System Block Diagram of Near Real Time Intrusion Detection System

Receive.py

Send.py

Capture 100 packets

from interface

Create Queue in

that channel

Create Channel

Save packets with

unique filename

Send filename

to the queue

Accept filename

from the Queue

Locate pcap file

from filename

Convert to

CSV format

Data

processing

Data

Encoding

Evaluate the record

with classifier model

Label the data

Write to

database

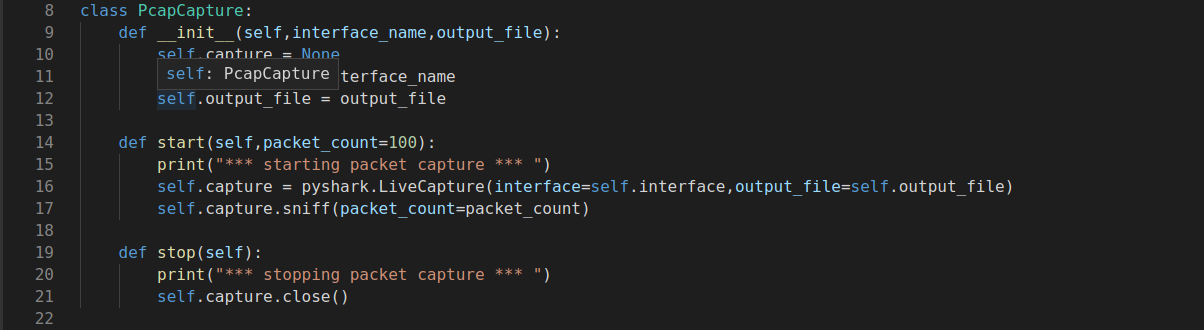
Queue

# code for send.py

Importing relevant package inside send.py script



This class helps us to select an interface(wifi or ethernet) from which the packets are to be captured

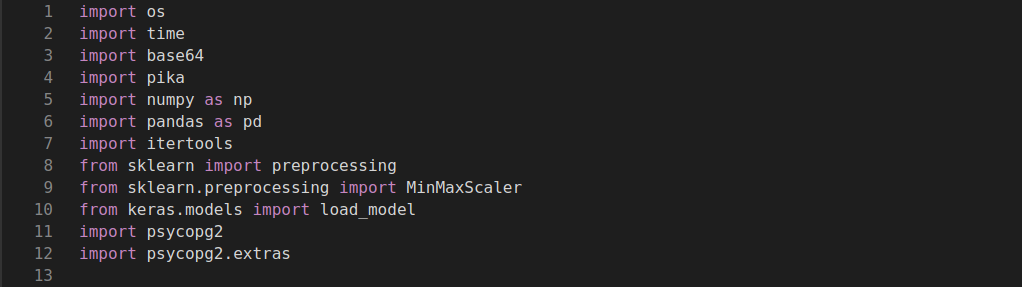


pika is used to create connection and provide channel for creating exchange queues. In this code we created an connection ato create channel inside the local host. Then we declared an exchange for sending captured packet’s filename.Inside the infinte loop the program captures 100 packets from wlp2s0 interface and sends it to exchange using the channel. The process works on loop continuously sending the packets filename

