AN EFFICIENT XGBOOST–TABNET BASED CLASSIFICATION METHOD FOR NETWORK INTRUSION DETECTION SYSTEM

A Project Report

Submitted by

Ms. T SARANYA

REG NO : TKM20MEAI16

SEMESTER : IV

In partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY
IN
Mechanical Engineering (Artificial Intelligence)

Under the guidance of
Prof. SUMOD SUNDAR

Thangal Kunju Musaliar College of Engineering
Kerala

JULY 2022

DECLARATION

I undersigned hereby declare that the project report "An efficient XGBoost–TabNet based classification method for Network Intrusion Detection System", submitted for partial fulfillment of the requirements for the award of degree of Master of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Prof. Sumod Sundar. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Kollam
Date: T SARANYA

Thangal Kunju Musaliar College of Engineering
Centre for Artificial Intelligence

C E R T I F I C A T E

This is to certify that, this report titled ***AN EFFICIENT XGBOOST–TABNET BASED CLASSIFICATION METHOD FOR NETWORK INTRUSION DETECTION SYSTEM*** is a bonafide record of the **Project** presented by **T SARANYA (TKM20MEAI16)**, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **M.Tech in Mechanical Engineering (Artificial Intelligence)** in **APJ Abdul Kalam Technological University** .

Project coordinator & Internal supervisor

Prof. Sumod Sundar
Assistant Professor
Centre for Artificial Intelligence

Head of the Department

Dr. Imthias Ahamed
Professor & HOD
Centre for Artificial Intelligence

Internal examiner

External examiner

ACKNOWLEDGEMENT

A successful project is a fruitful culmination of efforts by many people, some directly involved and some others indirectly, by providing support and encouragement. Firstly I would like to thank the almighty for giving me the wisdom and grace for making my project a successful one. I thank him for steering me to the shore of fulfillment under his protective wings

I express my sincere gratitude to **Dr. T A Shahul Hameed**, Principal of TKMCE, and **Dr. Imthias Ahamed**, Professor and Head of the Department, Centre for Artificial Intelligence, TKMCE, for their constant support and encouragement throughout the project work.

I would like to express my heartfelt thanks to our project coordinator cum internal supervisor **Prof. Sumod Sundar**, Assistant Professor, Centre for Artificial Intelligence, TKMCE, for his expert guidance and cooperation. With a profound sense of gratitude, I would like to thank **Dr. Santhi Natarajan**, Honorary Professor, for her immense encouragement. I would like to express my gratitude to **Mr. Rajeev Azhuvath**, Mentor, Tata Consultancy Services (TCS), for his expert guidance, and cooperation. I also extend my thanks to the entire faculty and staff members of the Centre for AI, TKMCE, who has encouraged me throughout this work.

I also express my thanks to my loving parents, brother and friends, for their support and encouragement in the successful completion of this project work.

T SARANYA

ABSTRACT

With the advancement in internet technology, an increase in network-related threats has led to several privacy concerns. Automated Intrusion Detection Systems (IDSs) can detect malicious patterns where firewalls designed using conventional detection techniques fail to identify the relevant set of threat patterns. There are different types of IDS available, but those systems are prone to high False Alarm Rate (FAR) because anomalies can be new legitimate activities. Good quality and surplus network traffic patterns will make IDS systems more effective. Hence to reduce the FAR and enhance detection accuracy, a novel method to extract the best features and classify them as 'normal' and 'intrusive' is proposed. Feature extraction is carried out using XGBoost and Autoencoder techniques and classification is performed using TabNet. The proposed method is compared using four classifier models - XGBoost, Dense Neural Network, Convolutional Neural Network, and Temporal Convolutional Network. Temporal relations underlying the data are also analyzed. XGBoost feature extraction is found more efficient for feature extraction when compared to Autoencoder. Also, TabNet exhibited the top performance while comparing with other classifier models. The experiments are carried out using UNSW-NB15 and NSL-KDD datasets and the performance is evaluated using other techniques available in the literature.

Contents

List of Figures

List of Tables

ABBREVIATIONS

ACCS	Australian Centre for Cyber Security
CNN	Convolutional Neural Network
DDOS	Distributed Denial-Of-Service
DLSTM	Deep Long Short Term Memory
DNN	Dense Neural Network
FAR	False Alarm Rate
GA	Genetic Algorithm
IDS	Intrusion Detection System
IP	Internet Protocol
NIDS	Network Intrusion Detection System
RNN	Recurrent Neural Network
SCNN	Siamese Convolutional Neural Network
SMOTE	Synthetic Minority Oversampling TEchnique
SVM	Support Vector Machine
TCN	Temporal Convolutional Network
XGBoost	eXtreme Gradient Boosting

Chapter 1

INTRODUCTION

1.1 General Background

Internet is a universal system of interconnected networks that use Internet Protocols (IP) to interact between different computer devices and networks. With an increase in accessibility of the internet, it became possible to fetch and transfer data and information over multiple networks easily. But the vital task is to protect such systems and networks from data disclosure. This led to the development of automated techniques in cybersecurity domain. Here security is offered in online mode, and thus confidential data is protected. But the spike in network-related threats such as worms, malware, viruses, etc., demands a system that can adequately monitor the network activity to detect threats that interrupt the security systems.

Intrusion Detection System (IDS) is one such model that properly monitors and studies the pattern of a 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 typical user and an intruder is different, which makes it possible to distinguish between normal and malicious activities. Typically, there are three types of IDS systems [1]. The first type is a signature-based (misuse) IDS which tries to identify the signature patterns in the network traffic data and match them with pre-existing intrusive and non-intrusive traffic data. This model cannot detect new intrusions, resulting in a high false Alarm Rate (FAR). The second type is based on specific rules set by experts. High FAR is a challenge in these systems because anomalies can be legitimate activities. Another type of IDS was developed with the advancements in machine learning techniques. For accurate detection of intrusions in such models, it is necessary to have good quality and quantity of traffic data [2]. So, one of the crucial tasks of researchers while developing IDS is to extract relevant features from such large quantities of data. To identify the best features and improve the detection accuracy, researchers proposed several approaches involving machine learning techniques, deep learning techniques, and hybrid techniques. Techniques of feature extraction and classification using transfer learning approach is also found effective [3].

1.2 Objective

This work focuses on developing an efficient IDS system that can reduce FAR and improve the detection rate in terms of accuracy. Hence, a novel method is proposed using XGBoost

feature extraction and TabNet-based classification to identify the best features and classify them as 'normal' and 'intrusive'. Two standard datasets, UNSW-NB15 and NSL-KDD, are used for experimentation with the proposed system. Initially, all the categorical features in the dataset are encoded with five sets of encoders namely: Ordinal, One Hot, Frequency, Combined one hot and frequency, and Bayesian, and the best encoding technique is investigated. After further data preprocessing, the relevant features are extracted using XGBoost and Autoencoder models. These features are analyzed and fed as input to TabNet classifier for classifying intrusive and normal traffic. The proposed model is compared with XGBoost, Dense Neural Network (DNN), Convolutional Neural Network (CNN), and Temporal Convolutional Network (TCN) classifiers. The performance of the model is also compared with existing machine learning and deep learning techniques.

The remainder of this report is organized into the following chapters: Chapter 2 reviews different machine learning and deep learning techniques for developing automatic IDS. IDS, types and challenges are described in chapter 3. Chapter 4 includes dataset description and detailed methodology. The experimental results after performing various preprocessing, feature extraction, classification algorithms, and comparison with existing techniques are discussed in Chapter 5.

Chapter 2

RELATED WORKS

In this section, several studies of intelligent network IDS utilizing both machine learning and deep learning techniques are discussed.

Devan et al. [2] developed a feature selection model with XGBoost (eXtreme Gradient Boosting) followed by a DNN based classifier. The main problem addressed is it's high FAR. Simple machine learning models require superior domain knowledge for identifying relevant patterns. With deep learning techniques, faster detection of attacks is possible. Hence a model combining both machine learning and deep learning techniques is proposed. NSL-KDD dataset is used for experimentation in this work. More in-depth study of attack classes is yet to be analyzed.

Kasongo et al. [4] proposed a Deep Long Short-Term Memory (DLSTM) based classifier incorporating information gain-based feature scoring. The directional loop in Recurrent Neural Network (RNN) memorize the previous state, but the vanishing gradient issue reduce the accuracy in such models. Forget gate in LSTM, which can omit the irrelevant data, is utilized to develop the model. The NSL-KDD dataset is used to train and evaluate the model. But the work generally focused on the malicious class rather than studying individual classes separately.

Moustakidis et al. [5] developed a deep learning-based Siamese Convolutional Neural Network (SCNN) for extracting the relevant features from surplus network data. This model is proposed to develop a user-friendly risk indicator for identifying cyber-attacks. The NSL-KDD dataset is used to train and evaluate the model. Pre-processing is carried out with a fuzzy allocation scheme whose output is fuzzy values. These fuzzy values are then converted into images with a Vec2im-based modality transformation technique. The output obtained from Vec2im then serves as input to SCNN. The results revealed that the proposed system ensure capability in detecting attacks. But it is difficult to apply the proposed model where decisions should be taken based on complex and heterogenous data.

Alhajjar et al. [6] developed a model to evaluate the sensitivity of various machine learning models when they are undergone adversarial attacks. Adversarial systems are systems that are developed intentionally to fool the original model. So, to test the capacity of simple machine learning classifiers, training is done in both NSL-KDD and UNSW-NB15 datasets. Initially, adversarial examples are generated with Particle Swarm Optimization, Genetic Algorithm (GA), Generative Adversarial Network, and Monte Carlo Simulation. Each model produces malicious vectors, which are then input into classification models, including Support Vector Machine (SVM), Naïve Bayes, Decision Tree, Random Forest, K-Nearest Neighbor,

Multilayer perceptron, Gradient Boosting, Linear regression, Bagging, etc. With the proposed model, SVM and DT are identified as more vulnerable, and hence it is suggested to hold back these models from using it in automatic IDS systems. But it is impossible to analyze why some models are more robust than others as there is no system to identify the internal operations of the machine learning models.

Nguyen et al. [7] proposed an extensive feature selection method with a GA and K-Nearest Neighbor incorporating a 5-fold cross-validation fitness function along with Fuzzy C-means Clustering to develop an Improved Feature Subset (IFS). 3 CNN models are selected, and the IFS is served as input to these models. Further, in-depth features are extracted from these CNN models to form a Deep Feature Subset (DFS). The training and testing are done in the NSL-KDD dataset. Performance comparison is carried out by applying IFS and Original Feature Subset (OFS) in different classification models. But the system consumed more time to implement the GA algorithm and select the best 3 CNN models in the initial phase.

