# Detection of DDoS attacks using RNN-LSTM and Hybrid model ensemble.

MSc Internship
CyberSecurity

Siva Sarat Kona
Student ID: 18170366

School of Computing
National College of Ireland

Supervisor: Christos Grecos

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Siva Sarat Kona |
| **Student ID:** | 18170366 |
| **Programme:** | CyberSecurity |
| **Year:** | 2019 |
| **Module:** | MSc Internship |
| **Supervisor:** | Christos Grecos |
| **Submission Due Date:** | 12/12/2019 |
| **Project Title:** | Detection of DDoS attacks using RNN-LSTM and Hybrid model ensemble. |
| **Word Count:** | 5230 |
| **Page Count:** | 21 |

| | |
|---|---|
| **Signature:** | |
| **Date:** | 29th January 2020 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

| | |
|---|---|
| **Office Use Only** | |
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Detection of DDoS attacks using RNN-LSTM and Hybrid model ensemble.

Siva Sarat Kona
18170366

**Abstract**

The primary concern in the industry is cyber attacks. Among all, DDoS attacks are at the top of the list. The rapid increase in cloud migration also increases the scope of attacks. These DDoS attacks are of different types like denial of service, distributed denial of service, Slowloris, and so on. There are many implementations to detect the attacks. There are different types of detection systems and using popular machine learning techniques. A lot of research is going under the improvisation of machine learning techniques. The existing implementations are proved to predict better results with classifiers like Decision Trees, Support Vector Machine(SVM), Logistic Regression, and Neural networks. Also, many types of research proved to be more efficient by combining the algorithms to achieve high accuracy. Usually, network data is immense. So, generating and maintaining a hybrid model requires high execution time and more resources. My model is a solution with the hybrid implementation of the model using ensembling. Recurrent neural networks are used in weather, share prices, e-commerce and typing prediction. Many big players in this industry were adopting to this prediction model. This widely used time series analysis algorithm is used to predict the anomaly within the dataset. Based on the prediction period, the data is sent to a hybrid model to detect the attack record. This hybrid model is built on the high accurate prediction model "Random Forest" along with high customizable algorithm "Neural Network". With this implementation, I can achieve an accuracy of 95.2% and 83%, respectively. The ensemble model with these two algorithms chooses the best model based on model voting. The final model built with an accuracy equal to the Random forest.

## 1 Introduction

Nowadays, Cloud is a highly adopted technology due to its high flexibility and low maintenance cost. Most of the devices, such as the Internet of Things(IoT), Mobile devices, computers, and so on, are actively connecting to the internet to access Cloud. This massive shift of demand towards public Cloud is attracting attackers. Public Cloud is de-perimetrized, and anyone with internet connectivity can access. These factors are pushing security specialists to find new ways to detect the attack in advance.

There are few worst attacks of all time published by the popular mitigation platform "CloudFlare". These are classified based on the amount of traffic generated during the period. In 2000, a young hacker named Mafiaboy attacked major websites of Dell, eBay, Yahoo, CNN and E-Trade created stack market chaos. The attackers targeted the government services and financial institutions of Estonia in 2007. Spam related attack was in 2013 on the famous Spam filtering company Spamhaus. GitHub attack in 2015, using HTTP requests from malicious code. Dyn company was attacked in 2016; This attack was made using the system from the most prominent attack in the history ”Mirai”. These attacks were targeted to create havoc. Some of them are mitigated, and some attacks created a massive loss for the victims[1].

In the computer security field, Intrusion detection is an automated ”intruder alarm”. There are three major detection principles of IDS. The classification is Anomaly-based, Signature-based and signature inspired. The anomaly-based will raise an abnormal flag when the percentage of the activity throttles over the defined levels. Signature-based detects the anomalies based on the signatures(series or events) already classified in the model. These are much reliable than anomaly-based detection, but signatures need to be updated regularly to get the most reliable detection. Signature-inspired or compound detection is a combination of the normal behavior of the system and signature behavior of the attacker. This model is much stronger and more precise than other models. Also, it will carry the short comes of the above two models [1]. Denning [2] in the year 1987 introduced real-time intrusion detection. He created a profile to identify intrusions based on the past anomaly activities with specific rules.

The research of Denning [2] is further improvised in 1990 by Heberlein et al., [3]. The real-time detection is moved ahead to real-time monitoring and detection by the end of the year 1990. Research is further improved by the Dowell et al. [4] by creating a tool ”ComputerWatch” with all the above functions. Mukherjee et al., [5] in 1994 added the reporting to administrator functionality to the tool. The first leap in research is from Okazaki et al. [6] in 2002 by introducing a signature-based detection system. These researches helped the industry to automate the intrusion detection process.

Intrusion detection systems help in detecting vulnerabilities ahead. Most popular platform GitHub was attacked in February 2018, with traffic flow in terabytes per second. This attack is one of the largest DDOS attacks. In technical terms, the attack is ”Memcached DDoS attack” which involves no botnets. Still, the attackers amplified the traffic 50000 times than usual. Luckily, Mitigation alert triggered on time, and the company escaped without any loss[1].

The data plays a significant role in data mining. We all know, no model is ideal and has its disadvantages. But the considerable percentage of the models will give false alarms because of biased models. There are different types of bias like confirmation bias, Interpretation bias, prediction bias and information bias.[2] The confirmation bias is dependent on the data and will occur if the analyzed data doesn’t represent the full scenario. Example of this is the US presidential election. Besides, there are other biases which will depend on the dataset considered. They are Availability bias, which will occur only when the available data is limited and Selection bias, which will be based on the selection of data sample.[3] So, Accuracy of the output value has a vital impact on the data considered.

To minimize the false alarm rate, many researchers used machine learning(ML) techniques. The main goal of ML is to train the model to detect the new attacks consistently. The study of researchers on the datasets and the neural networks proved that the combination of both would result in highly accurate anomaly detection. One of the related research by Jia et al. [7] in 2019 with a deep neural network on NSL-KDD resulted in

[1]Famous DDoS Attacks:`https://www.cloudflare.com`

[2]Avoiding bias in data analytics: `https://www.allerin.com`

[3]Four cognitive biases that affect big data analysis: `https://www.bigdata-madesimple.com`

99% accuracy using test data. Wu et al. [8] research in 2017 proved RNN FAR values are better than traditional classifiers. Ensembling the above techniques will results in more accurate values, based on the Riyad.A [9] research in 2017

This whole evolution of IDS still needs some improvement in the detection strategy. This paper will combine the capability of deep neural networks (RNN-LSTM) along with random forest and neural network ensemble with new dataset CICIDS from the Canadian Institute of Cybersecurity to achieve a better detection model.

The research of this paper is well ordered as follows. In section 2, I have reviewed and added the previous research related to intrusion detection, how the deep Neural networks like RNN helps in enhancing the detecting the attack time frame, How the dataset or the data sample used to affect the accuracy and so on. The briefing of tools in proposed architecture, The research architecture and methodology, measurable parameters are explained in section 3 & 4. Section 5 highlights the experimental measures and comparison between random forest and Neural networks. The conclusions and the scope of future work are discussed in section 6.

