Table of contents

Table of figures

Introduction

Chapter I
Problematic

The rise of Distributed Denial of Service (DDoS) attacks poses a significant threat to network security, making it crucial to develop effective detection mechanisms. In the context of machine learning, the challenge lies in accurately identifying DDoS attacks from normal network traffic data.

Detecting DDoS attacks presents several challenges due to the dynamic and evolving nature of these attacks. Traditional detection methods often fall short in accurately identifying DDoS attacks, leading to increased risk and potential downtime for organizations. There is a critical need for an effective and efficient DDoS detection solution that can adapt to the changing nature of attacks. Machine learning offers a promising approach to DDoS detection, thanks to its ability to analyze large volumes of data and identify complex patterns that may indicate an ongoing attack.

In this project, our objective is to develop a web application that provides users with an interface to upload their Packet Capture (PCAP) files. The application will then process these files and present the results in a tabular format, indicating whether the traffic is classified as NORMAL or as a DDoS attack.

1. What is a DDoS attack ?

A distributed denial-of-service (DDoS) attack is analogous to a group of people crowding the entry door of a shop, making it hard for legitimate customers to enter, thus disrupting trade and losing the business money.

It is a malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic.

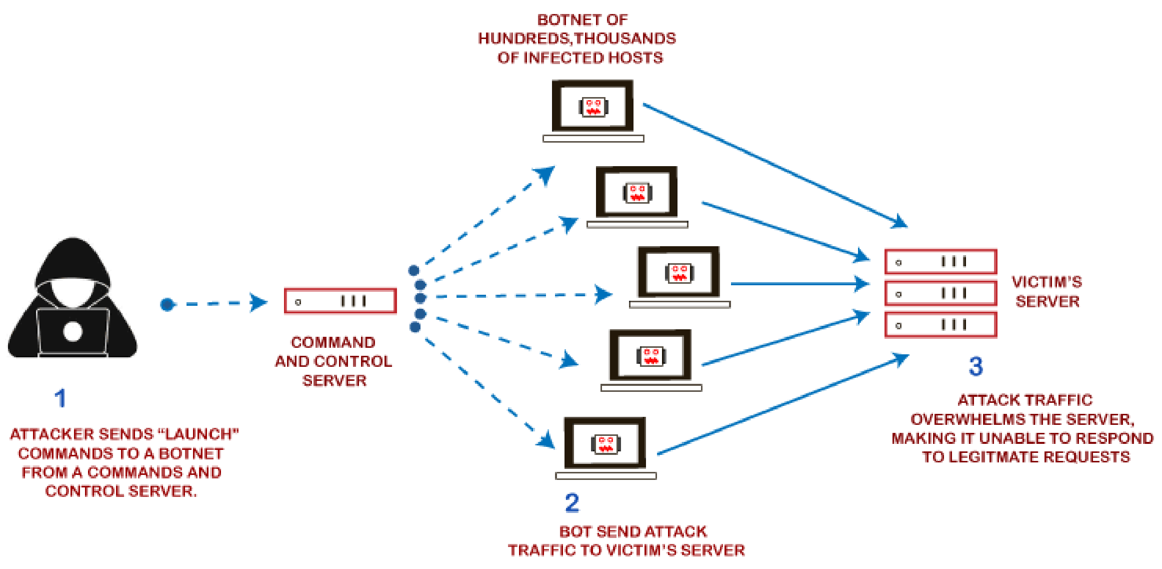
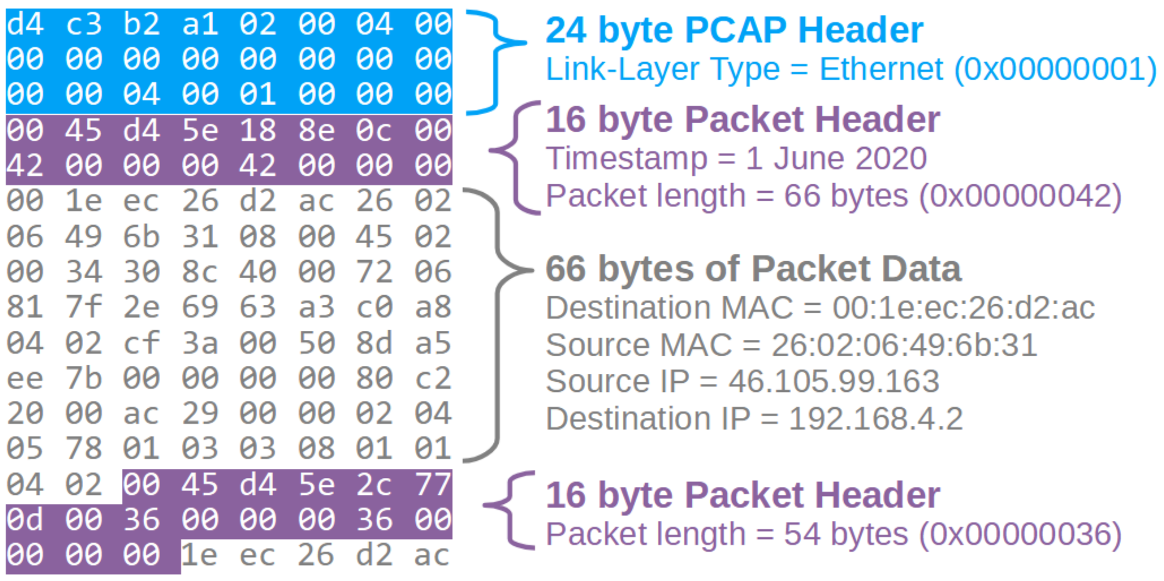


