Building a NIDS Using a Two-Stage Classifier and Feature Reduction through Statistical Methods

RAGHUVANSH RAJ, SOMYA GUPTA, MANAN LOHIA, H C TANEJA
Department of Mathematics
Delhi Technological University
Shahbad Daulatpur, Main Bawana Road, Delhi - 110042
INDIA
somya_bt2k16@dtu.ac.in http://www.dtu.ac.in

Abstract: - Today, data is produced, transmitted and stored across the internet in abundant quantities. In such a world, modern-day network security systems have had to develop at an equally astounding rate in order to keep up with the data deluge. Subsequently, companies spend a substantial amount of money, time and effort in developing intrusion detection systems to ensure timely detection and prevention of malicious activity in order to preserve system security. In our paper, we propose a two-stage algorithm to solve the problem of NIDS resulting in fewer false positives and false negatives. We built and evaluated the performance of our IDS using the benchmarked NSL-KDD dataset, which consists of 41 features, of which 3 are categorical and the remaining numeric. Categorical features have been transformed via One Hot Encoding due to which the feature space explodes to 122 features. Our aim henceforth is to reduce this feature space through methods of feature selection and dimensionality reduction to develop a computationally inexpensive classifier capable of operating on the reduced feature space using sequential models. The work has involved using autoencoders for feature space reduction followed by a feed forward network, and has delivered encouraging results. We have then extended our analysis to identify features which can be eliminated without any substantial loss of information available for the classification algorithm. The remaining set of features can then be input into a different model to possibly provide better results or reduce training and evaluation time.

Key-Words: Intrusion Detection, Feature Reduction, Artificial Neural Networks, NSL-KDD

Received: January 2, 2020. Revised: April 2, 2020. Accepted: April 16, 2020. Published: April 30, 2020.

1 Introduction

In the current scenario, the Internet has become a necessity for daily life and is used for various purposes such as education business and banking. The Internet is extremely vast and diversified, connecting millions of devices with each other to create and save data but at the same time making it vulnerable to attacks on sensitive information. Exposure of one system could lead to the subsequent contamination of another. With the advancement of the Internet, the equipment for network security has also evolved. This field is extremely versatile and has grown in complexity to prevent the various attacks that have been created over time and put into action. Intrusion Detection Systems (IDS) monitor networks for malicious activity or privacy breaches to protect data confidentiality and integrity, i.e. any form of intrusion that may negatively affect privacy or safety. Depending upon the deployed location, IDS are majorly of two types; host-based and network-based. Host-based IDS are installed on PCs and network-based IDS are situated on networks.

These systems employ machine learning techniques differentiating the malicious usage patterns from the benign ones. In these techniques, a model is trained on a training set consisting of data points of both classes then applied on the test set to compare the predicted classes to the actual ones then tuned upon to improve the accuracy and complexity. Once the desired output is achieved, the trained model is employed in real-life scenarios to prevent malicious data packets from going through the network. To keep the model up-to-date, it is trained in further samples to identify new types of malicious attacks.

This paper employs the NSL-KDD dataset which is a balanced resampling of the KDD CUP 99 dataset [1]. This dataset contains 5 classes with 4 of them malicious attacks and 1 benign. The malicious classes are DoS, Probe, R2L and U2R.

The main algorithms in effect make use of rule-based methods such as SNORT [2], clustering methods such as fuzzy [3], [4] and mini batch kmeans [5], and neural networks [6], [7], [8]. These methods make use of the direct dataset which has a high variance and no clear decision boundary between the different classes, making it a tougher classification process and requiring a denser model network.

In this paper, we set out to propose and implement a new method which first consists of pre-processing via feature engineering using one hot encoding followed by dimensionality reduction using Principal Component Analysis (PCA) to improve the efficiency of the resultant dataset for the following steps. Secondly, outlier detection is implemented as a binary classification problem with the two classes being malicious and benign to classify the malicious attacks with less complexity. The last stage consists of multi-class classification with all 5 classes in an effort to further classify the attack and reduce the rate of false positives.

2 Related Work

Due to the increasing demand of computationally inexpensive and extremely accurate network intrusion detection systems, many algorithms have been recently proposed that yield unbeatable accuracies. Since the proposal of deep learning as a method for intrusion detection [9] by Professor Hinton in 2006, there have been huge advancements in every field of computational intelligence, as has been the case with intrusion detection systems.

[3], [4], make use of fuzzy clustering augmented with Artificial Neural Networks (ANN) which causes a drastic reduction in false positives. The technique used by [4] results in extremely less false positives due to the presence of an aggregating ANN after the classification ANNs.

[6] uses autoencoders to reduce the feature space and executes instructions on parallel processors. This paper focuses more on the complexity of the algorithm and the time taken by the algorithm, which is why feature space reduction is done on a rather drastic scale. Experiments show that such drastic reduction in feature space result in stark information loss which results in a less than ideal situation.

[7], [10] make use of Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) to introduce context among the features. These kinds of neural networks are implemented by these papers in a bidirectional sense and since context is introduced in the first layer itself, the depth of the network can be limited to 2-3. Although forward propagation in this kind of network takes a substantial amount of time due to bidirectional connections in each layer, it causes minimal space overhead due to a radical reduction in the number of nodes and layers.

[11] uses genetic algorithms like crossover and mutation to search for the optimal feature space in neural networks. The idea of genetic algorithms is inspired by evolution and how it results in the fittest population (features in this case). Genetic algorithms simply need a fitness measure that results in the optimal feature space when run for multiple epochs but the drawback with this method lies in the complexity and time taken by the model as it is essentially running over the entire dataset multiple times.

In this paper we have attempted to create a model that runs on a reduced feature space and works as a robust classifier while providing minimum false negatives. For this, we use a two-step approach. First, anomaly detection is conducted which detects malicious data points. Then, multiclass classification which classifies data points into respective classes and reduces the number of false positives.

3 Preliminaries

In this section, a basic overview is provided of all the related terms such as IDS, dimensionality reduction, decision trees, ensemble methods and the NSL-KDD dataset.

3.1 Intrusion Detection Systems

Intrusion Detection Systems are classified into two major categories based on their approach and the techniques and algorithms used to analyze traffic data and detect intrusions, namely anomaly-based and signature-based.

