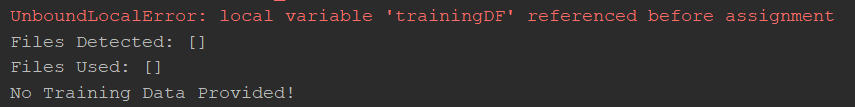
**Guide to Decision Tree Packet Algorithm**

In this how-to guide I will explain how the decision tree machine learning program works, and how to use it.

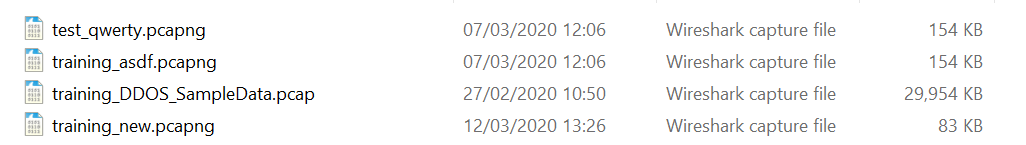
# Setup the “PacketFiles” Folder

You will need to create a folder called “PacketFiles” in the same directory as the script as can be seen here.

Note that if you don’t create the PacketFiles folder you will see this error message.

Inside the folder we are going to store the packet files. These packet files will need to be named according to their purpose.

* **training\_filename** – This packet file contains training data for the machine learning algorithm, which **does not contain** any DDOS packets.
* **training\_DDOS\_filename** – This packet file contains training data for the machine learning algorithm, which **does contain** DDOS packets, and the **vast majority** of those packets are DDOS packets.
* **test\_filename** – This packet file contains the packets which the machine learning algorithm will be tested against. This packet file **does not contain** any DDOS packets.
* **test\_DDOS\_filename** - This packet file contains the packets which the machine learning algorithm will be tested against. This packet file **does contain** any DDOS packets.

Here is an example of the packet files found in the PacketFiles folder.

We can now look at the program.

# Breaking Down the Program

For this program break down I will break down each function/line, in order of call in the program.

The first line executed is:

For creation of the visual analysis of the model, we will use a program called Graphiz (<https://www.graphviz.org/>). The filepath to this software must be specified in this line. You do not need to include this if you do not plan to make a visual representation of the algorithm.

**Main()**

The main method will be where the program flow will be. This means that from this function, all other functions are called.

**createPacketFileLocation()**

Checks if the PacketFiles folder has been created. If not, the folder is created.

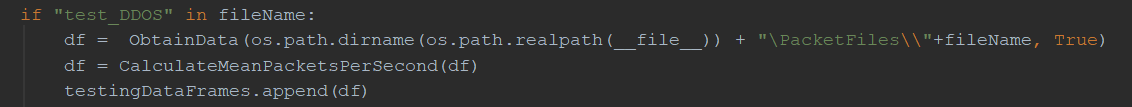
A screenshot of a cell phone

Description automatically generated**validateFiles()**

Checks if the files in the PacketFiles folder are valid i.e. are of filetype .pcapng or .pcap. If you need other filetypes supported, simply add the file extension to the list comprehension (Line 179).

Back in the main method, now that we have the valid files, we iterate over them in the for loop,



and check what the purpose of each file is, by checking the prefix of each file. E.g for test\_DDOS files:

As can be seen in the above image, we call the ObtainData method, which returns a dataframe, and then we call CalculateMeanPacketsPerSecond upon that dataframe. Finally, we append the dataframe to the appropriate list. This set of calls occurs for each packet file.

**ObtainData(PacketFile,DDOSStatus)**

This function reads the packet file passed into it and stores the data read in a variable called cap. We then iterate through the data. This can be seen in the screenshot below.

A screenshot of a cell phone

Description automatically generated

For each packet in the packet file, we need the pertinent information from said packet. This is why we define a list called packetFileData which will contain this data. The data contained within packetFileData will be our **features** for are machine learning model

List of Features:

* Timestamp – Refers to when the packet was sent. (Note that this feature will not be included in the model, but is necessary for calculating packets per second.)
* SourceIP – Refers to the IP of the sending device
* SourcePort – Refers to the port used to send the packet.
* DestinationIP – Refers to the IP of the receiving device.
* DestinationPort – Refers to the port used to receive the packet.
* Protocol – Refers to the protocol of the packet.
* DDOS? – Either True or False and refers to whether or not the packet is a DDOS packet or not.
* Size – The size of the packet.

I use PyShark to obtain all of this data(apart from DDOS? which is obtained from the other parameter passed into the ObtainData method). PyShark is very easy to use, as it is simply a wrapper for tshark. It allows for easy access to the packets data by using . accessors, e.g

packet.ipv6.src

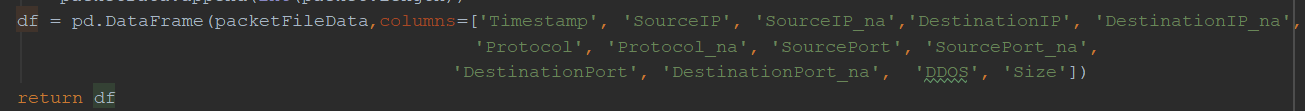
refers to the SourceIP (src) of the packet using IP version 6 (ipv6).

For special case protocols such as ARP, which do not contain a Source Port etc, we need to identify them, and assign null values to the respective features they are missing e.g

A screen shot of a computer

Description automatically generated

You may note that a integer value of 1 is appended to the list when a null value occurs, or when the null value does not occur a 0 is appended. This is because the machine learning algorithm can only accept numbers as input, not null values, so in order to detect these null values, a separate collumn, simply called columnname\_na is made. A 1 means the corresponding value is null, a 0 means that it is not null.

