Intrusion Detection System

and classification of network attacks

On Software Defined Networks

By-
Abhilash Kumar Chaudhary
Akshay Pimpalgaonkar
Gaurav Tyagi
Pooja Gupta
Yash Gupta

Instructor

Dr Mehdi Sookhak

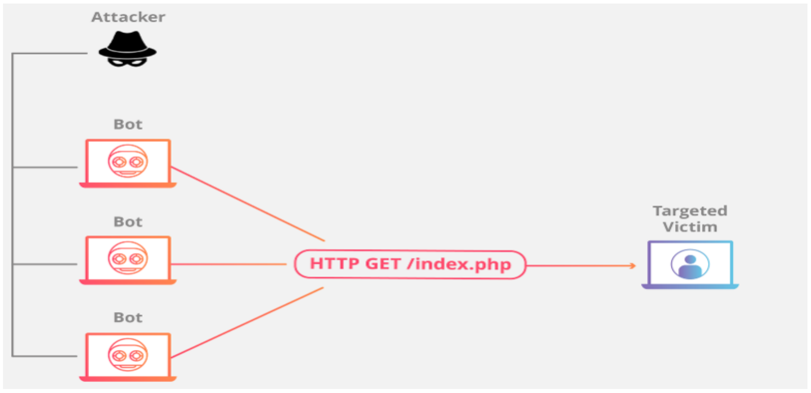
Arizona State University

Table of Content:

Introduction

In this project, we have implemented an Intrusion Detection System to detect the Distributed Denial of Service Attack in a Software Defined network. We have trained multiple learning models such as K-Nearest Neighbour, Logistic regression and Neural Network to distinguish the DDOS attack from normal traffic by extraction network parameters. We then classified the network activity into benign or DDOS attack by observing the different features of the traffic such as flow packet rate, standard deviation and packet entries ratio.

In a Distributed Denial of Service attack, a large number of network traffic is targeted towards a server with the aim to disrupt its services. The attacks disrupt the normal working of the system, sometimes even crash the victim's system. It not only affects the client' time but it leads to monetary loss. Not only this, the reputation of the service provider is also put at stake. In a DDoS attack, a computer system is infected with viruses and converted into 'Bot' by sending multiple HTTP requests from the attacker to the Targeted victim as shown below (Figure 1)[6].

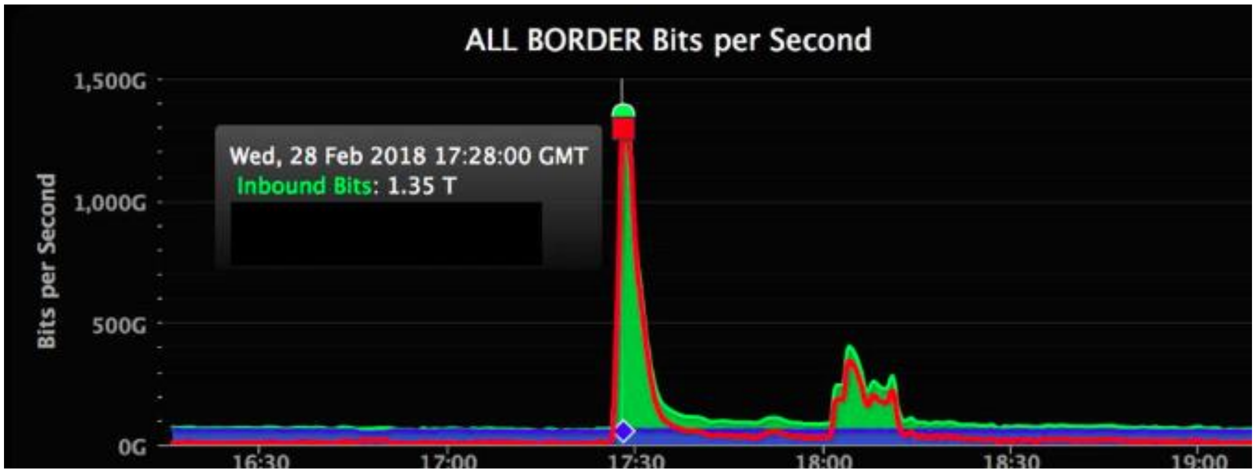


Software Defined Networks are becoming more popular and more accompanying in today's world. SDN employs OpenFlow protocol between the control plane and application plane of the SDN architecture[6]. As per the SDN architecture, most of the SDN based attacks are focussed on SDN Controllers, SDN control-application interface and SDN control data interface. SDN networks are prone to Denial Of Service attacks, especially DDoS. We will be using SDN to detect the attack rather than traditional network comes for the many advantages that SDN brings with it. The biggest of all would be the centralized control over the network i.e., control on the data flow and ability to alter the characteristics of the switching device from the centralized location.