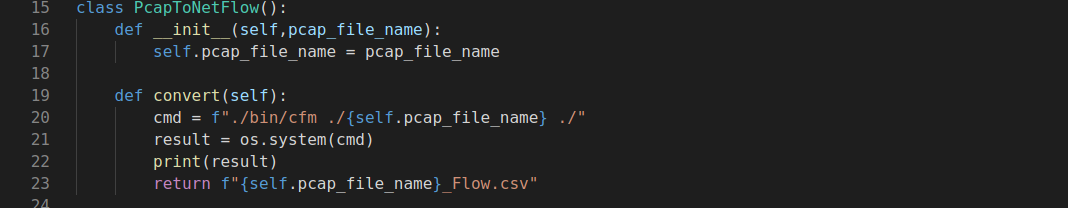


# code for receive.py

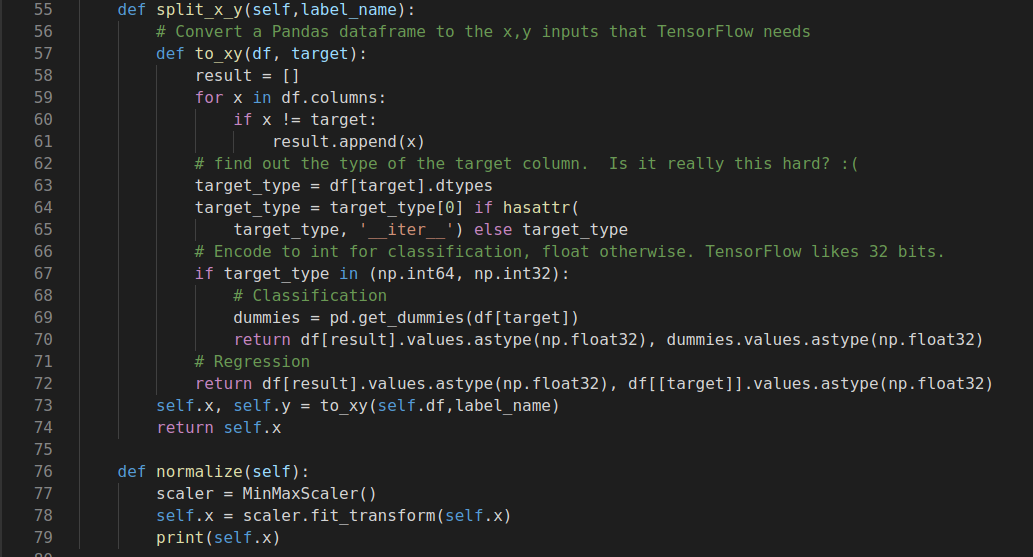
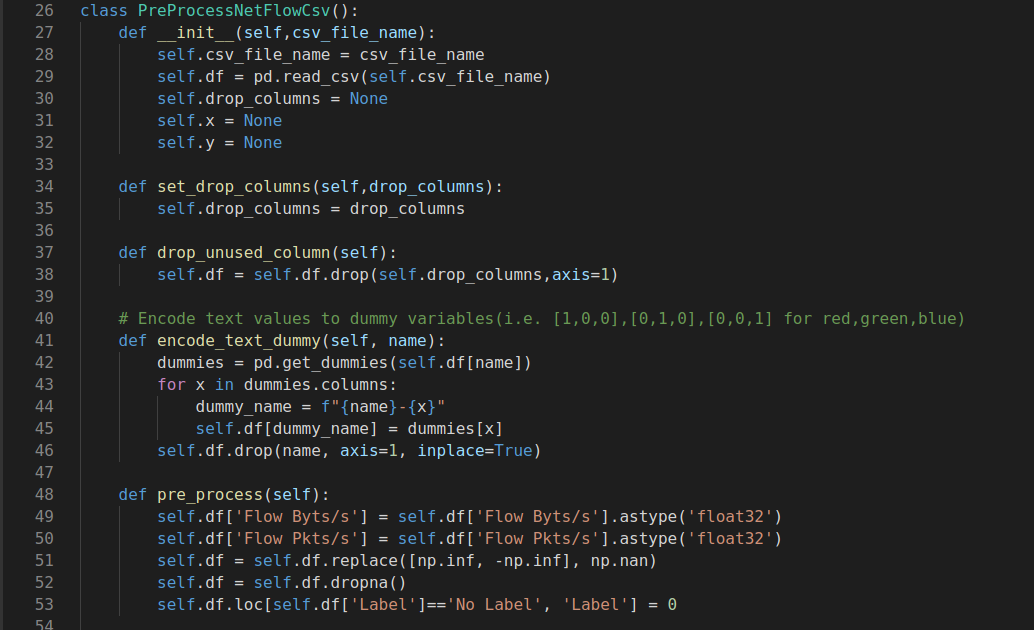
Importing relevant package for receive.py