Rajagopal et al. [8] proposed an ensemble-based classification model for IDS systems incorporating hashing and information gain techniques for feature selection. The detection accuracy of hybrid models is comparatively more than simple machine learning models. Random Forest, K-Nearest Neighbor, and Logistic regression are used as the base classifiers in this system, and SVM is chosen as the meta-model. Training and testing are performed in the UNSW-NB15 dataset. The proposed model obtained an accuracy of 92.85%. Since the classifier output is based on the maximum voting scheme, the imbalance in the dataset has not become an issue. Even though FAR is substantially reduced the model takes enough time for processing the data.

Asahi-Shahri et al. [9] developed a genetic engineering-based feature selection model with embedded parameter optimization. Finally, the classification is done with SVM. KDD Cup dataset is used to check the effectiveness of the system. Thereby, the FAR is reduced. But the training time and testing time required are more than other models.

Hsu et al. [10] incorporating two deep learning architectures such as simple LSTM model and CNN – LSTM hybrid model. Both binary and multi-class classification is done using instances from the NSL-KDD dataset. Both models performed much better than a simple RNN-based IDS system. The research proved that incorporating CNN before LSTM improved the accuracy as convolutional layers can extract the local features. LSTM time-series data is also studied.

Gao et al. [11] developed a system that can detect both temporally uncorrelated and correlated attacks. Three approaches were introduced for the temporal feature study. These include: - Feed Forward Neural Network (FNN), LSTM, and an architecture combining FNN and LSTM by ensemble approach. Each data packet has different features of network traffic, and in-packet features such as sequencing data, error codes, etc., are not enough to detect time-correlated attacks. So, from a simple FNN model, the accuracy in detecting correlated attacks is less. The accuracy of detection is increased by using LSTM that predict future timestamps when the previous time stamps are known. But the proposed system did not effectively utilize convolutional layers to study the local features.

The advantages and limitations of 5 recent related works are summarized in table 2.1.

Table 2.1: Review

Reference	Technique used	Advantages	Disadvantages
[2]	Feature selection:- XGBoost and classification:- DNN [2020]	This model reduces overfitting and helps in regularization. It provides faster detection of network intrusions. They also outperforms simple ML models like SVM and Naïve Bayes.	The model is deployed as a simple binary classification problem. More detailed analysis of different works could have been done to deal with several other attacks in depth which includes: - Probe, U2R, R2L, DoS.
[4]	Feature selection:- IG and classification:- SVM, KNN, NB, RF, ANN, FFDNN, DLSTM RNN [2020]	DLSTM RNN method outperforms ML methods and FFDNN. Information gain discover non-linear relationship between variables in the dataset.	Individual classes of attacks in the dataset is not studied.
[5]	Feature selection:- Siamese CNN and classification:- NB, AdaBoost, Random forest, KNN, Decision tree [2020]	The model has increased discrimination capability. This can be potentially used as a risk indicator to identify cyber attacks.	It is challenging to apply the model where decisions should be taken based on complex and heterogeneous data.
[6]	Feature selection:- GA,PSO and GAN and classification:- SVM, Decision tree (DT), NB, KNN, Random Forest, MLP, Gradient Boosting, Linear Regression [2021]	Identified SVM as most vulnerable hence SVM models can be refrained from using in NIDS.	The proposed system cannot analyze the internal operations of ML models and why those models are more robust than some other models.
[7]	Feature selection:- Genetic algorithm (GA) with KNN and classification:- CNN [2020]	With Genetic algorithm high quality feature set are identified. Improved final detection rate. Model is well-fitted in practical computer network environments.	With regard to implementation of GA and identify 3 best fit CNN model the time consumption is more. Repetition of 5-fold cross-validation consumes even more time and power.

Chapter 3

INTRUSION DETECTION SYSTEM

An IDS monitors network traffic patterns and helps to distinguish between legitimate and malicious traffic. As its name suggests, this is used to detect different attacks. An IDS system can be deployed in small scale systems or large scale systems, i.e., its scope can range from single computers to large networks of computers. An attack can be either host-based or network-based. Host-based attacks occur in a single system. Worms, viruses, trojans, and backdoors are some examples. To prevent these attacks, Host-based IDS is developed. They are deployed within the host, and they monitor the network flow, which originated from a particular host only. These systems also analyze the file systems, login/off activities, data processing, etc. Network-based at- tacks occur in an inter-connected network. These can be either a DDOS attack, Internet Protocol (IP) spoofing, Port sniffing or Man-in-the-Middle attack. Now to mitigate such attacks, Network-based IDS is developed. Fig. 3.1 shows different types of IDS. It is positioned in the network so that any attack in the host connected to that particular network will be detected, i.e., it will be placed in the entry and exit point of data from that network.

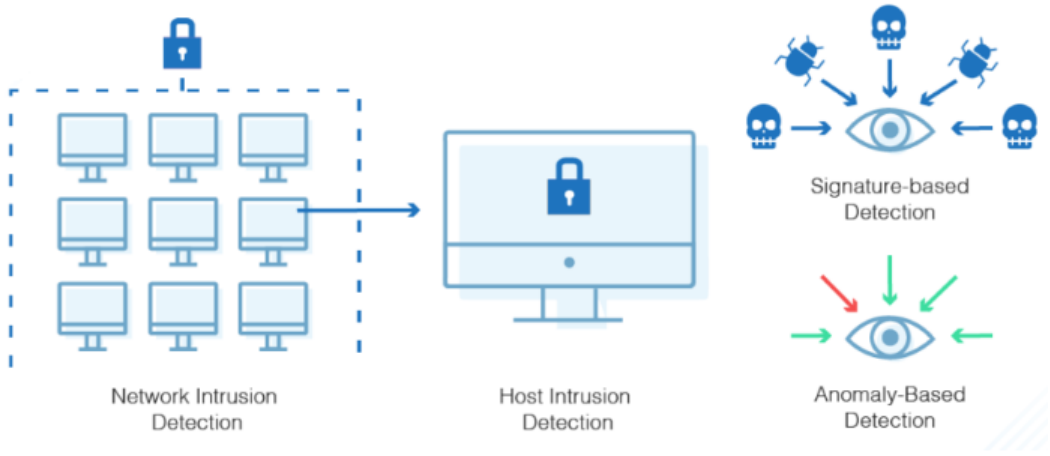

Figure 3.1: Types of IDS

3.1 Challenges of IDS

All these types of IDS can either be signature-based or anomaly-based. Fig. 3.2 shows the flowchart of different IDS techniques. Signature-based IDS, otherwise known as misuse-based IDS system, is developed so that any pre-defined patterns of network traffic will be easily identified. In contrast to that anomaly-based IDS detects attacks based on variations from the normal instances which are already defined. The main disadvantage of a signature-based system is that it will not be able to identify unseen attacks. But the deployment of both techniques could not reduce the problem of false alarms, i.e., all these techniques are based on strict rules, and they are vulnerable to false positive and false negative alarms. Apart from that, the computing cost is high as the network traffic types increase day by day. The network characteristics are becoming complex as the attackers are changing the features periodically.

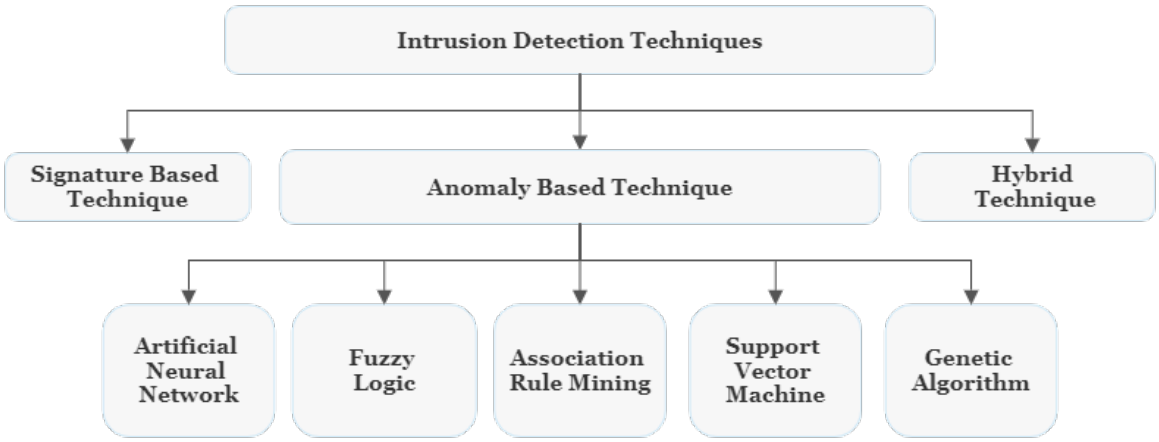

Figure 3.2: Flowchart of IDS techniques

3.2 Network Intrusion Detection System (NIDS)

Conventional NIDS works by building statistical tools or models around the network traffic they monitor, as shown in fig. 3.3. The current network traffic flow will be captured, and this data will be sent to the server. The NIDS server will then process the data to detect the threat. It will detect a threat whenever there is a significant deviation from the baseline models, and once such a threat is detected, it will raise alerts in the form of alarms. The main issue with conventional NIDS is the high false-alarm rates, which reduces the detection rate. As part of improving the system, the researchers started to work on NIDS models. Advancements in machine learning techniques paved the way for making these conventional models intelligent. Thus, intelligent NIDS was developed with shallow machine learning techniques like SVM, Naive Bayes, Logistic Regression, Decision tree, Random Forest, etc. Several deep learning techniques were also employed to improve the detection rate and reduce the false alarm rate. The development has reached the extent of fusing machine learning and deep

learning approaches.