Problem Statement:

According to "What we Know About Friday's Massive East Coast Internet Outage" [1], in 2016, a group of non-authorized users installed botnets on multiple network devices distributed across the US through internet service provider Dyn and drastically altered the network by generating high volume of network towards a server. There are several major websites such as Twitter, GitHub, Spotify, CNN which were affected by this attack which is reported as one of the largest DDOS attack ever seen. This massive internet outage took several hours to be restored.

Another famous DDOS attack happened in Feb 2018, when GitHub was attacked by a high volume of traffic at an approximate rate of 1.35 terabytes per second disrupting the internet service for several minutes(Figure 2)[2].



Another attack is from 2012, when 6 large banks of US system became victim of the DDoS attack. The attack disrupted mobile and online banking functionalities of the bank for an extended period of time.[2][3]

These are just a few recent attacks that greatly affected the working of a server. There are several works in this area to detect and mitigate DDoS attack and the above scenarios defines the severity of effects attacks could have on a system ranging from monetary loss for companies to physical damage and safety threat. As the number of devices connected to the internet increase at a startling rate, so does the vulnerability of the system to be attacked by unauthorized users and hence arises the need of a system which could detect and mitigate this attack.

OBJECTIVE:

After reading through the threats that DDoS attacks has on the economy and the safety of the feature, we planned on creating a system that could be helpful in detecting DDOS attacks in the system by observing various network packet features such as packet flow rate, standard deviation of the flow packets, flow bytes and the ratio of Pair flow entries with an aim to reduce the number of internet criminal activity and the inconvenience caused to the users due to DDoS attack.

We have implemented our Software Defined Network using Mininets. In this project, we have designed an IDS system that would help us to study the network activity and classify it to be benign or a DDoS . We are using ‘Signature based network intrusion detection system’ which would help us identify the new attack on the network using byte sequence in the network or malicious instruction sequence. This type of IDS collects signature i.e. characteristics of an attack to detect the attacks in the future based on the historical signature data collected. A severity is assigned to each packet and if the severity increased beyond a threshold Intrusion Detection System 10 level, an alert is triggered to signify a malicious activity. This type of network technique based on signature is known as Signature based network detection System[4].

For classifying the incoming network traffic, we trained our model on training data using three different machine learning model i.e. K- Nearest Neighbour, Neural Network and Logistic Rgression. After training our data using the above models, we have tested the incoming traffic and calculate the accuracy of the system to detect the type of network activity. The accuracy is calculated by comparing the predicted and actual value of the dependent variable.

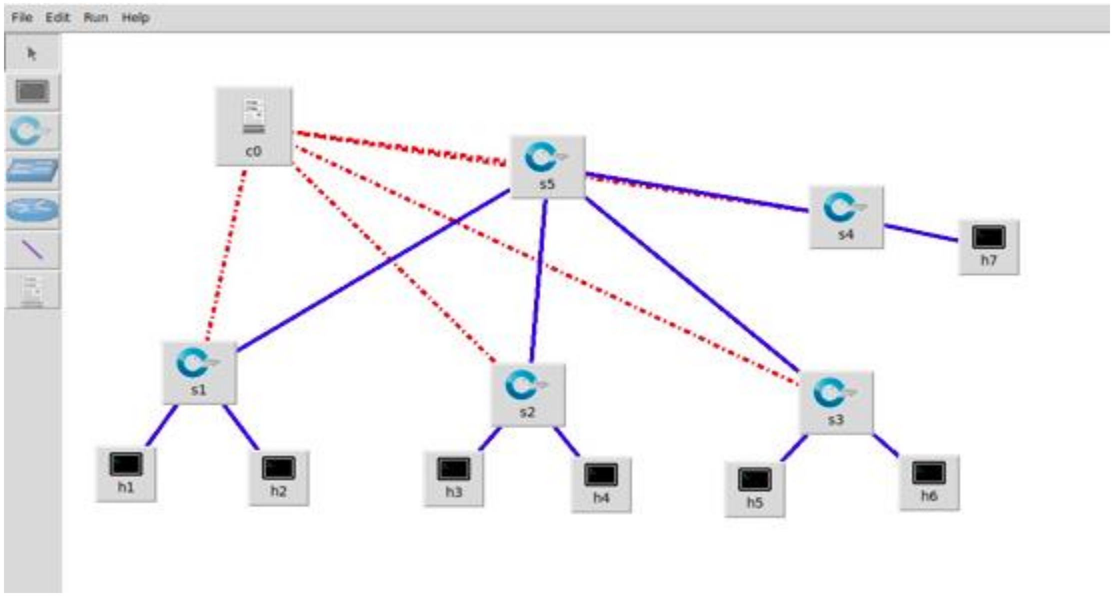
To summarize, we have created an Intrusion detection system with the following objectives:

