Optimized Intrusion Detection System with Feature Extraction for effective Network Traffic Classification

With the increase in accessibility of internet, it became possible to fetch and transfer data and information over multiple networks easily. But the spike in network related threats such as worms, malware, viruses, etc. has demanded the necessity of a system which can properly monitor the network activity to detect threats that interrupt the security systems. Intrusion Detection System (IDS) is one such system that properly monitor and study the pattern of network traffic which is vulnerable to malicious activity and makes the operators vigilant by alarms or alerts. This system works on the principle that the conduct of a normal user and an intruder is different, which makes it possible to distinguish normal and malicious activities. So, an efficient IDS system which can accurately detect the attacks with proper feature selection for minimizing False Alarm Rate (FAR) and thereby maximizing the detection rate is highly on demand.

1. Problem under analysis

Enhancing the detection rate of IDS with optimized FAR

An authentic IDS can be established with a reliable feature selection technique. Several features can be extracted from data packets. Efficient detection of attacks requires identification of relevant features. This can be extracted using machine learning algorithms. But in such cases, there is chance that some intrusion attacks are misclassified as normal, and some normal flows are misclassified as intrusions and thereby increasing the FAR.

Hence, in this work, feature extraction is done with XGBoost and Autoencoder, and classification is done with TabNet. The proposed method is compared with four classifier models - XGBoost, Dense Neural Network (DNN), Convolutional Neural Network (CNN), and Temporal Convolution Network (TCN). Results prove that XGBoost feature extraction is a more efficient method for reliable feature extraction when compared to Autoencoder. Temporal relations in the dataset are also analyzed. The proposed model, XGBoost feature extraction, and TabNet-based classification provided maximum detection accuracy in UNSW-NB15 and NSL-KDD datasets.

2. Contribution

The contribution of this work includes the following: -

1. Two standard datasets: - UNSW-NB15 and NSL-KDD, were used for experimentation with the proposed system.
2. Select the best encoding technique.
3. XGBoost and Autoencoder-based feature extraction.
4. Implementing TabNet classifier model for classifying intrusive and normal traffic.
5. Comparison of the proposed model with XGBoost, DNN, CNN, and TCN classifiers.

3. Proposed system architecture for network traffic classification

The proposed system architecture for effective network traffic classification is shown in fig 3.1. Data preprocessing, parameter tuning, feature extraction, and classification are the four essential tasks for accomplishing this work. The steps include: -

1. Data preprocessing includes encoding, resampling, and normalization.
2. Perform parameter tuning to assign the best values to the model.
3. Find the best correlations and develop the feature score graph using the feature extraction technique.
4. Apply features to the classification model.
5. Compare their performance and evaluate the FAR with performance metrics like accuracy, precision, recall, F1-score, and ROC-AUC score.