## 2 Related Work

### 2.1 Machine learning

Machine learning techniques have been using for a long time. Without the need for explicit programming, it extracts the patterns and has the ability of learning. The representational data [10] and domain knowledge is the core of traditional machine learning techniques to get the best fit from raw data. Using the best fit model, deep learning can solve many problems. Many kinds of research already proved that machine learning capabilities are showing exceptional results [10] in many applications.

#### 2.1.1 Deep belief network

According to Hinton et al. pre-training Deep Belief Network [11] effective strategy, when traced back in 2006, the deep networks research interest wave can be seen. Deep belief networks are generally used for the motion-capture data, generate images and clusters, video sequences. The extension for the deep belief network, i.e. the continuous deep belief network that supports a continuum of decimals other than binary data. The Graphical Processing Unit (GPU) advances are motivated further by a surge of interests. The heart of deep learning is the Graphics Processing Unit(GPU). To free the CPU cycles for different jobs, a single-chip processor for mathematical and graphical computations are used. Deep network training for the efficiency in GPU is very efficient for experimentation [11].

#### 2.1.2 Multi layer learning - RBM

A generative probabilistic model is a Restricted Boltzmann Machine (RBM). For the reconstruction of the inputs, the probability distribution of learning capability is high. The two layers of RBM are Hidden layer and Visible layer. The Visible layer will change into the hidden layer by stacking the RBM. The higher-level representations [12] can arrive by forming the Deep Belief Network(DBN). The Deep Belief Network, which is a class of deep neural networks that is a composition of latent variables of multiple layers that has connections in between the layers and each layer not in between the units.

In contrast to past writing, hubs in the concealed layers are presently "completely associated". The creator has contemplated the presentation on both twofold and multi-class grouping. The exploratory outcomes show that the presentation is of progressively predominant. In another examination, Kim et al. [13] embrace sans hessian advancement calculation to address the trouble of taking care of complex long-haul conditions in RNN. Sans hessian enhancement permits quicker intermingling without calculation of Hessian grid. The investigation of profound learning approach in NIDS has gotten developing consideration as of late. The more significant part of the current intelligent learning strategies for NIDS is based on KDDCup'99 and NSL-KDD informational indexes. On account of stream-based NIDS, the endeavour of profound learning study is inadequate.

#### 2.1.3 Combinations and hidden layers

According to Salama et al. [14], The earliest form of implementation for DBN in the NIDS is proposed. To deduct the 41 features, DBN from 2 RBM layers were used from 5 output features of NSL-KDD. DBN is trained with multiple configurations and is also trained with backpropagation. The two different configurations are: Classifier by itself and applying the classifier Support Vector Machine(SVN) after performing the dimension reduction. The DBN and standalone SVM are outperformed when the DBN-SVN combination results are initially compared. The DBN-based models are much more successful and received more recognition when compared with the different approaches. Later the performance was increased by performing hidden layers of DBN and demonstrated them. Higher performance was achieved on KDDCup'99 data set, for a combination of 4 hidden layers.

## 2.2 Neural Networks

#### 2.2.1 Recurrent neural network

More considerable attention is received in the domain for the network intrusion detection using the Recurrent Neural Network (RNN). The Recurrent Neural Network (RNN) consists of a linear graph between nodes with a temporal sequence which is an artificial neural networks class. This class allowed the dynamic behaviour for a temporal exhibit. To process the inputs of sequences, the internal memory state RNN's can be used. RNN is capable of memorizing and perceive by introducing the feedback loop, which relates to the previous time step, which is entirely different from the feed-forward neural network. A three-layer diminished size RNN is proposed by Sheikhan [15], in which the hubs are incompletely associated between layers. The execution of this structure permits both compelling preparing speed and improved characterization rate on KDDCup'99 informational index.

#### 2.2.2 LSTM - Long short term memory

According to Ralf C. Staudemeyer [16], the RNN proposed variant to eliminate the severe vanishing and exploding problems by using Long Short-Term Memory (LSTM). Not at all, like standard feed-forward neural systems, LSTM has input associations. It cannot just process single information focuses, (for example, pictures), yet additionally whole arrangements of information, (for example, discourse or video). There exist many proposed variations of LSTM engineering. Most current vanilla LSTM has since involved adjustments, including overlook forget gate [16] and peephole associations. Other eminent variations likewise incorporate a lesser complex Gated Recurrent Unit (GRU) design. Results acquired in has demonstrated the strength of vanilla LSTM in taking care of different informational indexes when contrasting with different variations of it.

The formulae of non-peephole implementation that proposed forward pass:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3}$$

$$c_t = (c_{t-1}) \odot (f_t + i_t \odot tanh(W_c[h_{t-1}, x_t] + b_c)) \tag{4}$$

$$h_t = tanh(c_t) \odot o_t \tag{5}$$

The three main scenarios where each LSTM cells are: forget gate $\mathrm{f_t}$, input gate $\mathrm{i_t}$, output gate $\mathrm{o_t}$ at time t step. The entryways actuation utilizes insightful calculated component sigmoid capacity ($\sigma$). The loads and biases are W and b, for the entryways or state (eqn:**??**). $\mathrm{x_t}$ is the input information, and $\mathrm{h_{t\text{-}1}}$ is the hidden concealed state from past time step. The refreshed cell state filled in as a memory and is spoken to utilizing $\mathrm{h_t}$. At last, $\odot$ is a shrewd component multiplication of two vectors. Values in between 0, 1 are the outputs of a forget gate (eqn:**??**). To forget the previous steps to set the decision. The LSTM cells can learn when performing complex tasks and unreasonably long tasks when the memory contents are resettled.

## 2.3 Datasets for IDS

### 2.3.1 DARPA

Some of the oldest datasets, i.e. the DARPA'98 and DARPA'99 [17] are generally designed for IDS assessments. While extending DARPA'98 to DARPA'99, various Windows NT victim machines and different attack types are included. DARPA'98 & DARPA'99 are created similarly, with the traffic captured from a simulated offline military environment. The simulated data is categorized into four categories. The categories are Remote to Local, User to Root, Probes and Denial of service(DOS) [17]. The three different type of features that the author has processed with different types of connection.

### 2.3.2 KDDCup

The KDDCup'99 data set was derived using the DARPA'98 TCP dump data. The considered features of DARPA'98 dataset are Domain features, data and characteristics of traffic from the past 2 seconds, basic features of Single TCP connection. From the creation, the remaining used data remains in the KDDCup'99 [17]. Problems exist with the above datasets, and they are mentioned in different studies. Because of the prominence of KDDCup'99, few studies and investigations have been endeavoured to refine the data index. The data index: gureKDDCup was created by a similar system to duplicate KDDCup'99 as exact as could be expected under the circumstances while keeping up included highlights. gureKDDCup informational collection conveys every one of the highlights of KDDCup'99 with additional payload data, IP locations and port numbers. NSL-KDD informational index is another endeavour to solve a portion of the intrinsic shortcomings in KDDCup'99. All the records are redundantly removed by the author in testing and training data sets to eliminate many records in a biased way. Moreover, the chose subsets of records are conversely relative to their trouble level of expectations in the first KDDCup'99 data sets using different machine learning methods.