1. Simulate an SDN network in the mininet.
2. Create the machine learning model using KNN, Neural network and Logistic regression.
3. Created virtual network dataset using mininet to capture normal traffic and attack to classify the network activity.
4. Feature extraction from the dataset by sending packet at different rate.
5. Testing our model with real time data for DDOS attack detection by comparing the parameter extracted from the network.
6. Generating alert message whenever the system is on DDoS attack.

IMPLEMENTATION:

MININET NETWORK:

Below is our network topology we have created a Software defined network using mininet in which we have added one POX controller (C0), 5 switches (s1,s2,s3,s4) and 7 hubs (h1,h2,h3,h4,h5,h6,h7).



C0 : POX controller

S1-S5: Open Flow Switches

H1-h6 : Normal Host

H7 : Target Host

Performing DDOS attack

For performing the attack on our SDN we used hping3, which is a pen-testing tool. The command is as follows -

```
hping3 --faster --rand-source 10.0.0.7
```

The options used in the hping3 command are -

--faster for sending 100 packets per second.

--rand-source for sending packets from random source IPs.

Feature selection

We selected five features for our model learning[11]. They are as follows:

1. Speed of Source IP
2. Standard Deviation of Flow Packets
3. Standard Deviation of Flow Bytes
4. Speed of Flow Entries
5. Ratio of Pair-Flow Entries

Speed of Source IP

It is the rate of Source IP, i.e., number of source IP per unit time. Mathematically, it is defined as:

$$SpSourceIP = TotalIP/T$$

where TotalIP is the total number of Source IPs in sampling time period T.

We chose a time period to be 3 seconds. That means, after every 3 seconds we recorded the number of source IPs in that duration. During the DDoS attack the number of source IPs increases and so does the Speed of Source IP.

Standard Deviation of Flow Packets

It is the standard deviation of number of flow packets in time period T. Mathematically, it is defined as:

$$SDFP = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(packet_i - MeanPackets)^2}$$

where $packet_i$ is the number of packets in i^{th} flow and MeanPackets is the average number of packets in time period T.

In normal traffic, there is a lot of variation in number of packets that pass through a switch in a given time period. But during a DDoS attack, this variation decreases to a large extent and thus, the standard deviation of flow packets decreases.

Standard Deviation of Flow Bytes

It is the standard deviation of number of flow bytes in time period T. Mathematically, it is defined as:

$$SDFB = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(bytes_i - MeanBytes)^2}$$

where $bytes_i$ is the number of bytes in i^{th} flow and MeanBytes is the average number of bytes in time period T.

Like the standard deviation of flow packets, this feature also behaves in the same way. It decreases during a DDoS attack.

Speed of Flow Entries

It is the ratio of number of flow entries and the time period T, the time period in which the number of flow entries were recorded. Mathematically, it is defined as:

$$SFE = N/T$$

where N is the number of flow entries and T is the time period.

Since during a DDoS attack, the flow entries increase compared to the normal traffic. In turn, the speed of flow entries increases.

Ratio of Pair-Flow Entries

It is the ratio of number of flow entries which were interactive and the time period T, the time period in which the number of flow entries were recorded. Mathematically, is defined as:

$$RFIP = NI/T$$

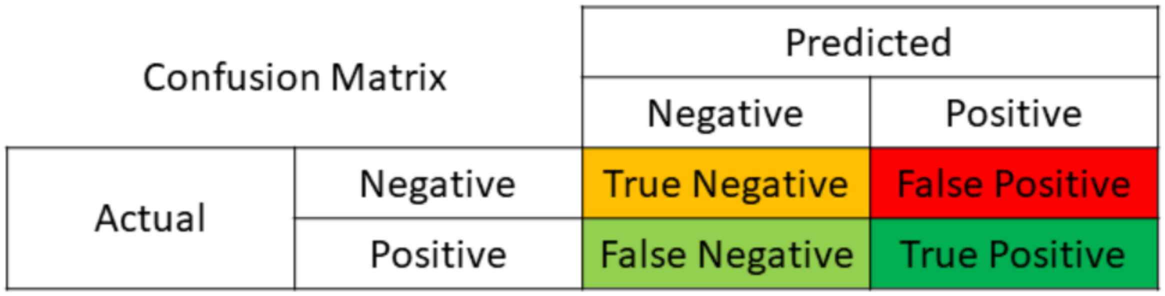
where NI is the number of interactive flow entries and T is the time period in which those flow entries were recorded. During DDoS attack the number of interactions decreases as the server is not able to respond to the flood of requests. Therefore the Ratio of Pair-Flow Entries also decreases.