Figure I.1: DDoS attack

2. Structure of a PCAP file

A PCAP file is a binary file format that stores network traffic data. It captures packets in a structured manner, preserving the details of each communication unit traversing a network. These files are instrumental for network administrators, analysts, and cybersecurity professionals in diagnosing network issues, monitoring activities, and investigating security incidents.

The PCAP's file structure is defined by three fundamental components : the Global Header (PCAP Header), the Packet Headers, and the Packet Data.



```
d4 c3 b2 a1 02 00 04 00
00 00 00 00 00 00 00 00
00 00 04 00 01 00 00 00
00 45 d4 5e 18 8e 0c 00
42 00 00 00 42 00 00 00
00 1e ec 26 d2 ac 26 02
06 49 6b 31 08 00 45 02
00 34 30 8c 40 00 72 06
81 7f 2e 69 63 a3 c0 a8
04 02 cf 3a 00 50 8d a5
ee 7b 00 00 00 00 80 c2
20 00 ac 29 00 00 02 04
05 78 01 03 03 08 01 01
04 02 00 45 d4 5e 2c 77
0d 00 36 00 00 00 36 00
00 00 00 1e ec 26 d2 ac
```

24 byte PCAP Header
Link-Layer Type = Ethernet (0x00000001)
16 byte Packet Header
Timestamp = 1 June 2020
Packet length = 66 bytes (0x00000042)
66 bytes of Packet Data
Destination MAC = 00:1e:ec:26:d2:ac
Source MAC = 26:02:06:49:6b:31
Source IP = 46.105.99.163
Destination IP = 192.168.4.2
16 byte Packet Header
Packet length = 54 bytes (0x00000036)

