International Journal of Scientific Research and Engineering Development-– Volume 4 Issue 3,May- June 2021

Available at www.ijsred.com

Detection of Anomalous TCP/IP Network Requests using Autoencoders

Ayush Agarwal[1], Prableen Singh[2] ,Rohan Shetty[3], Aditya Raj[4], Mohit Singh[5],Pavithra H[6],Vinay V Hegde[7]

[1](*Computer Science & Engineering, R V College of Engineering,Bengaluru, India,ayushagarwal.cs17@rvce.edu.in*)
[2](*Computer Science & Engineering, R V College of Engineering,Bengaluru, India,prableensingh.cs17@rvce.edu.in*)
[3](*Computer Science & Engineering, R V College of Engineering,Bengaluru, India,shettyrohan.cs17@rvce.edu.in*)
[4](*Computer Science & Engineering, R V College of Engineering,Bengaluru, India,adityaraj.cs17@rvce.edu.in*)
[5](*Computer Science & Engineering, R V College of Engineering,Bengaluru, India,.mohitsingh.cs17@rvce.edu.in*)
[6](*Computer Science & Engineering, R V College of Engineering,Bengaluru, India,.pavithrah@rvce.edu.in*)
[7](*Computer Science & Engineering, R V College of Engineering,Bengaluru, India,vinayvhegde@rvce.edu.in*)

----------**********----------

Abstract:

With the increasing number of devices connected to the Internet, the need for network security becomes very critical. One of the crucial steps in ensuring network security is the detection of anomalous requests. In this paper, we discuss the use of autoencoders, a type of artificial neural network (ANN), in the detection of anomalous network requests using the KDD Dataset. The dataset is preprocessed and reduced to five columns using the Principal Component Analysis technique and is fed to the autoencoder which classifies the input as anomalous or normal network request. The input classified as anomalous is further fed to a 3-layered ANN to classify it as DDoS, Probe, R2L or U2R type of network attack. The architecture enables robustness and high accuracies of 84% on the autoencoder classification and 84% on the ANN classification.

Keywords—autoencoders, artificial neural networks (ANN), principal component analysis, network anomalies

----------**********----------

I. INTRODUCTION

The number of devices connected to the Internet has increased from approximately 7 billion in 2018 to around 26.66 billion in the year 2019 and an estimation of 31 billion devices is made for the year 2020. With the growth of the Internet of Things (IoT), artificial intelligence (AI), cryptocurrencies and other financial technologies along with the increased dependency of businesses, government and common people on the Internet and terabytes of confidential data being stored and circulated around the network, it becomes very important to ensure the security of the network and provide the required immunity to the users of the Internet. With the increasing computational power and availability of computers, the threat of cyber-attacks is alarming. Amongst the various strategies to ensure network security and reliability, is the ability to detect the arrival of anomalous network requests in order to take action and defend against them. Some of the common network attacks that are encountered include Distributed Denial of Service (DDoS), Remote to Local (R2L), Probe and U2R.

In the proposed system, an autoencoder model has been trained over the NSL - KDD dataset to assist the detection .of network anomalies. NSL-KDD dataset consists approximately 1,074,992 single connection vectors each of which contains 41 features The detection of network anomalies follows a two-step process:

- Classifying the network request as a normal network request or an anomalous network request with the autoencoder.
- If the above classifier detects an anomalous network request, then it is further classified as DDoS, Probe, R2L or U2R type of anomaly using a 3-layered ANN.

This detection is done using Autoencoders, which attempt to recreate input, by extracting core features from the data, taking into consideration the various patterns that the data might have. The paper discusses the architecture of the system along with the analysis of its performance and the corresponding results. The architecture provides an accuracy of 84% on the autoencoder model and 84% on the ANN model.

II. LITERATURE SURVEY

Anomaly Detection in TCP/IP network connection has been researched for quite a time, due to the increasing number of devices that are always connected to the Internet, and most of the traffic is on TCP/IP protocols.

It becomes necessary to effectively monitor the network logs to detect any anomaly in the network. A new fast and efficient approach using Neural networks is needed for analyzing the increasing amount of data, and detecting anomalies in it. Some of the papers discussing such methods are included in this section.