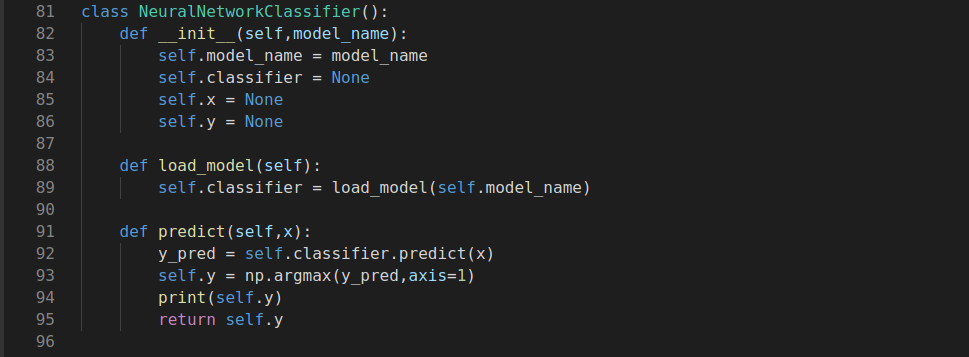
PcapToNetFlow class is used to get the pcap filename and convert it into netflow csv format for data processing.



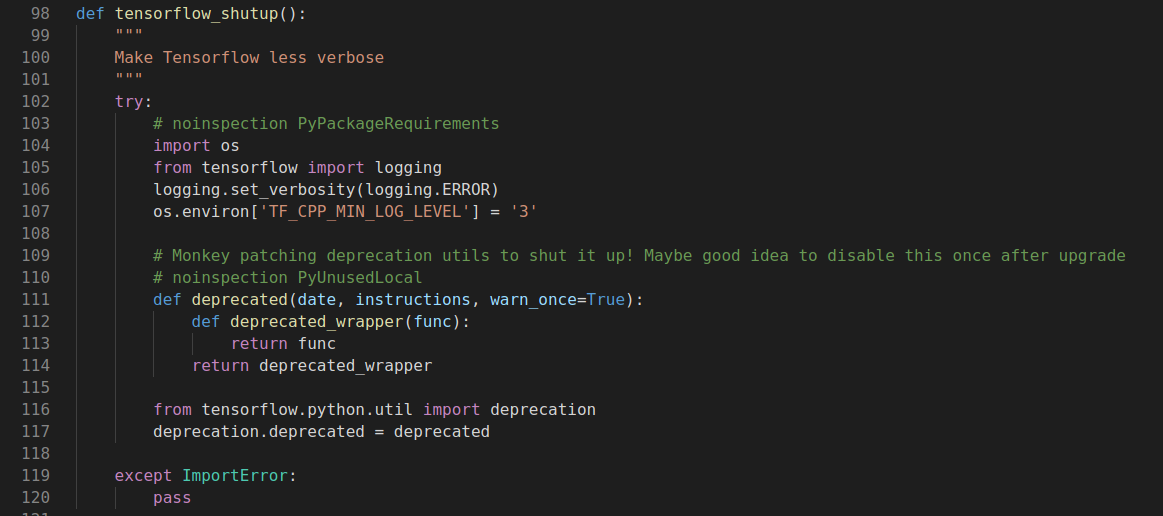
Class PreProcessNetFlowCsv is used to preprocess the raw data so that we can encode the data for classifier model to use it. It reads the CSV filename and converts it into dataframe. Unwanted columns are droped. One hot encoding is done to categorical data and data types are converted into float for machine for normalizing. Then data is splitted into X and y for data encoding and target labelling. In this process we have used MinMaxScaler which normalizes the data using min and max values of the data.