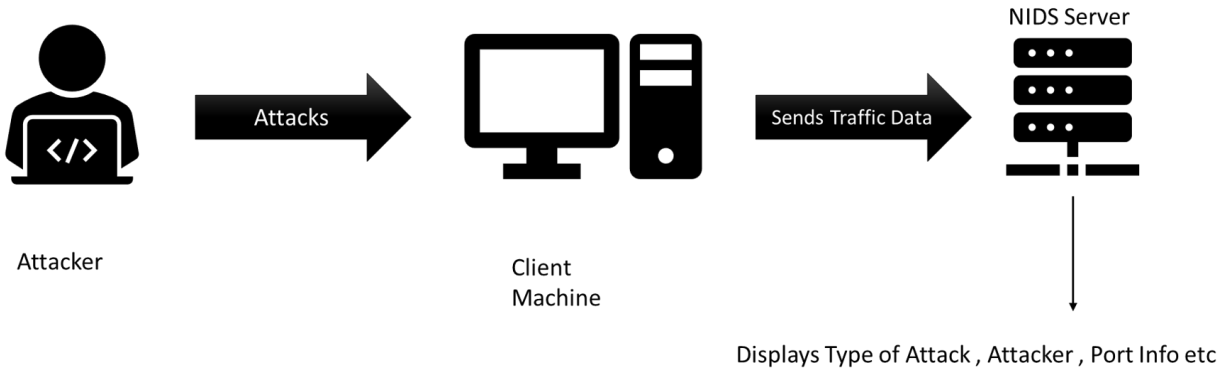

Figure 3.3: Network Intrusion Detection System

Chapter 4

METHODOLOGY

4.1 Dataset description

4.1.1 UNSW-NB15 Dataset

The raw network packets of the UNSW-NB15 data set are created by the IXIA Perfect Storm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of actual modern normal activities and synthetic recent attack behaviours. The tcpdump tool captures 100 GB of the raw traffic (e.g., pcap files). This dataset contains nine classes of attacks, namely: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The Argus and Bro-IDS tools are utilized, and twelve algorithms are developed to generate 49 features with the class label. A partition from this data set is configured as a training set and testing set. The training set contains 175,341 records, while the testing set contains 82,332 records from various classes of 'intrusive' and 'normal'.

4.1.2 NSL-KDD Dataset

The NSL-KDD data set is a new version of the KDD'99 data collection. This is a useful benchmark data set for academics to compare various IDS. The training and testing dataset were available in text format. It is then converted to csv format for our experiments. This dataset contains thirty-nine families of attacks, namely: Neptune, Satan, Upsweep, Smurf, Portsweep, Nmap, Back, Guess_passwd, Mscan, Warezmaster, Teardrop, Warezclient, Apache2, Processtable, Snmpguess, Saint, Mailbomb, Pod, Snmpgetattack, Httptunnel, Buffer_overflow, Multihop, Land, Rootkit, Named, Ps, Sendmail, Xterm, Imap, Ftp_write, Loadmodule, Clock, Phf, Perl, Xsnoop, Spy, Worm, Udpstorm, Sqlattack. All these categories were converted into a single "intrusive" category. The dataset is partitioned into training and testing sets. The training set contains 125,973 records, while the testing set contains 22,544 records from various classes of 'intrusive' and 'normal'.

This is a binary classification problem; hence all the attack categories are labelled into a single category of 'abnormal'. So, now the dataset consists of two labels: - '0' for normal and '1' for abnormal. Redundant columns are discarded manually. Tables 4.1, 4.2, 4.3 and 4.4 show the description of features in the UNSW-NB15 and NSL-KDD datasets.

4.2 Proposed framework for network traffic classification

The framework of proposed system to perform effective network traffic classification is shown in fig. 4.1.

- Initially all the categorical variables are encoded using Bayesian encoder as it exhibited highest accuracy on fitting the intrusion data. On further, resampling is done using SMOTE to avoid the class imbalance issue. Normalization using Min-Max normalizer makes the model learn the features more precisely. Three of these steps conclude the data preprocessing part.
- XGBoost parameter tuning is done to further improve the performance of the model.
- Feature extraction is carried out using XGBoost and the extracted compact features are then fed as input to the TabNet classifier model.
- Performance metrices, confusion matrix, and ROC curves of the proposed XGBoost-TabNet model is then analyzed.

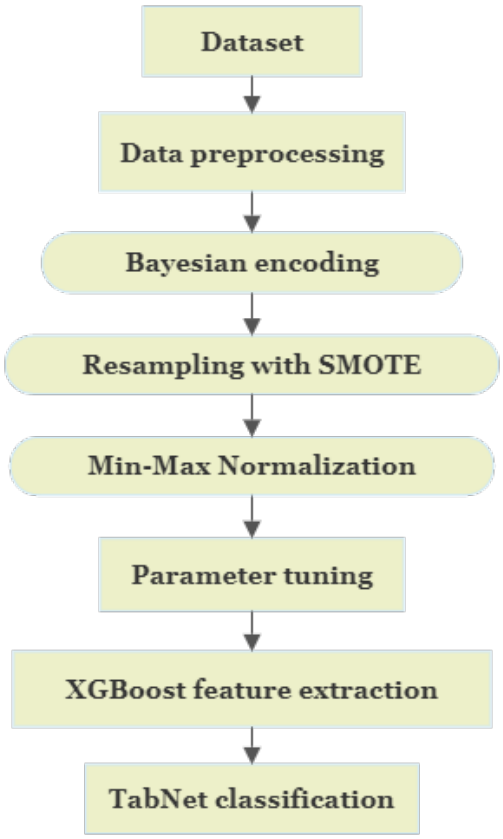

Figure 4.1: Proposed framework for network traffic classification

The overall view of the experiments is shown in fig. 4.2. Data preprocessing, parameter

tuning, feature extraction, and classification are the four steps in accomplishing the task. They are: -

1. Data preprocessing that performs 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 into the classification model.
5. Compare their performance and evaluate the FAR using performance metrics like accuracy, precision, recall, F1-score, ROC curve, and AUC score.

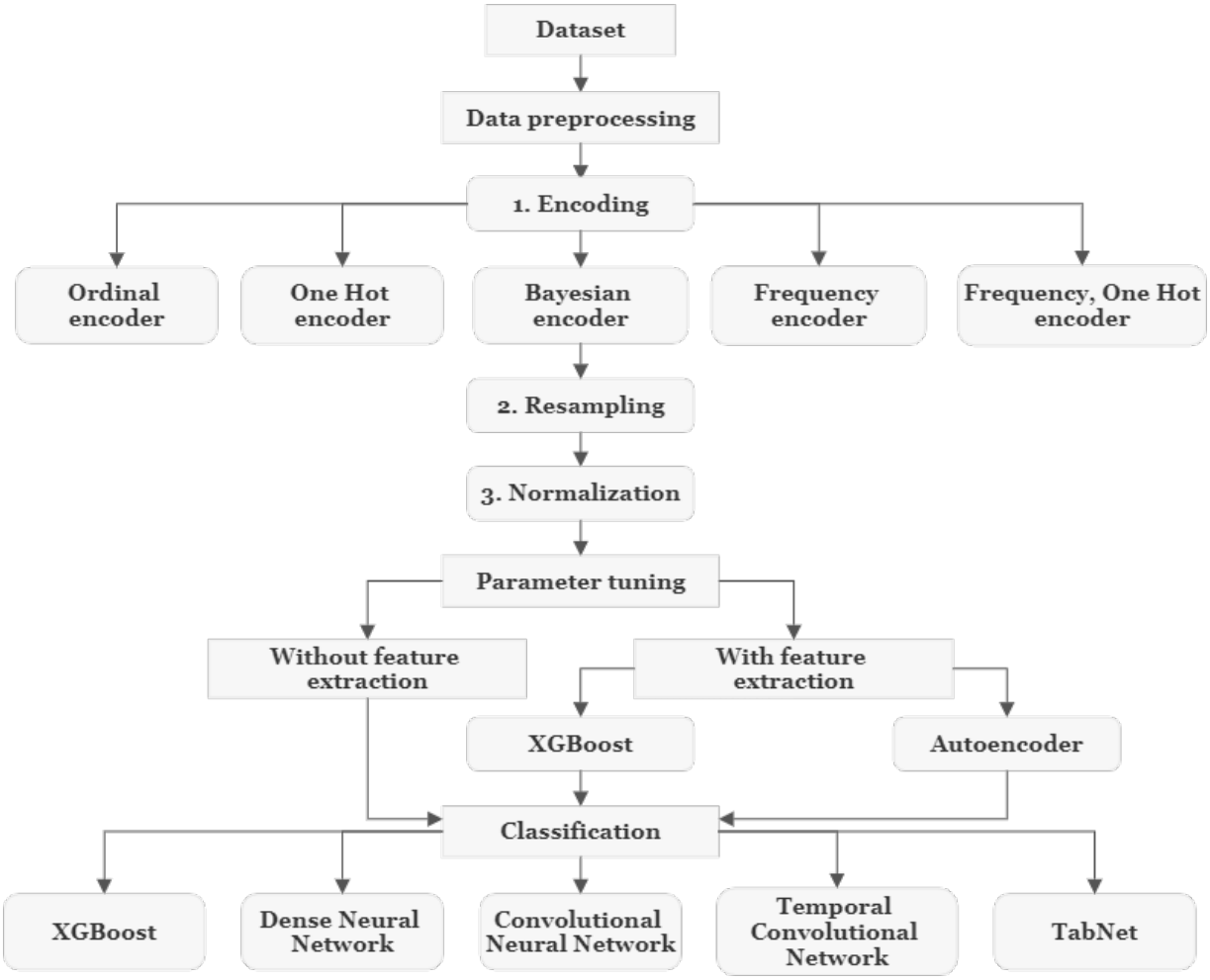

Figure 4.2: Overall view of experiments

4.2.1 Data preprocessing

- Encoding

  Most of the classification algorithms cannot deal with categorical data [12]. In UNSW-NB15 dataset, out of the 49 features three of them are categorical i.e., 'proto', 'service', 'state'. Similarly, in NSL-KDD dataset, 3 out of 41 features are categorical i.e., 'protocol_type', 'service', 'flag'. Hence it is necessary to convert these categorical data into numerical ones. This allows the model to perform better and return good performance results. Encoding techniques can be classic or advanced. To identify the best encoding scheme, experiments are carried out in the following sets of encoders: -

  1. Ordinal encoder: - This is usually used when the categories have an inherent order. So, encoding is done with an ordinal encoder to include these ordinal relations in the dataset.
  2. One hot encoder: - Each category in the categorical feature column is mapped as a separate column of '0's and '1's. Each row corresponding to the newly created column will allocate binary value '1' only if the original categorical column in the same row contains that category. If the number of categories for the feature is large, this approach produces many columns, increasing the dimension and slowing down learning.
  3. Frequency encoder: - Here, the categories are labelled based on their frequency. This enables the model to balance the weight among categories that occur more frequently and less frequently.
  4. Bayesian encoder: - To encode categorical data, Bayesian encoders use knowledge from dependent/target variables. The target variable's mean corresponding to a particular category is calculated, and the required category variable is replaced with the mean value in target encoding.

- Resampling

  The imbalance in the data can lead to reduced detection accuracy. Oversampling the minority class by duplicating the instances is a way to deal with imbalanced datasets. One such technique is known as SMOTE (Synthetic Minority Oversampling TEchnique). SMOTE is data augmentation for the minority population. The algorithm tries to create new data samples from the line obtained by mapping the neighbouring samples. This process does not add any relevant information to the data; instead, new instances are created by synthesizing old instances. Hence, the issue of data overfitting is resolved. Detection accuracy can also be improved as the model will learn both classes of data uniformly. Fig. 4.3 illustrates that the class imbalance in UNSW-NB15 dataset is more, whereas the 'intrusive' instances in NSL-KDD dataset is less indicating the necessity to resample both the datasets.