[1] goes in detail about the various machine learning models which can be used for anomaly detection using machine learning. It discusses in detail the use of Gaussian multivariate, K means, and LSTM models for anomaly detection. However, it does not tell about the best model which can be used for the majority of the cases.

[2] has listed Autoencoders as a suitable method for anomaly detection problems. It gives an architecture of the autoencoder model which can be used. We take the Autoencoder architecture from this paper. The architecture seems a bit complex here, and no discussion on reconstruction error has been made in this paper.

[3] is an exploratory paper about the NSL KDD dataset and is very useful for understanding the dataset and getting the most important parameters of it to be used in our model and thereby reducing the overall complexity of the model. The paper was used as a reference to extract features and confirm them with the results we achieve after Principal Component Analysis.

[4] leverages a streaming architecture based on ELK, Spark, and Hadoop in order to collect, store, and analyze network connection logs in real-time. The paper highlights the most important features and parameters to be displayed in a graphical manner which gives the most useful analysis of the real-time network connection logs

III. PROPOSED SYSTEM

The proposed system receives real-time network requests with the corresponding features as present in the dataset. The system pipeline (figure 1) is developed with the Python programming language using ElasticSearch and Kibana.The real-time data is preprocessed to transform it to the form that is expected by the anomaly detection system. The preprocessed data is then forwarded to the autoencoder model to determine if the request is anomalous or normal. If the request is identified as an anomalous request, then it is further forwarded to the 3-layered ANN classifier to determine the type of the anomalous request whether it is a DDoS, R2L, Probe, U2R or some other type of attack

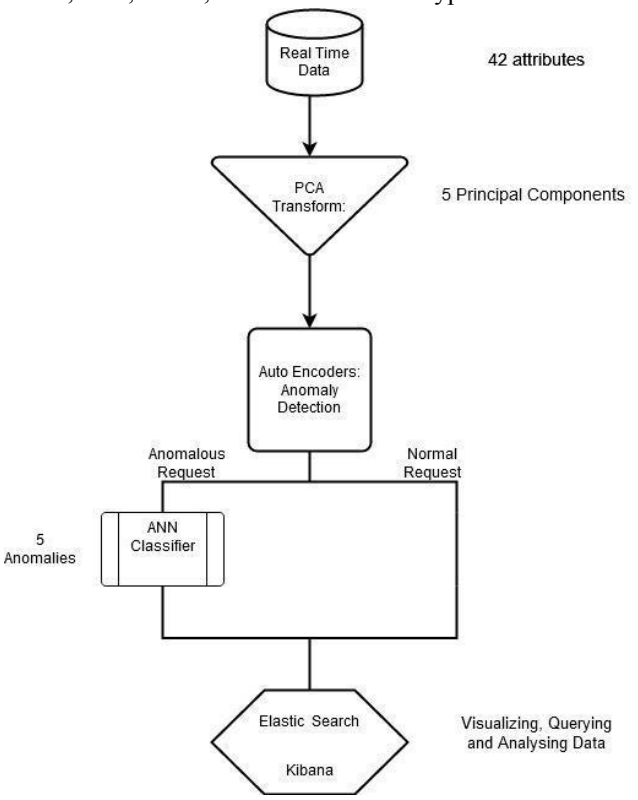

Figure 1 : Pipeline of the model

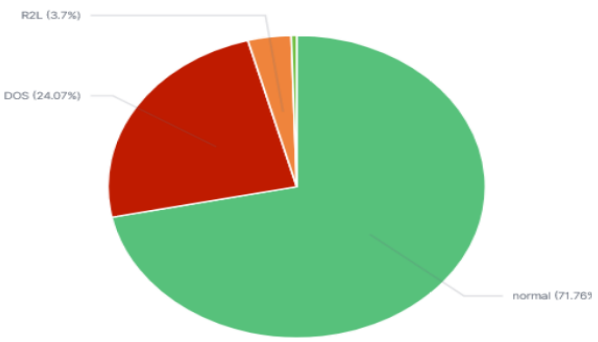

Figure 2: Pie Chart showing the ratio of types of network requests

The original network data obtained before preprocessing is appended with the request type predicted by the system and transferred to ElasticSearch, a NoSQL database. The process is repeated at regular intervals of time on the real-time data obtained during these intervals. The data and corresponding statistics are viewed on Kibana, a user-interface that allows creation and viewing of data in raw form or graphical visualizations. Kibana also allows a user to manage the Elastic Search database through it's simple

interface. This allows network managers or any other users who are not particularly experts in the field of machine learning and artificial intelligence to use the system for their network management and security requirements.