MACHINE LEARNING MODELS:

For our Intrusion detection system, we have trained our data set using three different models:

1. Artificial Neural Network
2. K-Nearest Neighbour
3. Logistic Regression

Result metrics used: Output of the confusion matrix is used to generate Accuracy and Recall.



Confusion Matrix		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Accuracy: Total correct prediction by the model.

Accuracy = True Positive + True Negative / Total

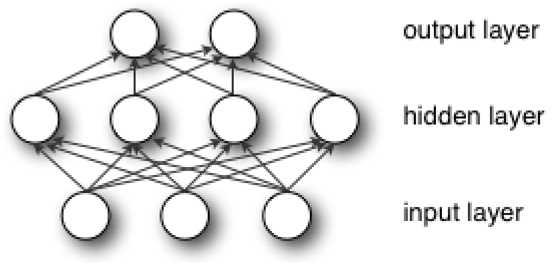
Recall : is the ability of the classifier to find all the positive samples.

Recall = True Positive / (true Positive + False Negative)

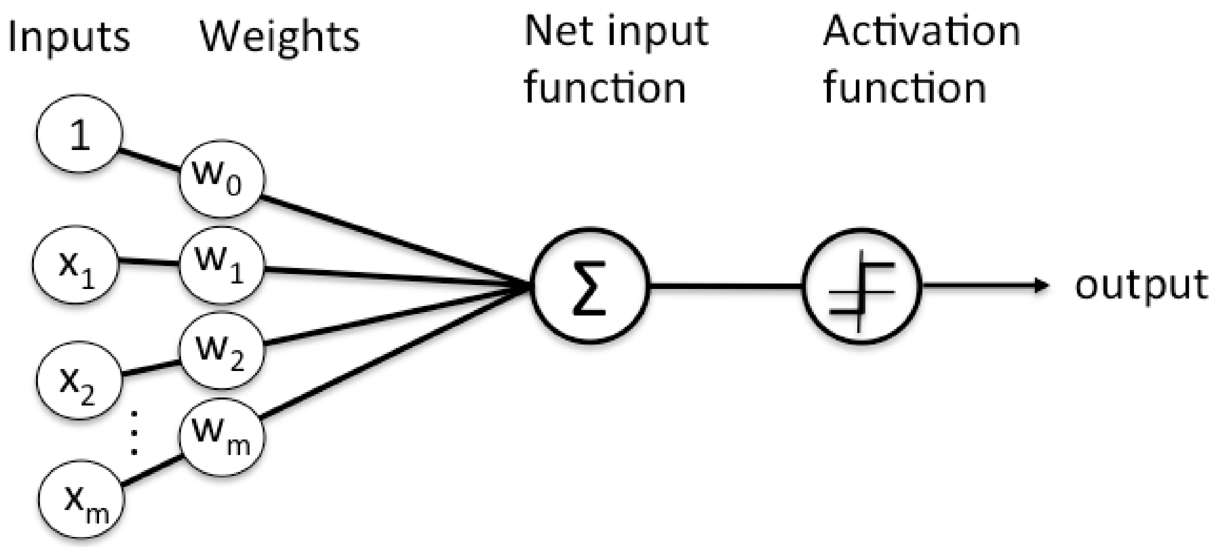
Artificial Neural Network

Artificial neural networks are one of the strongest tools in deep learning and as the name "neural" suggests, they are inspired by the brain and how they work when it comes to learning. Neural networks are a good tool to find the patterns in the recurring data string which can be invisible to traditional algorithms[8].

In a nutshell, neural network is composed of an input layer, several hidden layers and an output layer which work together as a factory line – feed the raw data in input layer, pass down the conveyor belt to extract high level features and then label the output(Figure 3)[9].

