**Lab 9 – Supervised Artificial Neural Network for Intrusion Detection**

Dr. Sana Ullah Jan

* **Objective:** Implement and analyse performance of Supervised Artificial Neural Network (ANN) for intrusion detection.
* **Dataset:** A subset of CIC-IDS2017 dataset will be provided to students (available on <https://www.unb.ca/cic/datasets/ids-2017.html> [Accessed on 27/10/2021]).
* **Exercise:** Vary the input parameters of ANN to see the effect on its performance.
* **Platform:** Google Colab (<https://colab.research.google.com>).

**Introduction**

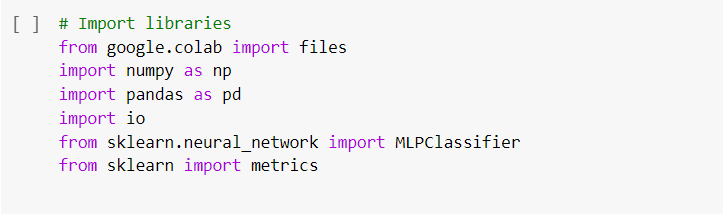
This lab will present implementation and performance analysis of Artificial Neural Networks (ANN) used for classification and identification of intrusion. We will briefly cover simple implementations of ANN with Python as well as convenient ways of validating them using tuning functions.

**Artificial Neural Networks**

Completely covering neural networks would take us a complete term (and we would still be missing a lot!). This lab will only present a simple way of building a neural network-based classifier using the *sklearn.neural\_network.MLPClassifier* library. We will use a small dataset of CIC-IDS2017 dataset for this problem.

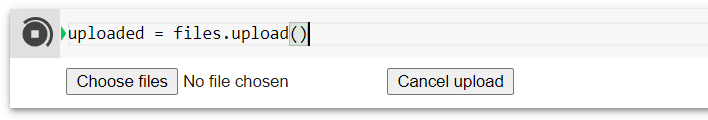
**Importing Necessary Libraries**

The necessary libraries are imported to the Python environment as follows



**Uploading data**

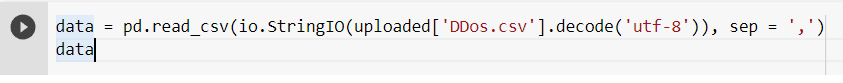
To upload data to the Google Colab environment, files.upload() function is used. Running it will generate an option to choose the files that you want to upload.



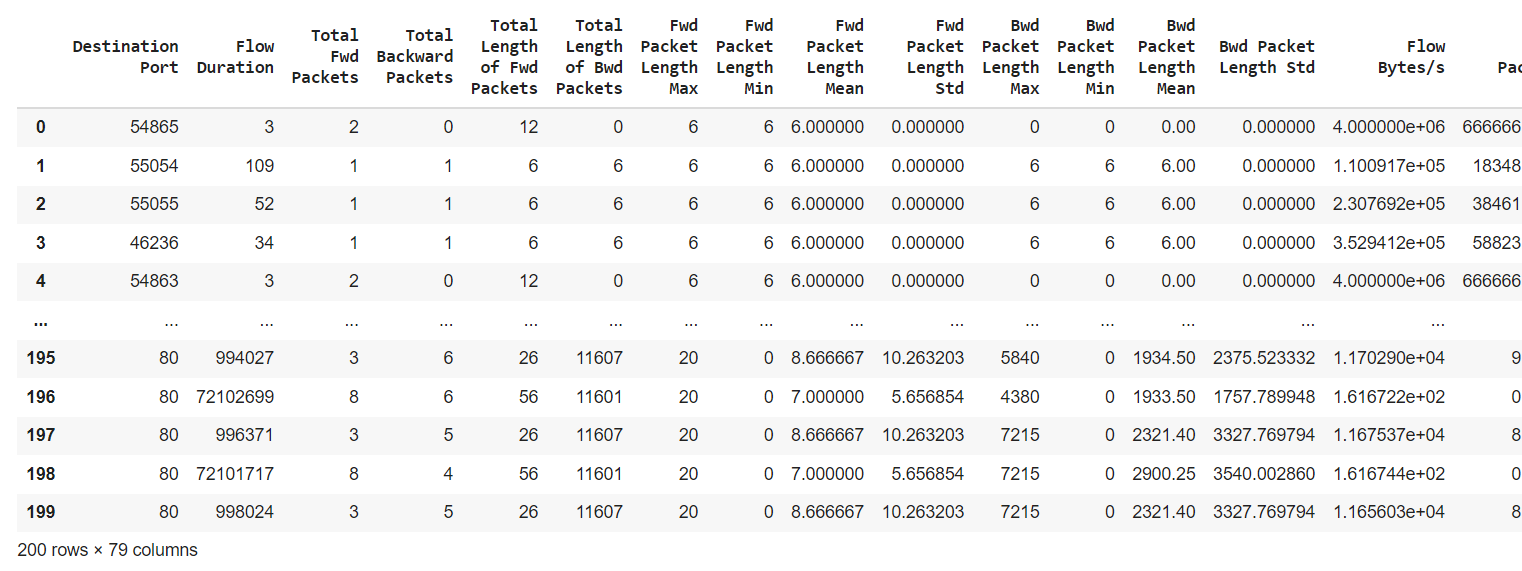
Click on the “Choose files” to navigate to the location of the data “DDos”.