Signature based IDS compare live traffic data to a stored database and try to match it with known intrusion patterns in order to detect attacks, i.e., they follow the well-known paradigm of rule-based systems. Signature-based IDS deliver high accuracy when detecting known intrusion patterns, but can fail to detect newer types of attacks due to a lack of any existing, match able patterns in the database. In an era where attackers are always finding newer ways to take control of a system or a network of systems, signature-based IDS fail to offer an adequate level of protection without a substantial degree of risk. Another drawback of such frameworks is the large administrative overhead required to regularly maintain and update the signature database to ensure that the system is able to detect even the newest types of attacks [12].

Contrarily, anomaly-based IDS rely on detecting deviations from what is considered to be normal network behaviour to identify malicious activity. Originally, anomaly-based systems used statistical methods to model data representing ‘normal behaviour’ and used these developed models to identify and flag any significant deviations as

suspicious. However, recent efforts have pivoted to focus on using machine learning and deep learning along with advanced mathematical techniques to develop improved anomaly-based IDS. Major challenges that arise when attempting to build such models is the high false alarm rates, owing to the difficulty of modelling the 'normal behaviour' due to non-linearity of data and a certain degree of obscurity present in the data [13].

3.2 Dimensionality Reduction

Dimensionality reduction is an umbrella term for the various processes aimed at reducing the number of random variables in a dataset. This serves multifarious purposes such as enabling visualization of data and easing computational requirements. It can also help remove contextual redundancies in the dataset and simplify the overall process of data analysis and model construction. Dimensionality reduction is split majorly into two kinds of techniques, feature selection and feature extraction.

Feature selection involves the selection of a subset of the existing features. The selection of features can be guided by strategies involving information gain, gain ration, scoring features based on their importance etc.

Feature extraction is a more complex set of techniques that involve combining the existing variables through various statistical, algebraic or deep learning techniques to form a new, lower dimensional set of features which can be used to represent the dataset effectively. However, this technique suffers with a loss of context as it is not always possible to determine what exactly the new features represent and some information is always lost due to the decrease in number of features.

3.2.1 Principal Component Analysis

Principal Component Analysis (PCA) [5], [14] is a kind of dimensionality reduction technique that combines concepts from statistics and linear algebra to perform feature extraction. The process involves the use of covariance matrix and eigenvalues to transform the existing dataset into a new dataset using principal components derived from eigenvectors. The following steps are involved in PCA:

- For the dataset $D_{m \times n}$ containing m datapoints with n features, subtract from each element the mean of the column to which it belongs to centre the datapoints at the origin. Let this new matrix be D`.
- Calculate the covariance matrix od D`, denoted by C.
- Obtain the eigenvalues and eigenvectors of the matrix C.
- If d denotes the required number of dimensions for the new dataset, select the first d eigenvalues with the largest magnitudes, and combine their corresponding eigenvectors to form the transformation matrix $E_{n \times d}$.
- The reduced dataset is given by $F_{m \times d} = [D`_{m x n}] \times [E_{n x d}]$. (1)

3.2 Decision Trees

A decision tree is a classification algorithm that forms a flowchart in a tree-like graph where at each internal node an attribute is chosen on which to split the dataset on. Consequently, each branch represents one outcome of the split and any subsequent leaf nodes constitute the classes. The entire path from the root node to a leaf node can be regarded as a classification rule [15]. They map both linear as well as non-linear relationships.

The split at each internal node can be done on the basis of information gain and Gini index. The model proposed in this paper uses Gini index. Gini index forces a binary tree which is ideal in our case as we are initially working on a binary classification model. The underlying principle for this splitting algorithm is based on the fact that the probability of two randomly chosen data tuples from the dataset belonging to the same class is 1 if the dataset is pure.

3.4 Ensemble Methods

The usage of ensemble methods is ideal in the combination of a number of base models to produce an optimal model. These methods take into consideration a number of decision trees and use those to find the ideal split at each internal node by calculating which features to use.

Bagging [16] is an example of an ensemble method that combines Bootstrapping and Aggregation. From a sample of the dataset, multiple bootstrapped samples are taken upon which a decision tree is formed for each one. An algorithm is then used to aggregate the trees and find the most efficient predictor.

Random Forests [17] are another example. These use the same concept as bagging except that they work with the entire dataset rather than a sample. Each tree in this method splits at different features thus overall providing a larger ensemble to aggregate over, eventually producing a more accurate predictor.

3.5 NSL-KDD

The NSL-KDD dataset [18] is a refined version of the original KDD-CUP 99 dataset, which, after a thorough analysis revealed, a number of redundancies and other issues such as imbalanced classes and a huge number of records which offered many challenges to the construction of a classification model. The improved NSL-KDD dataset consists of 41 features and is split into the test set and training set. The training set has 125,973 datapoints while the test set has 22,544 datapoints. The test data is classified into 38 different classes, of which 21 are present in the training data. 1 of these classes corresponds to 'normal activity' while the others are various attack types. For the sake of brevity, the 37 other classes are categorized into 4 different attack types, as follows [19]:

Table 1. Categorization of the 37 different classes into the 4 different attack types.

S. No.	Attack Type	Class in Dataset
1	DoS	back, land, neptune, pod, smurf, teardrop, mailbomb, processtable, udpstorm, apache2, worm
2	Probe	satan, ipsweep, nmap, portsweep, mscan, saint
3	R2L	guess_password, ftp_write, imap, phf, multihop, warezmaster, xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named
4	U2R	buffer_overflow, loadmodule, rootkit, perl, sqlattack, xterm, ps

We follow the aforementioned classification in our model. The following figures show the distribution of datapoints belonging to each class. As the distribution is highly skewed, we have used the log of the counts for each class where necessary.

Fig.1 Class distribution of the test set.