- Normalization

  A wide range of discrete and continuous values in the dataset may confuse the model. To avoid this, both UNSW-NB15 and NSL-KDD dataset is normalized. Thereby the detection accuracy is also improved. In this work, Min-Max Scaler is used to normalize the data. This makes the model learn each feature more precisely as the values are in

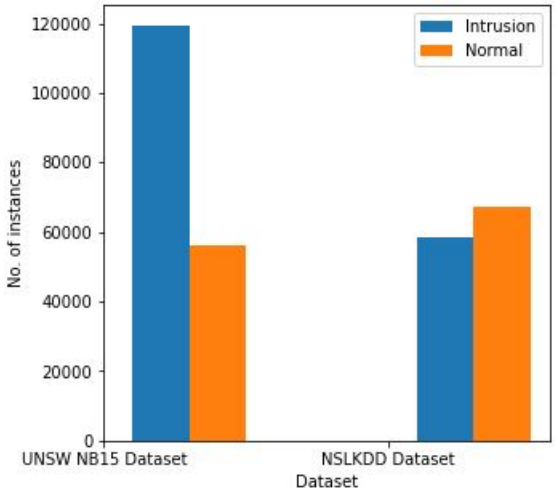

Figure 4.3: Class imbalance in UNSW-NB15 and NSL-KDD dataset.

the same format. One of the main reasons to choose the Min-Max normalizer is that even though the values are mapped to a new range, the data distribution remains the same. In this case, the new normalization range is fixed between 0 and 1. The min-max normalization is performed as follows:

$$data_{norm} = \frac{(data - data_{min})(range_{min} - range_{max})}{data_{max} - data_{min}} + range_{min} \tag{4.1}$$

where $data_{norm}$ is the normalized data, $data_{min}$ and $data_{max}$ is the minimum and maximum value in the original data, $range_{min}$ and $range_{max}$ are the new minimum and new maximum value.

4.2.2 Parameter tuning

Parameters in XGBoost are broadly classified into 3: -

1. Functioning / General parameters
   - booster (Type of model): - It can be gbtree (tree-based) or gblinear (linear) models. (default = gbtree)
   - silent: - It takes values 1 or 0. '1' means nu running messages are printed, and '0' means running messages are printed. (default = 0)
   - nthread: - It takes the number of cores required and is helpful in parallel processing. (default = maximum number of threads available)
2. Booster parameters: - Research have proved that tree boosters consistently outperform the linear boosters. So, the parameters under the tree booster category include: -
   - eta: - It helps in shrinking the weights on each step. (default = 0.3)

- min_child_weight: - It helps in controlling over-fitting. Higher values can lead to underfitting since proper relations may not be studied. Need to be tuned using CV. (default = 1)
- max_depth: - It also helps in controlling overfitting. More the depth, more specific relations are studied. Need to be tuned with CV. If max depth is specified, there is no need to specify max-leaf nodes and vice-versa. (default = 6)
- gamma: - A node can be split if there is a reduction in the loss function. Gamma value specifies how much loss reduction can lead to a split. This reduction varies with the type of loss function used, and hence it needs to be tuned. (default = 0)
- subsample: - It specifies the random number of samples (observations) assigned to each tree. It depends on the data and hence needs to be tuned. (default = 1)
- colsample_bytree: - It specifies the random number of samples (columns-features) assigned to each tree. It depends on the data and hence needs to be tuned. (default = 1)
- alpha: - It is the regularization parameter set usually when the dimensionality (number of features) is large. It is the L1 regularization term related to weight. (default = 0)
- scale_pos_weight: - It is used to solve the class imbalance issue. For high-class imbalance, a value greater than 0 is set. (default = 1)

3. Optimization / Learning task parameters

- objective: - It defines binary or multi-class classification. It also specifies whether it will return a specific class or probability. (default = reg: linear)
- eval_metric: - This is used for validating the data. (default = rmse for regression and error for classification depending upon the objective)
- seed: - Can be used for parameter tuning and generating good results.

Out of all these categories, some parameter depends upon the input, i.e., intrusion data. The functional and optimization parameters include options to print running messages, the objective (in this case: "classification"), and the model type (in this case: "tree-based"), which need not be tuned. But tuning booster parameters such as number of estimators, depth of the tree, weights assigned for each tree, number of data and samples required to be assigned in each tree, and regularization parameters, give better results. Hence parameter tuning is done to identify the best value that can be assigned to these parameters. The grid search algorithm [13] is used in this work, which will return the value with the highest rank, i.e., the best score. This is a technique of trying all the possible combinations within the range specified and find the optimal value that best fit to a parameter. Initially, a range of values that the parameter can use is given as input to the model. Training corresponds to the specified estimator and randomly provided parameter grid. This fitting process will iterate over the given number of crossvalidation folds and values within the parameter range. In each iteration, based on the mean fit time and score time, a different combination of values is formed, and the score corresponding to each combination is evaluated. Based on the score obtained, ranks are assigned. Finally, the value with the highest rank is returned as the optimal parameter value. The authenticity of the algorithm also depends on the initially assigned random parameter inputs.

4.2.3 Feature extraction

Several features like flow_id, source_address, destination_address and more can be extracted from the network traffic. All these features are not required to accurately classify the traffic as 'intrusive' and 'normal'. Features like id number, record time and more, identified as redundant by experts can be removed manually. But the problem is when we cannot distinguish between relevant and irrelevant data. Feature extraction identifies the relevant data and removes the redundant data from the dataset so that learning is made simpler. This will allow faster learning and reduce the computational complexity of the model. In this work, feature extraction is done using both XGBoost and Autoencoder.

- XGBoost feature extraction

  XGBoost or eXtreme Gradient Boost is a gradient boosting algorithm that comes under ensemble learning [14]. This is a flexible machine learning algorithm used for regression, classification, and feature scoring tasks. One of the main characteristics of the XGBoost algorithm is that it uses a gradient descent algorithm to minimize the loss function. Since this is an ensemble learning approach, the algorithm uses a decision tree as the predictor. Predictions are made sequentially by minimizing the error of the previous tree. Weights for the new tree are assigned based on their performance in the previous tree. The final output is the sum of outputs from each predictor. Apart from regression and classification, XGBoost also returns the feature importance score based on the number of times each feature is utilized in a tree. This feature score is mapped into some threshold values, which will return the number of features and accuracy when trained. So, XGBoost will help find the optimum number of features with maximum accuracy. In this work, XGBoost is selected for feature extraction and classification because of its high execution speed and enhanced performance.

- Autoencoder feature extraction

  Autoencoder is a neural network model which can be used for learning compressed data in latent space. If the low dimension structure has a non-linear relationship among the variables, autoencoders may be able to encode more information. Autoencoders for classification tasks usually consist of an encoder, bottleneck, and decoder. Here, the model is trained with both encoder and decoder for feature extraction. Since the compressed data is stored in the bottleneck, to extract the reduced features, the model is saved with the output as a 'bottleneck'. The decoder part is removed, and the reconstructed features are served as input to the classifier model. In this work, Autoencoder is selected as one of the feature extractors because of its ability to learn the non-linear relationship between the variables.

4.2.4 Classification using TabNet

Initially all the categorical variables are encoded using ordinal, one-hot, frequency and Bayesian encoder. The encoder which exhibited highest accuracy on fitting the intrusion data is taken for the next step. On further, resampling, normalization and parameter tuning is done to improve the performance of the models. Feature extraction is carried out using XGBoost and Autoencoder techniques, and the extracted compact features are then fed as input to the various classifier models. The features obtained from both XGBoost and Autoencoder are fed into XGBoost, DNN, CNN, TCN and TabNet classifiers for analyzing the

individual performance. The results obtained after various experimentation carried out are summarized in the tables 5.7 to 5.12. TabNet is a deep learning architecture that studies structured tabular data robustly and efficiently [15]. Each block in the TabNet contains a batch normalization layer, feature transformer, ReLU activation unit, fully connected layer for classification purposes, and an attentive transformer and mask for selecting the right attributes. The number of blocks depends upon the step size, which can be set as the input. For our experimentation, the step size is fixed to 10 which represents the usage of 10 blocks of TabNet. Mask in each block ensures that the prominent features are passed as input to the subsequent block. Hence, it 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. In this way, the model will ensure the effectiveness of the output. Fig. 4.4 shows the TabNet architecture.

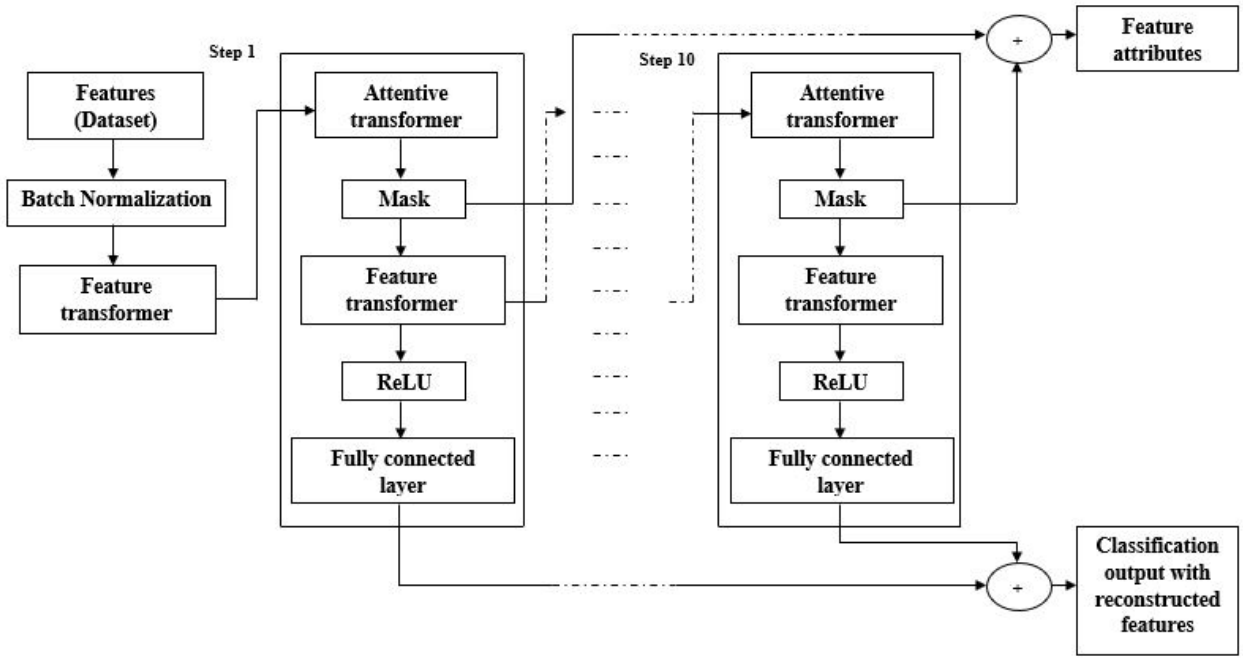

Figure 4.4: TabNet Architecture.