Figure I.2: PCAP file structure

Chapter II
Data exploration

1. Description of the dataset

Source of the dataset : Dataset link

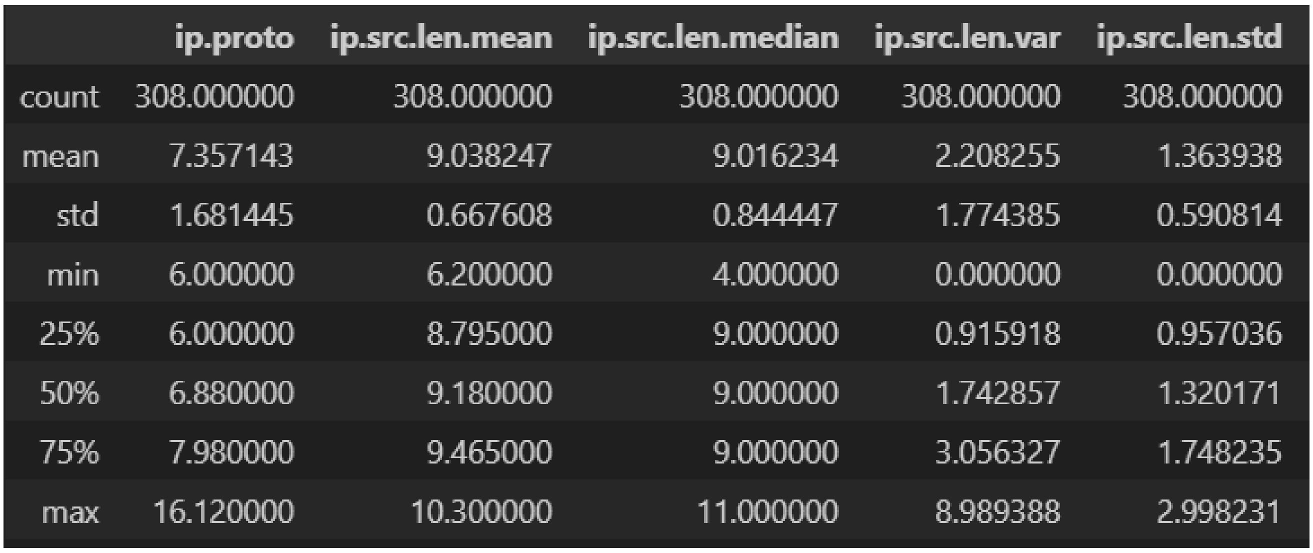
Our dataset is provided by the Canadian Institute for Cybersecurity of the University of New Brunswick (UNB). The dataset, is specifically designed for evaluating Intrusion Detection System (IDS) algorithms and systems on DDoS (Distributed Denial of Service) attacks.

More details on the dataset can be found in this paper : Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", IEEE 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019.

2. Data exploration

2.1. Describing the dataset

We used the **describe()** function in order to generate descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution.



	ip.proto	ip.src.len.mean	ip.src.len.median	ip.src.len.var	ip.src.len.std
count	308.000000	308.000000	308.000000	308.000000	308.000000
mean	7.357143	9.038247	9.016234	2.208255	1.363938
std	1.681445	0.667608	0.844447	1.774385	0.590814
min	6.000000	6.200000	4.000000	0.000000	0.000000
25%	6.000000	8.795000	9.000000	0.915918	0.957036
50%	6.880000	9.180000	9.000000	1.742857	1.320171
75%	7.980000	9.465000	9.000000	3.056327	1.748235
max	16.120000	10.300000	11.000000	8.989388	2.998231

Figure II.1: Describing the dataset

2.2. Distribution of the status column

We used **displot()** from **seaborn** to plot the distribution of the status column. 0 is the DDoS traffic and 1 is the normal traffic.