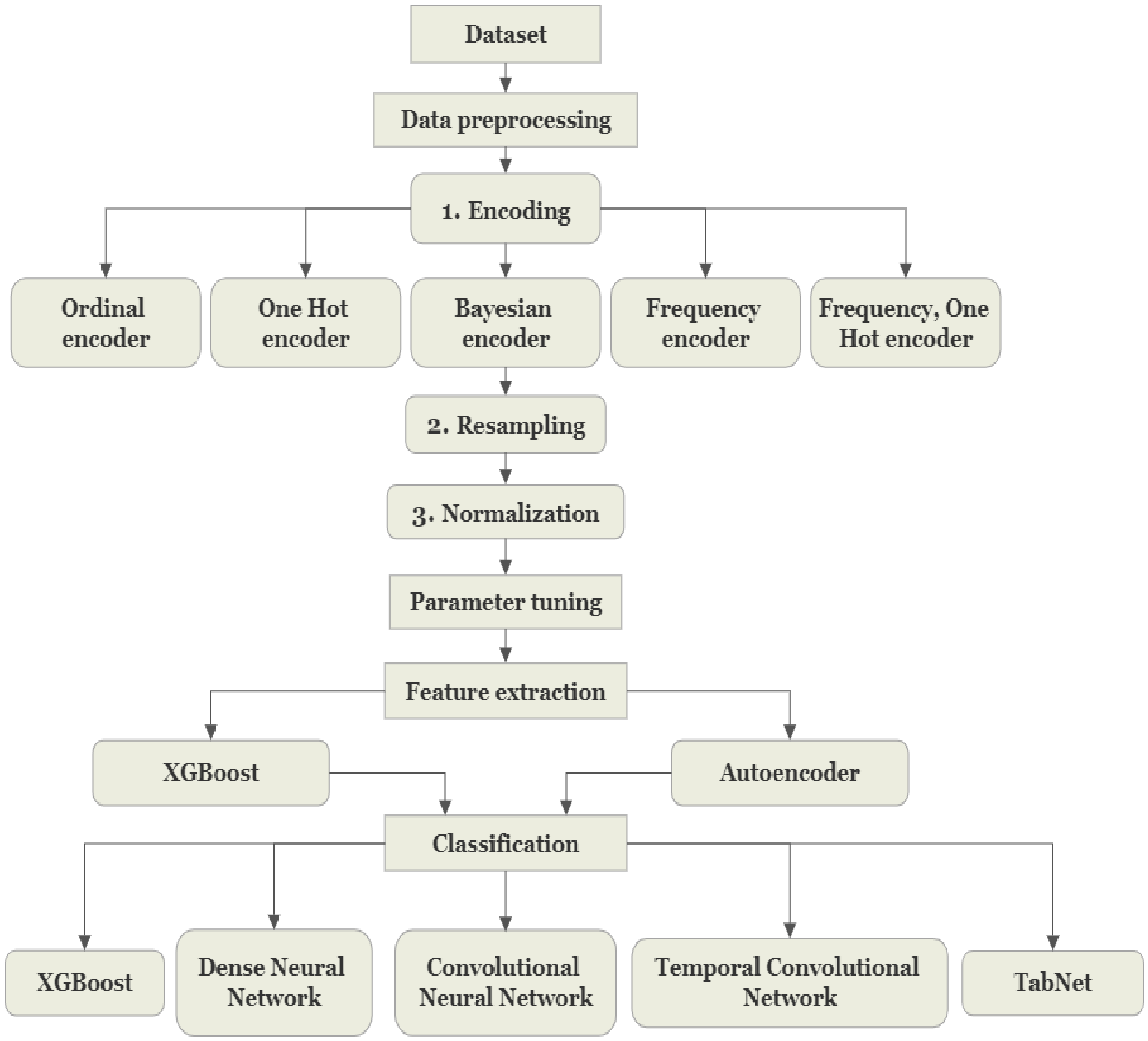

Fig 3.1: Proposed system architecture

4. Experimental setup and results

The hardware used for the experiments includes Windows 10 Pro OS, 64-bit operating system, x64-based processor, Intel(R) Core (TM) i3-5005U CPU @ 2.00GHz, 2.00 GHz, 4 GB RAM. The experimental environment was prepared by using Python 3.7 programming language. The framework used is Keras with TensorFlow as background in the Anaconda environment. Machine learning and deep learning libraries include - NumPy, Pandas, Matplotlib, and Scikit learn. Performance analysis identifies the best model having the highest detection rate. The general evaluation metrics such as Accuracy, Precision, Recall, F1 score, ROC AUC score, and confusion matrix are used.

4.1 Experiments on data preprocessing

Encoding is one of the main steps in data preprocessing. Every encoder has its characteristic feature, distinguishing it from every other encoder. So, to identify which characteristic feature suits our application, experiments on encoding are done with five different combinations: - Ordinal encoding, one-hot encoding, frequency encoding, combining one hot and frequency encoding, and Bayesian encoding. Initially, the required libraries and the dataset is loaded. Missing values and count of categories in each categorical feature column are identified. Each of these categorical features is encoded separately with five encoder combinations and is saved in a separate csv file. Without further preprocessing, the five sets of the UNSW-NB15 dataset and another five sets of the NSL-KDD dataset are applied to the XGBoost classifier model to identify the best encoder. Table 4.1 shows the performance metric values obtained after encoding. As shown in this table, the Bayesian encoder gives the highest accuracy.