### 2.3.3 NSL-KDD

According to Sperotto et al. [18], few problems were carried by the NSL-KDD. The KDDCup'99 dataset features consist of 14 varying features. The present of attacks is investigated by assisting multiple ten additional features like malware, IDS trigger and attacks. IDS dataset with the first labelled flow is generated in 2009. The glimpse of network traffic is obtained when the traffic data was captured in the University of Twente, in a honeypot, which is more similar to the old dataset. The labelling tasks log files are collected when the honeypot is executed using the services like Apache web server, SSH, and FTP.

The main aim is to catch the botnet traffic, alongside ordinary and regular traffic. Ordinary labels are known and controlled PCs in the system, while background names are allocated to other not known traffic. The informational set has 13 different scenarios, each with varying malware activities. Initially, unidirectional streams were utilized. However bidirectional stream is later moved to incorporate substantially more point by point names.

### 2.3.4 CIDDS-001

According to Ring et al. [19], the stream-based information is distributed in 2017, CIDDS-001. The data collection contains two different sources of traffic: outside server traffic and interior Open-Stack traffic which are presented to the Internet. Inside the controlled Open-Stack arrange, an aggregate of four distinctive subnets are intended to imitate the nature of an external server. Each subnet has its own endorsed practices and pursues the probability dispersion of working hours. Traffic which is not produced by the OpenStack customers is viewed as obscure for HTTP or HTTPS demands, or suspicious for remaining traffic at the external server.

### 2.3.5 CICIDS

According to Sharafaldin et al. [20], the hour of composing the latest data collection is CICIDS2017. The data collection covers every set of the eleven criteria of the assessment framework and utilizes the profile proposed in past work [20], Six distinctive assault profiles were made to incorporate normal and updated assaults. The NSL-KDD [12] and KDDCup'99 studies are performed to evaluate the performances on network detection intrusion methods. The real-world modern traffic is inferior in reflecting both of them as the essential DARPA'98 data set is 20 years old.

# 3 Methodology

## 3.1 High level design

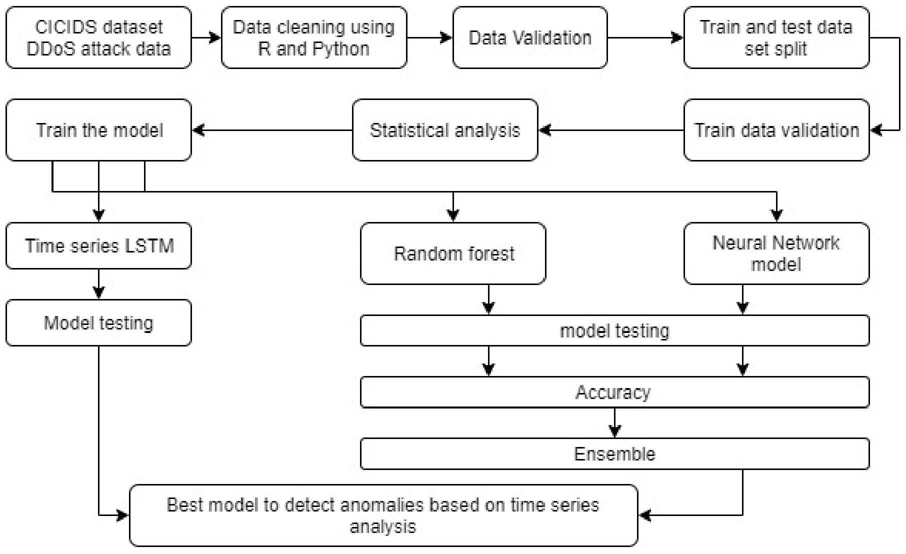
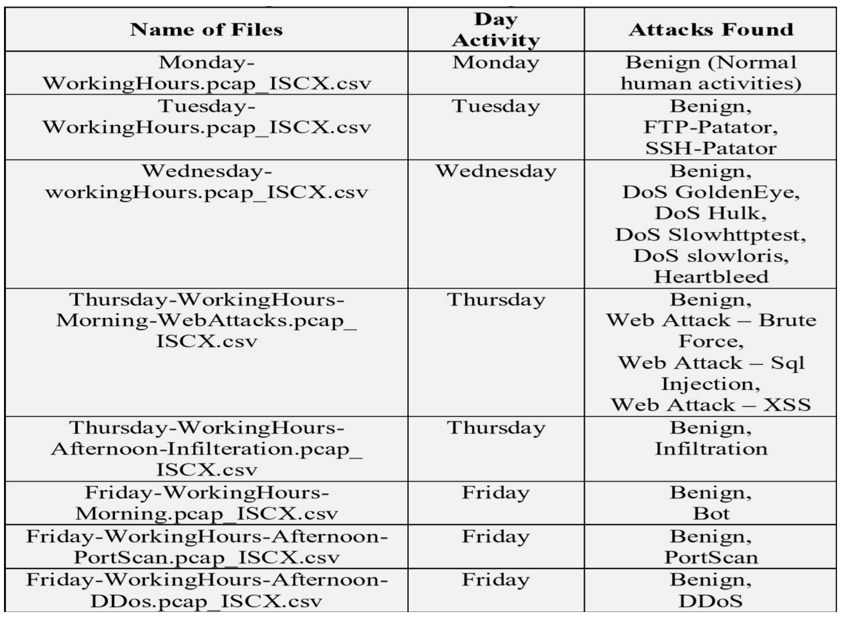


Figure 1: Representation of high level design

## 3.2 CICIDS2017 Dataset

The main advantage of CICIDS dataset is newly added attacks, this attracted researchers for the enhancement of their analysis and to develop improvised models. This collection has information of 5 days attack and general traffic distributed among eight files. The below table is the illustration of the data taken by the researcher.

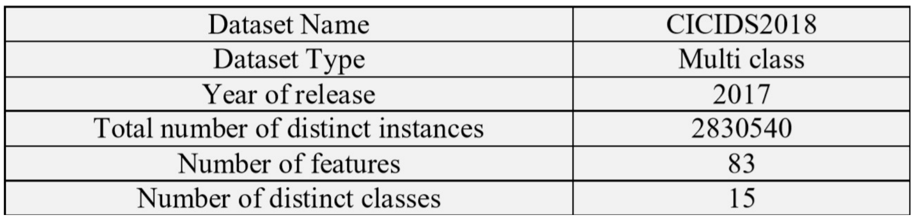


| Name of Files | Day Activity | Attacks Found |
|---|---|---|
| Monday-WorkingHours.pcap_ISCX.csv | Monday | Benign (Normal human activities) |
| Tuesday-WorkingHours.pcap_ISCX.csv | Tuesday | Benign, FTP-Patator, SSH-Patator |
| Wednesday-workingHours.pcap_ISCX.csv | Wednesday | Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed |
| Thursday-WorkingHours-Morning-WebAttacks.pcap_ ISCX.csv | Thursday | Benign, Web Attack – Brute Force, Web Attack – Sql Injection, Web Attack – XSS |
| Thursday-WorkingHours-Afternoon-Infilteration.pcap_ ISCX.csv | Thursday | Benign, Infiltration |
| Friday-WorkingHours-Morning.pcap_ISCX.csv | Friday | Benign, Bot |
| Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv | Friday | Benign, PortScan |
| Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv | Friday | Benign, DDoS |