It is to be noted that the model is trained on a particular dataset and expects a set of similar features to give accurate results. Although, the architecture may be used for any dataset, a corresponding system may be developed based on the data and it's features available to the end user

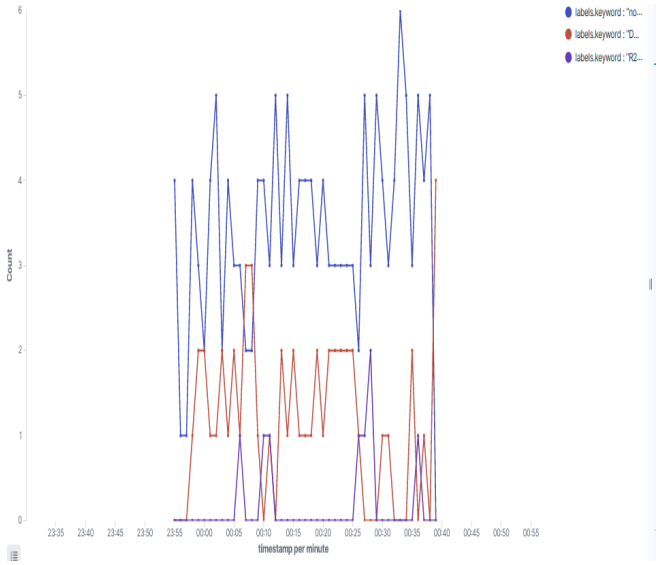

Figure 3:No of request per minute corresponding to each class (color code)

IV. ARCHITECTURE

A. Description of KDD Dataset

As earlier specified, the autoencoder model has utilized the NSL-KDD dataset for training. The original KDD dataset, published in 1999, was found to include some statistical issues which degraded the evaluation of anomaly detection and further affected the performance of security analysis. As a result, the NSL-KDD (published 2009) dataset was suggested keeping in mind the inherent problems with the original KDD dataset.

The training dataset of NSL-KDD consists of approximately 1,074,992 single connection vectors each of which contains 41 features (listed in Table below). Each connection is labelled as either normal or attack type. Feature reduction techniques can be applied to reduce the size of the dataset.

The dataset defines various high-level features that aid in distinguishing anomalous network connection requests from normal ones. Categories of the derived features include the "Same Host" features that only consider the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. The "Same Service" features examine only the connections in the past two seconds that have the same service as the current connection. Together, the "Same host" and "Same service" features constitute the time-based traffic features of the network connection records. Two attacks, namely the Denial of Service (DoS) and Probing attacks, involve a large number of connections to some hosts in a very short period of time, and can hence be identified using this dataset.

TABLE 1
LIST OF FEATURES

S.NO	FEATURE NAME	S.NO	FEATURE NAME
1	Duration	22	Is_guest_login
2	Protocol type	23	Count
3	Service	24	Serror_rate
4	Src_byte	25	Rerror_rate
5	Dst_byte	26	Same_srv_rate
6	Flag	27	Diff_srv_rate
7	Land	28	Srv_count
8	Wrong_fragment	29	Srv_serror_rate
9	Urgent	30	Srv_rerror_rate
10	Hot	31	Srv_diff_host_rate
11	Num_failed_logins	32	Dst_host_count
12	Logged_in	33	Dst_host_srv_count
13	Num_compromised	34	Dst_host_same_srv_count
14	Root_shell	35	Dst_host_diff_srv_count
15	Su_attempted	36	Dst_host_same_src_port_rate
16	Num_root	37	Dst_host_srv_diff_host_rate
17	Num_file_creations	38	Dst_host_serror_rate
18	Num_shells	39	Dst_host_srv_serror_rate
19	Num_access_shells	40	Dst_host_rerror_rate
20	Num_outbound_cmds	41	Dst_host_srv_rerror_rate
21	Is_hot_login		

Domain knowledge has been utilized to add features that look for suspicious behavior in the data portions, such as the number of failed login attempts. These features are known as "content" features. This proves to be very useful in predicting intrusions like R2L and U2R attacks since these are embedded in the data portions.