Table 4.1: Feature description of UNSW-NB15 Dataset - Part 1

Feature Name	Description
srcip	Source IP address
sport	Source port number
dstip	Destination IP address
dsport	Destination port number
proto	Transaction protocol
state	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
dur	Record total duration
sbytes	Source to destination transaction bytes
dbytes	Destination to source transaction bytes
sttl	Source to destination time to live value
dttl	Destination to source time to live value
sloss	Source packets retransmitted or dropped
dloss	Destination packets retransmitted or dropped
service	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
Sload	Source bits per second
Dload	Destination bits per second
Spkts	Source to destination packet count
Dpkts	Destination to source packet count
swin	Source TCP window advertisement value
dwin	Destination TCP window advertisement value
stcpb	Source TCP base sequence number
dtcpb	Destination TCP base sequence number
smeansz	Mean of the packet size transmitted by the src
dmeansz	Mean of the packet size transmitted by the dst
trans_depth	Represents the pipelined depth into the connection of http request/response transaction
res_body_len	Actual uncompressed content size of the data transferred from the server's http service.
Sjit	Source jitter (mSec)
Djit	Destination jitter (mSec)

Table 4.2: Feature description of UNSW-NB15 Dataset - Part 2

Feature Name	Description
Stime	record start time
Ltime	record last time
Sintpkt	Source interpacket arrival time (mSec)
Dintpkt	Destination interpacket arrival time (mSec)
tcprtt	TCP connection setup round-trip time, the sum of synack' and 'ackdat'.
synack	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
ackdat	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
is_sm_ips_ports	If source and destination IP addresses equal and port numbers equal then, this variable takes value 1 else 0
ct_state_ttl	No. for each state according to specific range of values for source/destination time to live.
ct_flw_http_mthd	No. of flows that has methods such as Get and Post in http service.
is_ftp_login	If the ftp session is accessed by user and password then 1 else 0.
ct_ftp_cmd	No of flows that has a command in ftp session.
ct_srv_src	No. of connections that contain the same service and source address in 100 connections according to the last time.
ct_srv_dst	No. of connections that contain the same service and destination addres in 100 connections according to the last time.
ct_dst_ltm	No. of connections of the same destination address in 100 connections according to the last time.
ct_src_ltm	No. of connections of the same source address in 100 connections according to the last time.
ct_src_dport_ltm	No of connections of the same source address and the destination port in 100 connections according to the last time.
ct_dst_sport_ltm	No of connections of the same destination address and the source port in 100 connections according to the last time.
ct_dst_src_ltm	No of connections of the same source and the destination address in in 100 connections according to the last time.
attack_cat	The name of each attack category. In this data set , nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
Label	0 for normal and 1 for attack records

Table 4.3: Feature description of NSL-KDD Dataset - Part 1

Feature Name	Description
duration	Length of time duration of the connection
protocol_type	Protocol used in the connection
service	Destination network service used
flag	Status of the connection – Normal or Error
src_bytes	Number of data bytes transferred from source to destination in single connection
dst_bytes	Number of data bytes transferred from destination to source in single connection
land	if source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0
wrong_fragment	Total number of wrong fragments in this connection
urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated
hot	Number of 'hot' indicators in the content such as: entering a system directory, creating programs and executing programs
num_failed_logins	Count of failed login attempts
logged_in	Login Status : 1 if successfully logged in; 0 otherwise
num_compromised	Number of 'compromised' conditions
root_shell	1 if root shell is obtained; 0 otherwise
su_attempted	1 if 'su root' command attempted or used; 0 otherwise
num_root	Number of 'root' accesses or number of operations performed as a root in the connection
num_file_creations	Number of file creation operations in the connection
num_shells	Number of shell prompts
num_access_files	Number of operations on access control files
num_outbound_cmds	Number of outbound commands in an ftp session
is_host_login	1 if the login belongs to the 'hot' list i.e., root or admin; else 0
is_guest_login	1 if the login is a 'guest' login; 0 otherwise
count	Number of connections to the same destination host as the current connection in the past two seconds
srv_count	Number of connections to the same service (port number) as the current connection in the past two seconds
serror_rate	The percentage of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in count
srv_serror_rate	The percentage of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in srv_count
rerror_rate	The percentage of connections that have activated the flag REJ, among the connections aggregated in count

Table 4.4: Feature description of NSL-KDD Dataset - Part 2

Feature Name	Description
srv_rerror_rate	The percentage of connections that have activated the flag REJ, among the connections aggregated in srv_count
same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in count
diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in count
srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count
dst_host_count	Number of connections having the same destination host IP address
dst_host_srv_count	Number of connections having the same port number
dst_host_same_srv_rate	The percentage of connections that were to the same service, among the connections aggregated in dst_host_count
dst_host_diff_srv_rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count
dst_host_same_src_port_rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count
dst_host_srv_diff_host_rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count
dst_host_serror_rate	The percentage of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in dst_host_count
dst_host_srv_serror_rate	The percent of connections that have activated the flag s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count
dst_host_rerror_rate	The percentage of connections that have activated the flag REJ, among the connections aggregated in dst_host_count
dst_host_srv_rerror_rate	The percentage of connections that have activated the flag REJ, among the connections aggregated in dst_host_srv_count
Label	0 for normal and 1 for attack records

Chapter 5

RESULTS AND DISCUSSION

5.1 Hardware and experimental environment

Parameter tuning is done on HPC Machine hardware, Ubuntu 18.04.05 LTS, NVIDIA Tesla V100 16G Passive GPU and rest of the experiments are carried out in 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 back-end in jupyter notebook. Machine learning and deep learning libraries include - NumPy, Pandas, Matplotlib, and Scikit learn. Performance analysis is performed to identify the best model that have the highest detection rate. The general evaluation metrics such as Accuracy, Precision, Recall, F1 score, AUC score, and Confusion matrix are used.

5.2 Experiments on data preprocessing

Encoding is necessary to convert the categorical data into numerical ones. Every encoder has its own characteristic feature that distinguish it from other encoders. For example, ordinal encoder is used to identify the variation in performance due to ordinal relations in the data. Hence, to analyze which characteristic feature is suitable for our experimentation, encoding is performed using five different combinations: - Ordinal encoder, one-hot encoder, frequency encoder, combining one hot and frequency encoder, and Bayesian encoder. After loading the dataset, missing values and count of categories in each categorical feature column are identified. Table 5.1 shows the count and name of categories in both datasets.

Table 5.1: Categorical data in UNSW-NB15 and NSL-KDD dataset

Dataset	Categorical feature	Count of sub-category
UNSW-NB15	Proto	133
	Service	13
	State	11
NSL-KDD	Protocol_type	3
	Service	70
	Flag	11

Using these categorical features, both datasets are encoded using five encoders separately and is then saved into different files. Table 5.2 and 5.3 shows the performance metric values obtained after encoding in UNSW-NB15 and NSL-KDD dataset respectively. As shown in this table, the Bayesian encoder obtained the highest accuracy. Bayesian encoding evaluates the mean value of a particular categorical features in UNSW-NB15 dataset namely Proto, Service and State. Similarly, mean of categorical features such as Protocol_type, Service, Flag in NSL-KDD dataset is taken. The categorical features consist of various subcategories within it. For e.g., the categorical feature - "Proto" have subcategories like "FDP", "UDP" and is of count 133. The count of other subcategories for each categorical features in both datasets are shown in table 5.1. The rest of the encoders carry out encoding by considering the occurrence of subcategories within the respective category independently. The Bayesian encoder exhibited good results because of its ability to study how a feature is dependent on the target data.

Table 5.2: Results of data preprocessing using different encoders in UNSW-NB15 dataset

Processing tool	Accuracy (%)	Precision	Recall	F1-Score
Ordinal Encoder	55.06	1.0	0.5506	0.3551
One Hot and Frequency encoder	87.59	0.8915	0.8759	0.8704
One Hot Encoder	87.61	0.8914	0.8761	0.8706
Frequency encoder	87.72	0.8928	0.8772	0.8717
Bayesian encoder	**90.12**	**0.9071**	**0.9012**	**0.8984**

Table 5.3: Results of data preprocessing using different encoders in NSL-KDD dataset

Processing tool	Accuracy (%)	Precision	Recall	F1-Score
Ordinal Encoder	77.44	0.8360	0.7744	0.7735
One Hot and Frequency encoder	77.59	0.8371	0.7759	0.7750
One Hot Encoder	78.21	0.8404	0.7820	0.7813
Frequency encoder	77.37	0.8354	0.7737	0.7728
Bayesian encoder	**79.79**	**0.8481**	**0.7979**	**0.7976**

Fig. 5.1 shows the comparison between different encoding techniques. Since Bayesian encoding is proved to be efficient, Bayesian encoded UNSW-NB15 and Bayesian encoded NSL-KDD dataset is used for further experimentation.

The next step in data preprocessing is resampling. As shown in fig. 4.3, UNSW-NB15 dataset consist of 1,75,341 instances (1,19,341 - "attack" and 56,000 - "normal" instances). The NSL-KDD dataset contains 125,973 instances (67,343 - "attack" 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 1,75,341 to 238,682 and 1,25,973 to 1,34,686 respectively. The minority class in both datasets is increased up to the count of majority instances i.e., both the classes are now in equal proportion. The graphical representation of the balanced dataset is shown in fig. 5.2.

