0 Introduction

0.1 DDoS attacks

In the cyber threat landscape, Distributed Denial of Service (DDoS) attacks are distinguished by their ability to cripple networks, services, and systems, denying access to legitimate users. These attacks exploit vulnerabilities inherent in the protocols underlying the World Wide Web, flooding targets with a disproportionate amount of traffic from multiple compromised devices, often thousands.

0.2 Project Objectives

The primary objective of this project is to develop a machine learning model for network traffic classification, with the intent to distinguish between benign traffic and DDoS (Distributed Denial of Service) attacks.

To achieve this goal, an approach based on machine learning techniques will be adopted. The model will be trained using a dataset provided by the instructors, which includes a variety of packet flow samples. This dataset has been specifically curated to encompass both legitimate traffic and data related to various DDoS attack scenarios, thereby ensuring an accurate and balanced representation of the possible situations the model will encounter in a real context.

0.3 Project structure

The project is divided into a series of comprehensive sections, each addressing a key component of the machine learning process for classifying network traffic. For organizational purposes, the code for each section is contained within a single notebook.

- **Section 1: Data Exploration and Pre-processing:** Employ statistical and visualization techniques to understand the data. Normalize features and remove highly correlated ones to enhance model performance and address the curse of dimensionality.
    - Explore the Dataset
    - Clean the Dataset
    - Visualize the Data
        - Network Traffic Overview
        - Ground Truth (GT) Analysis
    - Feature Engineering
    - Data Scaling or Standardization
    - Correlation Analysis and Dimensionality Reduction
- **Section 2: Supervised Learning – Classification**: Involves training and evaluating various machine learning models for on the dataset to identify the most effective approach for detecting and classifying flows, including performance analysis and hyperparameter tuning.
- **Section 3: Unsupervised Learning – Clustering**: Explores clustering algorithms to identify patterns in network data, emphasizing the discovery of natural groupings without labeled data.
- **Section 4: Clusters Explainability and Analysis**: Analyzes the identified clusters, examining their features and distributions to draw conclusions about network traffic characteristics and cybersecurity threats.

1 Data exploration and pre-processing

1.1 Explore the Dataset

The initial and crucial step we undertook was a visual analysis of the database. Using a simple online viewer to open the .csv file allowed us to navigate more efficiently. In particular, this approach allowed us to quickly sort the database by clicking on the column headers of the features we wished to sort.

From this initial exploration, we immediately assigned a definition to the term "flow," understanding it as an exchange of data between two devices during which packets are transferred bidirectionally. A closer examination of the flow ID suggests that it is composed of a concatenation of Source IP, Source Port, Destination IP, Destination Port and Protocol as outlined in the design specifications provided by our professors.

Examining the features, it was noted that many were expanded during the data collection process, as indicated by labels with added prefixes such as total, average, standard deviation, maximum, and minimum. This organization suggests the presence of some correlation among these features.

In addition, we observed that some fields consistently had a value of 0, thus bringing no meaningful information to our analysis.

After this preliminary overview, we proceeded to load the database into our Jupyter environment for a more detailed examination.

1.2 Clean the Dataset

Flow ID issue
Once the dataset is loaded, we want to make sure that the flow ID is unique as we expect, so we count the occurrences of the flow ID.

Contrary to what we expected, the flow ID is not unique; performing further analysis presented on notebook 1, which showed a discrepancy in the structure of some flow IDs, we interpreted this error as a data quality problem and decided to delete the flow ID and recreate one with the same pattern: Source IP - Destination IP - Source Port - Destination Port - Protocol.

Even this was not enough to make the flow ID a true unique ID, so we added an additional feature: the timestamp. Now the flow ID is unique and can be set as an index of the dataframe.

No variance features
While exploring the database, we noticed that some features had a value of 0. Therefore, we decided to eliminate all features that do not add information and thus have zero variance.