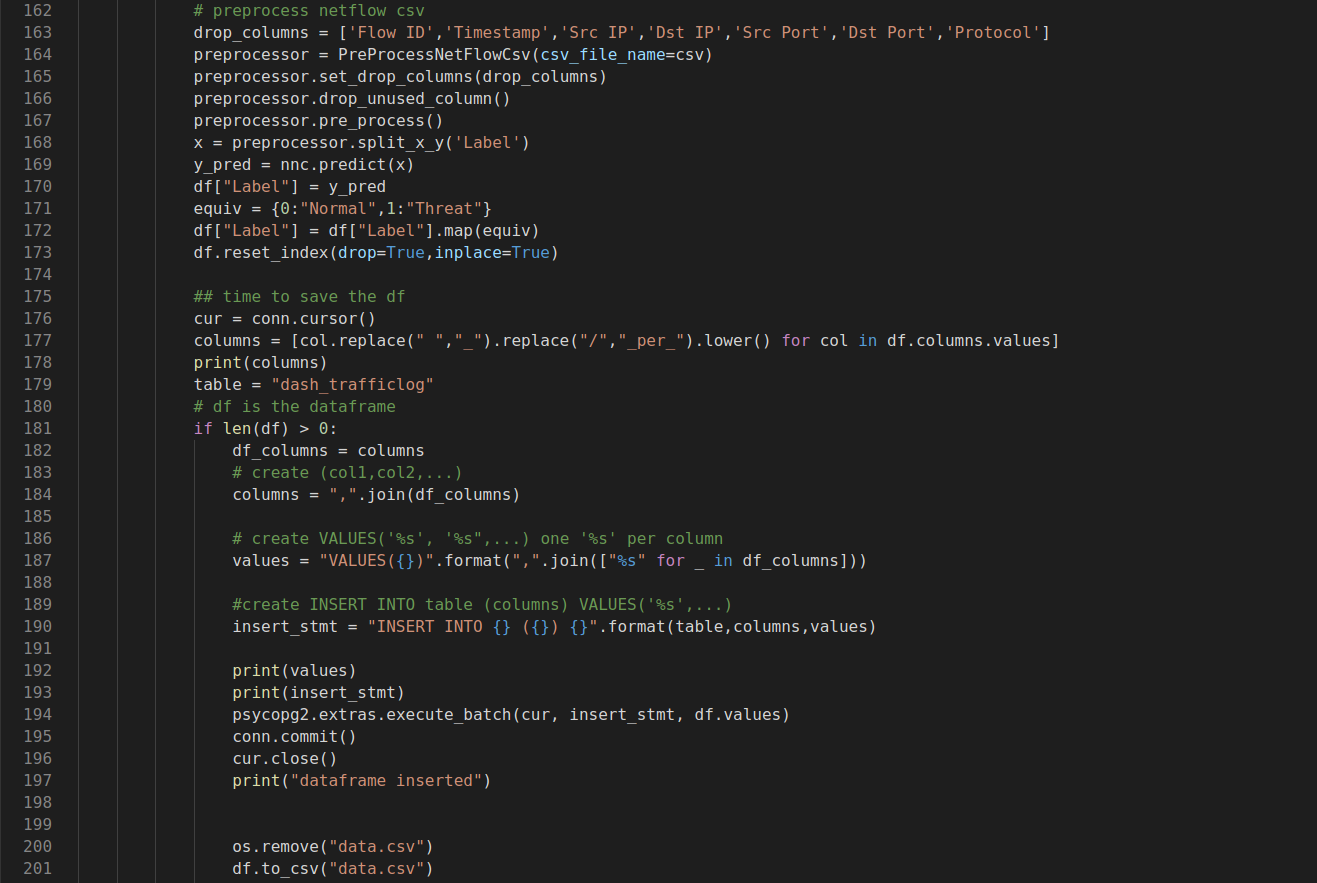
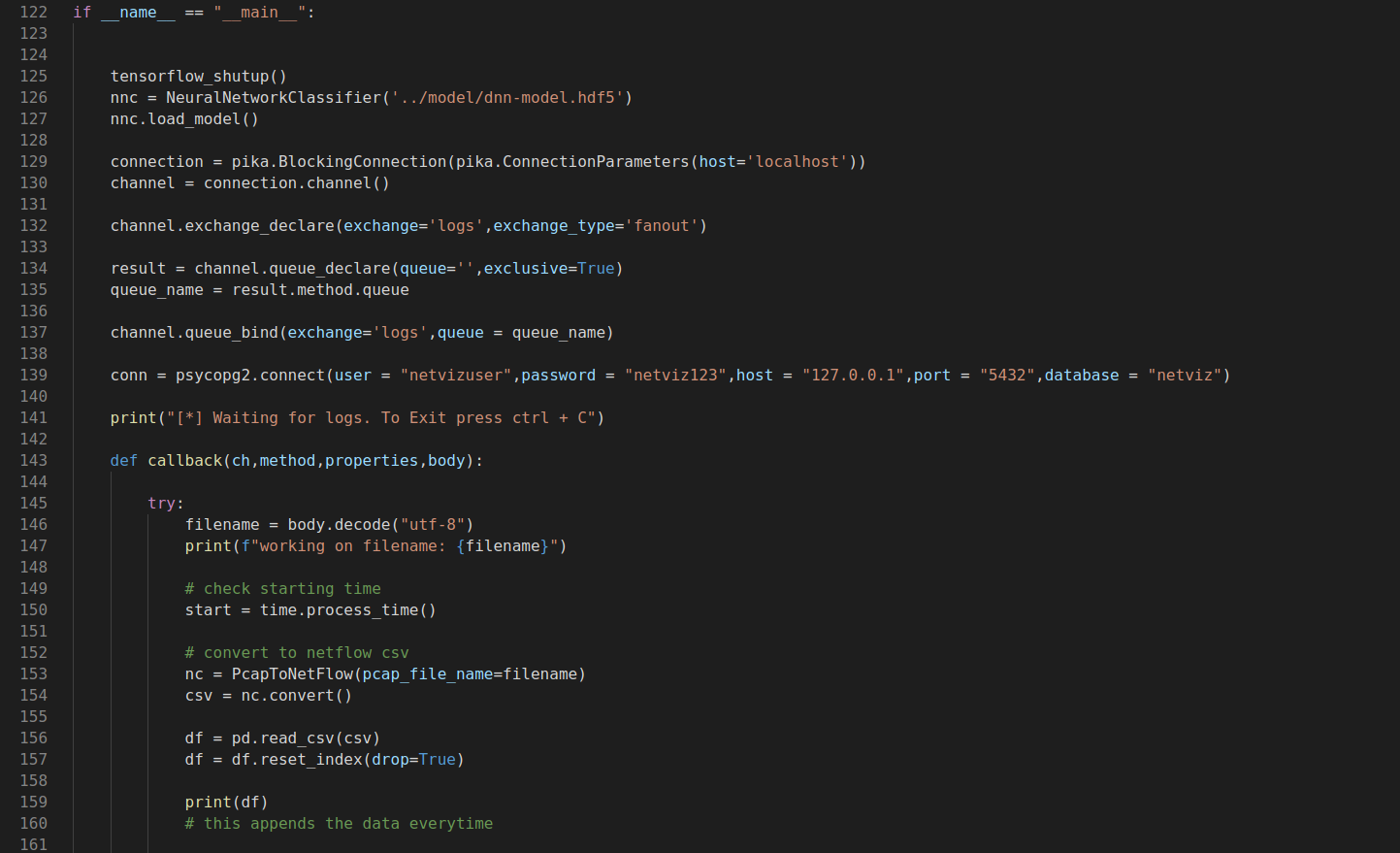