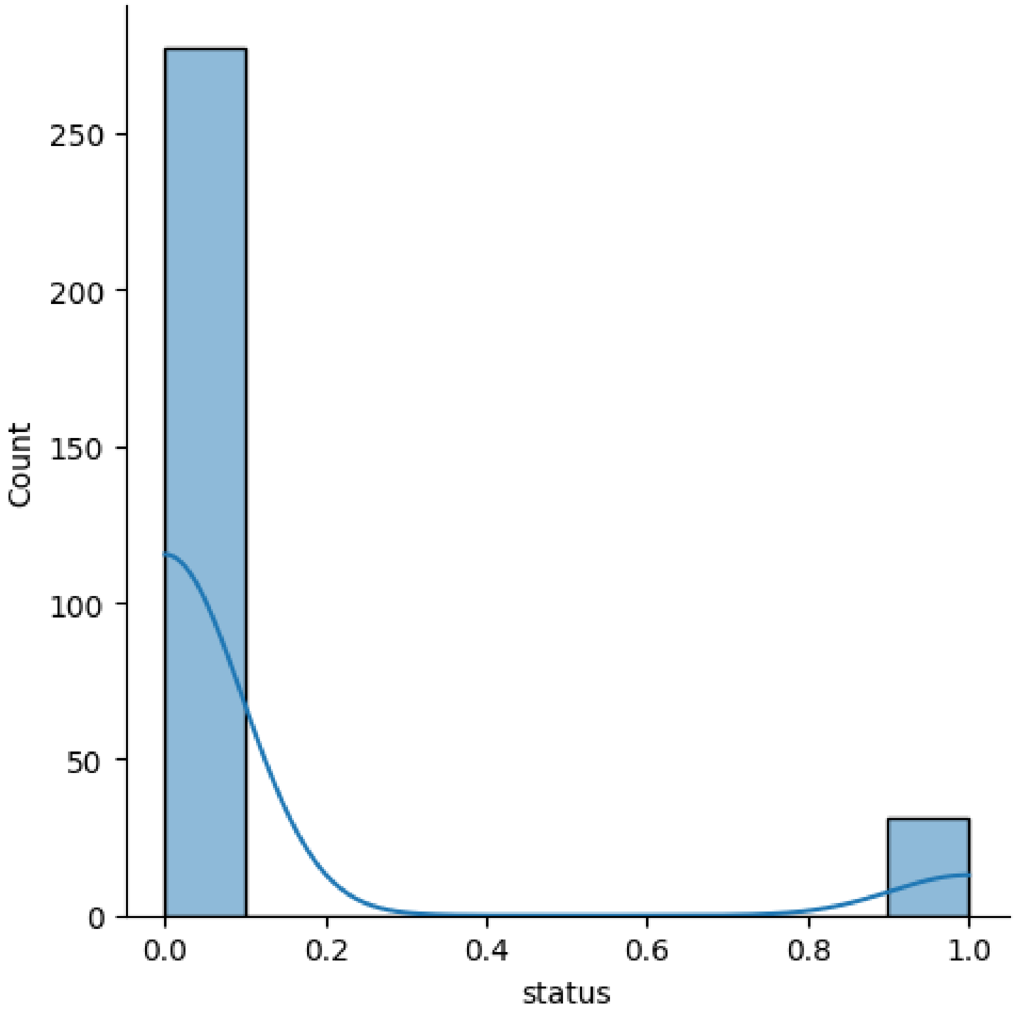


Figure II.2: Distribution of the status column

2.3. Histograms of the variables

Then we plotted the histograms for each variable :