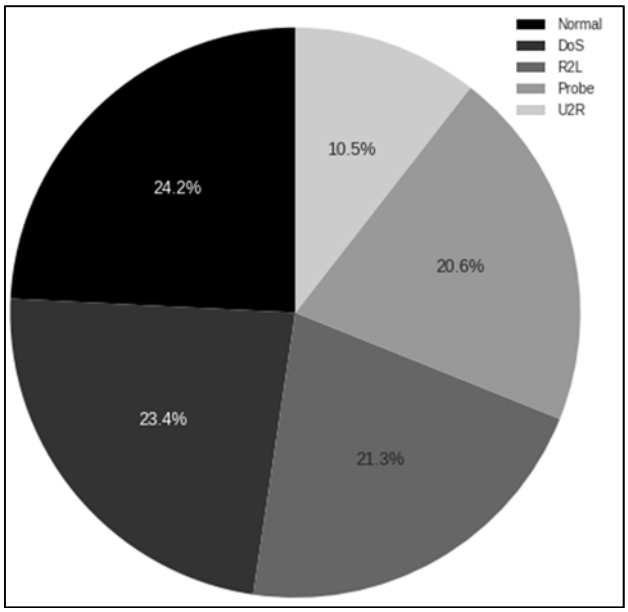
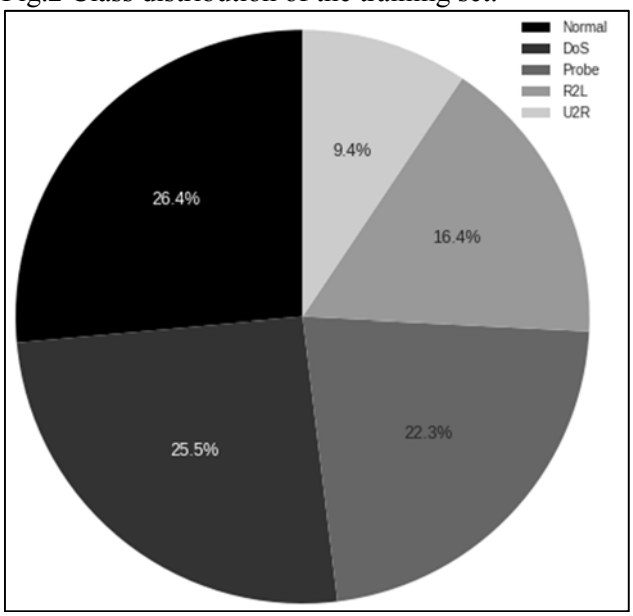


Fig.2 Class distribution of the training set.



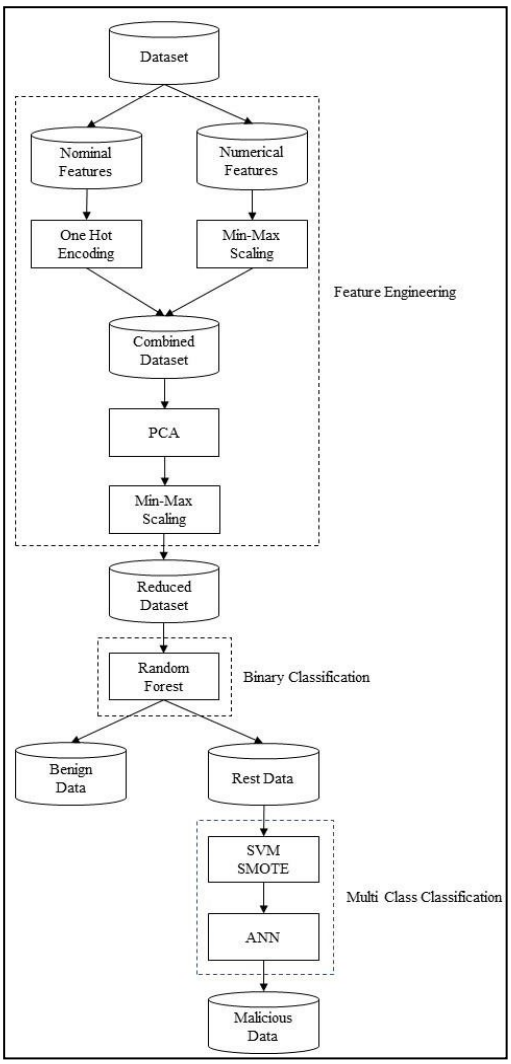
The 41 features can be divided into 4 major sets, consisting of the basic features for any network connection, content related features, time related traffic features and host-based traffic features [20]. A description of the basic features is given in [20] The remaining features are majorly binary variables which provide additional information about the network connection represented by the corresponding datapoints.

4 Proposed Methodology

In this section, we provide the proposed methodology for intrusion detection. The proposed schematic is shown in figure 3.

The proposed architecture consists of 4 steps; each of these are explained in detail further:

Fig.3 Flow of the model



4.1 Feature Engineering

The initial phase of the model is the feature engineering step. The NSL-KDD dataset consists of a training set and a testing set. The training set has 41 features, out of which the 20th feature, i.e., num_outbound_cmds which is the number of outbound commands in an ftp session [20], consists of only zeros so the column is dropped. After dropping the 20th column, we divide the dataset into nominal and numerical features depending on the data type. The nominal features undergo one-hot encoding which expands the number of features to become 84. The numerical features on the other hand undergo min-max scaling so that features with a broader range do not outweigh the features with a smaller range. They are a total of 37 features. The combined dataset of both numerical and nominal features gives a total of 121 features. The feature matrix formed is sparse due to one-hot encoding so dimensionality reduction is carried out using PCA on the 121 features. Retention of 67 features retains 99% variance of the original dataset. Then min-max scaling once again to bring all the features within the same range.

4.2 Binary Classification

In this step, binary classification is done to divide the dataset into two classes namely, malicious and benign. Ensemble methods have been tried and tested to see which one works best. Random Forest, AdaBoost and XGBoost have been compared in terms of precision, recall and area under the receiver operating characteristics (ROC) curve. The precision-recall curves have been plotted to calculate the threshold for the confidence score. The optimal threshold value is the intersection of the precision and recall curves in order to optimize both metrics. The aim of binary classification was to increase recall so as to reduce false negatives.

After applying binary classification, we have divided the dataset into two categories one with completely benign data points and another with both benign and malicious data points as the number of false positives are still relatively high.

4.3 Oversampling

Before we reach the step where we reduce the number of false positives, we oversample the dataset as the number of data points of U2R and R2L are extremely less in comparison to the rest, i.e., Normal, DoS and Probe. This could potentially lead to the problem of the network not being able to learn the datapoints of those 2 classes. Since the samples of U2R and R2L are less, the network is not able to classify points into these 2 classes properly resulting in false negatives. To solve this problem, we used SVMSMOTE which oversampled for the 2 classes and synthesized new data points.

4.4 Multi Class Classification

The last step is multi class classification. This is done to reduce the number of false positives occurring at the output of the model. An artificial neural network (ANN) is fitted onto the output class of binary classification that had both malicious and benign data points after oversampling.