B. Preprocessing Techniques

Data preprocessing is a critical step in machine learning, especially when we are dealing with large amounts of data. Preprocessing majorly consists of three steps which are, data cleaning and processing, feature reduction and feature construction. Data cleaning and processing involves tasks such as handling null values, encoding categorical

features and transforming data so that it can be used in the best possible manner. The dataset used has no null values. The dataset has three categorical features namely protocol, service and flag. Protocol has three distinct values namely tcp, udp and icmp. Service has 70 distinct values and Flag has 11distinct values which are SF, S0, REJ, RSTR .SH, RSTO, S1, RSTOSO, S3, S2, OTH. To encode these categorical values into numeric data label encoders have been used. Label encoders assign a unique numeric value to each of the distinct values. For example, if protocol feature is considered tcp is assigned value zero, udp is assigned one and icmp two. The main advantage of using label encoder over one-hot encoder is that label encoder uses much less memory when compared to one-hot encoder and since we are dealing with real time data label encoder is preferred. Feature extraction transforms high dimensional data to lower dimension. For feature compression PCA (principal component analysis) technique has been used. PCA is a linear technique in dimensionality reduction for data analysis and compression. PCA was preferred because it was compatible with real time data. In PCA we calculate covariance matrices. Find the eigenvectors and eigenvalues of the covariance matrix and then sort it in descending order to form the feature set. The eigenvector with highest eigenvalue represents the principal component. The scree plot is a graph which plots the accumulated variance against no of principal components. Since PCA involves projecting points onto a vector (eigenvector), it has to be made sure that each instance is a unit vector. To convert each instance to a unit vector l2 norm was used. By l2 norm it is meant that when we square each term and take its sum, we get the result as one. The process of converting each instance to a unit length is also called normalization. After data cleaning and encoding the categorical features the data was normalized and scree plot was built.

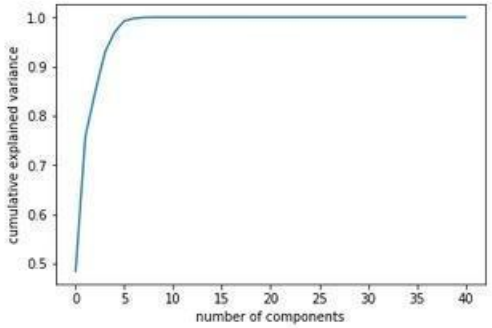

Figure 4:Scree Plot of PCA

In the above scree plot the elbow decides the selection of eigenvalues. There is one elbow in the scree plot, where k=5. Therefore 41 features of NSL KDD dataset are reduced to five features.

Apart from preprocessing the training data, the labels were also grouped as and when required. To train autoencoders the labels were grouped into two broad categories namely normal and anomaly. To train the ANN the labels were grouped into four broad categories namely DoS (denial of service), probe, R2L (root to local) and U2R (user to root). The grouping of labels into four major anomalies is done according to the table below.

TABLE 2
CLASSIFICATION OF ATTACKS

Attacks in Dataset	Attack Type (37)
DOS	Back,Land,Neptune,Pod,Smurf, Teardrop,Mailbomb,Processtable,Udpstorm,Apache2,Worm
Probe	Satan,IPsweep,Nmap,Portsweep,Mscan,Saint
R2L	Guess_password,Ftp_write,Imap,Phf,Multihop,Warezmaster,Xlock,Xsnoop,Snmpguess,Snmpgetattack,Httptunnel,Sendmail, Named
U2R	Buffer_overflow,Loadmodule,Rootkit,Perl,Sqlattack,Xterm,Ps

C. Architecture of Autoencoder

"An outlier is an observation in a dataset
which appears to be inconsistent with
the remainder of that set of data."
- Johnson 1992

It seems very reasonable to think of anomalies as deviations. Intuitively, the usage of clustering algorithms to detect anomalies looks particularly useful. However, they don't scale well and are not very promising if the number of attributes is high. Decision-tree based algorithms like Isolation forest have also been well documented for the purpose of Anomaly detection.