The final dataframe is my adding the packetFileData for each packet. You can see the column names of the dataframe here.

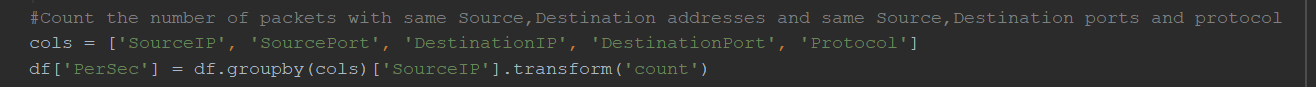
We then return the dataframe.

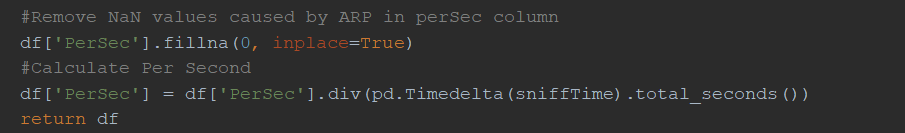
**CalculateMeanPacketsPerSecond(dataframe)**

This function calculates the mean packets per second sent from a Source IP using the same SourceIP,SourcePort,DestinationIP,DestinationPort and Protocol.This result (PerSec) is another **feature** that we use in the machine learning process.

A screen shot of a social media post

Description automatically generatedThe first step of calculating something per second is to determine the amount of time that has passed during the reading. In the below code we get the first and final entries in the dataframe and subtract their Timestamps from one another to determine the time period.

The next step is to count the occurrence of the number of packets with the same SourceIP,SourcePort,DestinationIP,DestinationPort and Protocol. This can be seen below.

Finally, divide the number of occurrences by the amount of time pas to obtain the number of packets per second.

**Proc\_df(dataframe)**

Proc\_df stands for process dataframe. It calls a series of functions that will ensure the dataframe is ready to be used in the machine learning algorithm. This function was listed from the fast.ai library. If you would like more information on this function please consult their documentation.

**numericalize(dataframe)**

Ensures that the category codes are not negative.

**train\_cats(dataframe)**

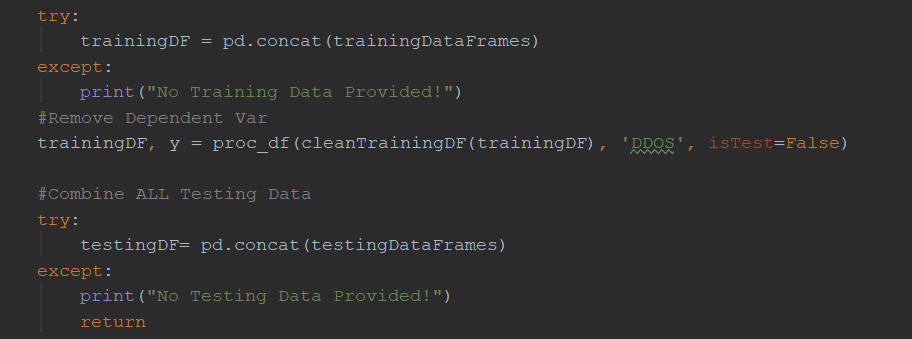
Converts collumns that are of datatype string into datatype category.

**cleanTrainingDF(dataframe)**

The machine learning algorithm will only accepts numbers as an input, thus things like Source IPs which are strings “192.56.43.21” need to be converted into a number. To do this each category which is not of datatype number is converted into a category datatype. When a collumn is of datatype category it stores the values it holds as numbers. These numbers are referred to as category codes. These category codes are assigned to their String counterparts in a separate internal table by pandas.

**cleanTestingDF(dataframe)**

Same as cleanTrainingDF.

The final step of the data processing is merging all of the testing and training pandas dataframes into their respective dataframe, which we will use to train the and test the model on, as can be seen below.

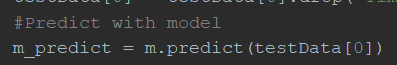
Now that all the data has been processed, we can now create the model. We are using the **Decision Tree algorithm from sklearn**. I have tried a variety of different classifier algorithms and this algorithm seems to be the best in terms of accuracy and not overfitting. For more information on the Decision Tree algorithm, I suggest reading the documentation for it on the sklearn website. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

Here is the creation and training of the model.

A picture containing holding, red, white

Description automatically generated

And here is the model predicting the identity of the packets (whether or not they are DDOS packets or not)

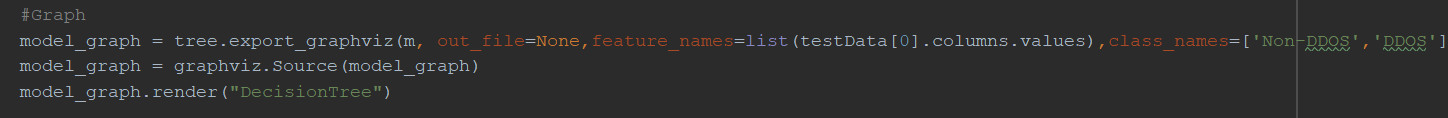


Finally, in order to get an accuracy score, we use the actual scores and the predicted scores. You don’t need the print statements.

A picture containing drawing

Description automatically generated

A screenshot of a cell phone

Description automatically generatedIf you would like a visual representation of the decision tree, then you can add the code below:

**Issues**

When reading multiple files, PyShark will give this run time error:



To resolve this, you will have to add asynchronistic functions to the program, something which is not in the scope of this tutorial. The program will still work correctly with this error present

This code will use graphiz to produce a decision tree diagram for your model. It will be produced a pdf document in the same directory as the script. Here is an example of the output.

A close up of text on a white background

Description automatically generated