200 entries are announced as a different group and each one of the remanining ones are added to the 'others'group. Utilizing this we were able to reduce the number of ip prefix groups to 27.

E. Splitting of dataset into train and test data

After the alteration of features in the data, it is randomly parted to train and test dataset. The partition is done stratifically. The stratify parameter helps to retain the same proportion of classes in the train and test sets that are found in the entire original dataset. This is done to prevent the scarcity of attack samples in any of the train or test sets, as attack samples are generally lower in amount. A train to test split proportion of 0.7:0.3 had been utilized. The random seed of 1010 had been utilized for parting. The value of 1010 had been chosen after intensive testing, as it gave best segregation of data.

F. Handling Class Imbalance

The given data is a real world data. Hence, it is subject to the restriction that the entries of actual detected attacks is very less compared to the total number of entries. One way to fight imbalance data is to generate new samples in the minority classes. The most naive strategy is to generate new samples by randomly sampling with replacement of the currently available samples. The RandomOverSampler offers such a scheme. Hence, in our experminent we have used the Random over-sampling with imblearn. Since we are already equipped with train and test sets, we can upscale the train set for training our model. Here the point of importance is that we had to only upscale the train data. If we would have performed the upscaling before segregation of data into train and test, it would have caused upscaling of test data too, which in desirable as it affect the evaluation of our model. Table I shows the counts for classes (0-non-attacks , 1-attacks) before and after handling imbalance in the training dataset.

TABLE I
CLASS VALUE COUNTS

Class notified	**Value Counts**	
	Before Upscaling	***After Upscaling***
0	26005	26005
1	1593	26005

Handling class imbalance by upscaling minor class

III. APPROACHES

We have performed various ML and DL approaches for building our IDS detection system. In this section we have discussed each of them one by one. We have also mentioned the optimizations available and the pros and cons for each of the algorithms. The hardware used for these experiments is Google Colab. Table II show the specification of the hardware used for our analysis

TABLE II
HARDWARE SPECIFICATIONS

Specification	***Info***
instance	n1-highmem-2
cpu model	Intel(R) Xeon(R) CPU
cpu MHz	2199.998
cache size	56320 KB
Available RAM	13 GB

To find the most suitable algorithm for a particular problem, there are few model evaluation techniques. The ones used by us in our analysis are listed below.

- *'roc curve' :* ROC (Receiver Operating Characteristics) is a probability curve and AUC (Area Under Curve) represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.
- *'precision score' :* Precision is the ratio of correctly predicted positive observations to the total predicted

positive observations.

$$Precision\ Score = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

- *'recall score' :* Recall is calculated as the number of true positives divided by the total number of true positives and false negatives.

$$Recall\ Score = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

- *'f1 score' :* The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

A. SVM Classifier

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

We utilized the sklearn library for training the SVM classifier for our IDS. The model took approximately 30 mins for training. Using this model on the preprocessed data we were able to achieve `roc_auc_score` as high as 0.718165.

TABLE III
RESULTS

Metric	*Score*
ROC AUC score	0.718165
Average precision score	0.115253
Mean Squared error	0.635995
Macro F1 score	0.320338
Micro F1 score	0.364005

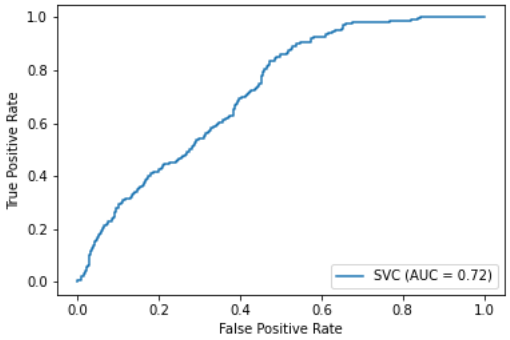

Fig. 2. ROC curve

Table III shows the results obtained for the model. From Fig 2 we can observe that the model recognized all the attacks with almost full accuracy which shows us that the model was very successful in recognizing the close margin between normal samples and the attacks. Although the model has detected almost all of the attacks, it had also detected a large number of normal samples as attacks. These high number of false positives had brought down its precision score to such a low value. This low precision makes SVM unfit for use in real life.