This paper proposes the usage of Autoencoders for distinguishing anomalous network requests from the normal ones. Autoencoders are a family of neural networks that essentially try to recreate the input. In other words, autoencoders accept the input with all of its core features and attempt to learn the pattern of data by learning an identity function $h(x) \approx x$. There is some loss (reconstruction error) that is involved in this process. But the intuition is that the encoder will be to accurately recreate the input corresponding to the normal network requests, and will incur a large loss in reconstructing the anomalous network requests.

To learn this identity function, an autoencoder uses two main components – an encoder and a decoder. As the names suggest, the encoder accepts the input and compresses it into a latent space representation, whereas the decoder aims to reconstruct the input from this latent space representation. An ideal autoencoder should be able to reconstruct the exact same input which was initially passed to it, but in the real world there is a finite "reconstruction error" corresponding to each input tuple. This reconstruction error can be utilized as a statistical metric to draw inferences. Comparing the observed "reconstruction error" with a threshold error (calculating during training of the model) can distinguish between the two types of requests.

A 3-hidden layer neural network has been used as the autoencoder, with 10, 2 and 10 neurons each, using Exponential Linear Unit (elu) as the activation function. The input and output layer have 5 units, corresponding to each Principal Component. We use the mean square error as a loss function, and train the model using the "Adam" optimizer. The distribution of the calculated loss is plotted for the training set, and then used to identify a suitable threshold value for identifying an anomaly. The set threshold is kept above the "noise" level.

D. Architecture of 3-layered ANN Classifier

The incoming network request is flagged either as a normal request or an anomalous one. In the latter case, the input is made to pass through a classifier, which classifies it into one among four network anomaly attack groups – DoS, Probe, R2L, U2R. An artificial neural network has been used as a classifier, trained on the same NSL-KDD dataset.

The classifier is a 3-layered ANN, with 8 neurons in the first 2 hidden layers. The hidden layers utilize the elu activation function. The output layer has 5 neurons (4 corresponding to the 5 anomaly groups + Others). It utilizes the SoftMax activation function. "Adam" optimizer and "cross-entropy" is used as the loss function.

V. RESULTS

A. Autoencoder

The input data is reduced into 5 principal components by the PCA transform. The autoencoder takes this 5-tuple as input and attempts to recreate it using the encoding-decoding mechanism. The following table shows the predicted values of the 5 Principal components for a set of input data.

```
[[ 0.6680492   0.54097927  0.0868991  -0.05402528 -0.00522541]
 [ 0.6531236  -0.01969635  0.45396066 -0.32774618 -0.18531074]
 [-0.60170186 -0.00917315  0.08081692  0.01417885  0.02402122]
 ...
 [ 0.35662824 -0.3270517  -0.05296606 -0.15523145 -0.2176622 ]
 [ 0.47292104  0.6330303  -0.02276951 -0.03537756  0.03083922]
 [ 0.6871091  -0.06219742  0.22894216 -0.31450894 -0.18044956]]
```

Figure 5: Predictions

To see the relationship between the predicted values and the actual input values, the following graphs were plotted for the Principal Component 2 for 20 input vectors.