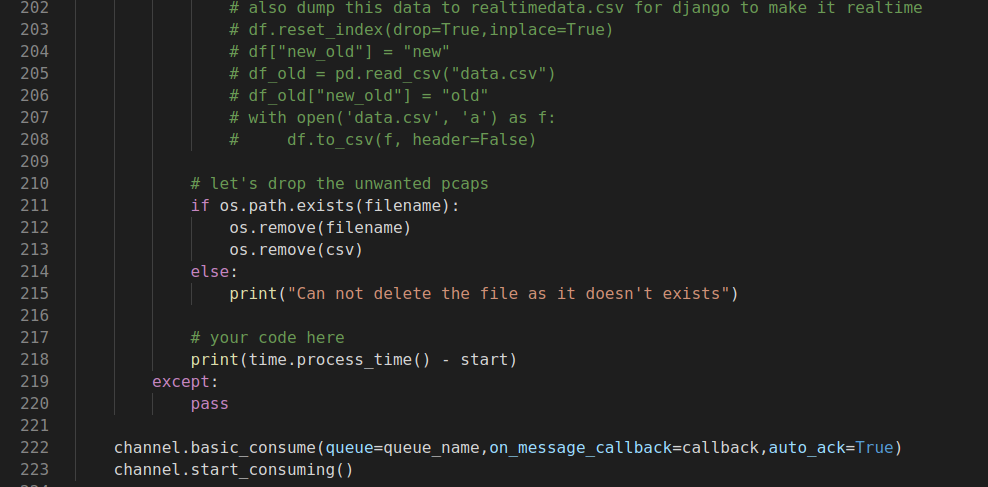
NeuralnetworkClassifier class loads the model and predicts the label of the record