1.3 Visualizations and statistical analysis

In this section, the goal is to visualize using various methodologies the data to highlight interesting patterns that may lead to the creation of new features and to get a general picture on the data

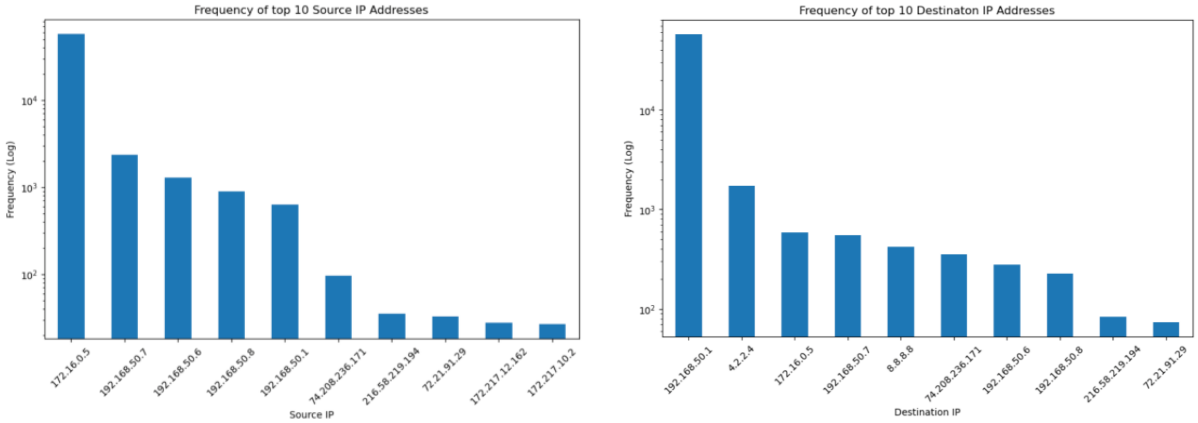
processed, flows, ports used, ip affected and more. The section is divided in turn into general traffic investigation and investigation with reference to ground truth.

Generic traffic level

In this part we analyze the generic network traffic without being aware of the label.

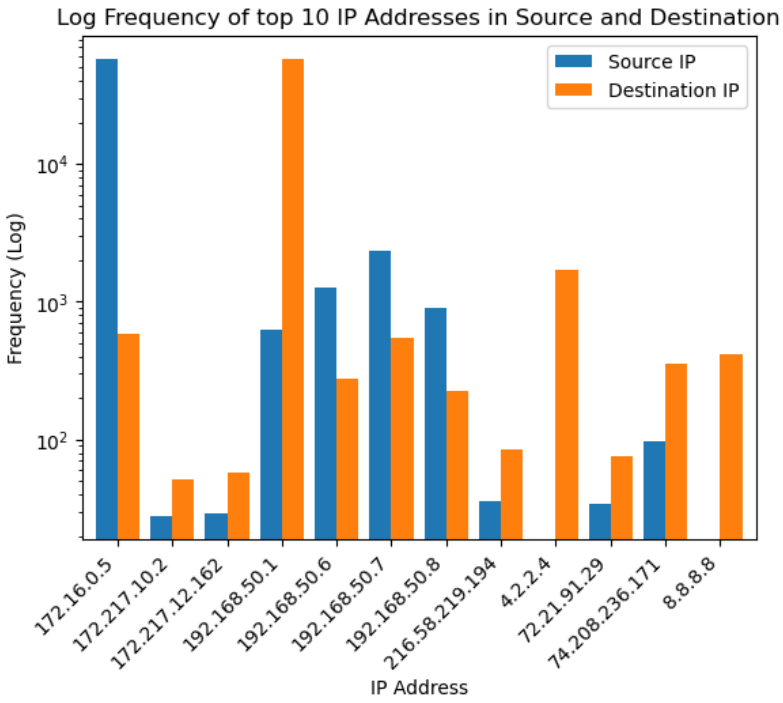
IP distribution

To accomplish this, we extract the most frequent source and destination IP addresses from our data.



Despite the y-axis being in a logarithmic scale, there are two particular addresses (one source and one destination) that appear with a significantly higher frequency compared to all others.

Additionally, it is noteworthy that many IP addresses appear both as destinations and sources. Therefore, we can consider merging the two graphs. To achieve this, we combine all the IP addresses into a single set and count the frequency of each address in both the destination IP and source IP fields.