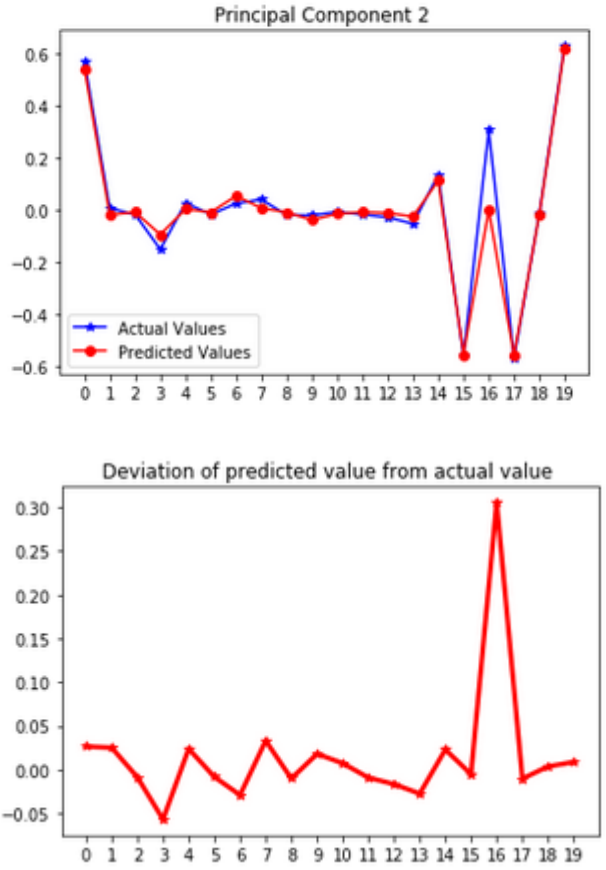

Figure 6 : deviation graph

The second graph shows the deviation or loss that the autoencoder incurs in the process of reconstruction for each input vector. During the training phase, all the 5 deviationsare averaged to yield a "reconstruction error" for the particular input vector. Quartile analysis is performed separately for the normal network connection requests and the anomalous network connection requests, to obtain a threshold value of reconstruction error.


```
count    67343.000000
mean         0.032493
std          0.034213
min          0.003228
25%          0.014153
50%          0.021965
75%          0.036279
max          0.480447
dtype: float64
```

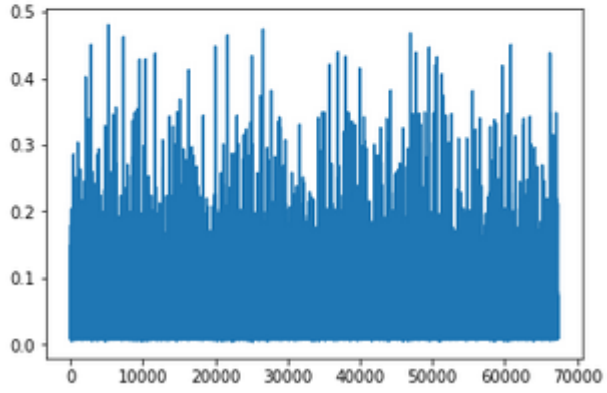

Figure 7: Quartile Analysis for normal request

```
count    58630.000000
mean         0.140746
std          0.068055
min          0.005873
25%          0.087103
50%          0.167009
75%          0.182102
max          0.454890
dtype: float64
```

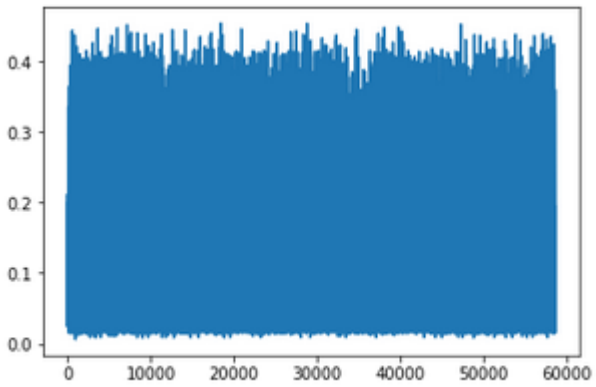

Figure 8: Quartile Analysis for anomalous request

As noted in the figure 7, 75% input corresponding to normal connection requests witness a reconstruction error of 0.036279 or below. Whereas, only 25% input corresponding to anomalous requests have a reconstruction error of 0.087103 or below. Since the 25th percentile of anomalous is greater than the 75th of the normal one, a good starting approximation for the threshold reconstruction error would be the mean of these two values, i.e. mean of 0.036279 and 0.087103. This amounts to 0.06169. All the subsequent predictions are done with respect to this threshold.The accuracy obtained for detecting anomalous requests using the proposed model is 84.66%