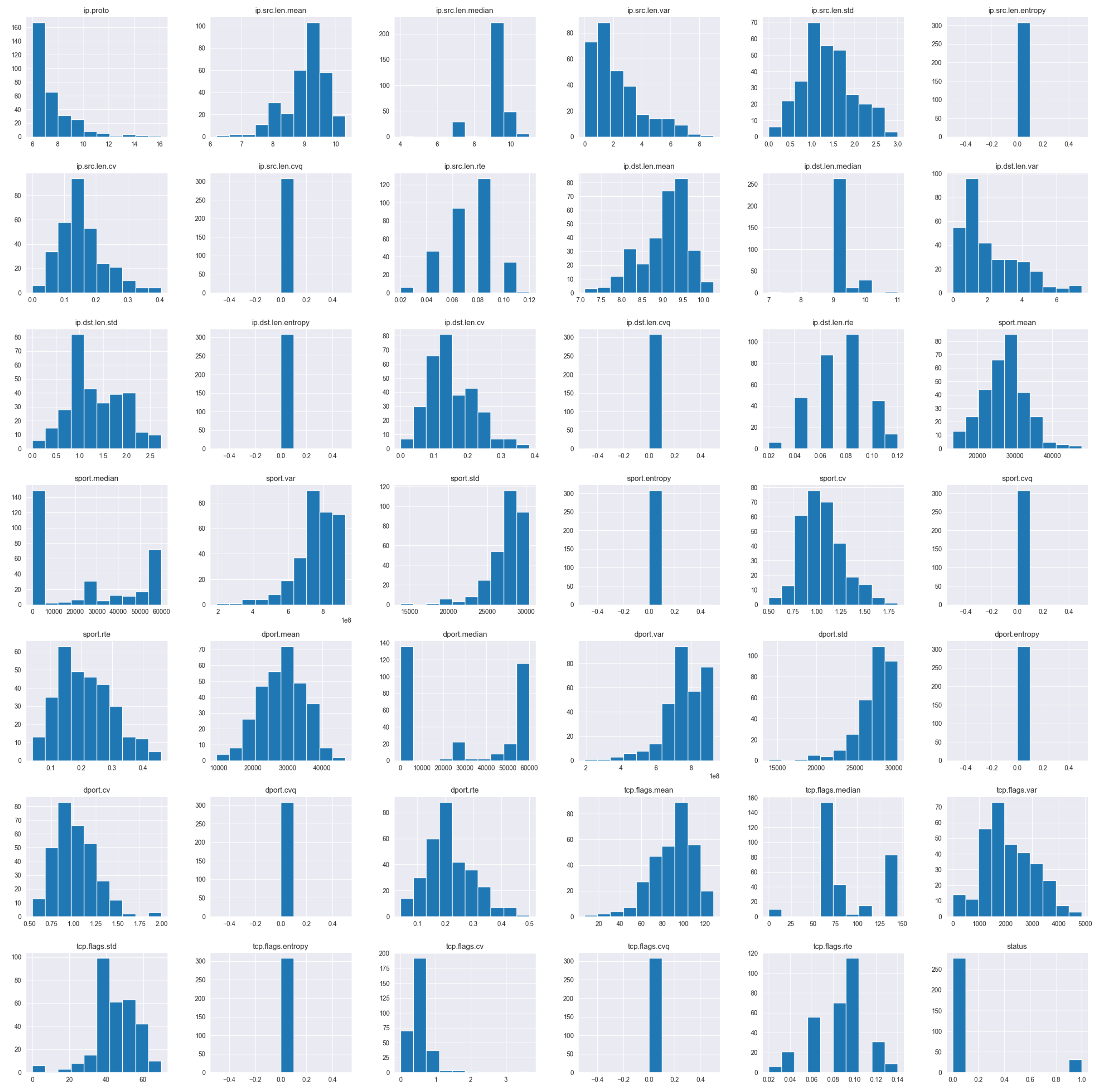


Figure II.3: Histograms of the variables

2.4. Boxplots

Boxplots are useful for visually summarizing the distribution of a dataset, identifying outliers and comparing the distribution of the variables. They provide a compact way to display key statistical information about the data.