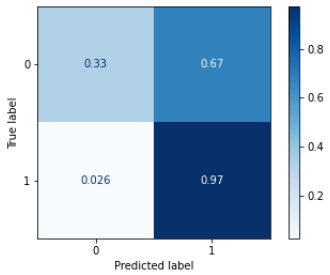

Fig. 3. Confussion Matrix

The main benefit of using SVM classifier is that, they works really well with a clear margin of separation at dimensional level. This makes them effective in high dimensional spaces [4]. Since the intrusion detection problems are generally high dimensional, SVM becomes a good potential solution for solving them.

The major short coming of SVM is that it doesn't perform well with bigger datasets. The reason for this is that, the required training time is higher. It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping, which is generally the case in Intrusion Detection problems.

B. Naive Bayes Classifier

Naive Bayes is a very nice example in support of the statement the simplest solutions are typically the most remarkable ones. Despite the advances in Machine Learning in the last years, it has proven to not only be simple but also fast, accurate, and reliable. The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the outcome.

$$P(y|x1, ..., xn) = \frac{P(y)P(x1, ..., xn|y)}{P(x1, ..., xn)} = P(y)\Pi_{i=1}P(xi|y)P(x1, ...,$$

TABLE IV
RESULTS

Metric	*Score*
ROC AUC score	0.732927
Average precision score	0.131071
Mean Squared error	0.088173
Macro F1 score	0.548249
Micro F1 score	0.911827

It is observed that the proposed technique performs better in terms of false positive rate, cost, and computational time compared to complicated algorithms. In some cases, speed is preferred over higher accuracy [5]. The model took just few seconds to train achieving `roc_auc_score` as high as 0.732927. Table IV shows the results obtained for the Naive Bayes model.

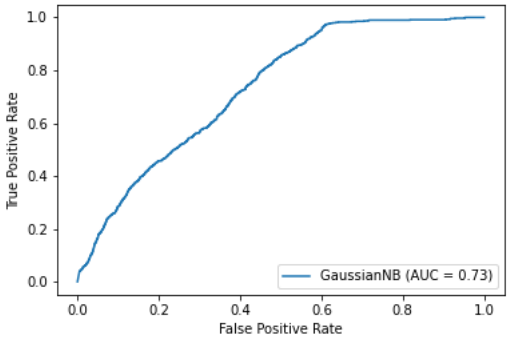

Fig. 4. ROC curve

In Fig 5 we can clearly observe that the model has been unsuccessful in recognizing a large amount of attacks. It is too biased towards normal samples reducing the number if true positives as well as increasing the number of false negatives. This brings down the precision of the model to a very low value, unfit for practical use. The can be explained as the assumption that all features are independent is quite limiting in real life so it makes Naive Bayes algorithm less accurate than complicated algorithms. Speed comes at a cost!

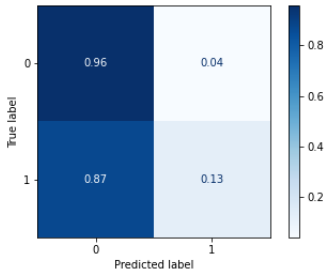

Fig. 5. Confussion Matrix

C. Neural Networks

In the recent decades, Neural networks have been effectively used to distinguish and foresee strange activities in any system. They have effectively been utilized to tackle numerous issues identified with pattern recognition, information mining and examination is as yet in progress concerning intrusion detection. Quick network convergence and unsupervised learning are the potential solutions that can help in designing better neural network based IDS systems.

TABLE V
HYPER-PARAMETER TUNING

Hyper-parameter	***Value****
activation	relu
alpha	0.0001
hidden_layer_sizes	(10, 30, 10)
learning_rate	adaptive
solver	adam

*Values in the best combination of parameters

For our analysis, we utilized grid search algorithm for hyper-parameter tuning. The variables optimized using Grid Search are activation, learning rate, hidden layers, learning type and solver. Table V shows the optimum values of the hyper-parameters obtained using the Grid Search algorithm.

TABLE VI
RESULTS

Metric	***Score***
ROC AUC score	0.760601
Average precision score	0.195958
Mean Squared error	0.352016
Macro F1 score	0.48963
Micro F1 score	0.647984

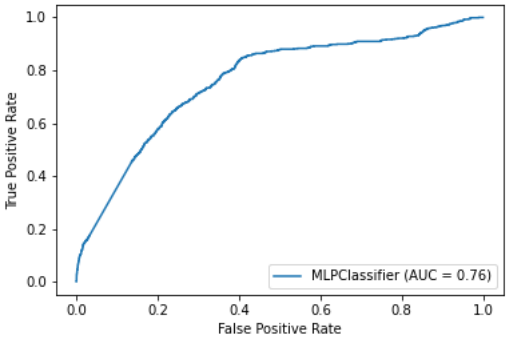

Fig. 6. ROC curve

Using the optimum set of hyper-parameters we were able to achieve `roc_auc_score` as high as 0.760601 within just 1 min of training of our model. Although the improvement in scores seems less, but actually the model has shown a lot of improvement than both of the previous models. From Fig 7 we can observe that the model not only shows an improvement in true positive rate, but also shows significant reduction in false negative rate contrary to the SVM model.