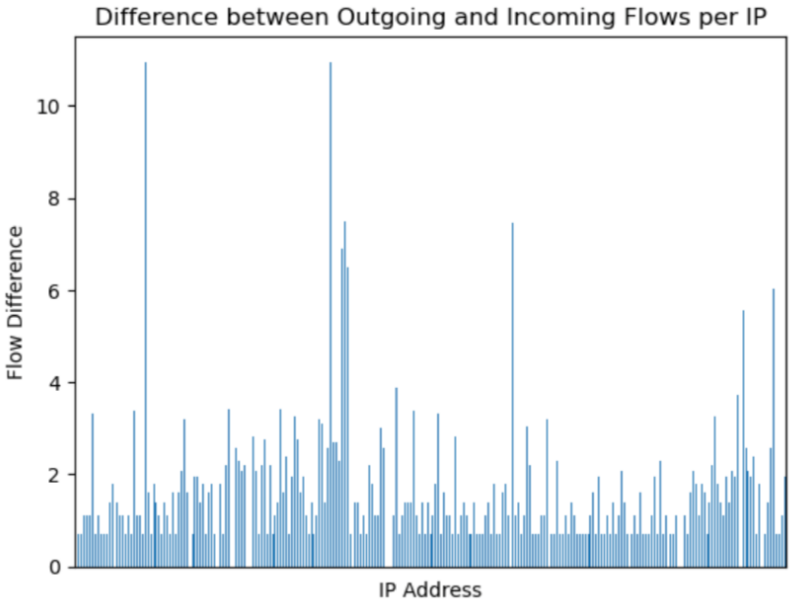
This graph is quite interesting as it highlights the presence of certain specific IP addresses such as 8.8.8.8. A high frequency of requests originating from specific IP addresses that appear both as

sources and destinations, but with significantly different frequencies, may suggest a botnet being used for a DDoS attack. However, some of these IP addresses might also be legitimate servers or network nodes with high traffic. For example, the address 8.8.8.8 appears in the flows only as a destination and never as a source. Despite the significant difference in frequency that might suggest the IP is being attacked, we know that 8.8.8.8 is Google's address. Therefore, it is normal for it to receive many requests but not send any.

Another interesting aspect highlighted by this graph concerns the two IP addresses with the highest frequency: 172.16.0.5 and 192.168.50.1. We observe a perfect symmetry: the source frequency of 172.16.0.5 is equal to the destination frequency of 192.168.50.1 and vice versa. This symmetry suggests that these two IP addresses might primarily communicate with each other.

Incoming and outgoing flows distribution

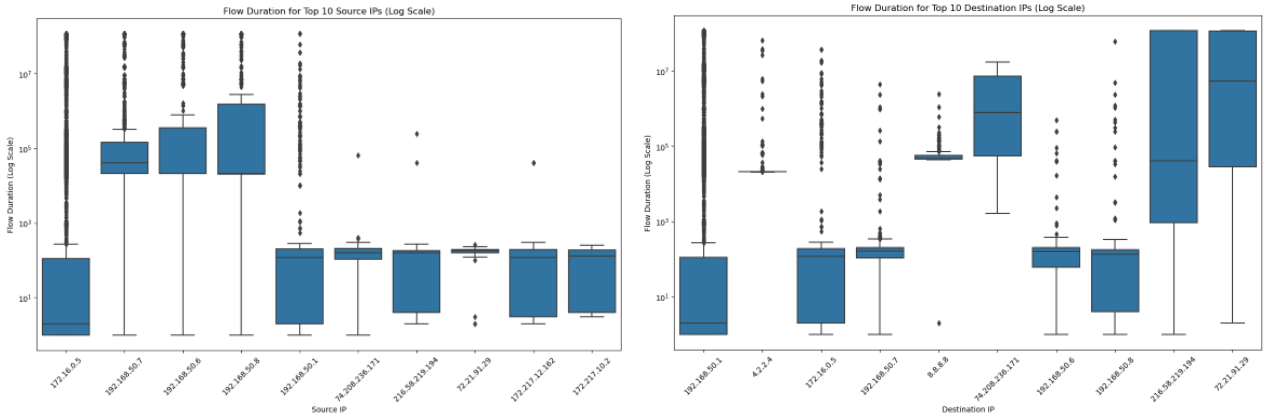
If we want to understand the potential variations in incoming and outgoing flows of an IP address, we can observe this graph.



Note: Our analysis will primarily focus on the 10 most significant IP addresses, as it is challenging to visualize a characteristic graph with a large amount of data.

Flow duration distribution of IPs

Let's shift our focus to the duration of the flow and see what insights can be derived.