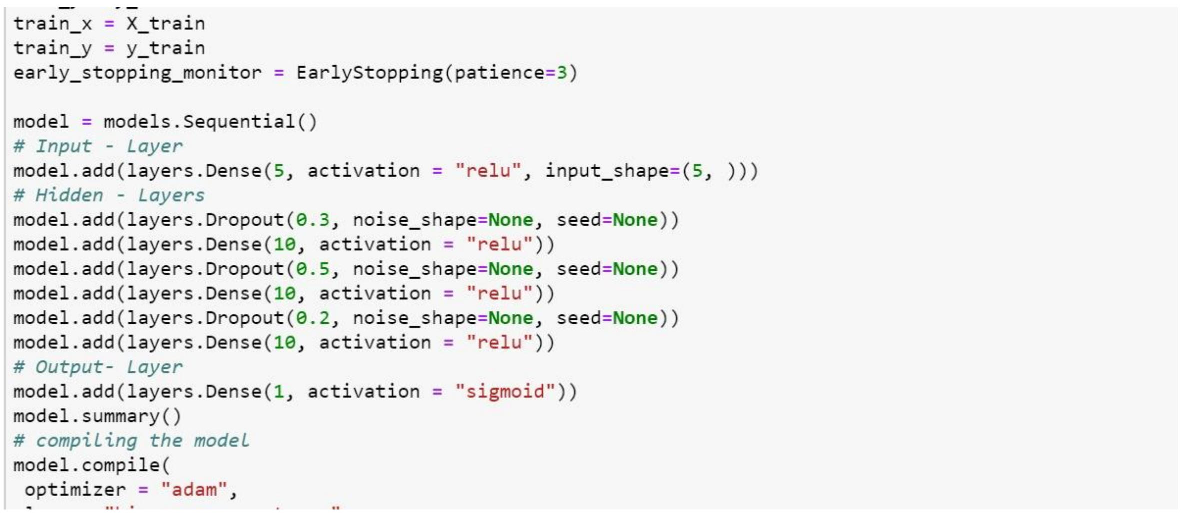


A layer consists of several nodes. A node is a mathematical representation of a perceptron which performs the computation by combining the input with coefficients to amplify or dampen that input with respect to the learning.



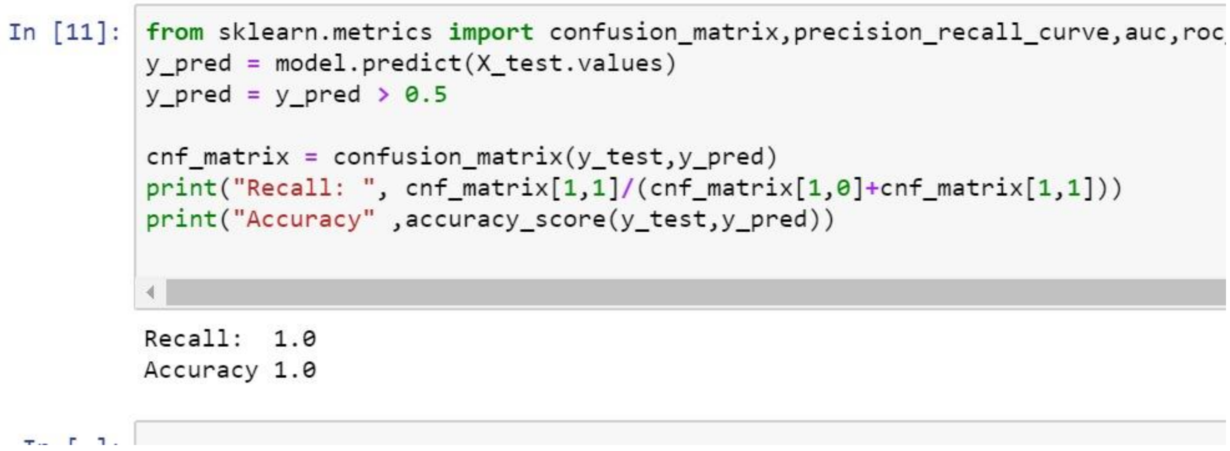
In our model, we have used ReLU (Rectified Linear Unt) Activation function in input layer for faster convergence which avoids the plateau of the slope, hence, linear identity for positive values and zero for negative values[10]. Our Model architecture is as follows:

1. Input layer with ReLU activation
2. Three hidden layers with input size of 10– combination of Dropout and Dense layers to avoid codependency of neurons, and,
3. An output layer, which uses sigmoid Activation function – a non-linear function for better adaptability with the wide variety of data.



```
train_x = X_train
train_y = y_train
early_stopping_monitor = EarlyStopping(patience=3)

model = models.Sequential()
# Input - Layer
model.add(layers.Dense(5, activation = "relu", input_shape=(5, )))
# Hidden - Layers
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(10, activation = "relu"))
model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
model.add(layers.Dense(10, activation = "relu"))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(layers.Dense(10, activation = "relu"))
# Output- Layer
model.add(layers.Dense(1, activation = "sigmoid"))
model.summary()
# compiling the model
model.compile(
 optimizer = "adam",
```