The major benefits of using a neural network is that they are effective at recognizing patterns. Also, settings of a neural network can be adapted to varying circumstances and demands. This makes them a potential solution for intrusion detection. As the tools and techniques of intruders keep on changing and we have to make an adaptable solution. In addition to this, attacks generally form a certain sort of pattern making neural network a powerful tool against them.

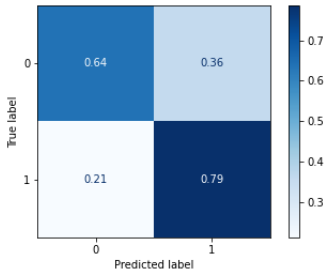

Fig. 7. Confussion Matrix

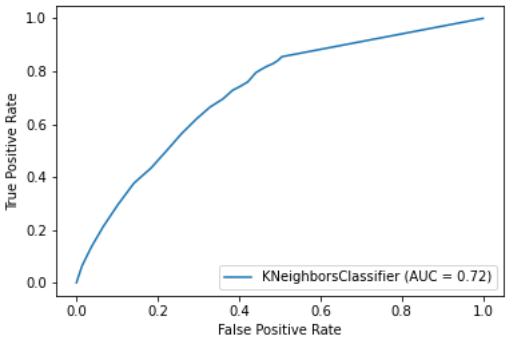

Fig. 8. ROC curve

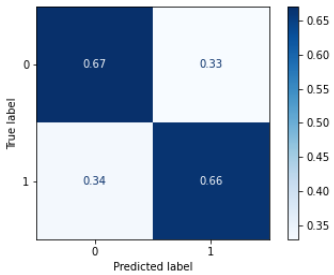

Fig. 9. Confussion Matrix

There are a few limitations too. The quality of the outcome depends greatly on the quality of the data used in the learning phase. The user has little influence on the function of the network (weighting and thresholds). This makes the use of neural network a little difficult in real world.

D. KNN Classifier

K- Nearest Neighbors or also known as K-NN belong to the family of supervised machine learning algorithms which means we use labeled (Target Variable) dataset to predict the class of new data point. The K-NN algorithm is a robust classifier which is often used as a benchmark for more complex classifiers such as Artificial Neural Network (ANN) or Support vector machine (SVM).

After hyper-tuning of the n_neighbours parameter in the KNeighborsClassifier it was found that 20 was the optimum value in considering accuracy and cost. After training the model using the optimum parameter we were able to achieve `roc_auc_score` as high as 0.720903 with a small training time 14 sec. Table VII shows the results obtained for the model. This model has not shown a significant improvement in results than the previous model but a peculiar inference is that it is more balanced than other models - Fig 9. It is not biased towards any of the classes and is able to maintain a good proportion between the true positives and true negatives.

TABLE VII
RESULTS

Metric	***Score***
ROC AUC score	0.720903
Average precision score	0.12532
Mean Squared error	0.329276
Macro F1 score	0.491236
Micro F1 score	0.670724

The major benefits of KNN classifier is that it is pretty simple and involves no implicit assumptions. Also, kNN is a memory-based approach. The classifier immediately adapts as we collect new training data. This makes it suitable for intrusion detection problems where adapting with the new data is one of our major concerns.

Even though KNN has several advantages but there are certain very important disadvantages or constraints of KNN. KNN works well with small number of input variables but as the numbers of variables grow KNN algorithm struggles to predict the output of new data point. Apart from this one of the major struggles is that KNN doesn't perform well on imbalanced data. If we consider two classes, A and B, and the majority of the training data is labeled as A, then the model will ultimately give a lot of preference to A. This might result in getting the less common class B wrongly classified.

E. Adaboost Classifier with Random Forests as base classfier

Boosting is a algorithm that helps in lessening difference and inclination in a machine learning ensemble. The algorithm helps in the transformation of weak learners into strong learners by consolidating N number of learners. Adaboost targets joining a few feeble learners to frame a solitary solid learner. It focuses on weaker learners, which are regularly choice trees with just one split and are generally alluded to as choice stumps. The main choice stump in Adaboost contains perceptions that are weighted similarly. Past mistakes are revised, and any observations that were grouped erroneously are appointed more weight than different perceptions that had no erroer in classification.