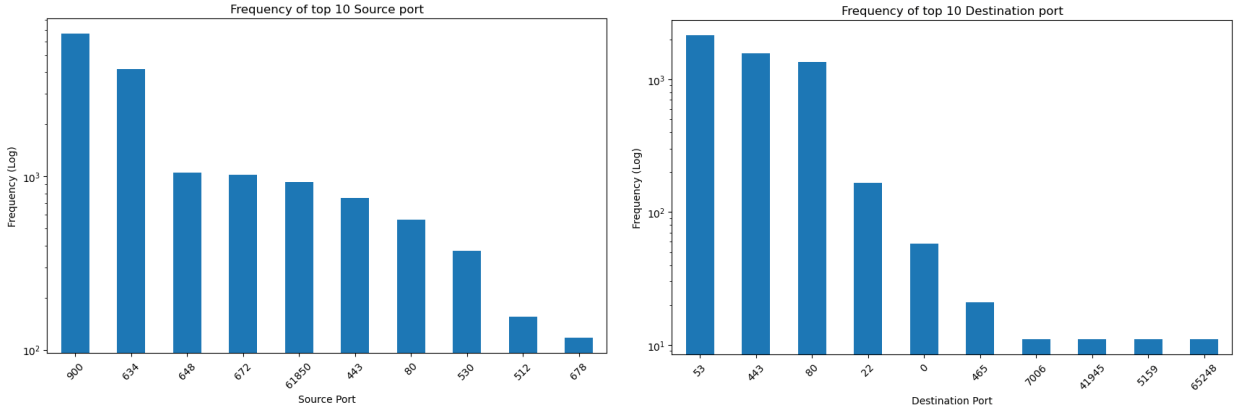


We observe a striking similarity when comparing 172.16.0.5 as a source to 192.168.50.1 as a destination. The same similarity is noted between other pairs of addresses, for example, 74.208.236.171 and 192.168.50.7.

However, if we examine the box plot of the flow duration for an IP address by looking at flows where it appears as a source and those where it appears as a destination, we do not see significant similarities. For instance, examining 192.168.70.7 first as a source and then as a destination shows a significantly different box plot.

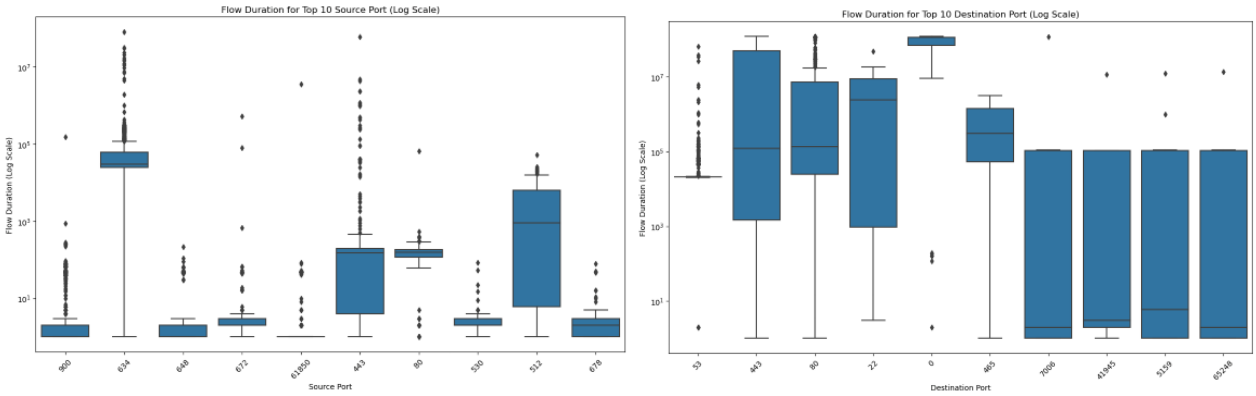
Port distribution

Another crucial aspect we wanted to analyze, given its importance in these types of attacks, is the distribution of ports.



We considered it appropriate to conduct further investigations on the ports, as they represent an interesting aspect. Therefore, similar to our analysis of IP addresses, we will analyze the flow duration for the ports.