Processing tool	**Accuracy (%)**		**Precision**		**Recall**		**F1-Score**	
	UNSW-NB15 dataset	NSL-KDD dataset	UNSW-NB15 dataset	NSL-KDD dataset	UNSW-NB15 dataset	NSL-KDD dataset	UNSW-NB15 dataset	NSL-KDD dataset
Ordinal Encoder	55.06	77.44	1.0	0.8360	0.5506	0.7744	0.3551	0.7735
One Hot and Frequency encoder	87.59	77.59	0.8915	0.8371	0.8759	0.7759	0.8704	0.7750
One Hot Encoder	87.61	78.21	0.8914	0.8404	0.8761	0.7820	0.8706	0.7813
Frequency encoder	87.72	77.37	0.8928	0.8354	0.8772	0.7737	0.8717	0.7728
Bayesian encoder	**90.12**	**79.79**	**0.9071**	**0.8481**	**0.9012**	**0.7979**	**0.8984**	**0.7976**

Table 4.1: Experimental result of data preprocessing on different encoders

The Bayesian encoder provides good results because of its ability to study how a feature is dependent on the target data. Compared to other classic encoders where the relation between

each category within the same feature is considered, target or Bayesian encoding evaluates the mean value of a particular category based on its occurrence with the target. Hence, it is the best among all the combinations in both datasets. Since Bayesian encoding is proved to be robust, Bayesian encoded UNSW-NB15, and Bayesian encoded NSL-KDD dataset is used for further experimentation.
The next step in data preprocessing is resampling. In UNSW-NB15 dataset, there are 119,341 "attack" instances and 56,000 "normal" instances and in NSL-KDD dataset, there are 67,343 "attack" instances and 58,630 "normal" instances. After performing SMOTE-based resampling, the number of training instances in the UNSW-NB15 and NSL-KDD datasets increased from 175,341 to 238,682 and 125,973 to 134,686, respectively, i.e., the minority class label is now in proportion with the majority class.
The final step in data preprocessing includes normalizing the data. Both UNSW-NB15 and NSL-KDD dataset is normalized to improve the detection accuracy. The new range of normalization is fixed as $\text{range}_{\min} = 0$ and $\text{range}_{\max} = 1$.

4.2 Experiments on parameter tuning

XGBoost has some intrinsic characteristics, which make it robust and efficient. But parameter tuning is done to identify the best parameters which will offer a sound detection rate corresponding to a particular dataset, here UNSW-NB15 and NSL-KDD dataset. Grid search algorithm is utilized for the same. The n-estimators are successfully identified by setting the initial n-estimate as ‘4000’ and ‘1000’ in UNSW-NB15 and NSL-KDD datasets, respectively, based on the total number of data samples in each dataset. Initially, the model is created by assigning random values. Finally, parameters like ‘max_depth’, ’min_child_weight’, ’gamma’, ‘subsample’, ‘col_sample_bytree’, and ‘alpha’ is tuned. Table 4.2 shows the values obtained after tuning parameters for both datasets.

Parameter	Value Obtained	
	UNSW-NB15 Dataset	NSL-KDD Dataset
n_estimators	1484	381
max_depth	4	5
min_child_weight	5	8
Gamma	0.4	0.0
Subsample	0.6	0.9
col_sample_bytree	0.6	0.6
Alpha	0.05	0.005

Table 4.2: Parameter tuning

4.3 Experiments on feature extraction

4.3.1 Feature extraction with XGBoost

After loading Bayesian encoded training and testing data with pandas, both dataset’s feature score graph and f-score value are obtained. The sort () imported from NumPy identified the thresholds in ascending order. Figs 4.1 and 4.2 show the feature importance graph of UNSW-NB15 and NSL-KDD datasets.