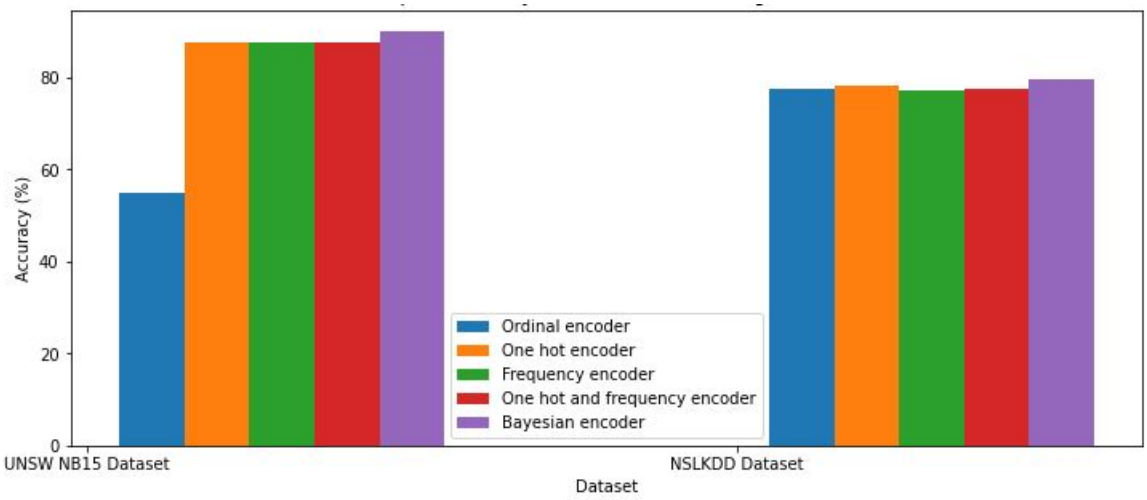

Figure 5.1: Performance of various encoding techniques

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 $range_{min} = 0$ and $range_{max} = 1$.

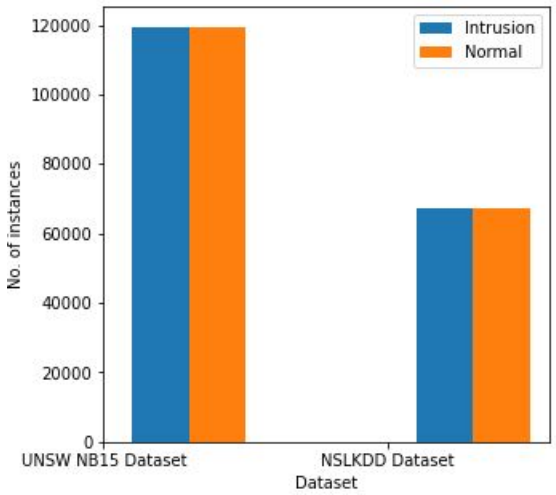

Figure 5.2: Balanced UNSW-NB15 and NSL-KDD dataset

5.3 Experiments on parameter tuning

XGBoost has some intrinsic characteristics which make it robust and efficient. Parameter tuning is done to identify the best parameters which will offer a good detection rate corresponding to UNSW-NB15 and NSL-KDD datasets. The n-estimators are successfully iden-

tified 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 for finding best values. Table 5.4 shows the values obtained after tuning these parameters on both datasets.

Table 5.4: Experimental result of XGBoost Parameter tuning

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

5.4 Experiments on feature extraction

5.4.1 Feature extraction using XGBoost

The F-score values of training and testing data of both UNSW-NB15 and NSL-KDD datasets which are Bayesian encoded is calculated. These results are illustrated as feature importance graphs and are shown in fig. 5.3 (UNSW-NB15 dataset) and fig. 5.4 (NSL-KDD dataset). A threshold is randomly generated, which is then evaluated using XGBoost classifier and a set of features satisfying that threshold. Similarly, different set of thresholds is randomly generated and evaluated using XGBoost classifier with different set of features satisfying the respective thresholds to identify minimum features providing maximum accuracy.

Tables 5.5 and 5.6 show the result of XGBoost feature selection with different thresholds. With the decrease in the number of features, the model's performance decreases. Hence, it is necessary 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.

5.4.2 Feature extraction using Autoencoder

The components of autoencoder includes input layer, encoder, bottleneck, decoder, and an output layer. Input layers consist of units as the number of features in the UNSW-NB15 and NSL-KDD dataset. The encoder part consists of two continuous sequences of dense layer - batch normalization - leaky ReLU layers. The first dense layer contains 86 neuron units, and the second dense layer contains 43 units. The bottleneck layer (code) has 21 units. The decoder part has two layers: a dense layer, a batch normalization layer, and a

leaky ReLU layer. Similarly, the decoder part is designed in a symmetrical fashion using two continuous sequences of dense layer - batch normalization - leaky ReLU layers, where dense layers contain neurons 43 and 86 respectively. The Adam optimizer and mse loss function is used along with the linear activation function in the output layer. The entire model is trained using the encoder and decoder parts. The outputs are saved from both bottleneck part and decoder part of the trained model. The output taken from the bottleneck part contains the features in a latent representation. These features are fed into various classifiers and the performance is analyzed.

Table 5.5: XGBoost feature extraction with different thresholds in UNSW-NB15 dataset

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 5.6: XGBoost feature extraction with different thresholds in NSL-KDD 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

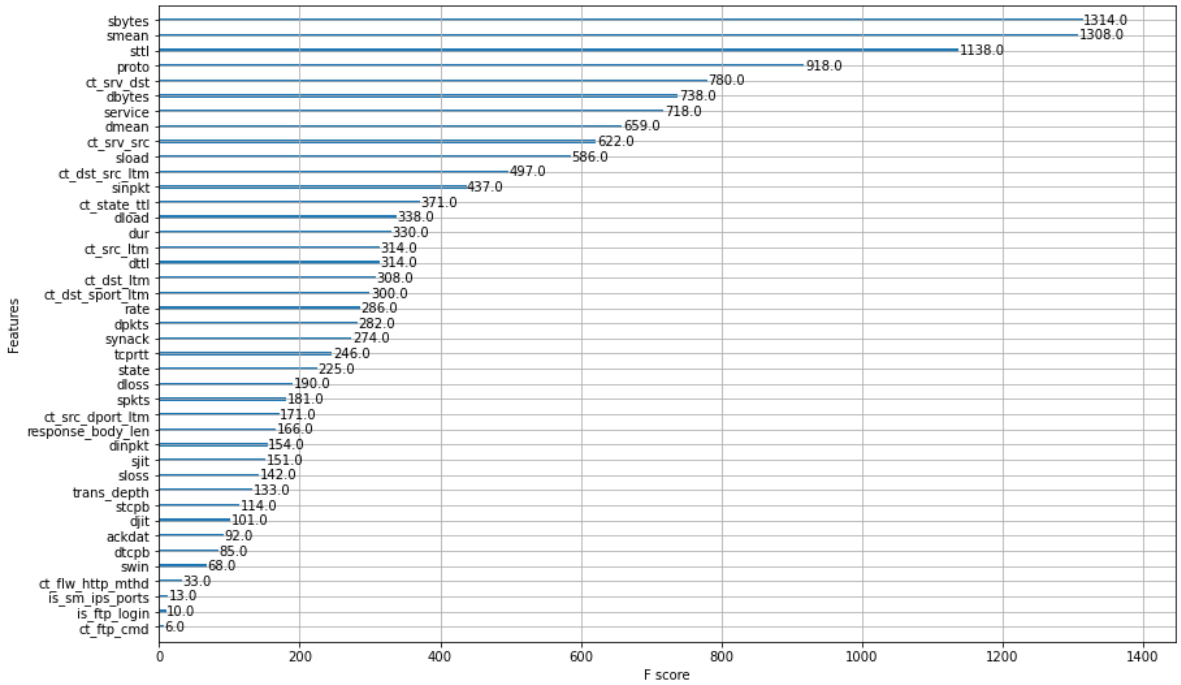

Figure 5.3: Feature importance graph of UNSW-NB15 dataset

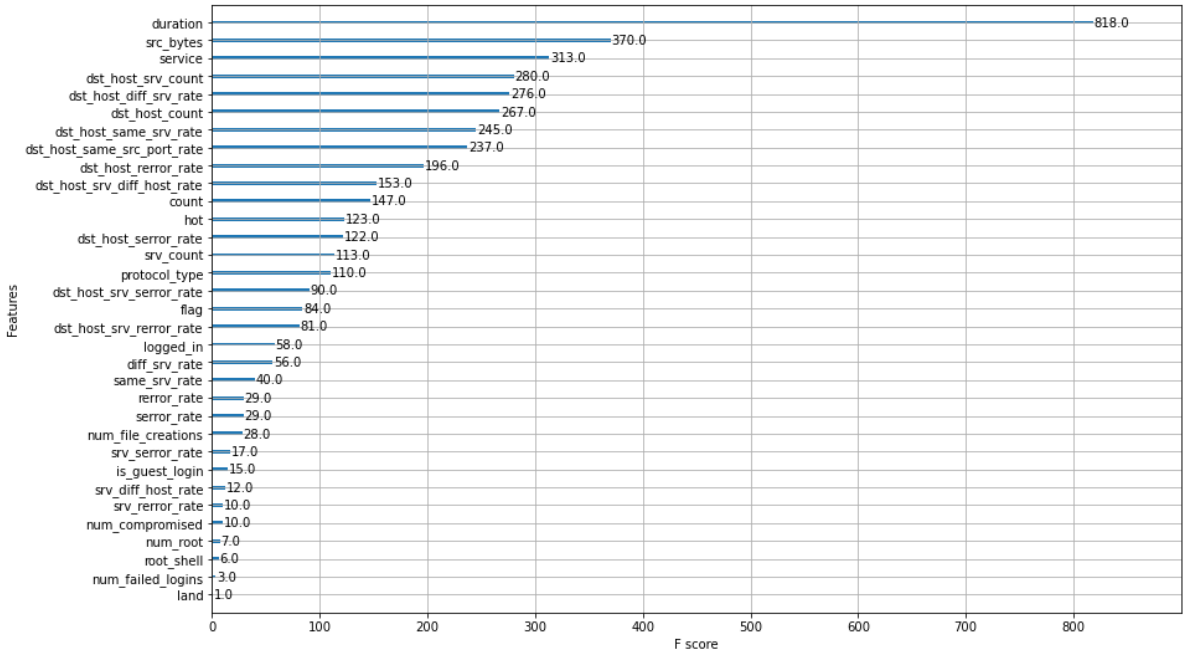

Figure 5.4: Feature importance graph of NSL-KDD dataset

5.5 Experiments on classification and comparative analysis

The features extracted using XGBoost and Autoencoder are fed into various classifiers. The performance analysis of various classifiers with and without feature extraction using XGBoost and Autoencoder on UNSW-NB15 and NSL-KDD datasets are summarized in tables 5.7 to 5.12. The detection rate of the proposed model XGBoost feature selection and TabNet classification technique yields a maximum accuracy of 93.02% compared to all the models in the UNSW-NB15 dataset. Similarly, in the NSL-KDD dataset, XGBoost feature selection and TabNet classification technique yield a maximum accuracy of 88.35%. Comparing XGBoost and Autoencoder feature extraction, the XGBoost feature extraction model provides the highest detection accuracy in both datasets. This is because, in case of structured tabular data, XGBoost consistently outperforms and provides good results by applying normalization to tabular data. XGBoost can minimize the loss of what is learned and then add the subsequent tree to it. This sequential learning allows XGBoost to extract features more efficiently. But if it was the case of unstructured data, neural networks might perform well. Hence, we infer that XGBoost feature selection is more effective than autoencoder-based feature extraction in intrusion datasets.