For this experminent, we have used adaboost as the main classifier and random forest as the weaker base classifier. The decision rules by the base estimator are provided for both categorical and continuous features. By combining the decision rules from weak classifier with the main learner, the relations between the features are handled more precisely

with proper weightage [7]. For the tuning of the hyper-parameters of both of them we have used Grid Search over all the parameter space and the best combination is chosen for training our model. The variables optimized using Grid Search are n_estimators, max_depth, random_state=0 for base random forest estimator and algorithm, n_estimators, learning_rate for adaboost classifier.

TABLE VIII
HYPER-PARAMETER TUNING

Hyper-parameter	***Value****
For base estimator	
n_estimators	2500
max_depth	80
random_state	0
For main classifier	
algorithm	SAMME.R
n_estimators	100
learning_rate	0.5

*Values in the best combination of parameters

Using the optimum set of hyper-parameters we were able to achieve `roc_auc_score` as high as with a small training time of 3 min. Table IX shows the results of the model. The model showed a significant improvement from the previous models. Also from Fig 10 we can observe that the model has been successful in minimizing the problem of false positives along with achieving high precision score.

TABLE IX
RESULTS

Metric	***Score***
ROC AUC score	0.92222
Average precision score	0.467119
Mean Squared error	0.052075
Macro F1 score	0.639627
Micro F1 score	0.947925

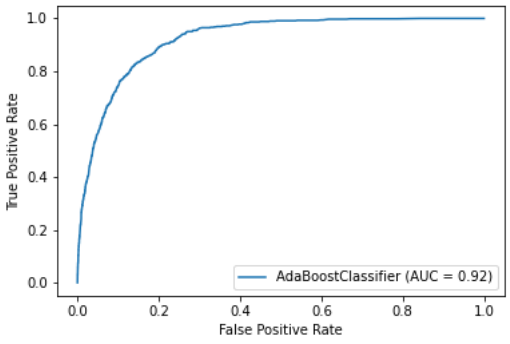

Fig. 10. ROC curve

The advantages of Adaboost incorporate that it is not prone to over-fitting in light of the stage-wise estimation which slows down the learning process. Likewise, AdaBoost can be utilized to improve the exactness of the weak classifiers henceforth making it adaptable to new changes.

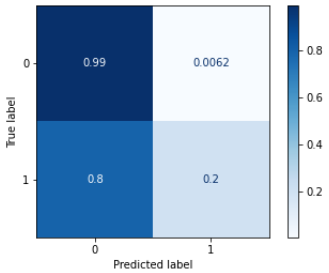

Fig. 11. Confussion Matrix

Adding to this, Boosting strategy adapts continuously, hence it becomes important that you have quality information. AdaBoost is also extremely sensitive to Noisy data and anomalies so when utilizing Adaboost, we must get rid of them.

IV. RELATED WORK

Using ML and DL approaches for intrusion detection has become a vital area of exploration in recent years. In lieu of which a few tecnhiques have been worked to improve different models. Researchers have created strategies to improve the accuracy, detection rate and decrease false positives of intrusion detection systems.

The paper by C. Yin and Y. Zhu [8], proposes a profound learning approach for intrusion detection utilizing Recurrent Neural Networks. RNN can recall past data and use it in for computation of newer ones which makes it more remarkable than other DL.

The paper by H. Zhang and C. Q. Wu [9], proposes a profound learning-based methodology utilizing denoising auto-encoder. It filters the significant features for decreasing dimensionality, utilizing a weight loss function. This is followed by the utilization of a MLP (multi-layer perceptron) for classification of data. The model constructed had exceptionally low memory and calculation necessities. Instead of these low prerequisites, the outcomes obtained utilizing this were astounding.

In the paper by M. Al-Qatf and Y. Lasheng a viable deep learning approach [10], self-trained learning (STL)- IDS is proposed. It consolidates Sparse Auto-Encoder (SAE) with SVM. This improves the accuracy of SVM as well as takes care of the issue of high training time in SVM. The comparison of this method is done with various classfication methods, for example, Naive Bayes, Random Forest, MLP, etc on KDD dataset.