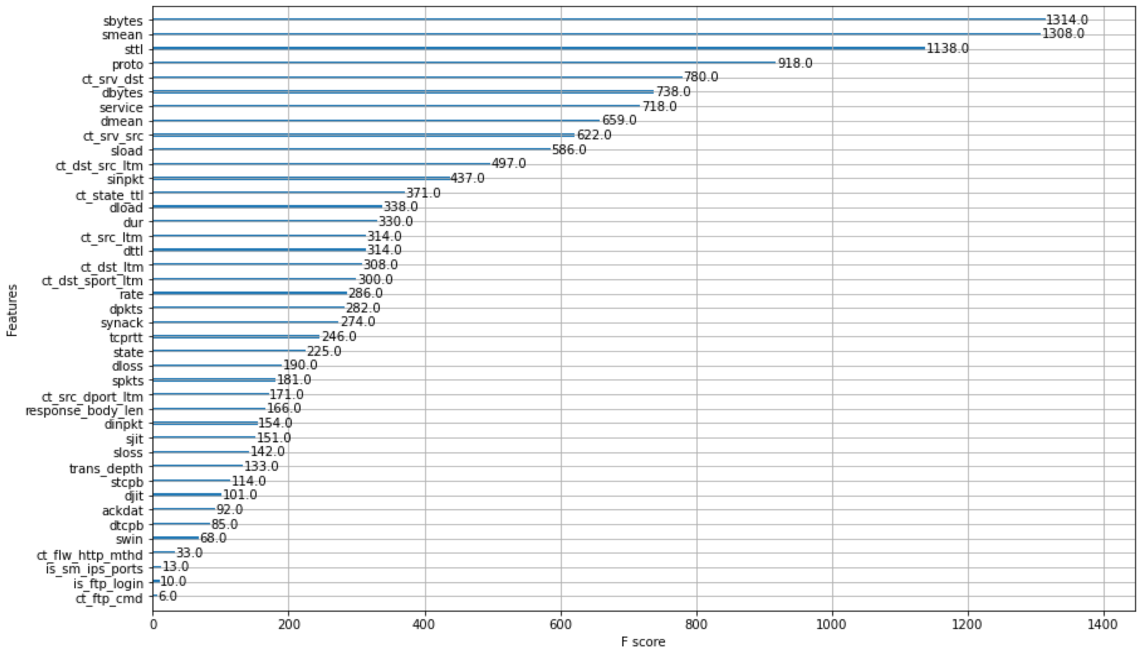

Fig 4.1: Feature importance graph of UNSW-NB15 dataset

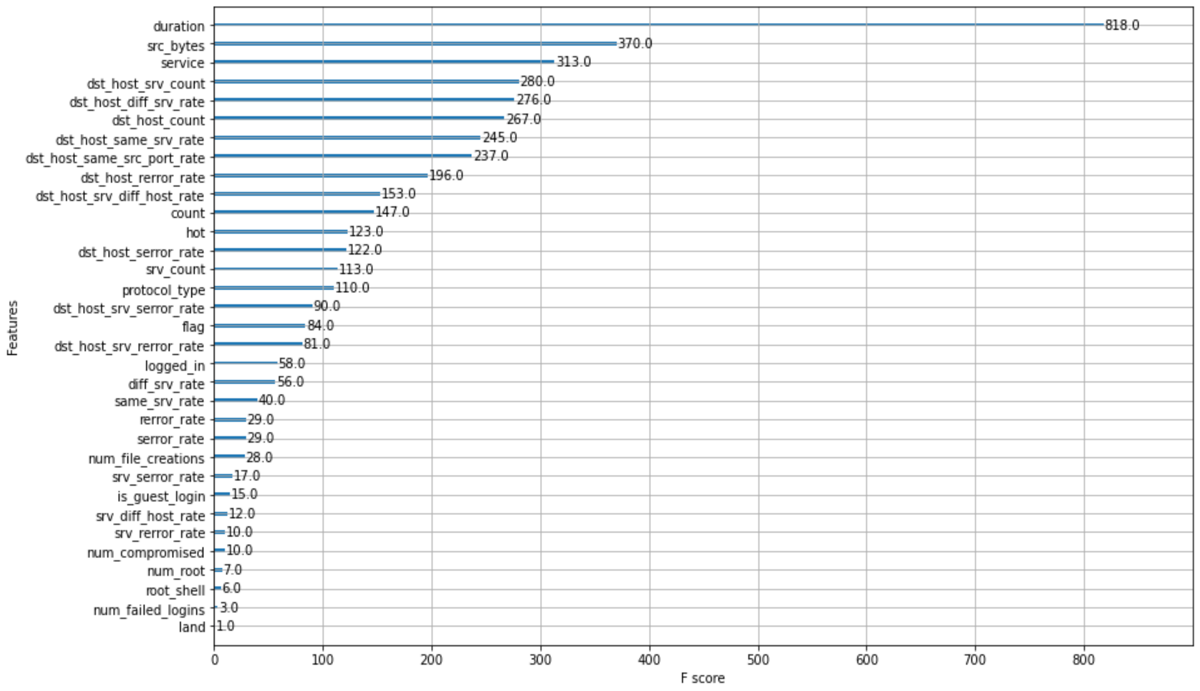

Fig 4.2: Feature importance graph of NSL-KDD dataset

Tables 4.3 and 4.4 show the result of XGBoost feature selection with different thresholds. With the decrease in the number of features, the model's performance decreases. Hence, we need to find the highest accuracy with the optimum number of features selected. The highest accuracy of

90.53% with a threshold = 0.016 and 15 selected features is the best combination in the UNSW-NB15 dataset. Similarly, in the NSL-KDD dataset highest accuracy of 82.10% with a threshold = 0.065 and 4 selected features are the best combination.

Threshold	Number of features	Accuracy (%)
0.000	42	90.20
0.001	40	90.22
0.002	38	90.20
0.003	33	90.39
0.004	26	90.48
0.004	24	90.34
0.004	23	90.37
0.004	20	90.18
0.016	**15**	**90.53**
0.016	11	86.91
0.016	6	81.03
0.016	2	80.57

Table 4.3: Experimental result of XGBoost feature selection with different thresholds in UNSW-NB15 Dataset

Threshold	Number of features	Accuracy (%)
0.000	40	79.46
0.000	33	80.14
0.002	31	79.58
0.002	29	80.23
0.003	26	80.31
0.005	24	80.47
0.006	20	80.38
0.006	15	79.76
0.023	11	80.57
0.023	7	81.27
0.065	**4**	**82.10**
0.065	2	81.34

Table 4.4: Experimental result of XGBoost feature selection with different thresholds in NSL-KDD Dataset

4.3.2 Feature extraction with Autoencoder

Imported all the required packages and datasets. Created input, encoder, bottleneck, decoder, and an output layer of the autoencoder. Input layers consist of units as the number of features in the UNSW-NB15 and NSL-KDD dataset. 