Table 5.7: Performance analysis of various classifiers without feature extraction (no. of features extracted =43) on UNSW-NB15 dataset

Classifier	Accuracy(%)	Precision	Recall	F1-Score	AUC score
XGBoost	90.12	0.9071	0.9012	0.8984	0.8935
DNN	75.08	0.8888	0.7508	0.7159	0.7227
CNN	78.68	0.8873	0.7867	0.7635	0.7628
TCN	90.94	0.9154	0.9094	0.9073	0.9033
TabNet	**92.09**	**0.9245**	**0.9209**	**0.9193**	**0.9161**

Table 5.8: Performance analysis using XGBoost feature extraction (no. of features extracted =15) + various classifiers on UNSW-NB15 dataset

Classifier	Accuracy(%)	Precision	Recall	F1-Score	AUC score
XGBoost	90.53	0.9116	0.9053	0.9025	0.8975
DNN	78.75	0.8879	0.7875	0.7643	0.7636
CNN	80.94	0.8897	0.8094	0.7917	0.7880
TCN	89.75	0.9108	0.8975	0.8941	0.8886
TabNet	**93.02**	**0.9309**	**0.9302**	**0.9293**	**0.9279**

TabNet is a tabular deep learning model that employs sequential attention to select a subset of relevant features and process them locally and globally at each decision-making step. TabNet classifier produced more accuracy when compared to XGBoost, DNN, CNN, and TCN classifiers. The 2-stage feature selector in the TabNet helps in more robust classification. The TabNet is found more effective than gradient boosting algorithms for classification purposes.

Table 5.9: Performance analysis using Autoencoder feature extraction (no. of features extracted =14) + various classifiers on UNSW-NB15 dataset

Classifier	Accuracy(%)	Precision	Recall	F1-Score	AUC score
XGBoost	85.32	0.8620	0.8532	0.8478	0.8427
DNN	81.03	0.8902	0.8103	0.7928	0.7890
CNN	85.10	0.8744	0.8510	0.8447	0.8392
TCN	89.06	0.8958	0.8906	0.8882	0.8848
TabNet	**89.25**	**0.8926**	**0.8925**	**0.8913**	**0.8909**

Table 5.10: Performance analysis of various classifiers without feature extraction (no. of features extracted =41) on NSL-KDD dataset

Classifier	Accuracy(%)	Precision	Recall	F1-Score	AUC score
XGBoost	79.79	0.8481	0.7979	0.7976	0.8191
DNN	77.85	0.8310	0.7785	0.7782	0.7991
CNN	74.16	0.8129	0.7416	0.7404	0.7656
TCN	78.42	0.8242	0.7842	0.7841	0.8019
TabNet	**86.39**	**0.8820**	**0.8639**	**0.8638**	**0.8760**

Table 5.11: Performance analysis using XGBoost feature extraction (no. of features extracted =4) + various classifiers on NSL-KDD dataset

Classifier	Accuracy(%)	Precision	Recall	F1-Score	AUC score
XGBoost	82.10	0.8568	0.8210	0.8209	0.8427
DNN	77.72	0.8064	0.7772	0.7771	0.7890
CNN	83.40	0.8405	0.8340	0.8332	0.8392
TCN	76.77	0.8011	0.7678	0.7675	0.8848
TabNet	**88.35**	**0.8845**	**0.8835**	**0.8807**	**0.8806**

The confusion matrix obtained with the proposed model, i.e., XGBoost feature extraction + TabNet classifier in UNSW-NB15 and NSL-KDD dataset, is shown in fig. 5.5 and 5.6 respectively. The matrix represents that the true positive rate, i.e., the rate of classifying intrusion as the intrusion, is high. Also, the false positives and false negatives are fewer. Hence, the model improves detection accuracy by reducing the FAR. The AUC score for TabNet is high, indicating good performance of the proposed model compared to other classifier models. Fig. 5.7 shows the feature importance graph of the proposed model (XGBoost feature extraction + TabNet classification in the UNSW-NB15 and NSL-KDD dataset).

Tables 5.13 and 5.14 show the comparison of the proposed model with the existing models using UNSW-NB15 and NSL-KDD datasets. The table indicates that FAR is high among machine learning models like SVM, whereas tree-based algorithm provides good detection accuracy in classifying the network intrusions. Effective feature extraction is possible with a gradient boosting algorithm rather than a neural network model because of the structured

Table 5.12: Performance analysis using Autoencoder feature extraction (no. of features extracted =13) + various classifiers on NSL-KDD dataset

Classifier	Accuracy(%)	Precision	Recall	F1-Score	AUC score
XGBoost	75.94	0.8079	0.7594	0.7589	0.7802
DNN	71.77	0.8074	0.7177	0.7153	0.7447
CNN	78.67	0.8437	0.7867	0.7863	0.8083
TCN	77.44	0.8141	0.7744	0.7743	0.7920
TabNet	**80.84**	**0.8341**	**0.8084**	**0.8083**	**0.8225**

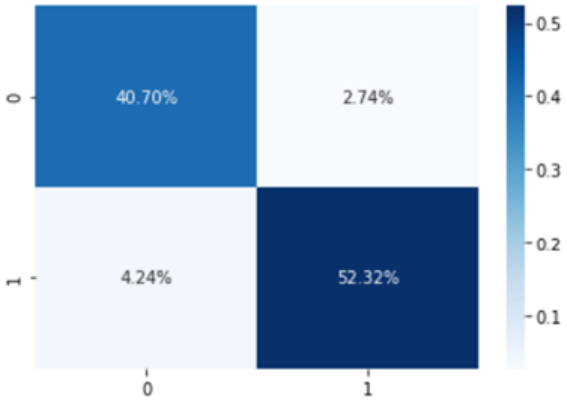

Figure 5.5: Confusion matrix of the proposed model in UNSW-NB15 dataset.

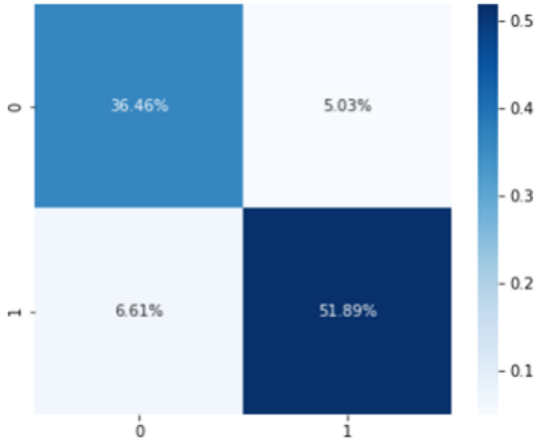

Figure 5.6: Confusion matrix of the proposed model in NSL-KDD dataset.

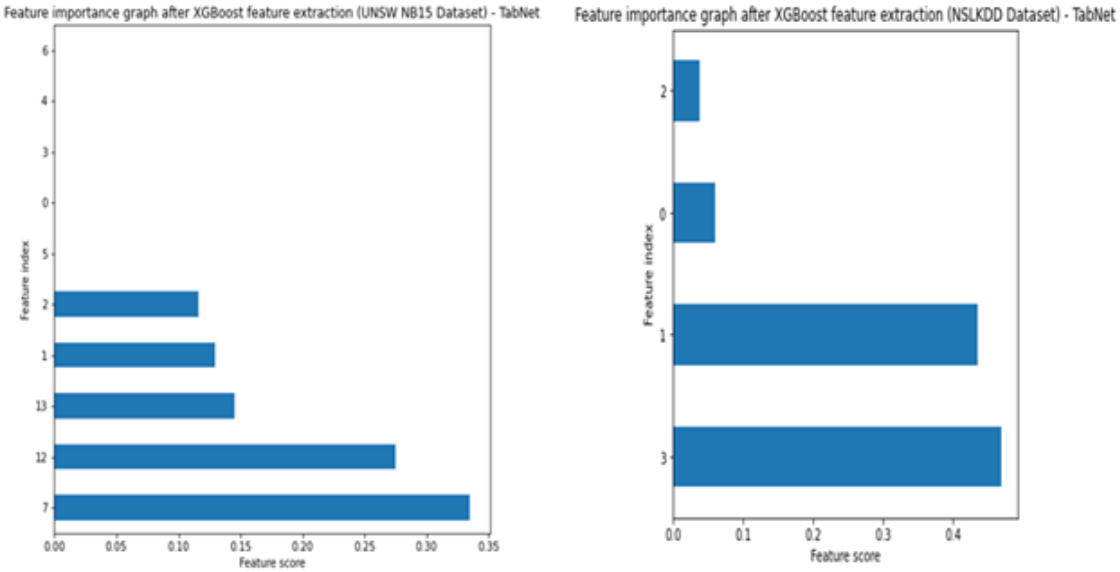

Figure 5.7: Feature importance graph of the proposed model.

nature of intrusion data. Even though DLSTM RNN is better than TCN in identifying temporal relations in the data, XGBoost feature extraction and TabNet classifier outperform, with high detection accuracy and reduced FAR. The ROC curves of classification using TabNet after XGBoost and Autoencoder feature extraction techniques in both datasets are shown in fig. 5.8 and 5.9. AUC Score of 0.9279 and 0.8790 is obtained using proposed model in UNSW-NB15 and NSL-KDD dataset respectively. This indicates exceptional discrimination among the classes in intrusion data.