After training the model based on the above architecture with our data, the results showed promise.



```
In [11]: from sklearn.metrics import confusion_matrix,precision_recall_curve,auc,roc
         y_pred = model.predict(X_test.values)
         y_pred = y_pred > 0.5

         cnf_matrix = confusion_matrix(y_test,y_pred)
         print("Recall: ", cnf_matrix[1,1]/(cnf_matrix[1,0]+cnf_matrix[1,1]))
         print("Accuracy" ,accuracy_score(y_test,y_pred))

         Recall:  1.0
         Accuracy 1.0
```

K-Nearest Neighbour

K nearest neighbours is an algorithm which classifies the feature based on the distance from the nearest neighbours. The most popular choice to calculate Euclidean distance which is represented as :

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_n - x'_n)^2}$$

These distances are calculated for the new cases for each label and the new case is assigned the label which has the least sum of distances.

KNN classifier goes through these steps:

1. It calculates the d for each input row (x).
2. It estimates the conditional probability for each class.

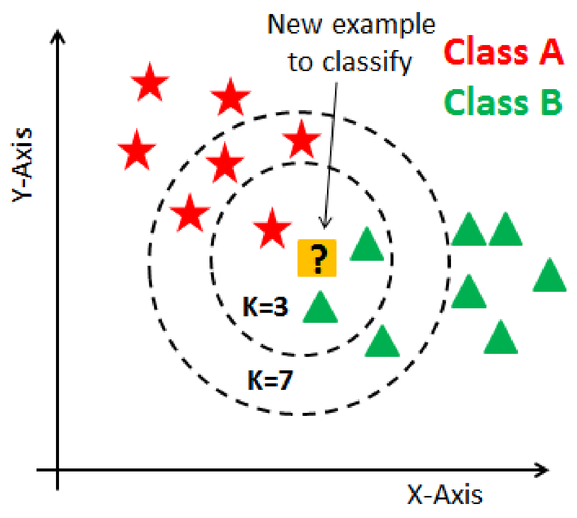
$$P(y = j | X = x) = \frac{1}{K} \sum_{i \in \mathcal{A}} I(y^{(i)} = j)$$

I : is the indicator function which results in 1 if the argument is true otherwise 0.

K : K parameter value

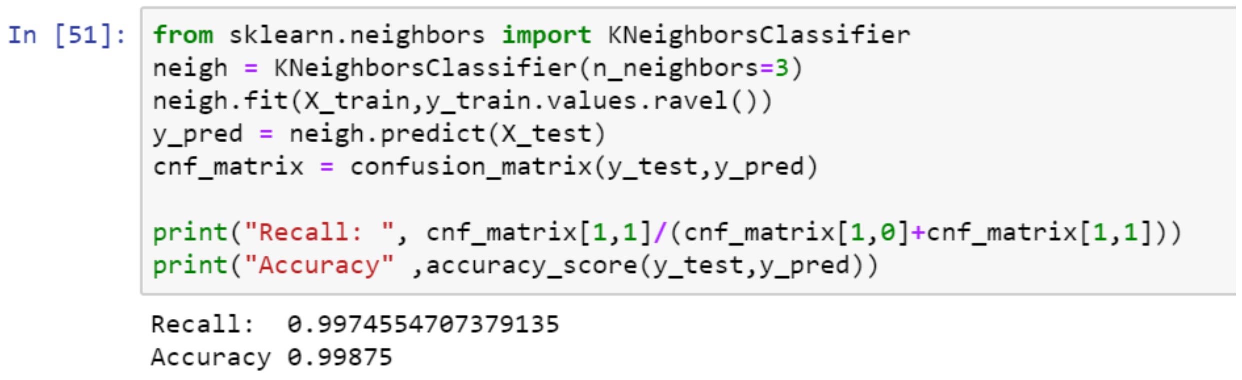
A: set of inputs

The important parameter here is the value of K. K is the number of neighbours from which the distance is calculated. For example, If K=1, only one distance from each class will be calculated and used to predict. However, lower values of K result in low bias and high variance which is equivalent to overfitting the model. Hence the value of K is chosen carefully.