**Reading and Decoding data**

The data is uploaded as ‘Dictionary’ type to the Python environment with on entry named ‘DDos.csv’. To change this data to a DataFrame, run the following command

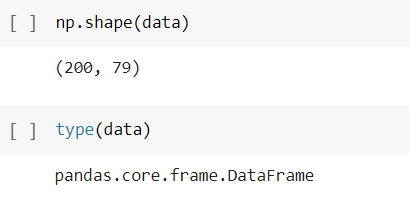


This will show the data as follows



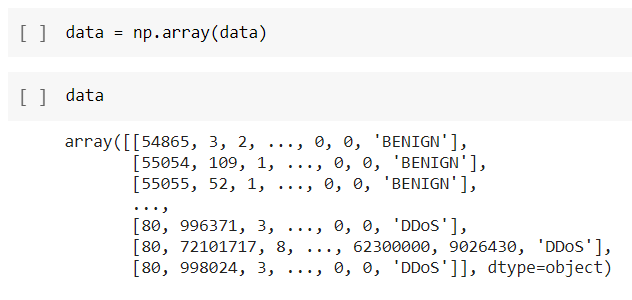
**Analysing data**

Following commands are used to check the size and type of data



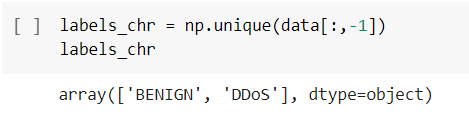
**Changing data type**

The data should be changed to array type in order to use it



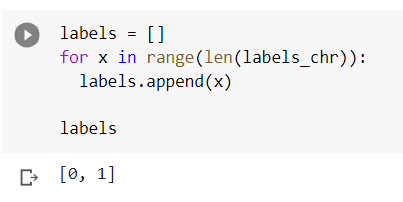
**Labels of data**

Obtain the list of labels given in the data. In this case, only two labels are given including ‘BENIGN’ and ‘DDoS’ which is the intrusion attempt.

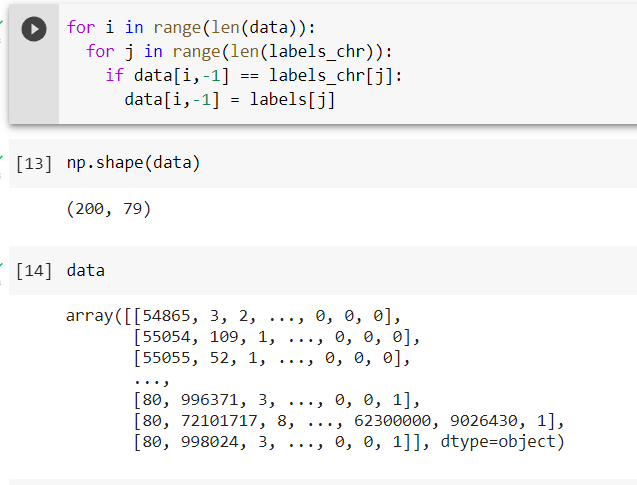


**Enumerating the labels**

As the ANN takes labels in form numbers 0,1,2,…; replace the labels with numbers where each label takes a number. In this case, ‘BENIGN’ is replaced with 0 and ‘DDoS’ is replaced with 1.