Table 5.13: Comparison of proposed technique with the existing techniques (UNSW-NB15 dataset)

Work	Technique	Accuracy (%)
Kasongo SM et al. (2020) [16]	XGBoost feature extraction and classification using Decision Tree	90.85
Meftah S et al. (2019) [17]	SVM	82.11
Proposed work	**XGBoost feature extraction and classification using TabNet**	**93.02**

Fig. 5.10 and 5.11 shows the comparative analysis between the various strategies used in experimentation. The performance of various classifiers with and without feature extraction using XGBoost and Autoencoder is shown these figures. From these graphs, we can infer that performance of classifier models is higher after extracting features from XGBoost feature extractor. The performance of Autoencoder feature extraction is found less than the results obtained from classification without feature selection. TCN exhibited a descent performance

Table 5.14: Comparison of proposed technique with the existing techniques (NSL-KDD dataset)

Work	Technique	Accuracy (%)
Kasongo SM et al. (2020) [4]	DLSTM RNN	86.99
Moustakidis S et al. (2020) [5]	Fuzz-Vec2im-Siamese CNN based feature extraction and classification using AdaBoost	86.64
Proposed work	**XGBoost feature extraction and classification using TabNet**	**88.35**

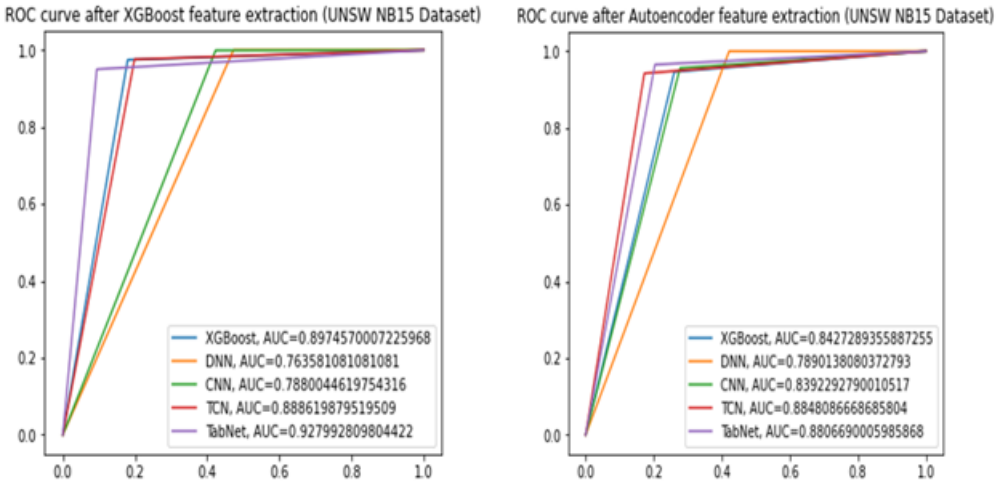

Figure 5.8: ROC curve of UNSW-NB15 dataset.

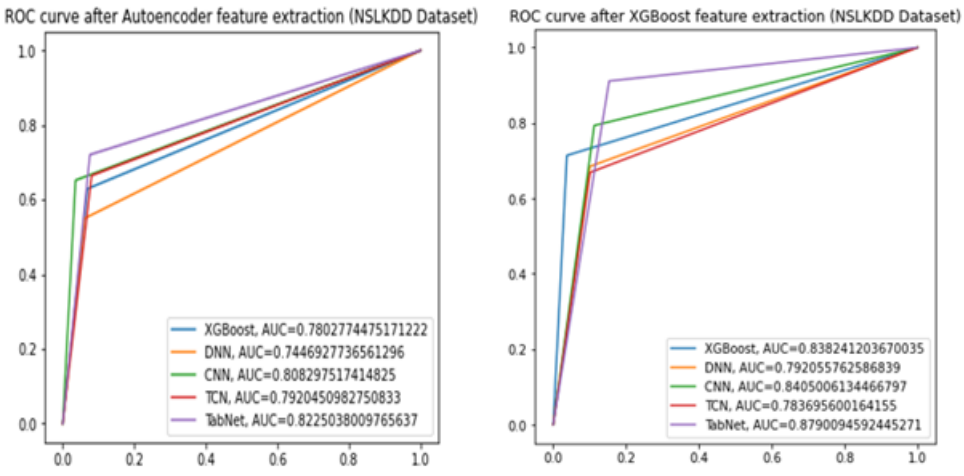

Figure 5.9: ROC curve of NSL-KDD dataset.

result on comparing to the top performed model, since it considers the temporal relation underlying the data. TCN layers such as causal convolutions, dilations, and skip connections utilize the present and immediate past inputs for output prediction. This allows the model to learn by considering the whole history of inputs.

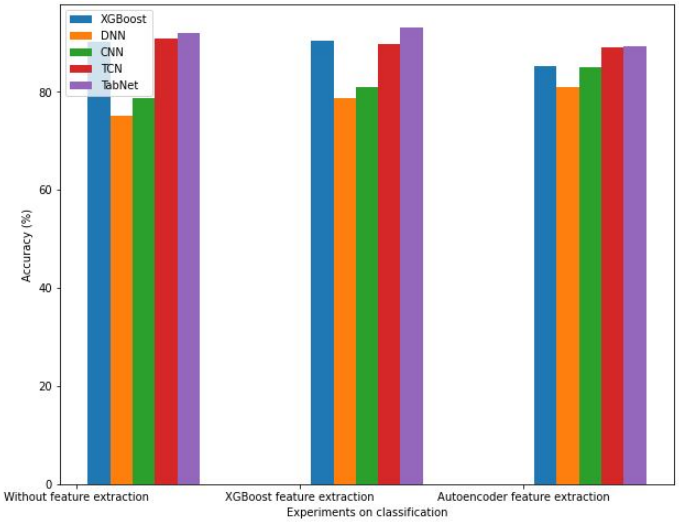

Figure 5.10: Comparison of classification experiments in UNSW-NB15 dataset.

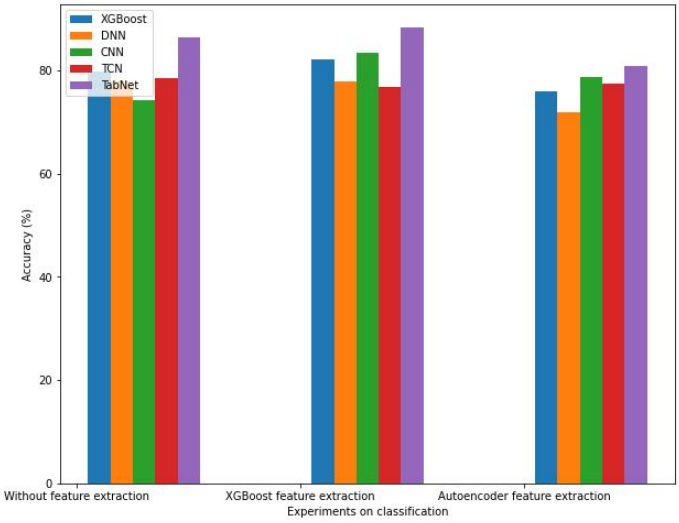

Figure 5.11: Comparison of classification experiments in NSL-KDD dataset.

Chapter 6

CONCLUSION

An automated IDS analyze the features of attack flow and usual network flow for effective prediction. It is necessary to have a reliable feature selection technique since the extraction of relevant features are very crucial in such classification tasks. In this work, feature selection is carried out using XGBoost and Autoencoder techniques, and classification is performed using XGBoost, DNN, CNN, TCN, and TabNet classifiers. Also, how various encoding scheme representations contributes to the performance is analyzed. Bayesian-encoder yields the highest classification accuracy of 90.12% and 79.79% in the UNSW-NB15 and NSL-KDD datasets when compared to other classic encoders. The Bayesian encoder could accurately map the relation between a categorical variable and a target variable. Hence further experiments were carried out with a Bayesian-encoded UNSW-NB15 and NSL-KDD dataset.

Features extracted from XGBoost and Autoencoder model are fed as input to DNN, CNN, TCN, and TabNet classifiers. Experimental results indicate that Autoencoder-based feature extraction requires more tuning to enhance the detection rate. Hence, XGBoost outperforms in extracting relevant features from the data.

In structured tabular data, XGBoost gives better results because of its ability to consider relevant features while passing through each subsequent tree sequentially. On comparing with the rest of classifiers experimented, TabNet is proved to give good performance 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 indicates the relevance of temporal relations in the data. Finally, a performance comparison is done with other works in literature. XGBoost feature selection + TabNet classification algorithm gave the best results with a maximum detection rate of 93.02% and 88.35% in UNSW-NB15 and NSL-KDD datasets respectively. Few limitations of the proposed work include - XGBoost feature extraction takes more time in extracting features corresponding to each threshold. Also, TCN was the only classifier that exhibited almost same accuracy before and after feature extraction. This indicates the least dependence on temporal features being extracted. This work can be extended for a more in-depth analysis of temporal features in the future.

References

[1] Tait KA, Khan JS, Alqahtani F, Shah AA, Khan FA, Rehman MU, Boulila W, Ahmad J. Intrusion detection using machine learning techniques: an experimental comparison. In 2021 International Congress of Advanced Technology and Engineering (ICOTEN) 2021 Jul 4 (pp. 1-10). IEEE.

[2] 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.

[3] Sundar S, Sumathy S. Transfer learning approach in deep neural networks for uterine fibroid detection. International Journal of Computational Science and Engineering. 2022;25(1):52-63.

[4] 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.

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

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] Potdar K, Pardawala TS, Pai CD. A comparative study of categorical variable encoding techniques for neural network classifiers. International journal of computer applications. 2017 Oct;175(4):7-9.

[13] Liashchynskyi P, Liashchynskyi P. Grid search, random search, genetic algorithm: A big comparison for NAS. arXiv preprint arXiv:1912.06059. 2019 Dec 12.

[14] Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining 2016 Aug 13 (pp. 785-794).

[15] Arık SO, Pfister T. Tabnet: Attentive interpretable tabular learning. InAAAI 2021 (Vol. 35, pp. 6679-6687).

[16] Kasongo SM, Sun Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. Journal of Big Data. 2020 Dec;7(1):1-20.

[17] Meftah S, Rachidi T, Assem N. Network based intrusion detection using the UNSW-NB15 dataset. International Journal of Computing and Digital Systems. 2019 Sep 1;8(5):478-87.

[18] Garcia-Teodoro P, Diaz-Verdejo J, Maciá-Fernández G, Vázquez E. Anomaly-based network intrusion detection: Techniques, systems, and challenges. computers security. 2009 Feb 1;28(1-2):18-28.

[19] Ahmad I, Basheri M, Iqbal MJ, Rahim A. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. IEEE access. 2018 May 30;6:33789-95.

[20] Manzoor I, Kumar N. A feature reduced intrusion detection system using ANN classifier. Expert Systems with Applications. 2017 Dec 1;88:249-57.

[21] Wang Z. Deep learning-based intrusion detection with adversaries. IEEE Access. 2018 Jul 9;6:38367-84.

[22] Behera S, Pradhan A, Dash R. Deep neural network architecture for anomaly based intrusion detection system. In2018 5th International Conference on Signal Processing and Integrated Networks (SPIN) 2018 Feb 22 (pp. 270-274). IEEE.

LIST OF PUBLICATIONS

Santhosh A, Saranya T, Sundar S, Natarajan S. Deep Learning Techniques for Brain Tumor Diagnosis: A Review. In2021 Fourth International Conference on Microelectronics, Signals Systems (ICMSS) 2021 Nov 18 (pp. 1-6). IEEE.