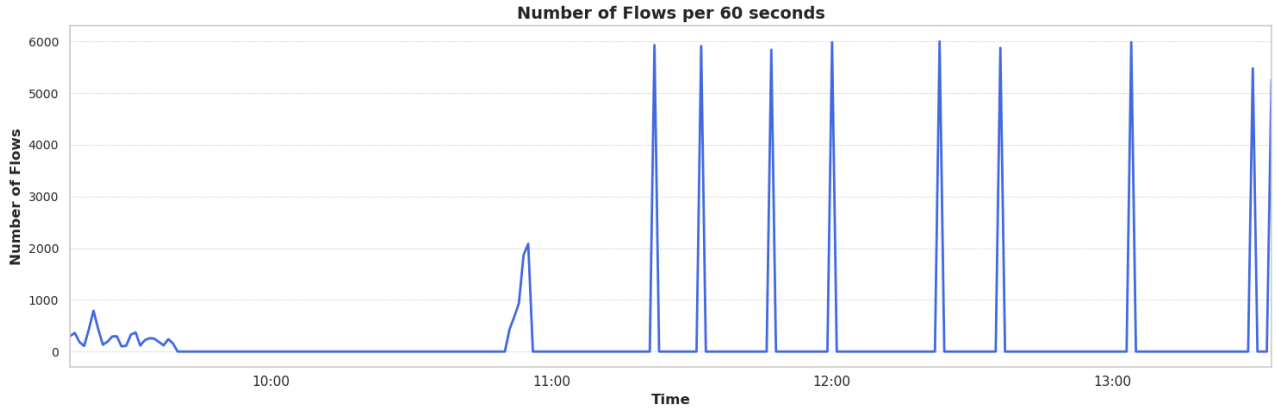
Flow duration distribution of ports



It is evident that source ports have less dispersion in flow duration compared to destination ports, indicating that the flow duration varies less for a particular source port.

Timestamp distribution

Given our domain knowledge and understanding that DDoS attacks are characterized by an enormous amount of data from different sources concentrated in a short time interval, we find it appropriate to thoroughly explore the temporal characteristics using timestamps:



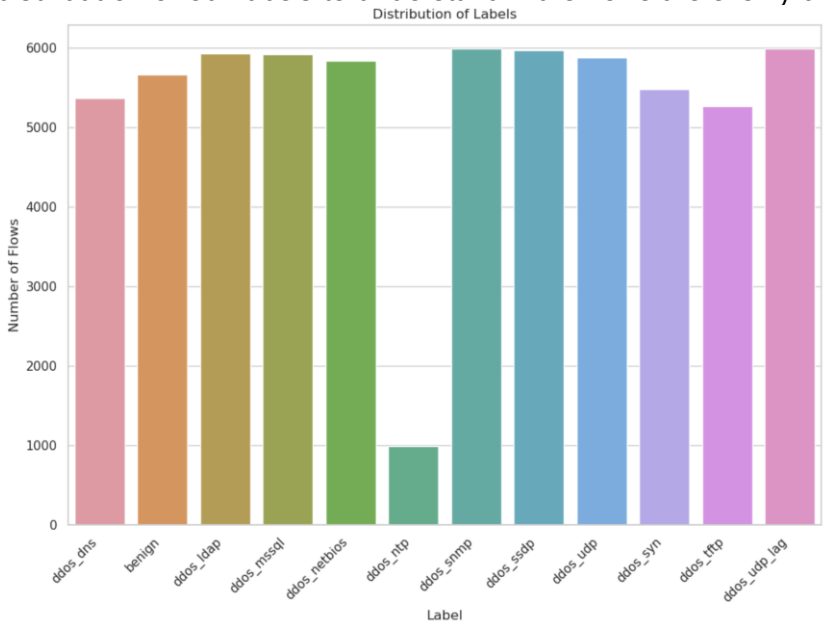
We consider this graph very interesting. In fact, having peaks where many flows occur in a short time can be an alarming signal. We will explore this characteristic further.

Ground truth level

We have obtained an overall view of our database and know what to expect. However, we have a crucial piece of information that allows us to conduct more in-depth analyses: the label ground truth. The objective of this second part of the section is to highlight additional characteristics that we can observe by using the label as a feature.