https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Result: We used sklearn KNNclassifier to train out the model using 4000 instances of the dataset. Dataset was divided into 80% training and 20% for testing. Here are the results:



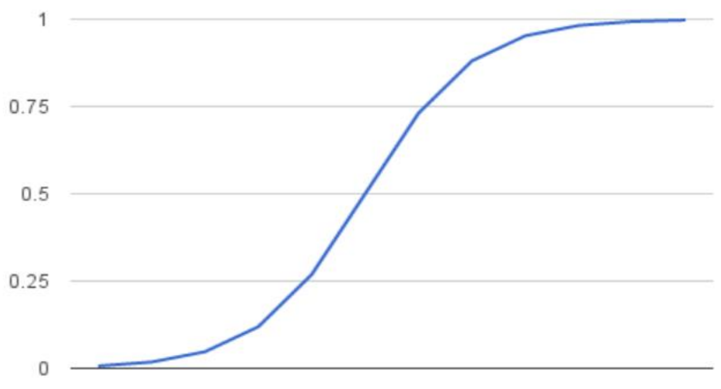
```
In [51]: from sklearn.neighbors import KNeighborsClassifier
         neigh = KNeighborsClassifier(n_neighbors=3)
         neigh.fit(X_train,y_train.values.ravel())
         y_pred = neigh.predict(X_test)
         cnf_matrix = confusion_matrix(y_test,y_pred)

         print("Recall: ", cnf_matrix[1,1]/(cnf_matrix[1,0]+cnf_matrix[1,1]))
         print("Accuracy" ,accuracy_score(y_test,y_pred))

         Recall:  0.9974554707379135
         Accuracy 0.99875
```

Accuracy : 99.875%

Recall : 99.745547%

Logistic Regression

Logistic regression is another algorithm which we used. It is a *go to* algorithm when the value to be predicted is binary (in our case attack or not). It is an extension to linear regression and makes use of the logistic function, also known as the sigmoid function. This function plots an S shaped curve having values between 0 and 1[7].



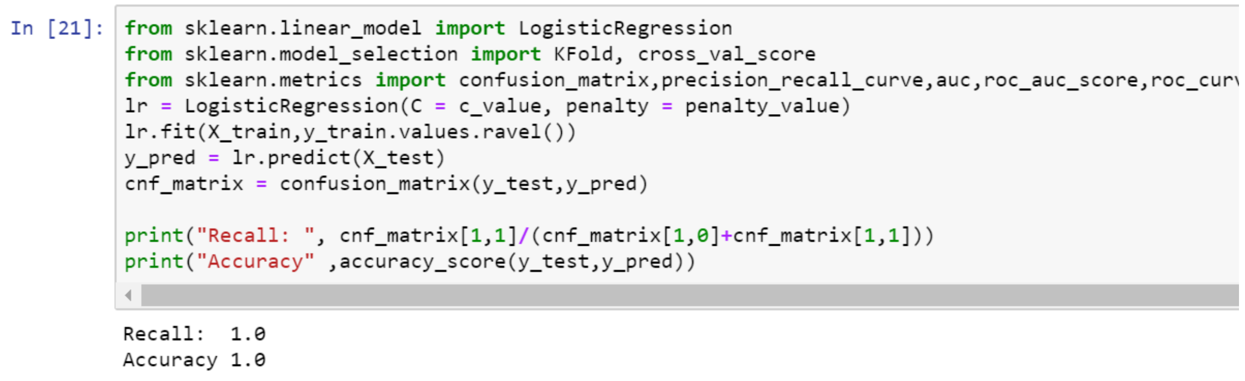
Logistic function

This algorithm basically uses probabilities of the default class, for example, the probability of a request being classified into the attack class. These probabilities are then transformed into binary values. Here is an example of an equation of logistic regression[7]:

$$Y = \exp(b0+b1*x)/(1+\exp(b0+b1*x))$$

The above equation gives us the probability depending on the one feature x. B0 is considered as the intercept or the bias and b1 is the coefficient. Each of the feature value of the dataset will have a coefficient that is derived from the training data. Now as we are interested in classification and not probabilities, these probability values are put into any one of the binary classes. For example classified as attack if P(attack)> 0.5 and as benign if P(attack) < 0.5.