The datapoints are classified into the 5 classes, namely Normal, DoS, Probe, U2R and R2L after being fed through an ANN. The ANN architecture that we have implemented consists of Convolutional and LSTM layers and outputs the class probabilities for all the 5 classes using SoftMax Activation at the final layer and Rectified Linear Unit (ReLU) activations at every other layer. Datapoints

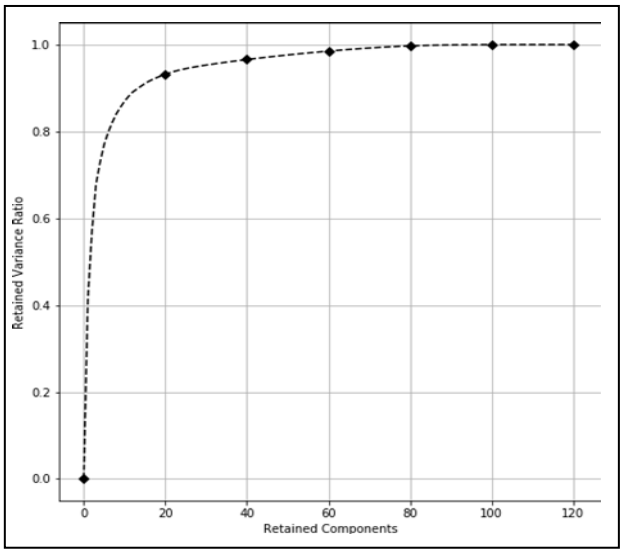
classified as normal here as well as at the binary classification stage form the combined population of benign data points. Every other data point is categorized first as malicious and then further into the type of attack.

5 Experiment and Results

In this research, we have used scikit-learn and keras with a tensorflow backend for the construction of our models and we have used matplotlib for generating graphs. All of the models have been trained on Google Colaboratory using a Tesla K80 single core GPU with 25.51 GB RAM, although RAM usage throughout our experiments was limited to 10 GB.

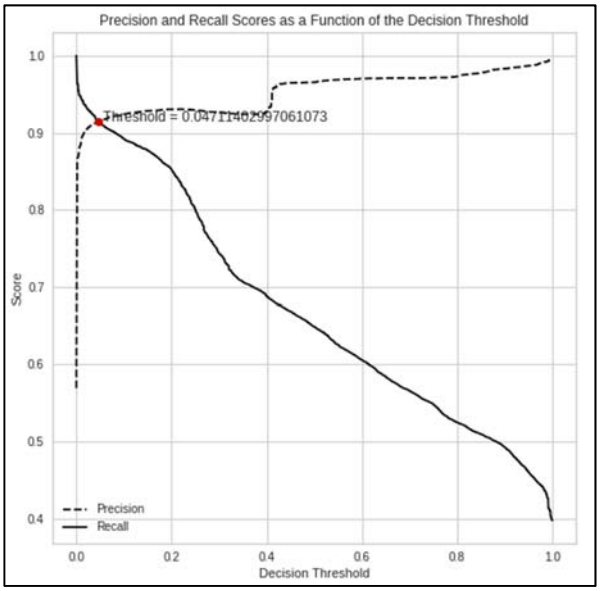
During feature engineering, we applied PCA for dimensionality reduction. The following graph shows the retention of variance corresponding to the number of components retained.

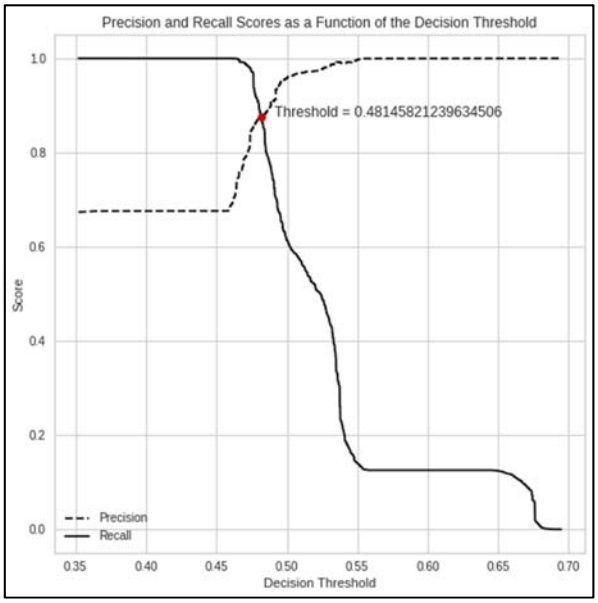
Fig.4 Variance Retention vs. Number of Components.

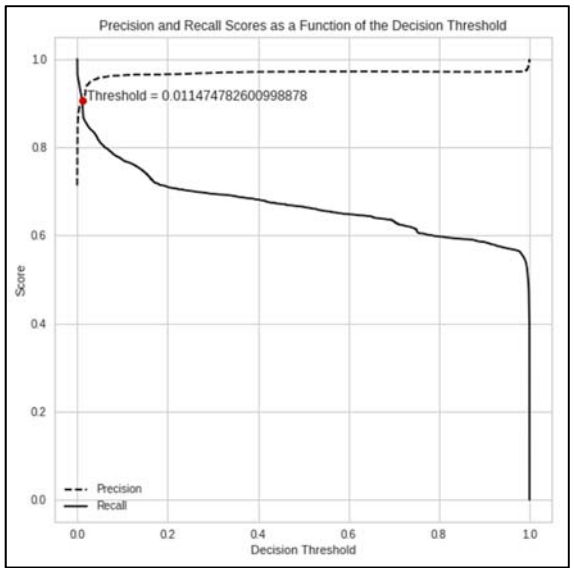


For binary classification we have compares 3 methods: Random Forest Classifiers and boosting algorithms AdaBoost and XGBoost. To obtain the optimal set of parameters for each of these classifiers, grid search was run on a suitable search space with recall as the optimizing metric. This was followed by comparing plots (Figure 5) of precision and recall and finding the optimal threshold for a point to be classified as malicious. Threshold tuning was based on the fact that we weren't looking for very high precision values at this stage; recall was of primary importance. Experiments suggested that threshold values that brought precision and recall closest together could be considered as optimal threshold points as the relation between these 2 metrics is almost inverse in nature, which means that increasing recall via threshold tuning would result in reduces precision.

Fig.5 Precision and recall scores as a function of the decision threshold of Random Forest, AdaBoost, and XGBoost respectively.