2.5. Correlation between different attributes

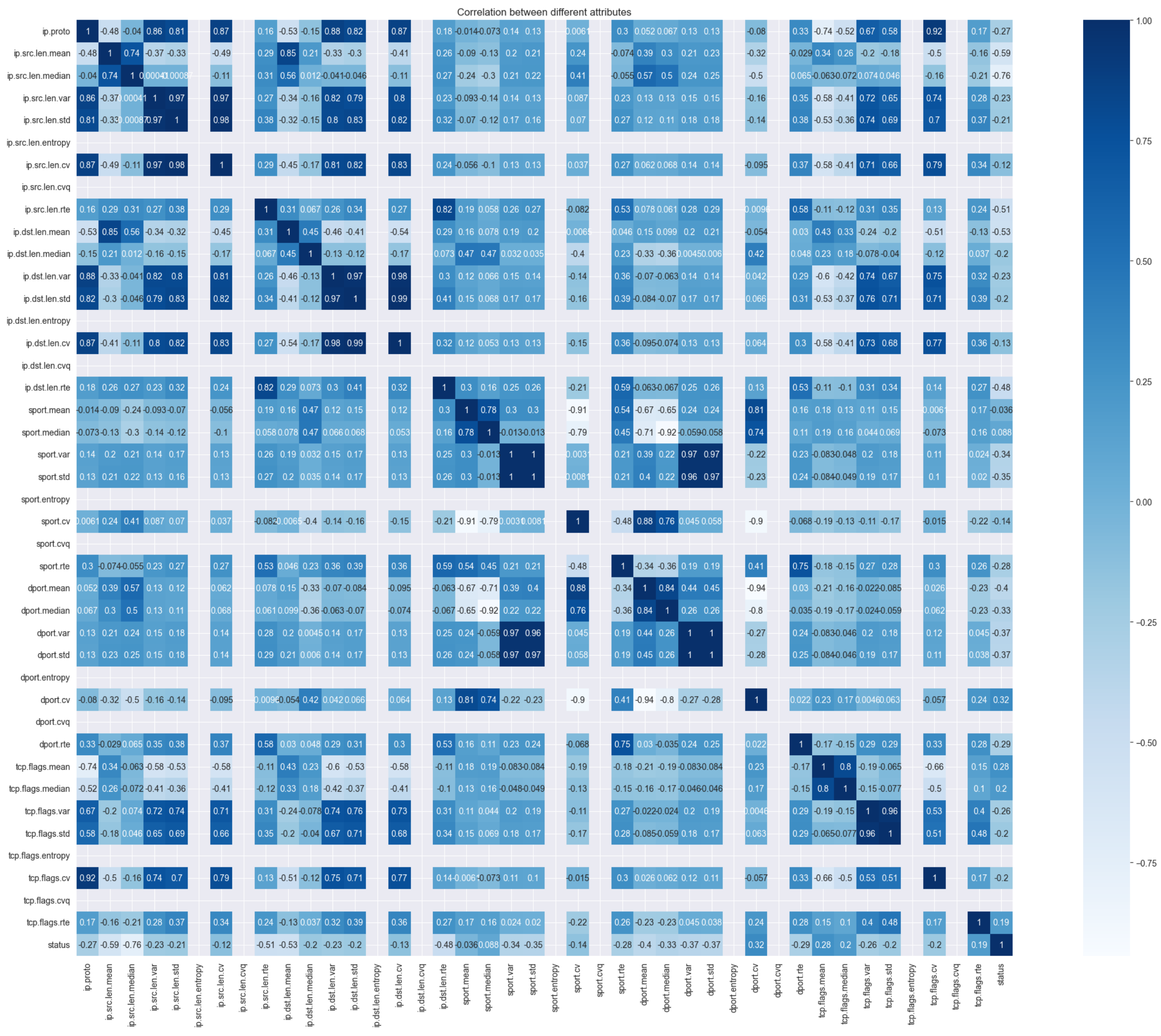


Figure II.4: Correlation between different attributes

Chapter III
Model training

1. Random Forest algorithm

1.1. Decision Trees

A decision tree is a type of supervised learning algorithms that is used for both classification and regression tasks. Decision trees learn a series of hierarchical ‘if/else‘ questions to classify data or predict outcomes.

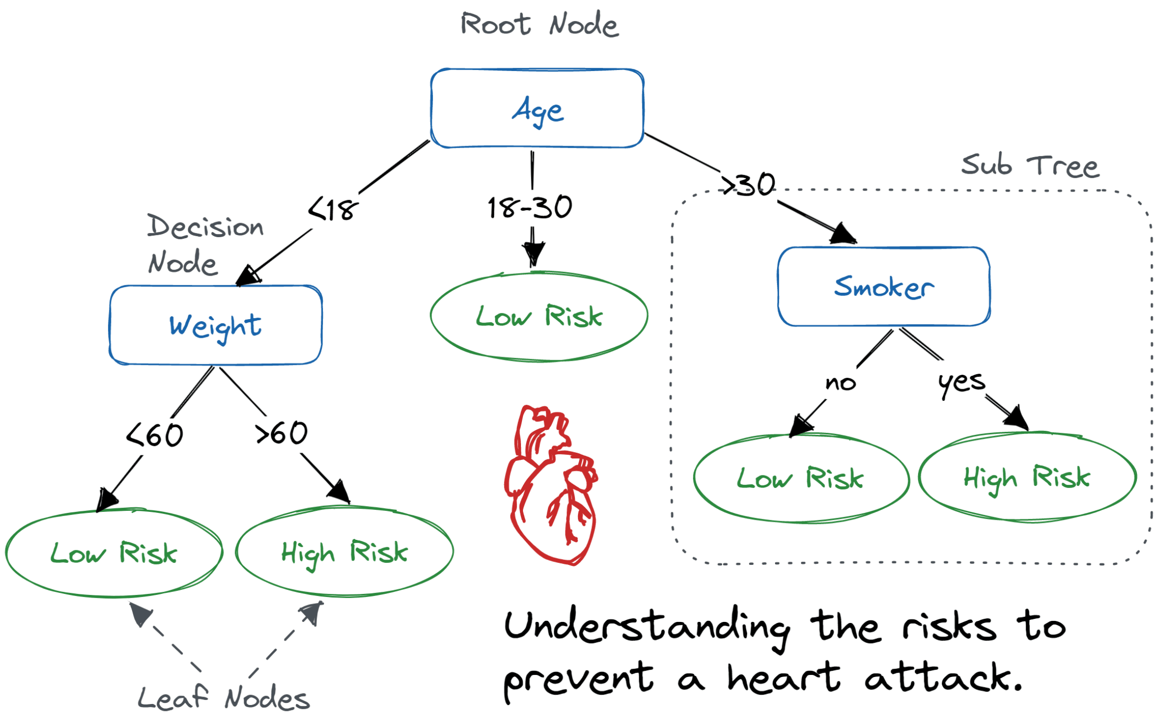


Figure III.1: Decision Tree example

1.2. Random Forests

A random forest is an ensemble learning method that uses a collection of **decision trees** to make predictions. It builds multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