Result: We used sklearn Logistic Regression classifier to train out the model using 4000 instances of the dataset. Dataset was divided into 80% training and 20% for testing. Here are the results:

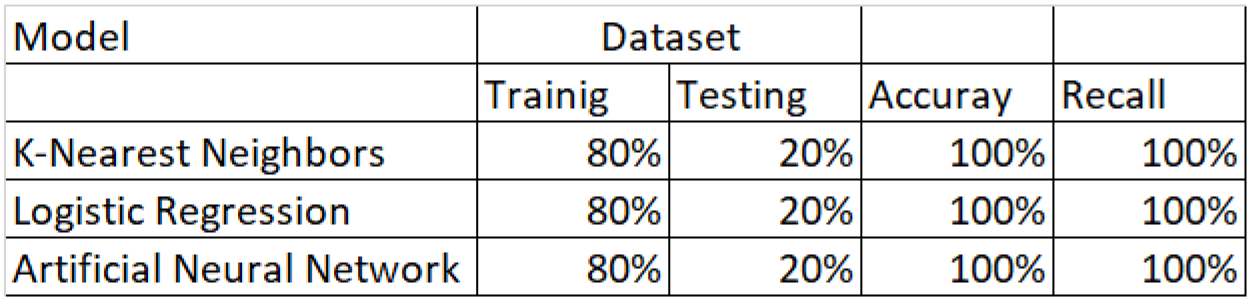


```
In [21]: from sklearn.linear_model import LogisticRegression
         from sklearn.model_selection import KFold, cross_val_score
         from sklearn.metrics import confusion_matrix,precision_recall_curve,auc,roc_auc_score,roc_curv
         lr = LogisticRegression(C = c_value, penalty = penalty_value)
         lr.fit(X_train,y_train.values.ravel())
         y_pred = lr.predict(X_test)
         cnf_matrix = confusion_matrix(y_test,y_pred)

         print("Recall: ", cnf_matrix[1,1]/(cnf_matrix[1,0]+cnf_matrix[1,1]))
         print("Accuracy" ,accuracy_score(y_test,y_pred))

         Recall:  1.0
         Accuracy 1.0
```

Accuracy : 100%

Recall: 100%

Comparison of 3Models:



Model	Dataset			
	Trainig	Testing	Accuray	Recall
K-Nearest Neighbors	80%	20%	100%	100%
Logistic Regression	80%	20%	100%	100%
Artificial Neural Network	80%	20%	100%	100%

CONCLUSION

To conclude, in our project we have successfully created an Intrusion detection System in a Software Defined network which was able to identify the DDoS attack with 99.8 % accuracy in KNN network and 100% accuracy and recall with logistic regression and neural network model.

CHALLENGES:

Limited Dataset: This is the biggest challenge we faced while working on this project. In this project Dataset is generated using the SDN network created using mininet. However, the dataset contains only 4000 instances which is not ideal for Real world application.

Model Training: There are multiple parameters that can be used to fine tune the mode. Since the dataset was collected in a controlled environment, the trained model provide very high accuracy and the model seems to be overfitted. To overcome this issue, dataset needs to be acquired from the real world.

References:

[1]. Newman, L.H. *What We Know About Friday's Massive East Coast Internet Outage*. Available online: *https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/* (accessed on 6 March 2019)

[2]. Ahmad Nassiri (Aug 15, 2018) *5 Most Famous DDoS attacks* Available online : *https://www.a10networks.com/resources/articles/5-most-famous-ddos-attacks*

[3]. Lucian Constantin, *DDoS Attacks Against US Banks Peaked At 60 Gbps* *https://www.cio.com/article/2389721/ddos-attacks-against-us-banks-peaked-at-60-gbps.htm*

[4] Red Hat Enterprise Linux 4. *Chapter 9 Intrusion Detection*. Retrieved from *http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-sg-en-4/s1-ids-net.html*

[5] Central, SDX. *Understanding the SDN Architecture – SDN Control Plane & SDN Data Plane*. Retrieved from *https://www.sdxcentral.com/networking/sdn/definitions/inside-sdn-architecture/*

[6] *Cloudfare* Retrieved from *https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/*

[7] Browniee, Jason. *Logistic Regression for Machine Learning* (April 1, 2016) Retrieved from https://machinelearningmastery.com/logistic-regression-for-machine-learning/

[8] Skymind. *A Beginner's Guide to Neural Networks and Deep Learning* Retrieved from https://skymind.ai/wiki/neural-network

[9] Digital Trends. *What is an artificial neural network? Here's everything you need to know*. Retrieved from https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/

[10] TinyMind. *A Practical Guide to ReLU* Retrieved from https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7

[11] Issac, William. Iye, Suraj *Software Defined Security* (April 2018) Retrieved from *https://www.researchgate.net/publication/324716038_SOFTWARE-DEFINED_SECURITY*