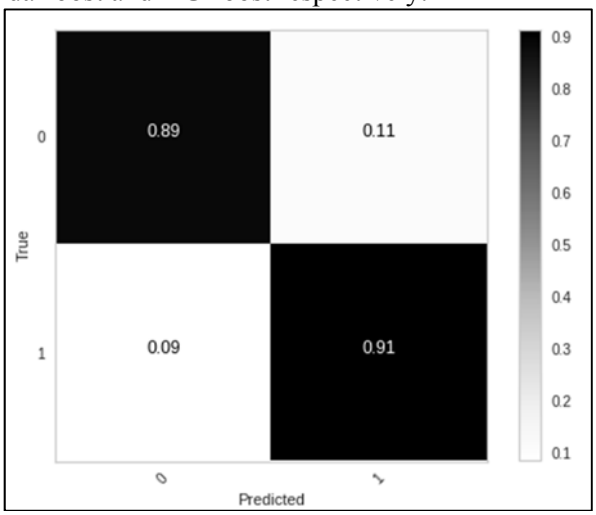


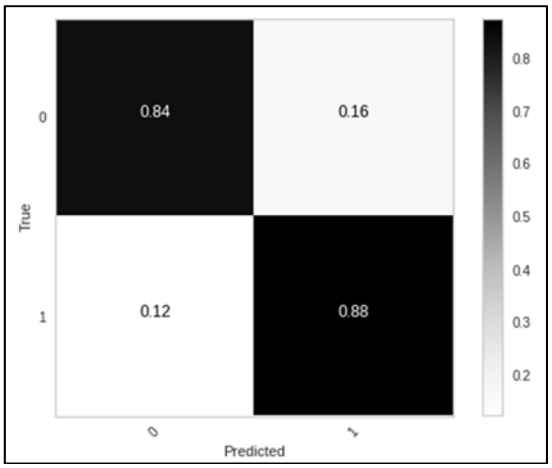


Random Forest Classifiers gave the best classification output overall (Fig.6) It and XGBoost had a recall of 91%, while AdaBoost gave a recall of 88%. Random Forest Classifiers gave a maximum precision of 89% while AdaBoost and XGBoost gave a precision of 84% and 88% respectively.

XGBoost had the maximum area under curve (AUC) of 0.96275, while Random Forest Classifiers and AdaBoost had an AUC of 0.94587 and 0.94139 respectively (Fig.7).

Fig.6 Confusion matrices of Random Forest, AdaBoost and XGBoost respectively.





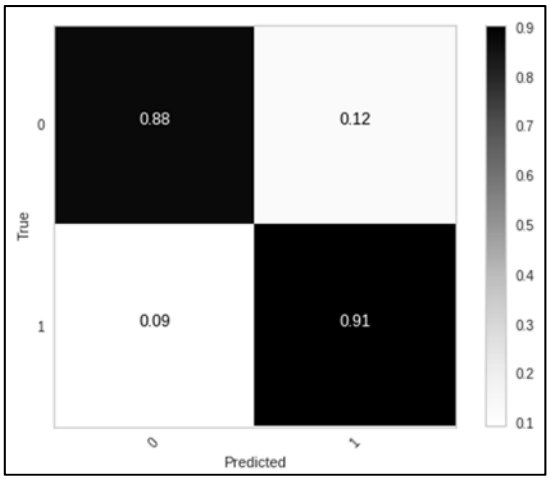
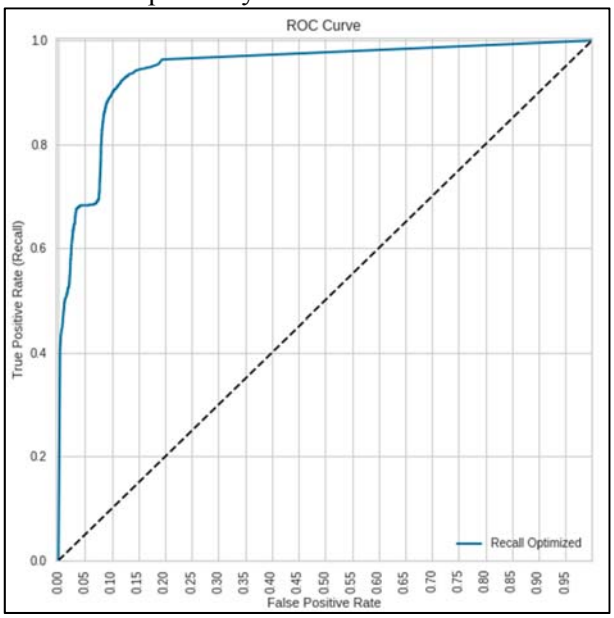
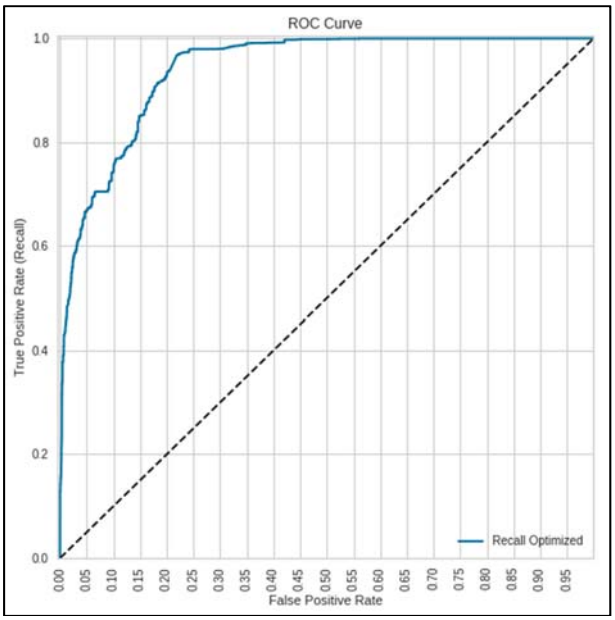
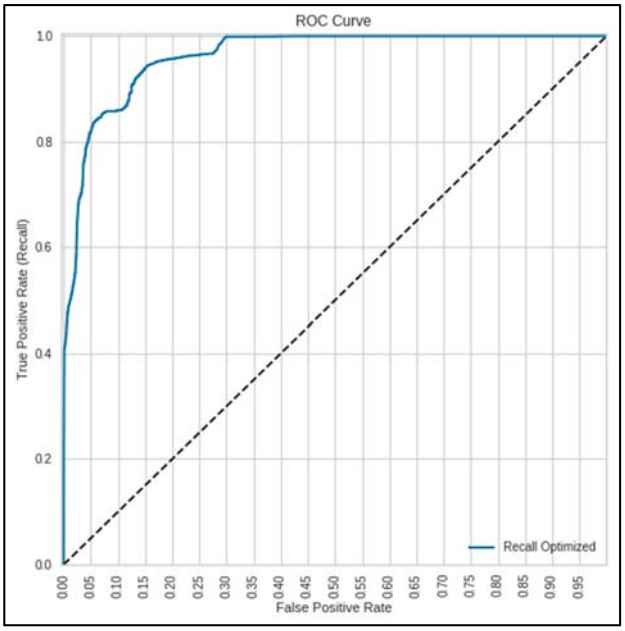