Figure 2: Data files of CICIDS2017 dataset. [21]

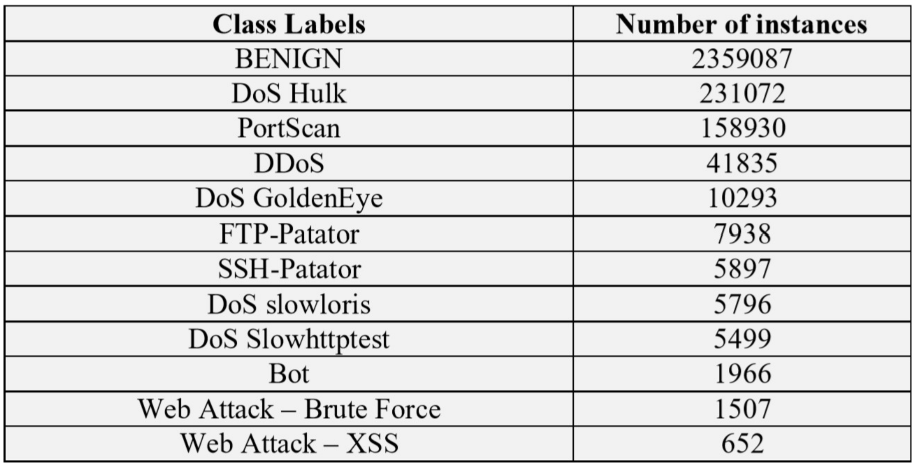
### 3.2.1 Advantages of CIC dataset

The official website and the research by Panigrahi [21] explain that defining new attacks along with the classification of information, is one of the significant advantages. This dataset is a combination of multiple files with fifteen class labels(14 attacks and one benign), 83 features and 3119345 instances. This dataset needs pre-processing. Data characteristics and data classes are represented in xx and yy, respectively.



| | |
|---|---|
| Dataset Name | CICIDS2018 |
| Dataset Type | Multi class |
| Year of release | 2017 |
| Total number of distinct instances | 2830540 |
| Number of features | 83 |
| Number of distinct classes | 15 |

Figure 3: Data characteristics of CICIDS2017 dataset. [21]



| Class Labels | Number of instances |
|---|---|
| BENIGN | 2359087 |
| DoS Hulk | 231072 |
| PortScan | 158930 |
| DDoS | 41835 |
| DoS GoldenEye | 10293 |
| FTP-Patator | 7938 |
| SSH-Patator | 5897 |
| DoS slowloris | 5796 |
| DoS Slowhttptest | 5499 |
| Bot | 1966 |
| Web Attack – Brute Force | 1507 |
| Web Attack – XSS | 652 |

Figure 4: Data classes of CICIDS2017 dataset. [21]

### 3.2.2 Shortcomings with the CICIDS dataset

Scattered data: The huge data is distributed among eight files and we should filter the data to pick the right right dataset with right attacks.
Huge volume: Huge volume of data has both advantages and disadvantages. The main disadvantage is data processing. It requires more resources and high amount of time.
Missing Values: There are a total of 288602 instances with missing information. this data should be excluded before processing. [21]
Class imbalance: We should be careful when considering huge data, if data has one pattern as majority , the model will bias towards the majority class and will be error prone. 6 & 5.

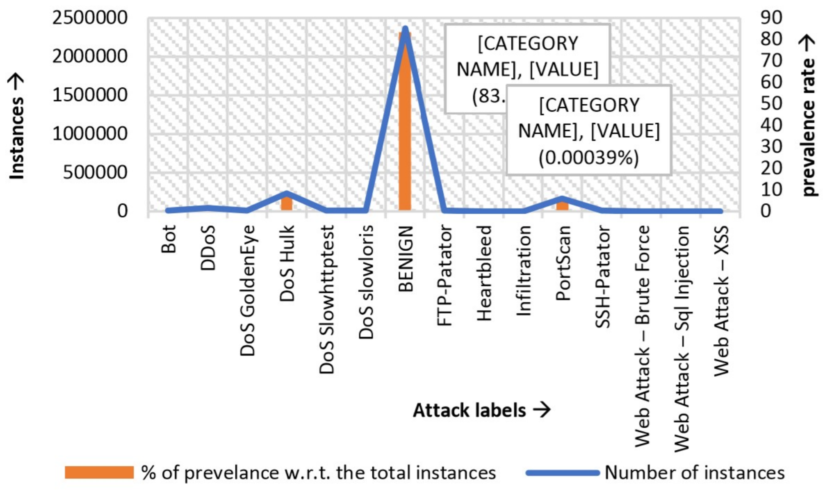
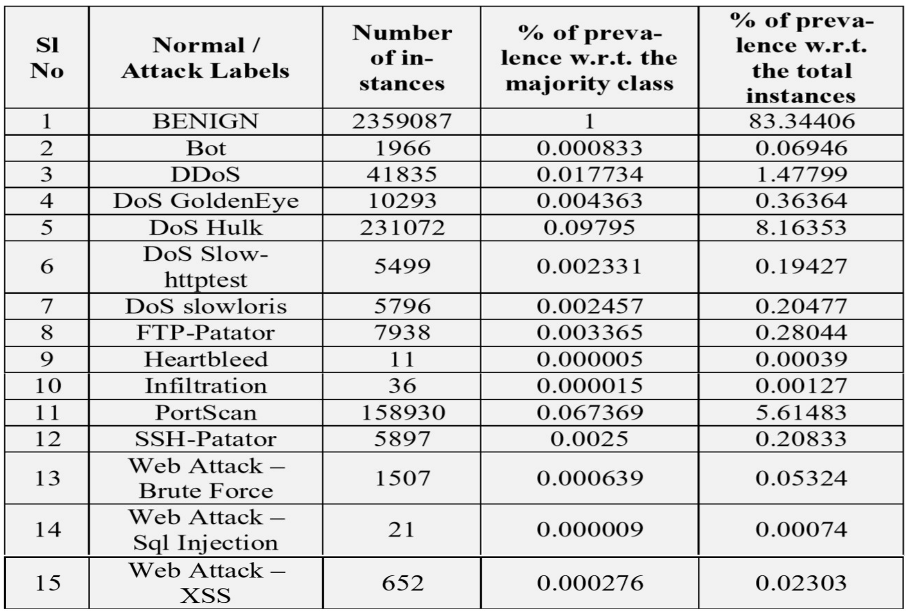