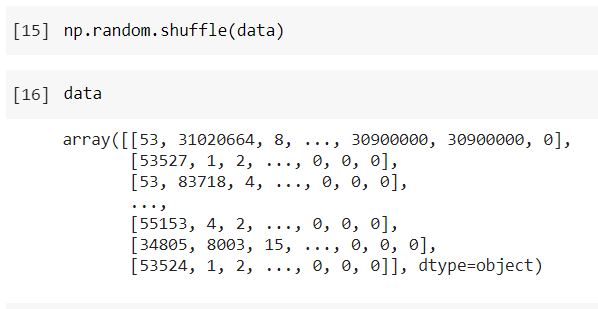


Now, transform this into the data as follows



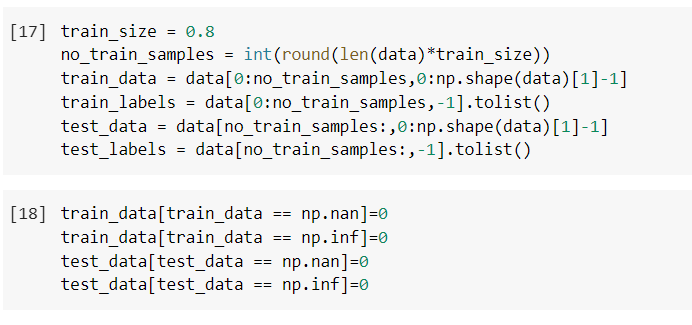
**Shuffle data**

Shuffle the data to mix samples from both classes. Shuffling is also important to increase the generalization capability of the classifier. Also, It may help reduce biasness which ultimately help reduce the chances of overfitting and underfitting classifier.



**Training and Testing Subsets**

Now that data is preprocessed and ready for being used to train the classifier, we divide data into two subsets including train\_data and test\_data with respective train\_labels and test\_labels. Here, train\_size represent the percent of data utilized in training. In this case, train\_size = 0.8 means that 80% of the data is used training while remaining 20% data is used for testing or analysing the performance of classifier.



**Train Classifier**

The train\_data along with train\_labels is used to train the classifier. The parameters used are

* **solver:** {‘lbfgs’, ‘sgd’, ‘adam’}, default=’adam’

The solver for weight optimization.

* + ‘lbfgs’ is an optimizer in the family of quasi-Newton methods.
  + ‘sgd’ refers to stochastic gradient descent.
  + ‘adam’ refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba

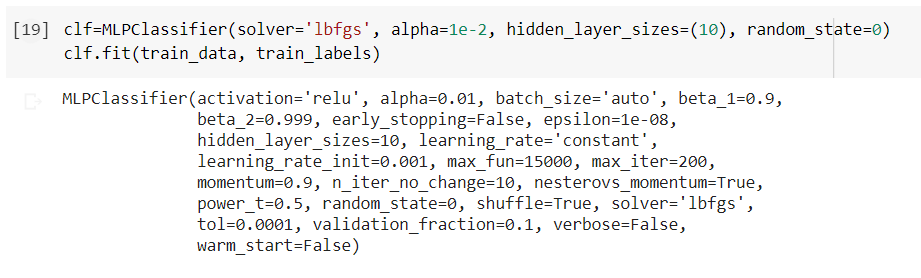
Note: The default solver ‘adam’ works pretty well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score. For small datasets, however, ‘lbfgs’ can converge faster and perform better.

* **alpha** : float, default=0.0001

L2 penalty (regularization term) parameter.

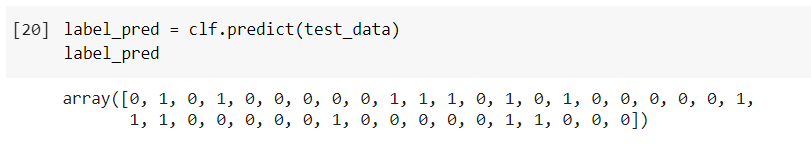
* **Hidden\_layer\_sizes:** tuple, length = n\_layers - 2, default=(100,)

The ith element represents the number of neurons in the ith hidden layer.

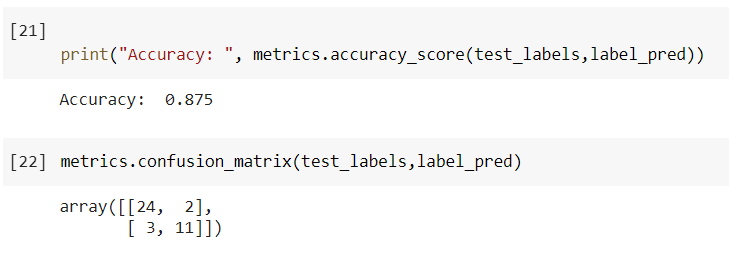


**Prediction with Trained Classifier**

Now, its time to assess the performance of trained classifier using the test\_data



**Performance Assessment**



**Exercise**

Change the following values to see its effect on performance

Solver = ‘sgd’, ‘adam’

Alpha = 1e-1, 1e-3, 1e-4

Hidden\_layer\_sizes = [20], [40, 20], [60, 40, 20]

Train\_size = 0.5, 0.6, 0.7, 0.9