Fig.7 ROC curves of Random Forest, AdaBoost and XGBoost respectively

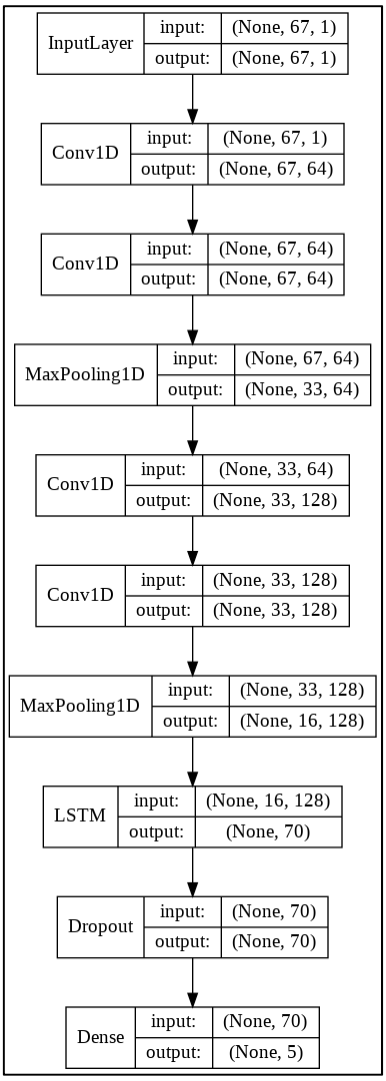






For multi-class classification, multiple neural network architectures were considered comprising of CNNs, RNNs, LSTMs, etc. The following architecture, as suggested by [21] yielded best results when trained on 100 epochs.

Fig.8 Architecture of the ANN



The loss convergence and accuracy over a course of 100 epochs are shown by the following graph.

Fig.9 Loss convergence of the model

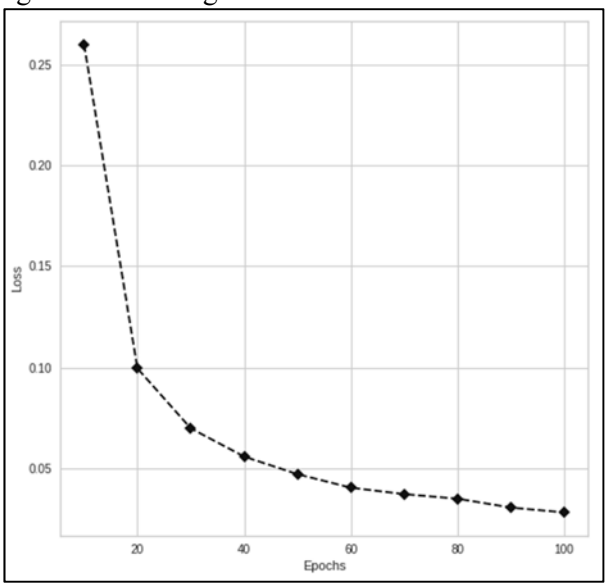
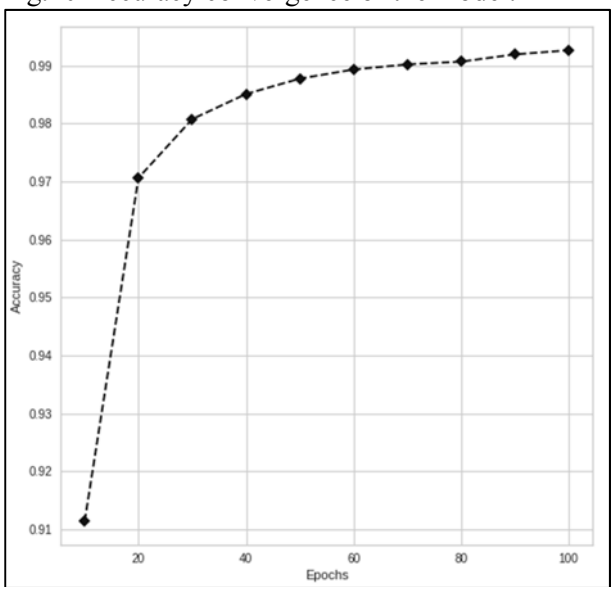
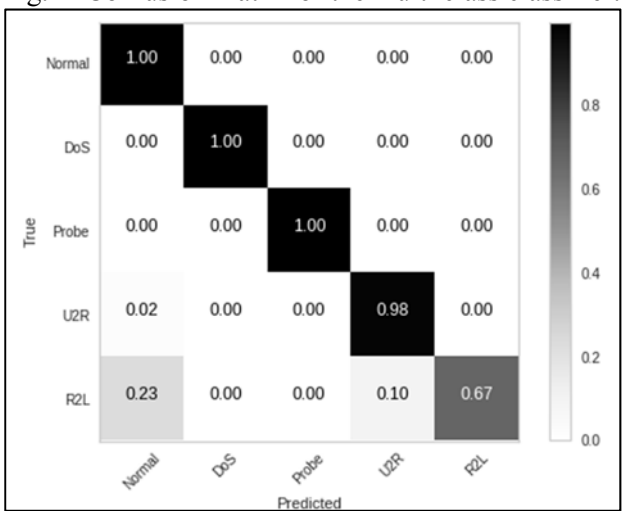


Fig.10 Accuracy convergence of the model.



The combined model, along with all binary classifiers in consideration combined with the multi class classifier gave the following results.

Fig.11 Confusion matrix of the multi class classifier.