Figure 5: class prevalence graph. [21]



| Sl No | Normal / Attack Labels | Number of instances | % of prevalence w.r.t. the majority class | % of prevalence w.r.t. the total instances |
|---|---|---|---|---|
| 1 | BENIGN | 2359087 | 1 | 83.34406 |
| 2 | Bot | 1966 | 0.000833 | 0.06946 |
| 3 | DDoS | 41835 | 0.017734 | 1.47799 |
| 4 | DoS GoldenEye | 10293 | 0.004363 | 0.36364 |
| 5 | DoS Hulk | 231072 | 0.09795 | 8.16353 |
| 6 | DoS Slow-httptest | 5499 | 0.002331 | 0.19427 |
| 7 | DoS slowloris | 5796 | 0.002457 | 0.20477 |
| 8 | FTP-Patator | 7938 | 0.003365 | 0.28044 |
| 9 | Heartbleed | 11 | 0.000005 | 0.00039 |
| 10 | Infiltration | 36 | 0.000015 | 0.00127 |
| 11 | PortScan | 158930 | 0.067369 | 5.61483 |
| 12 | SSH-Patator | 5897 | 0.0025 | 0.20833 |
| 13 | Web Attack – Brute Force | 1507 | 0.000639 | 0.05324 |
| 14 | Web Attack – Sql Injection | 21 | 0.000009 | 0.00074 |
| 15 | Web Attack – XSS | 652 | 0.000276 | 0.02303 |

Figure 6: table of class prevalence results. [21]

## 3.3 Training Algorithms

### 3.3.1 LSTM anomaly detection process

The traditional (RNN) networks are generally capable of learning high complex patterns. In many tasks RRN's are proven successful. For eg: Speech recognition and text generation. On long temporal sequence, it is difficult for RNN's to learn and train. This is generally due to the exploding and vanishing gradient problem that propagates through various layers of RNN. This results the network not to learn in an effective way. The hidden vector sequence $h = (h_1, h_2, ..., h_t)$ is computed by RNN through iteration of the equations from t = 1 to T to compute the output vector $y = (y_1, y_2, ..., y_t)$.

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = W_{hy}h_t + b_y$$

Figure 7: [22]

Here the b denoting bias vectors, W denotes weight matrix and H denotes the recurrent hidden layer function. The above limitation contains "memory cells" which is solved by LSTM that allow the network to learn and understand when to forget and leave the previous memory states or when should the hidden states updated when new information is given. This allows the network to be good at exploiting and finding long range context. The memory blocks that store temporal state of the network are memory cells that additionally multiplicative units called gates that are used to control the flow of information. In each and every block, there is an output as well as input and forget gate. The output gate controls and limit the output flow of cell activation into rest of the network where the input gate controls and limits the flow of input activation into the memory cell. The forget gate variates the self-recurrent connections of the cells. This was designed and created to allow the cell to reset or remember its previous state depending on what is needed. In the memory block, peephole connections are found that connects the internal cells with the gates in same cell to learn the precise timing output.

With LSTM, we can calculate function H with the below equations:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$
$$h_t = o_t \tanh(c_t)$$

Figure 8: [22]

On addition, modelling of complex temporal sequences are stacked by multiple layers of LSTM. The lower level LSTM layer output is the input to upper LSTM layer. The i/p layer will evaluate through the fully connected layer above it. This is through a feed forward network. Here, we used 3 LSTM stacks with 100 hidden units each and 100 seconds for input sequences of data. The mean-square loss function is the loss function used.

### 3.3.2 Neural networks and deep learning

To mimic in a way that a human brain deals with the problems, neural networks are used. To infer and learn relationships on the observed data, layers of interconnected units are used. A neural network consists of various connected layers. The neural networks are called Deep Learning, when neural network consists of more than 1 hidden layer. The neural networks are easily adjustable and can learn various data changes. When data is unstructured or unlabeled , Neural networks are used. The computer vision is one of the key used for neural networks.

In a variety of applications, Deep learning is being leveraged today. To help the vehicles understand the whole environment around a car, deep learning concepts are used. When the camera's capture the surrounding environment images, the unstructured data is interpreted by deep learning algorithms to help the system make real-time decisions. In this way, applications that are used by radiologists to interpret medical images, deep learning is used [23].

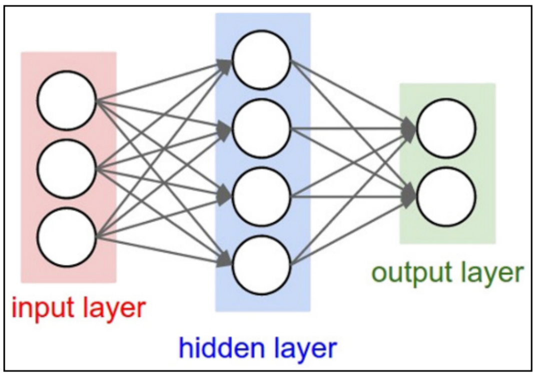


Figure 9: Architecture of neural networks [23]

## 4 Design Specification

### 4.1 Process flow diagram

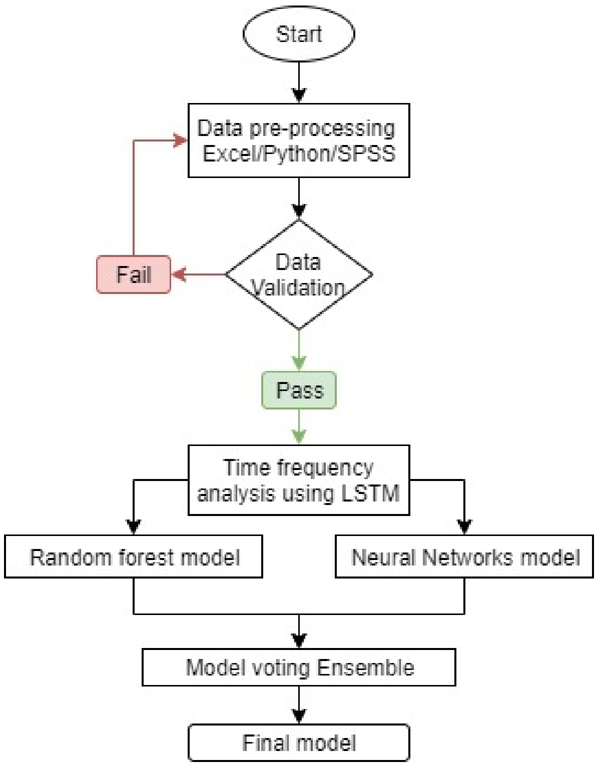


Figure 10: Architectural design flow of implementation

### 4.1.1 Random forest - Why?

Usually, if we consider decision trees which are good classifiers may lead to overfitting as they highly memorize the training data and predict the test data that matches very closely to training values. This type of overfitting can be reduced by using Random forests.

Due to no limit in the flexibility for Decision trees, these algorithms are highly prone to overfitting. On the contrary, if we want to keep a limitation on the flexibility, it may result in biased values.

To overcome biased values and also overfitting issue, we can lean on random forests which creates an individual ensemble model using a combination of decision trees. The process involved in random forests is they create hundreds of decision trees by training each tree with a different split of training observations, and the resultant predicted values would be the average/voting value of all the predictions.

### 4.1.2 Neural networks - Why?

The basic idea for considering Neural Networks in our model building is to combine input information in a sophisticated & flexible neural network model. The neural network model will tweak coefficients continually in an interactive process.