this function is used to hide tensor-flow warnings that are unnecessary

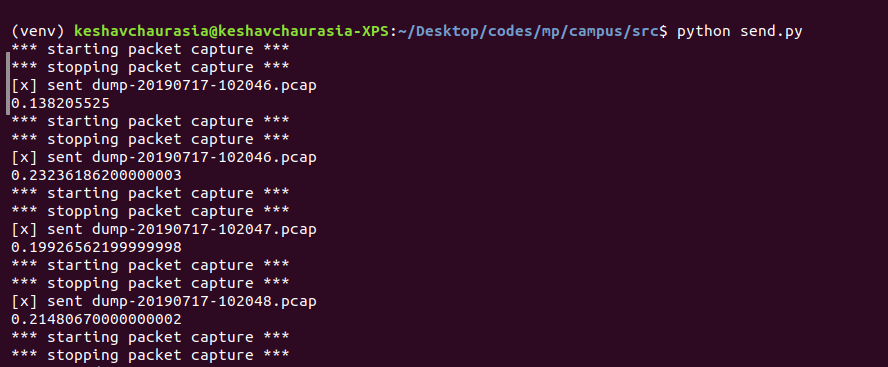




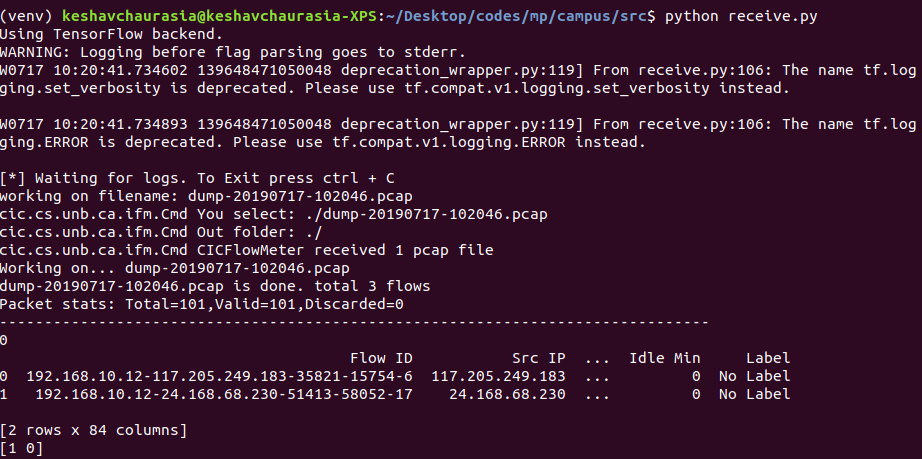


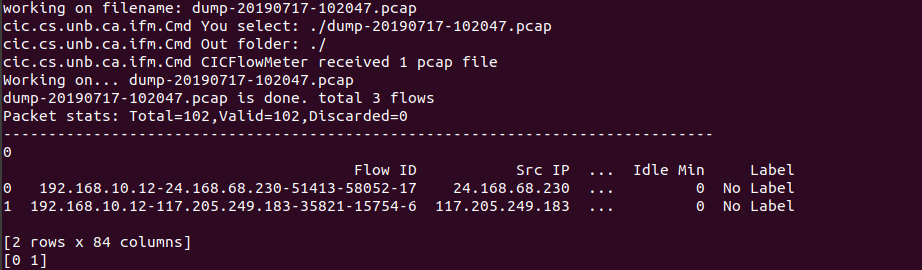
In this code we are utilizing all the class we created to receive the filename of the packet captured and converting to csv format. After that the data is preprocessed and normalized and evaluated using neural network classifier. After evaluation the data is labelled if the data is normal or attack and the final output is stored in the database for web viewing.