2 encoder layers, each with a dense layer, batch normalization layer, and leaky ReLU layer. The first layer contains 86 units, and the second layer contains 43 units. The bottleneck layer (dense layer) has 21 units. The decoder part has two layers: a dense layer, a batch normalization layer, and a leaky ReLU layer. Output layer with Linear activation function, optimizer = 'adam', loss='mse'. Trained the model and then saved the model with output as a bottleneck. Fit the saved autoencoder model data into the XGBoost classifier model and perform classification to identify the best feature extraction algorithm. Table 4.5 shows both dataset's Autoencoder and XGBoost feature selection model results. Accuracy of 85.32% and 75.94% is obtained in UNSW-NB15 and NSL-KDD datasets, respectively.

Feature extraction	**Accuracy (%)**		**Precision**		**Recall**		**F1-Score**	
	UNSW-NB15 dataset	NSL-KDD dataset	UNSW-NB15 dataset	NSL-KDD dataset	UNSW-NB15 dataset	NSL-KDD dataset	UNSW-NB15 dataset	NSL-KDD dataset
Auto Encoder	85.32	75.94	0.8620	0.8079	0.8532	0.7594	0.8477	0.7589
XGBoost	**90.53**	**82.10**	**0.9116**	**0.8568**	**0.9053**	**0.8210**	**0.9025**	**0.8209**

Table 4.5: Experimental result of Autoencoder and XGBoost feature extraction model in UNSW-NB15 and NSL-KDD dataset

From table 4.5, the XGBoost feature extraction model provides the highest detection accuracy in both datasets. This is because, for structured tabular data, XGBoost consistently outperforms. Just applying normalization to tabular data and training a model, XGBoost gives better results. XGBoost can minimize the loss of what is learned and add a new tree to it. This sequential learning allows XGBoost to extract features more efficiently. But in the case of unstructured data, neural networks outperform. Hence, we can infer that XGBoost feature selection is more effective than autoencoder-based feature extraction in intrusion datasets.