The network's interim performance in classification or prediction informs successive tweaks. Neural network structure has multiple layers (Input layers, Hidden layers and output layer), Nodes, Weights (Coefficients) and Bias values (Constant term).

In a neural network, the output of the input layer acts as the input for hidden layers. We can adjust our hidden layer nodes, and each node receives input from all the input nodes. The output of each hidden node is a function of the weighted sum of inputs.

The advantage of using neural networks is they calculate the difference for the predicted and the actual value. This error is propagated back and distributed to all the hidden nodes and used to update their weights.

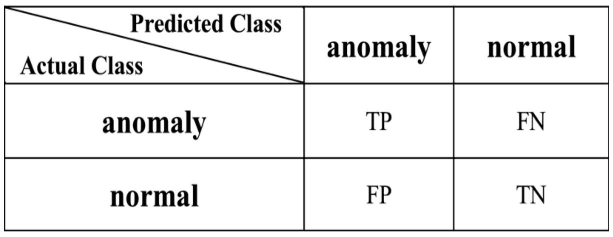
There is a chance of overfitting data with neural networks. To avoid this, we can track error in test data, limit iterations and limit the complexity of the network.

## 4.2 Evaluation Metrics

Accuracy is the key indicator to decide the efficiency of the model. In addition to this indicator, there are also few measures which should be considered; they are

- detection rate (DR)
- Error rate (ERR)
- Sensitivity or Recall or True positive rate(TPR)
- Specificity or True negative rate (TNR)
- Precision (Positive predictive value)
- False positive rate (FPR)
- F-score/measure

Figure 11 shows the definition of confusion matrix.



| Predicted Class / Actual Class | anomaly | normal |
|---|---|---|
| anomaly | TP | FN |
| normal | FP | TN |

Figure 11: Metrics confusion matrix.

Accuracy: correct classification of the records with the model generated. Accuracy is calculated with the equation shown below. This can also be defined as 1 - Error rate(ERR)

$$AC = \frac{TP + TN}{TP + TN + FP + FN}$$

Mis-classification rate (error):
Error rate = percentage of wrongly classified records within the total records.

$$ERR = \frac{(FP + FN)}{n}$$

True Positive Rate (TPR): The percentage of positives with in the correctly identified records.this can also be called as Detection Rate (DR), as shown below. This is also called as Sensitivity.

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR): the total percentage of the wrongly identified records with in all negative records as shown below. This can also be calculated as 1 - Sensitivity and called as Specificity.

$$FPR = \frac{FP}{FP + TN}$$

Precision(Positive predictive value):

$$PPR = \frac{TN}{(TN + FP)}$$

F-Score: It is calculated by taking the harmonic mean of the precision and the recall.

$$F - measure = 2\frac{precisionrecall}{precision + recall}$$

**The Idea behind hybrid IDS is to achieve peak of accuracy curve and detection rate with a minimum FPR.**

## 4.3 ROC - ROC curve analysis

The ROC curve is plotted based on the values generated by the model. the ROC model tuning helps using identifying the best model the above metrics.

## 4.4 AUC - Area under the curve

The AUC is the volume of records that are classified with the model.The higher the value reaches the max limit 100, will result in the best predicted value.

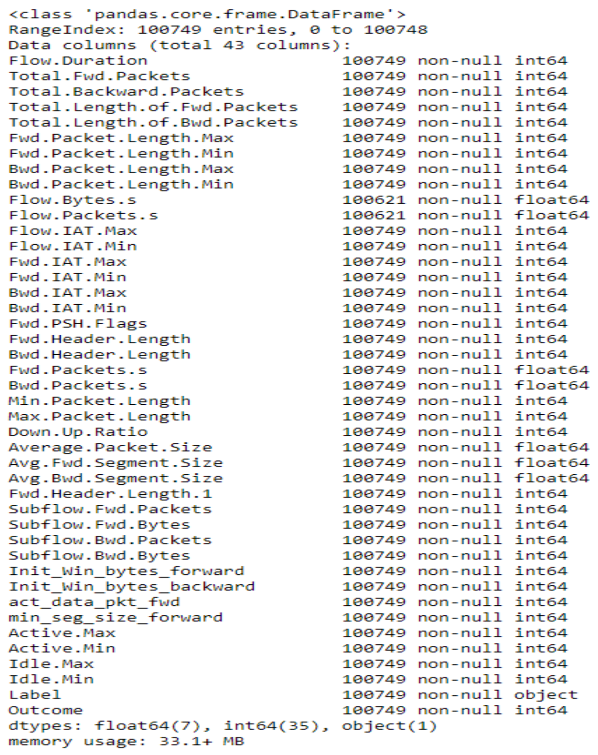
## 4.5 Design tools

I have used Python for building and implementing our algorithms, as Python is easy for implementation and handles hundreds of deep learning and machine learning techniques very quickly. I used Jupyter notebook for step by step implementation of our code. As our data has a binary outcome, we used two classifiers for building our model and then proceeded with Ensembling technique for finalizing our final best model.