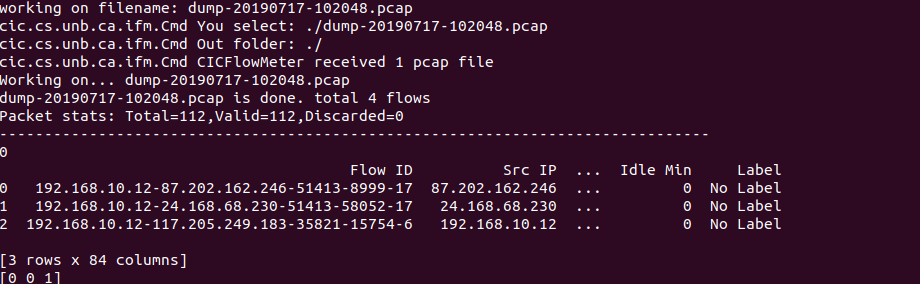
# Output from send.py

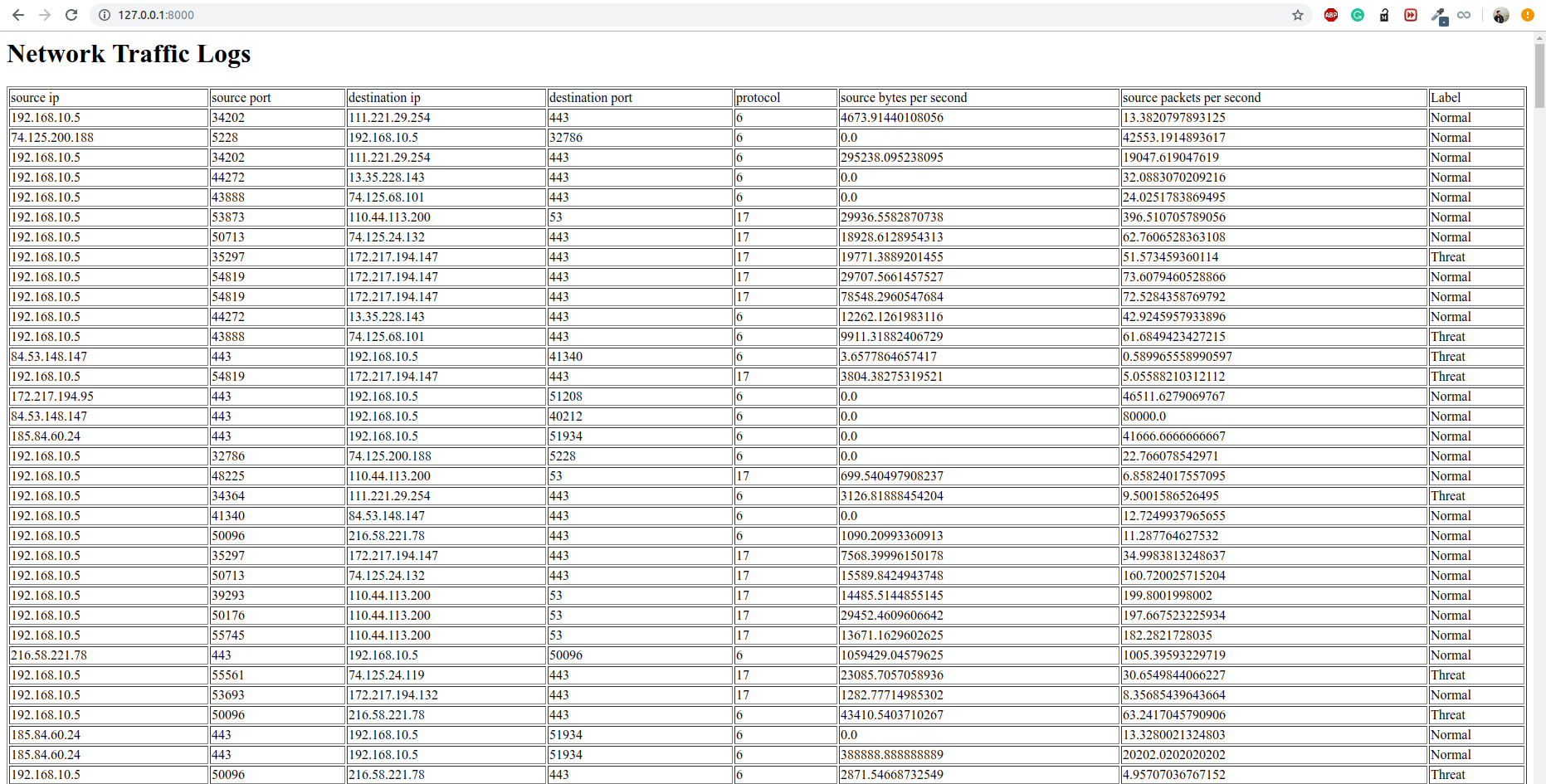


# output from receive.py







Web View of traffic logs with their output Label as Normal or threat