4.4 Experiments on classification and comparative analysis

The detection rate with the proposed model XGBoost feature selection and TabNet classification technique yields a maximum accuracy of 93.02% and 88.35% compared to all the models in the UNSW-NB15 and NSL-KDD dataset. Structured tabular data and a normalization help XGBoost perform well. This is because of its ability to minimize the loss of what is learned and then add a new tree. Autoencoder requires further optimization to enhance the effectiveness of feature extraction in tabular data.

TabNet is a tabular deep learning model. Each decision step employs sequential attention to select a subset of relevant features and process them locally and globally, i.e., for a single input and the whole dataset. TabNet model provides more accuracy when compared to XGBoost, DNN, CNN, and TCN. The 2-stage feature selector in the TabNet model helps in more robust classification. Hence, even though XGBoost provides good feature extraction results, in classification, TabNet outperforms. Nowadays, the TabNet model is proving to be more effective than gradient boosting algorithms for classification purposes. Similarly, after Autoencoder feature extraction, the TabNet model proved to be the best when the classification was done.

Fig 4.3 and 4.4 shows the comparative analysis among the classification categories: - Before feature selection, XGBoost feature selection, and Autoencoder feature selection. From this graph, we can infer that performance on classification is higher after XGBoost feature selection. Autoencoder feature selection performance results are less than results of classification before feature selection. TCN also proved to provide good accuracy even though not the best, indicating that the classification model also extracts the temporal relations. Causal convolutions, dilations, and skip connections incorporated in the network allow the use of present and just past inputs to predict an output. This allows the model to learn and consider the whole history of inputs.