1. Recurrent neural network - LSTM
2. Random forests
3. Neural networks
4. Model voting ensemble

## 4.6 Data Variables

Below is the list of variables that are present in our dataset. There is a total of 100748 records current and a total of 43 variables, including our outcome variable. Variables listed in figure 12.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100749 entries, 0 to 100748
Data columns (total 43 columns):
Flow.Duration                   100749 non-null int64
Total.Fwd.Packets               100749 non-null int64
Total.Backward.Packets          100749 non-null int64
Total.Length.of.Fwd.Packets     100749 non-null int64
Total.Length.of.Bwd.Packets     100749 non-null int64
Fwd.Packet.Length.Max           100749 non-null int64
Fwd.Packet.Length.Min           100749 non-null int64
Bwd.Packet.Length.Max           100749 non-null int64
Bwd.Packet.Length.Min           100749 non-null int64
Flow.Bytes.s                    100621 non-null float64
Flow.Packets.s                  100621 non-null float64
Flow.IAT.Max                    100749 non-null int64
Flow.IAT.Min                    100749 non-null int64
Fwd.IAT.Max                     100749 non-null int64
Fwd.IAT.Min                     100749 non-null int64
Bwd.IAT.Max                     100749 non-null int64
Bwd.IAT.Min                     100749 non-null int64
Fwd.PSH.Flags                   100749 non-null int64
Fwd.Header.Length               100749 non-null int64
Bwd.Header.Length               100749 non-null int64
Fwd.Packets.s                   100749 non-null float64
Bwd.Packets.s                   100749 non-null float64
Min.Packet.Length               100749 non-null int64
Max.Packet.Length               100749 non-null int64
Down.Up.Ratio                   100749 non-null int64
Average.Packet.Size             100749 non-null float64
Avg.Fwd.Segment.Size            100749 non-null float64
Avg.Bwd.Segment.Size            100749 non-null float64
Fwd.Header.Length.1             100749 non-null int64
Subflow.Fwd.Packets             100749 non-null int64
Subflow.Fwd.Bytes               100749 non-null int64
Subflow.Bwd.Packets             100749 non-null int64
Subflow.Bwd.Bytes               100749 non-null int64
Init_Win_bytes_forward          100749 non-null int64
Init_Win_bytes_backward         100749 non-null int64
act_data_pkt_fwd                100749 non-null int64
min_seg_size_forward            100749 non-null int64
Active.Max                      100749 non-null int64
Active.Min                      100749 non-null int64
Idle.Max                        100749 non-null int64
Idle.Min                        100749 non-null int64
Label                           100749 non-null object
Outcome                         100749 non-null int64
dtypes: float64(7), int64(35), object(1)
memory usage: 33.1+ MB
```

Figure 12

# 5 Implementation

## 5.1 Data Cleansing & Pre-processing

As part of data cleansing, we observed there are lot values which are NAN values. We replaced all those NAN values with zeros. We also dropped columns which have infinity values (if any) in it .

## 5.2 Variable Selection

Generally, an increased number of variables for building any model would increase the overfitting issue and cause complexity while building and interpreting the model results. To avoid this situation, we cleansed the data and prepared it before passing it through machine learning algorithms.

### 5.2.1 Removing highly correlated variables:

As part of data exploration, we identified some independent variables that are highly correlated to each other. This high correlation will cause multi collinearity issue, and this will cause difficulty in fitting and building a regression model. Using domain knowledge and also by checking the variance inflation between the variables, we dropped some of them before proceeding for model building.

### 5.2.2 Removing unnecessary variables

Variables that have the same value repeated throughout the dataset do not impact the outcome. There is no point in adding these variables into our model other than increasing complexity. We omitted these variables to reduce unnecessary impact on the outcome.

We also removed variables that have no relation with our outcome variable. We identified FLAG variables which has neither much impact on the outcome nor help us drawing better data insights. Deleted these variables as part of data cleansing.

## 5.3 Data partitioning

We are dealing with supervised data, as we have a defined outcome variable that needs to be predicted using classifier algorithms. The next step involved in supervised data is to partition the data. We divided the entire data into training and test with 75:25 split. We considered 75% of the data as training data and the remaining 25% as test data.

## 5.4 Implementation of LSTM

### 5.4.1 Model Interpretation

We used LSTM layers with in the recurrent neural network. As discussed earlier LSTM has less vanishing and exploding effect compared to RNN. This helps in remembering values for longer and shorter periods. I tried multiple variations of LSTM implementations and captured the values shown in table below.

| Model Parameters | RMS | Mean Abs | RMS | Mean Abs | Time |
|---|---|---|---|---|---|
| Batch size 32<br>No of neurons 32<br>time steps 30<br>epoch 1 | 0.48 | 0.4526 | 0.44 | 0.3619 | 683s |
| Batch size 32<br>No of neurons 32<br>time steps 1<br>epoch 1 | 0.48 | 0.4561 | 0.49 | 0.4672 | 959s |
| Batch size 1<br>No of neurons 32<br>time steps 1<br>epoch 1 | 0.48 | 0.4516 | 0.49 | 0.4671 | 936s |

From the above table we can that more we train the model the more accurate we can get the results. the loss and var_loss values are also decreasing with more iterations of learning.

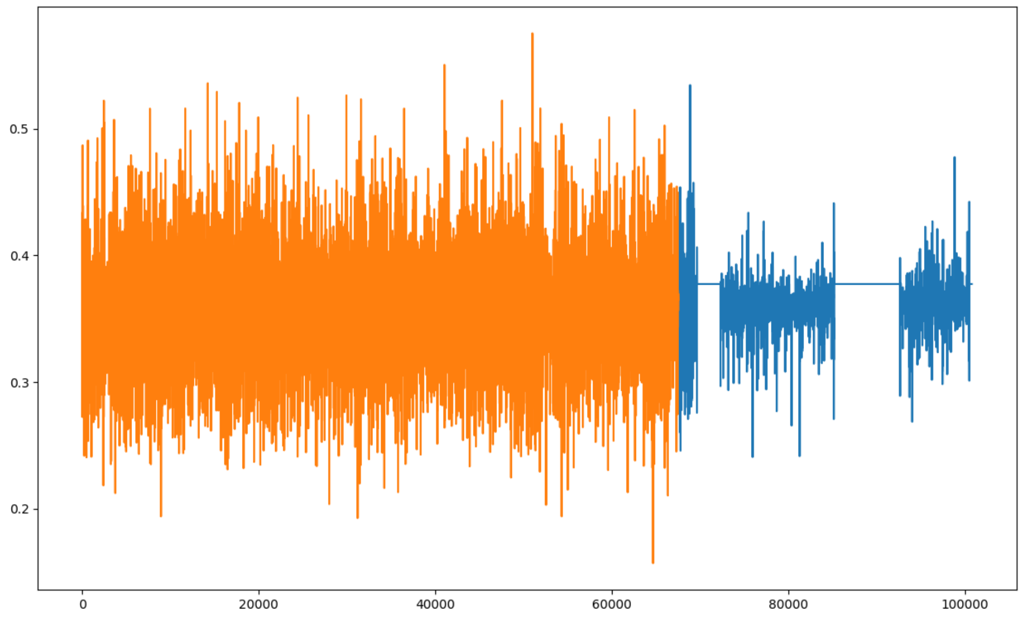


Figure 13: Learning & detection of attacks

This graph shows the training data and test prediction of the data. The orange represents the train data and the blue peaks represents attacks and blue lines represents benign data. Using this representation we can clearly pick the data from the timeline represented in green because of its efficiency and prediction ability

The filtered data is further fed into the hybrid model to get the exact records which are not benign.

## 5.5 Implementing Random Forest Algorithm

### 5.5.1 Model interpretation

We used random forest classifier in python with 'random state=0' and criterion as 'gini' which helps in measuring homogeneity contributed by each variable nodes and leaves. The target variable has (0, 1) as an outcome, and the first index shows us the probability for the data being 0 and the next one refers to 1.

### 5.5.2 confusion matrix

| | Anomaly(1) | Benign(0) |
|---|---|---|
| Anomaly(1) | 15231 | 638 |
| Benign(0) | 582 | 8737 |

## 5.6 Neural network

### 5.6.1 Model Interpretation

We also built a Neural Network classifier model for our data. For this data set, we created three hidden layers by using 'Dense' function. Also, for building a powerful NN model, we used ‘adam’ one of the powerful Stochastic Gradient Descent (SGD) and compiled the entire neural network model using logarithmic loss function known as ‘binary_crossentropy’ (works for binary outcome variable with two categories)

### 5.6.2 Confusion matrix

| | Anomaly(1) | Benign(0) |