V. CONCLUSION AND FUTURE WORK

The above analysis shows us that SVM doesn't fit as well as other solutions. SVM has the strength of beating others when we need to focus on the marginal distinction between our examples. Since, SVM doesn't seem to outperform others, we can infer that our dataset doesn't have a place with that case. Yet, in real life scenarios this case is very vital to investigate, as attackers consistently attempt to bring their attacks as close as

conceivable to typical traffic. Consequently, SVM with a little bit tinkering utilizing decision trees (used for feature reduction, which will help overcome the high training time) turns into a potential solution in such scenarios

Then again Naive Bayes is absolutely inverse to SVM. It takes a heavy assumption of independency of properties, which makes it incredibly quick. In network interruption datasets we have numerous traits which are really free of one another like attributes origin of attack, which settles on Naive Bayes a decent decision. But, parallely there are a few attributes, like the attributes related to type of message, which are profoundly related to one another. Here we may actually loose certain significant data utilizing that assumption. The solution to this is to use a hybrid model utilizing Naive Bayes and Decision Table Classifier. The other property of attacks is

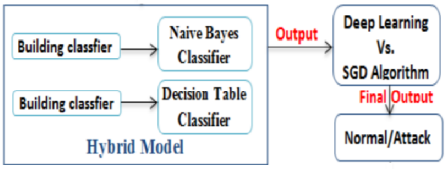

Fig. 12. Proposed Model

that they generally form patterns and Neural Networks are masters in perceiving patterns. For our dataset also, the neural organizations showed great outcomes and took a lot lesser time than SVM. The only limitation is that for Neural Networks to play out their best, they require a lot of information. Furthermore, by and large the records of attacks are a lot lesser, so it gets hard for them to perceive patterns. Here what should be possible is to make some counterfeit data for attacks utilizing some generation algorithm.

Coming up to our next proposition KNN classifier.KNN being a memory based classifier give us a portion of the advantages of neural networks alongside an extraordinary benefit of easy adaptability. The classifier quickly adjusts as we gather new trainig data. This makes it reasonable for intrusion detection problems where adjusting with the new information is one of our significant concerns.

Finally we have proposed the last call to Adaboost technique. As we have examined above single calculations are fruitless for our errand. Adaboost outperforms them by utilizing numerous weak learners for accomplishing our objective. By combining the decision rules from weak classifier with the main learner, the relations between the attributes are taken care of more precisely with legitimate weightage. Not just this, the advantage of variation is also their in adaboost. It adjusts continously as we kick in new information, making it the best performing model in our trials.

REFERENCES

[1] M. Almseidin, M. Alzubi, S. Kovacs and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), 2017, pp. 000277-000282, doi: 10.1109/SISY.2017.8080566.

[2] Ahmad, Zeeshan Khan, Adnan Shiang, Cheah Ahmad, Farhan. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies. 32. 10.1002/ett.4150.

[3] Potdar, Kedar Pardawala, Taher Pai, Chinmay. (2017). A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. International Journal of Computer Applications. 175. 7-9. 10.5120/ijca2017915495.

[4] Mulay, Snehal Devale, P.R. Garje, Goraksh. (2010). Intrusion Detection System Using Support Vector Machine and Decision Tree. International Journal of Computer Applications. 3. 10.5120/758-993.

[5] F. Gumus, C. O. Sakar, Z. Erdem and O. Kursun, "Online Naive Bayes classification for network intrusion detection," 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014), 2014, pp. 670-674, doi: 10.1109/ASONAM.2014.6921657.

[6] J. Shun and H. A. Malki, "Network Intrusion Detection System Using Neural Networks," 2008 Fourth International Conference on Natural Computation, 2008, pp. 242-246, doi: 10.1109/ICNC.2008.900.

[7] Hu, Weiming Hu, Wei Maybank, Steve. (2008). AdaBoost-Based Algorithm for Network Intrusion Detection. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society. 38. 577-83. 10.1109/TSMCB.2007.914695.

[8] C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," in IEEE Access, vol. 5, pp. 21954-21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[9] H. Zhang, C. Q. Wu, S. Gao, Z. Wang, Y. Xu and Y. Liu, "An Effective Deep Learning Based Scheme for Network Intrusion Detection," 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 682-687, doi: 10.1109/ICPR.2018.8546162.

[10] M. Al-Qatf, Y. Lasheng, M. Al-Habib and K. Al-Sabahi, "Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection," in IEEE Access, vol. 6, pp. 52843-52856, 2018, doi: 10.1109/ACCESS.2018.2869577.