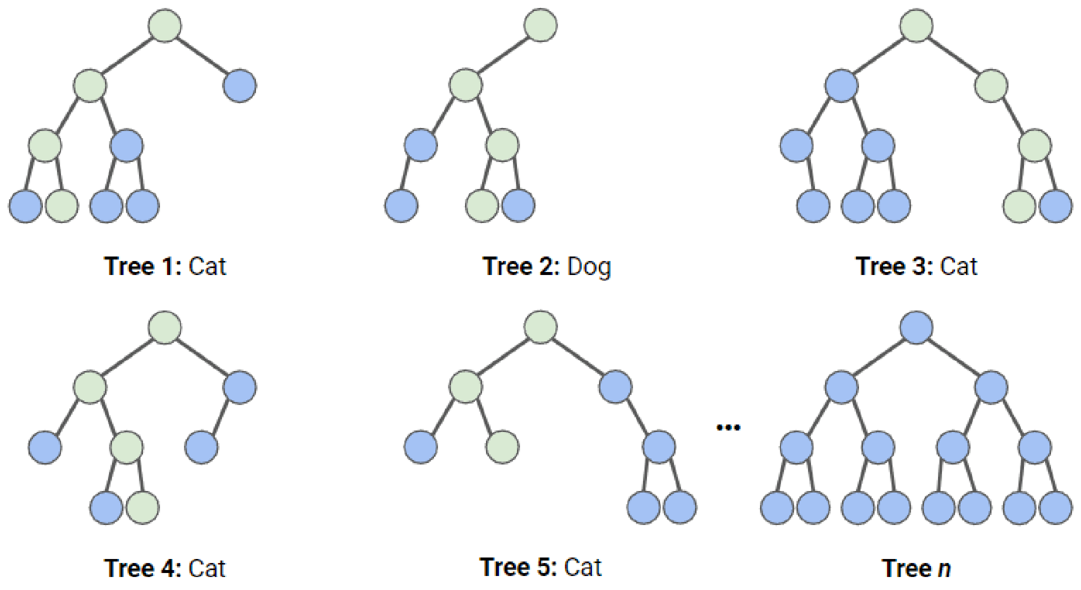


Figure III.2: Random Forest

2. Feature extraction and engineering

2.1. Feature extraction

In this step, we extract relevant information from the **'pcap'** files to create a dataset suitable for machine learning.

Each entry in the dataset represents a network packet and contains attributes such as :

```
["ip_source", "ip_destination", "frame_len", "protocol", "port_source",
 "port_destination", "flags", "frame_number"]
```

By converting the **pcap** files to **csv** format, we can easily analyze and process the data.

To manipulate packets in order to extract our features, we used **scapy**. Scapy is a powerful interactive packet manipulation library written in Python. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies.

2.2. Feature engineering

Because we cannot detect a DDoS attack by examining each individual row of our dataset alone, we reduced the size of our dataset by dividing it into batches of 100 rows. To do this, we calculated a series of statistical functions (such as mean, median, variance, standard deviation, etc.) for each batch using **pandas**.

The results of these calculations will serve as the new rows in our dataset, effectively summarizing our previous data.

3. Training the model

To train our model, we used the **RandomForestClassifier** in scikit-learn. The RandomForest-Classifier is a collection of decision trees. Each decision tree is built using a subset of the training data and a subset of the features.

To create each decision tree, the algorithm randomly selects a subset of the training data. Additionally, for each split in the decision tree, only a random subset of features is considered. This helps to introduce randomness into the trees and reduces the correlation between them.

The scikit-learn implementation of RandomForestClassifier uses an optimized version of the Random Forest algorithm to improve efficiency and scalability. The algorithm supports parallelism, allowing it to take advantage of multi-core processors for faster training.

Our RandomForestClassifier constructor to create our Random Forest classifier model in scikit-learn with the following parameter: n_estimators: This parameter specifies the number of decision trees to be used in the Random Forest. For us we chose n_estimators=10 wich means that the Random Forest will consist of 10 decision trees.

Chapter IV
App Development

1. **Backend**
2. **Frontend**

Conclusion

This is a genral conclusion