|---|---|---|
| Anomaly(1) | 11684 | 4185 |
| Benign(0) | 73 | 9246 |

## 5.7 Implementation of Ensemble Model

### 5.7.1 Model Interpretation

We implemented Ensemble technique as an ensemble of methods help us predict more accurately. This technique helps us in reducing variance in predictions. Generally, the ensemble model facilitates parallel processing and performs better than individual models. In an ensemble approach, overfitting issue will be mitigated, and multiple machine learning techniques are used initially, and their classification/prediction values will be tabulated. This will result in reducing bias, variance and improvise predictions. There are three major methods in Ensembling – Average, Maximum and Voting (used only for classification models). For our Ensemble model, we used a model voting technique from sklearn.ensemble class.

### 5.7.2 Why Ensemble with Voting Classifier?

In general, voting classifiers exhibit higher accuracy than individual classifiers. We used voting classifier technique and specifically we tried the “hard voting’ process rather than ’Soft voting’. This hard voting works by calculating the majority of the voting for accuracy. Using this method, we were able to get an accuracy of 95%, which is very good. Our ensembling model perfectly fit our data.

| | Anomaly(1) | Benign(0) |
|---|---|---|
| Anomaly(1) | 15231 | 638 |
| Benign(0) | 582 | 8737 |

## 5.8 Discussion

Using all confusion matrix, we can evaluate the accuracy of the developed classification models. Using metrics.accuracy_score, We were able to know the accuracy percentage for test data among all models.

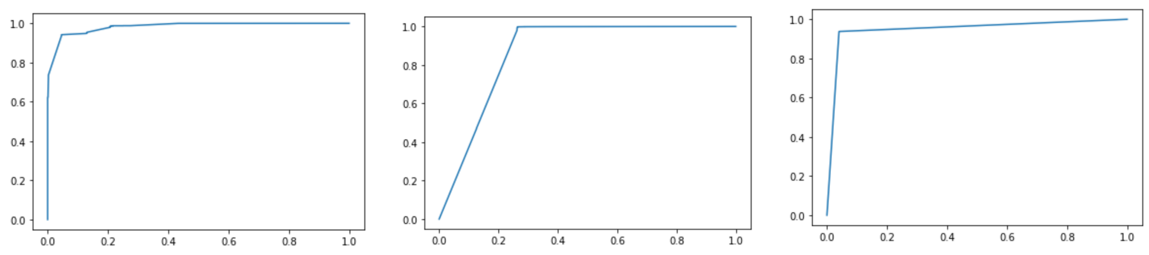
### 5.8.1 Calculated metrics of all models

| Metrics | RandomForest | NeuralNetwork | Ensemble |
|---|---|---|---|
| Accuracy (ACC) | 95.156 | 83.095 | 95.156 |
| Error rate (ERR) | 4.843 | 16.904 | 4.843 |
| Recall/Sensitivity (TPR) | 96.319 | 99.379 | 96.319 |
| Specificity (FPR) | 6.805 | 31.159 | 6.805 |
| Precision (PPV) | 95.979 | 73.627 | 95.979 |
| F-Score | 96.149 | 84.586 | 96.149 |

From the high precision score, we can see that the values are close to 1. The more the values closer to 1, the upper the ability of the classifier to predict or label a negative or positive value as the same. Lesser precision values lead to an inaccurate result

### 5.8.2 ROC analysis

ROC curve is important in identifying the behavior of any model. From the below plots, we can see that AUC has occupied a decent area in our ROC plots. But the small variation is considered by ensemble to improve the model.



(a) Random forest (b) Neural Network (c) Ensemble

Figure 14: ROC of three models

Random forest ROC clearly shows that if we capture 10% error in our prediction, we would be able to achieve 95% of the True Positive values. From the ROC of Neural network, we can capture 20% of the False Positive rate, we would be to capture 70% of the True Positives.

### 5.8.3 Area under the curve

| Metrics | Random Forest | Neural Network | Ensemble |
|---|---|---|---|
| Area under the curve (AUC) | 0.98300 | 0.85886 | 0.98352 |

The AUC value is the percentage of right predictions out of all records. If the value is one, then the model is predicting 100% correct values. From the above table, the ensemble has the highest value, which is reaching 1. So, we can consider the ensemble as the best model among the three models.

## 6 Conclusion and Future Work

After looking at the accuracy, ROC chart and AUC, we clearly observe that ensembling technique chose Random forests as the best model for achieving highest accuracy. Neural network model is less sensitive to changes on the training data. Hence, we can call NN models as stable learners. Adding to this, Random forests has inbuilt ensembling techniques which makes them highly efficient in providing us better and accurate results. We can also use GRU-Gated recurrent unit instead of LSTM. This will use two gates instead of one in LSTM. Also we can add more algorithms to the hybrid model to get high accuracy if needed. All this can be automated and can be taken as future work for better efficiency and accuracy.

## References

[1] S. Axelsson, *Intrusion detection systems: A survey and taxonomy*, 2000.

[2] D. Denning, "An intrusion-detection model," *An Intrusion-Detection Model*, vol. SE-13, no. 2, pp. 222–232, Feb 1987.

[3] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," in *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1990, pp. 296–304.

[4] C. Dowell and P. Ramstedt, "The computerwatch data reduction too1." Washington, DC.: Proc.. 13th National Computer Security Conference, Oct 1990, pp. 99–108.

[5] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," vol. 8, no. 3, pp. 26–41, May 1994.

[6] Y. Okazaki, I. Sato, and S. Goto, "A new intrusion detection method based on process profiling," in *Proceedings 2002 Symposium on Applications and the Internet (SAINT 2002)*, Jan 2002, pp. 82–90.

[7] Y. Jia, M. Wang, and Y. Wang, "Network intrusion detection algorithm based on deep neural network," vol. 13, no. 1, pp. 48–53, 2019.

[8] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[9] M. Riyad.A and M. S. I. Ahmed, "An ensemble classification approach for intrusion detection," 2013.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, p. 1527–1554, 2006.

[12] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 14–36.

[13] J. Kim and H. Kim, "Applying recurrent neural network to intrusion detection with hessian free optimization," in *Information Security Applications*, H.-w. Kim and D. Choi, Eds. Cham: Springer International Publishing, 2016, pp. 357–369.

[14] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme," in *Soft Computing in Industrial Applications*, A. Gaspar-Cunha, R. Takahashi, G. Schaefer, and L. Costa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 293–303.

[15] M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size rnn based on feature grouping," *Neural Computing and Applications*, vol. 21, no. 6, pp. 1185–1190, Sep 2012. [Online]. Available: https://doi.org/10.1007/s00521-010-0487-0

[16] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," vol. 0, no. 56, Jul 2015. [Online]. Available: https://doaj.org

[17] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579 – 595, 2000, recent Advances in Intrusion Detection Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128600001390

[18] A. Sperotto, R. Sadre, F. van Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection," in *IP Operations and Management*, ser. Lecture Notes in Computer Science, G. Nunzi, C. Scoglio, and X. Li, Eds. Springer, pp. 39–50.

[19] A. C. a. P. Limited, *ECCWS 2017 16th European Conference on Cyber Warfare and Security*. Academic Conferences and publishing limited, google-Books-ID: uFA8DwAAQBAJ.

[20] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization:," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, pp. 108–116. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006639801080116

[21] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," vol. 7, pp. 479–482, Jan 2018.

[22] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, Jan 2017, pp. 140–145.

[23] J. P. Mueller and L. Massaron, *Machine learning for dummies*, 2018.