B. 3-layered ANN

The following confusion matrix was obtained after testing the ANN classifier. Each row corresponds to one type of anomaly group, namely – DoS, R2L, Probe, U2R and Other (in order from top to bottom). The diagonal elements indicate the True Positives for that particular class. Clearly a large chunk of the anomalous requests is due to DoS attacks (7587) and Probe attacks (1748), indicating that the dataset is skewed in terms of anomalies.

```
[[7587  169  338    1    0]
 [   2  101    4    0    0]
 [ 400    5 1748    4    0]
 [ 123  106  646   25    2]
 [   2   11    4   18    3]]
```

Figure 9 : Confusion matrix

The precision, recall, f1-score and support corresponding to each anomalous class are summarized below

	precision	recall	f1-score	support
0	0.94	0.94	0.94	8095
1	0.26	0.94	0.40	107
2	0.64	0.81	0.71	2157
3	0.52	0.03	0.05	902
4	0.60	0.08	0.14	38
accuracy			0.84	11299
macro avg	0.59	0.56	0.45	11299
weighted avg	0.84	0.84	0.82	11299

Figure 10 : metrics of the model

The accuracy obtained for classifying into anomaly sources is 84%

VI. CONCLUSIONS

The proposed system addresses the objective of detecting anomalous network requests and classifying it to the corresponding anomaly type.

The use of autoencoders along with the ANN classifier is able to effectively determine the type of the network request with a combined accuracy of 84%.. This ensures reliability on the outcomes provided by the system and to allow necessary actions to be taken to ensure network safety.

The architecture discussed could be used on other network request datasets with reduced or varied features to obtain similar results.

The development of a real-time network data pipeline with ElasticSearch and Kibana allows the end user to effectively create inferences from the data and the statistics obtained from it. The system may be integrated with existing network safety management software for improved performance in network anomaly detection

ACKNOWLEDGMENT

We would like to thank RV College of Engineering for their constant support in the field of Research and DevelopmentWe would like to extend our gratitude to Nokia, Bengaluru for guiding us during the research. Also, we would like to thank all the contributors of the NSL-KDD Dataset for providing apt data and thus encouraging research in the field of computer networking.

REFERENCES

[1] [1] Chakraborty, Niloy & Nair, Roshan & Kasula, Chaithanya Pramodh & Vankayala, Sravanthi. (2019). IP Network Anomaly Detection using Machine Learning. 10.1109/I2CT45611.2019.9033545.

[2] M. Sakurada and T. Yairi, “Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction,” presented at the MLSDA 2014 2nd Workshop, 2014, doi: 10.1145/2689746.2689747.

[3] Chae, Hee-su, Byung-oh Jo, Sang-Hyun Choi and Twae-Kyung Park. “Feature Selection for Intrusion Detection using NSL-KDD.” (2013).

[4] Mehta, Swapneel, et al. ‘Anomaly Detection for Network Connection Logs’. ArXiv:1812.01941 [Cs, Stat], Nov. 2018. arXiv.org, http://arxiv.org/abs/1812.01941.

[5] S. B. Wankhede, "Anomaly Detection using Machine Learning Techniques," *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, 2019, pp. 1-3, doi: 10.1109/I2CT45611.2019.9033532.

[6] K. Limthong and T. Tawsook, "Network traffic anomaly detection using machine learning approaches," 2012 IEEE Network Operations and Management Symposium, 2012, pp. 542-545, doi: 10.1109/NOMS.2012.6211951.

[7] Y. Imamverdiyev and L. Sukhostat, "Anomaly detection in network traffic using extreme learning machine," 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT), 2016, pp. 1-4, doi: 10.1109/ICAICT.2016.7991732.

[8] T. Wahyono, Y. Heryadi, Lukas, A. S. Achmad, H. Soeparno and B. S. Abbas, "Anomaly detection to evaluate in-class learning process using distance and density approach of machine learning," *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, 2017, pp. 1-6, doi: 10.1109/INNOCIT.2017.8319138.