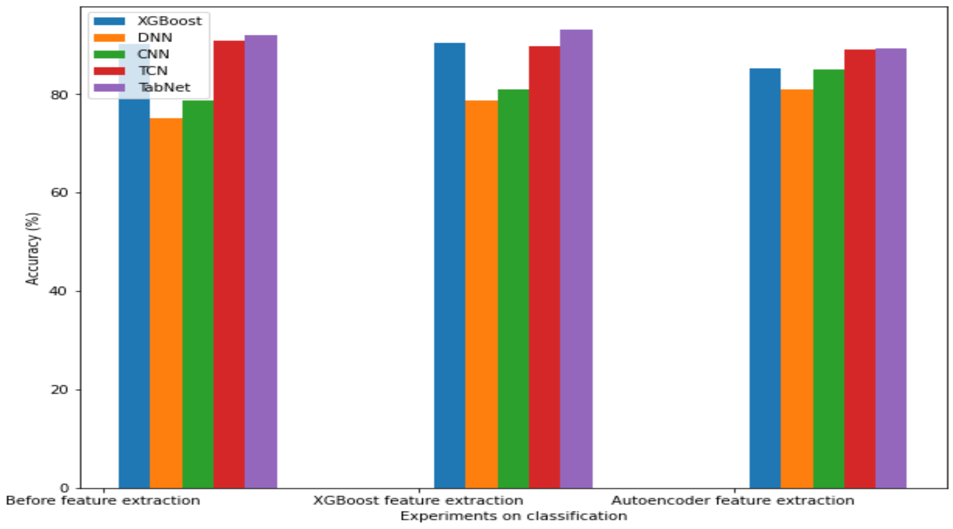

Fig 4.3: Comparison of classification experiments in UNSW-NB15 dataset

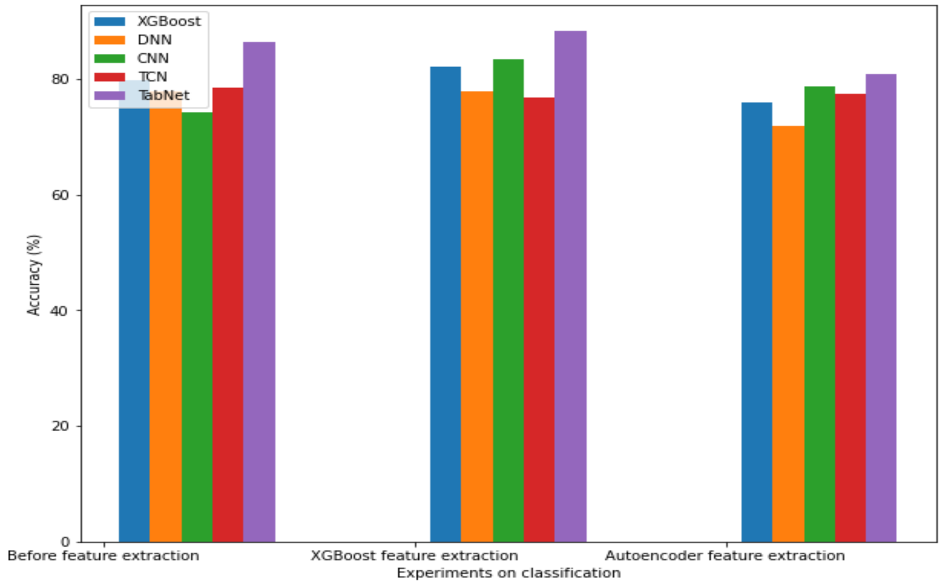

Fig 4.4: Comparison of classification experiments in NSL-KDD dataset

5. Inference

- Preprocessing with a Bayesian-encoder yields the highest classification accuracy of 90.12% and 79.79% in the UNSW-NB15 and NSL-KDD datasets. This is because the Bayesian encoder could accurately map the relation between a categorical variable and a target variable which was not possible with other encoders.
- Features extracted from XGBoost and Autoencoder model are fed as input to XGBoost classifier, DNN, CNN, TCN, and TabNet. Results indicate that XGBoost outperforms in extracting features efficiently from the data. In structured tabular data, XGBoost gives better results because of its

ability to minimize the error and thereby consider relevant features while passing through each subsequent tree sequentially.

- TabNet is proven to give good classification results because of its 2-stage feature transformer and mask that ensures only relevant features are passed to the subsequent blocks. So, it is evident that TabNet performs better than gradient boosting algorithms.
- Promising results from TCN also prove the relevance of temporal features in the data.
- The proposed model, XGBoost feature selection, and TabNet classification algorithm proved to be the best, with a maximum detection rate of 93.02% and 84.22% in UNSW-NB15 and NSL-KDD datasets, respectively.

REFERENCES

[1] Devan P, Khare N. An efficient XGBoost–DNN-based classification model for network intrusion detection system. Neural Computing and Applications. 2020 Jan 19:1-6.

[2] Kasongo SM, Sun Y. A deep long short-term memory-based classifier for wireless intrusion detection system. ICT Express. 2020 Jun 1;6(2):98-103.

[3] Moustakidis S, Karlsson P. A novel feature extraction methodology using Siamese convolutional neural networks for intrusion detection. Cybersecurity. 2020 Dec;3(1):1-3.

[4] Alhajjar E, Maxwell P, Bastian N. Adversarial machine learning in network intrusion detection systems. Expert Systems with Applications. 2021 Dec 30;186:115782.

[5] Nguyen MT, Kim K. Genetic convolutional neural network for intrusion detection systems. Future Generation Computer Systems. 2020 Dec 1;113:418-27.

[6] Rajagopal S, Kundapur PP, Hareesha KS. A stacking ensemble for network intrusion detection using heterogeneous datasets. Security and Communication Networks. 2020 Jan 24;2020.

[7] Aslahi-Shahri BM, Rahmani R, Chizari M, Maralani A, Eslami M, Golkar MJ, Ebrahimi A. A hybrid method consisting of GA and SVM for intrusion detection system. Neural computing and applications. 2016 Aug;27(6):1669-76.

[8] Hsu CM, Hsieh HY, Prakosa SW, Azhari MZ, Leu JS. Using long-short-term memory based convolutional neural networks for network intrusion detection. In International wireless internet conference 2018 Oct 15 (pp. 86-94). Springer, Cham.

[9] Gao J, Gan L, Buschendorf F, Zhang L, Liu H, Li P, Dong X, Lu T. Omni SCADA intrusion detection using deep learning algorithms. IEEE Internet of Things Journal. 2020 Jul 14;8(2):951-61.

Guide details

Rajeev Azhuvath
Mentor
Tata Consultancy Services
Email: rajeev.azhuvath@tcs.com

Prof. Sumod Sundar
TKM College of Engineering, Kollam
Email: sumodsundar@tkmce.ac.in
Mob: 8086515716