6 Further Analysis

In this section, we resort to using various methods to help understand the importance of each feature viz a viz the information it provides in aiding the classification process. Three standard techniques have been used for the same, which are mentioned and further elaborated upon below:

6.1 Decision Tree Classifiers

For categorical variables, we used the weighted Gini index criterion after doing one hot encoding on the variables. Each individual categorical variable was separated into binary attributes. The weighted Gini index was calculated using the following formula:

$$W_G(A) = \sum_{a_i \in A} w(a_i) G(a_i)$$

$$w(a_i) = \frac{entries\ with\ value\ a_i\ in\ column\ A}{total\ entries\ in\ dataset}$$

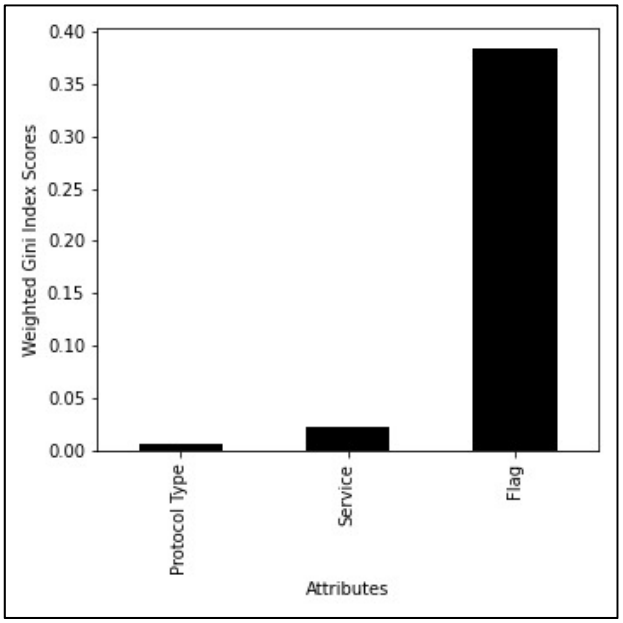
$$G(a_i) = Gini\ Index\ calculated\ for\ a_i$$

(2)

Categorical variables in the dataset have been inputted into a decision tree classifier to evaluate the importance of each in the classification process. This is measured as the total reduction of the Gini importance criterion caused due to each feature. The Gini Importance, also known as the Mean Decrease in Impurity is calculated as the number of times a feature is used to split a node, weighted according to the number of samples split at each node in a decision tree.

The conclusion in this regard is that while the Flag and Service variables are crucial in detecting intrusions, while the Protocol Type is not so important owing to its low Gini scores.

Fig.12 Feature Importance's using Decision Tree Classifier



6.2 Univariate Selection

For nominal features, we compared their importance using a chi square test (χ^2) which is commonly used for testing relationships between categorical variables. The null hypothesis of the Chi-Square test is that no relationship exists on the categorical variables in the population; they are independent.

We see from the following graph that these are the top 10 features we have identified in order of their importance.

Fig.13 Univariate Selection – Top 10 Features

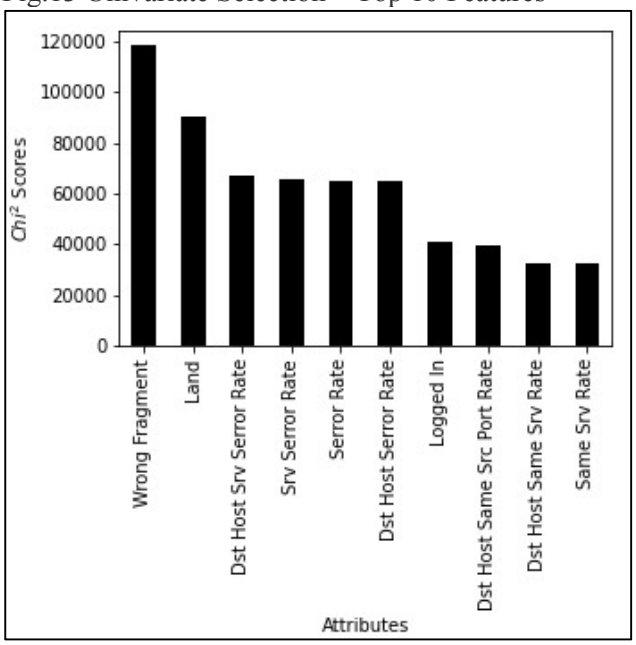
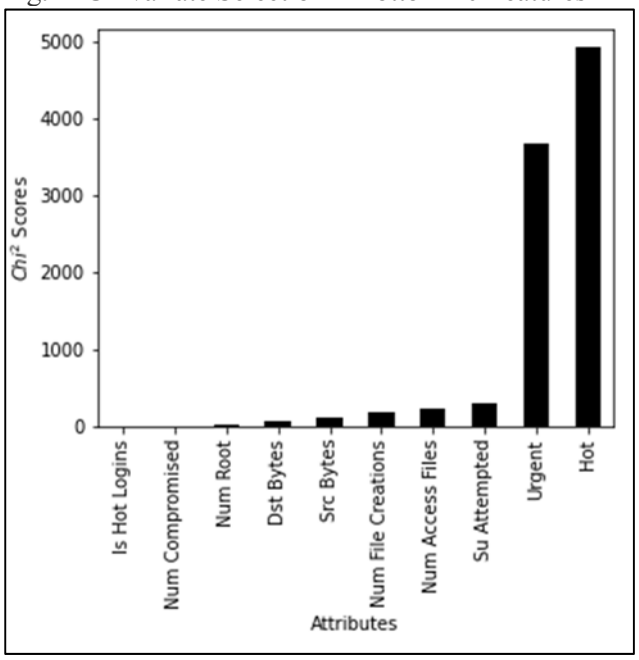


Fig.14 Univariate Selection – Bottom 10 Features



6.2 Correlation Plot

Correlation plots are used to determine the degree of correlation between every pair of features in a dataset. Each element corresponds to the Pearson's correlation coefficient between the two variables. Naturally, it is a symmetric matrix. Higher the correlation coefficient between two features, the stronger the linear relationship between them. Hence, each coefficient can be considered as a measure of the linear dependency between two variables. Ideally, we want a linearly independent set of features to maximize information present in the data and minimize redundancy. Assuming a threshold of 0.95 (or 95% correlation), we can identify 5 pairs of highly correlated features.

Fig.15 Correlation Heatmap with 95% Threshold

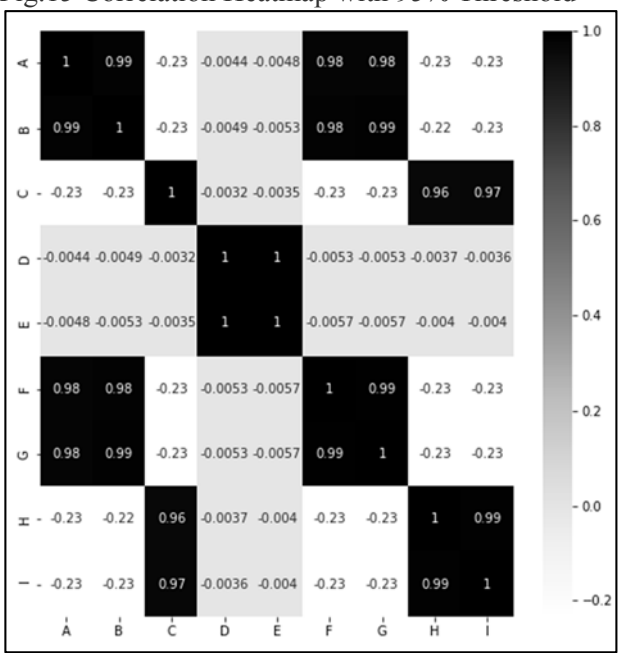


Table 2. Legend for Correlation Heatmap

A	Dst_Host_Serror_Rate
B	Dst_Host_Srv_Serror_Rate
C	Dst_Host_Srv_Rerror_Rate
D	Num_Compromised
E	Num_Root
F	Serror_Rate
G	Srv_Serror_Rate
H	Rerror_Rate
I	Srv_Rerror_Rate

7 Conclusion and Future Scope

The two-stage classification algorithm we considered here has resulted in much better evaluation results than the methods we explored. A reduced feature space ensures forward propagation through the model to work fast so that detection overhead is minimized.

Based on the further analysis, a total of 9 features: 8 numerical and 1 nominal, can be eliminated, reducing our feature space from 122 to 111. As the elimination has helped reduce redundancies in the dataset, we can expect our

model to perform better due to advanced deep learning algorithms on the new dataset. There is further scope for removal based on correlation plot data which would require further analysis.

Future scopes for this architecture include using GANs for generating adversarial samples that can compensate for the lesser proportion of U2R and R2L samples and generate a more global feature space that would result in better model parameters and would detect a larger proportion of attacks. A number of other over sampling techniques can be compared to the results we have obtained in this paper. The model can be trained on the new feature space for an expected enhanced performance.

References:

[1] A. Divekar, M. Parekh, V. Savla, R. Mishra and M Shirole, Benchmarking datasets for Anomaly-based Intrusion Detection: KDD CUP 99 alternatives, *Proceedings on 2018 IEEE 3rd International Conference on Computing, Communication and Security, ICCCS 2018*, 2018

[2] M. Roesch, Snort – Lightweight Intrusion Detection for Networks, *Proceedings of LISA '99: 13th Systems Administration Conference,* Seattle, Washington, USA, 1999.

[3] G. Wang, J. Hao, J. Ma, L. Huang, A New Approach to Intrusion Detecting using Artificial Neural Networks and Fuzzy Clustering, *Expert Systems with Application,* vol. 37, no. 9, pp. 6225-6232, 2010

[4] S. Shraddha, Intrusion Detection using Fuzzy Clustering and Artificial Neural Network, *Advances in Neural Networks, Fuzzy Systems and Artificial Intelligence,* pp. 209-217

[5] K. Peng, V. C. M. Leung and Q. Huang, Clustering Approach Based on Mini Batch Kmeans for Intrusion Detection System Over Big Data, *IEEE Access,* vol. 6, no. 1, pp. 11897-11906, 2018

[6] S. Potluri and C. Diedrich, Accelerated Deep Neural Networks for Enhanced Intrusion Detection System, *IEEE International Conference on Emerging Technologies and Factory Automation, EFTA,* 2016

[7] C. Yin, Y. Zhu, J. Fei and X. He, A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks, *IEEE Access,* vol. 5, no. 1, pp. 21954-21961, 2017

[8] J. Kim, N. Shin, S. Y. Jo and S. H. Kim, Method of Intrusion Detection Using Deep Neural Network, in *IEEE*, 2017

[9] Y. LeCun, Y. Bengio and G. Hinton, Deep Learning, *Nature,* vol. 521, no. 1, pp. 436-444, 2015

[10] J. Kim, J. Kim, H. L. T. Thu and H. Kim, "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection," *IEEE*, 2016

[11] Z. Chiba, A. Noreddine, K. Moussaid, A. E. Omri, M. Rida, Intelligent and Improved Self-Adaptive Anomaly Based Detection System for Networks, *International Journal of Communication Networks and Information Security,* vol. 11, no. 2, pp. 312-330, 2019

[12] P. Manandhar and Z. Aung, Intrusion Detection Based on Outlier Detection Method, *International conference on Intelligent Systems, Data Mining and Information Technology,* Bangkok, Thailand, 2014

[13] J. Vacca, Computer and Information Security Handbook*, Elsevier*, 2009.

[14] M. E. Tipping and C. M. Bishop, Mixtures of Probabilistic Principal Component Analysers, *Journal of the Royal Statistical Society,* pp. 443-482, 1999

[15] K. Rai, M. S. Devi and A. Guleria, Decision Tree Based Algorithm for Intrusion Detection, *Advanced Networking and Applications,* vol. 7, no. 4, pp. 2828-2834, 2016

[16] D. P. Gaikwad and R. C. Thool, Intrusion Detection System using Bagging Ensemble Method of Machine Learning, *2015 International Conference on Computing Communication Control and Automation*, Pune, India, 2015

[17] J. Zhang, M. Zulkernine and A. Haque, Random Forests Based Network Intrusion Detection Systems, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 38, no. 5, pp. 649-659, 2008

[18] Atthapol Ngaopitakkul, Chaiyan Jettanasen, Dimas Anton Asfani, Yulistya Negara Application of Discrete wavelet transform and Back-propagation Neural Network for Internal and External Fault Classification in Transformer, *International Journal of Circuits, Systems and Signal processing*, pp.458-463, Volume 13, 2019

[19] Dehong Ding, Sisi Zhu, A Method of Forest-Fire Image Recognition based on Ada Boost-BP Algorithm, *International Journal of Circuits, Systems and Signal processing,* pp.312-319, Volume 13, 2019