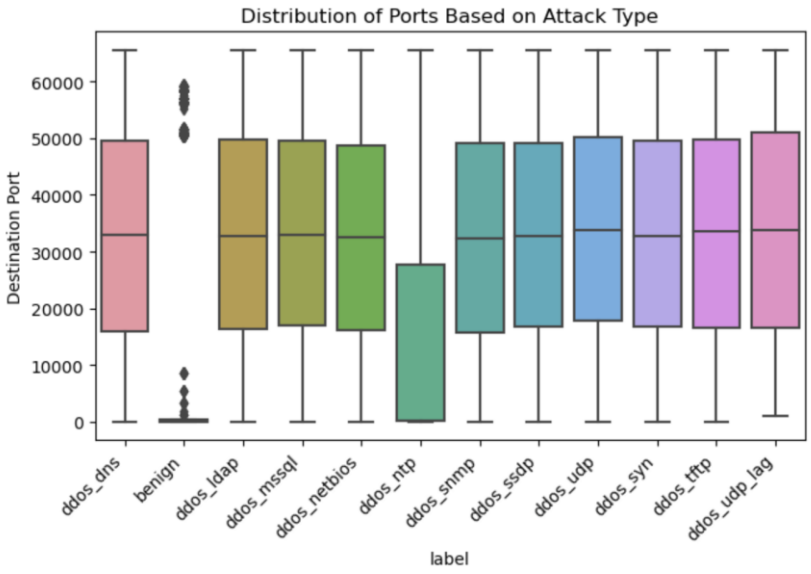
Label distribution

Let's analyze the distribution of our labels to understand if the flows are evenly distributed.

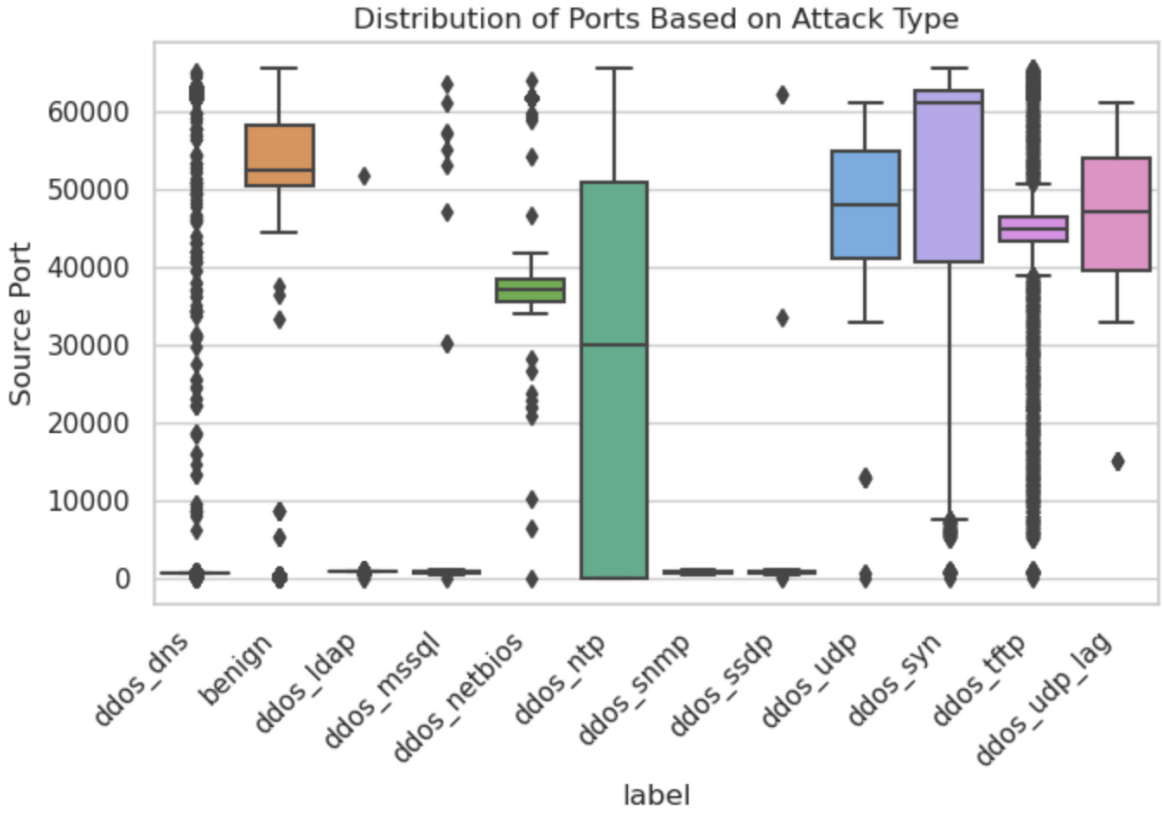


The dataset is almost homogeneous except for the ddos attack class ntp, which is underrepresented; this is no small detail and we should take it into account.

Port distribution



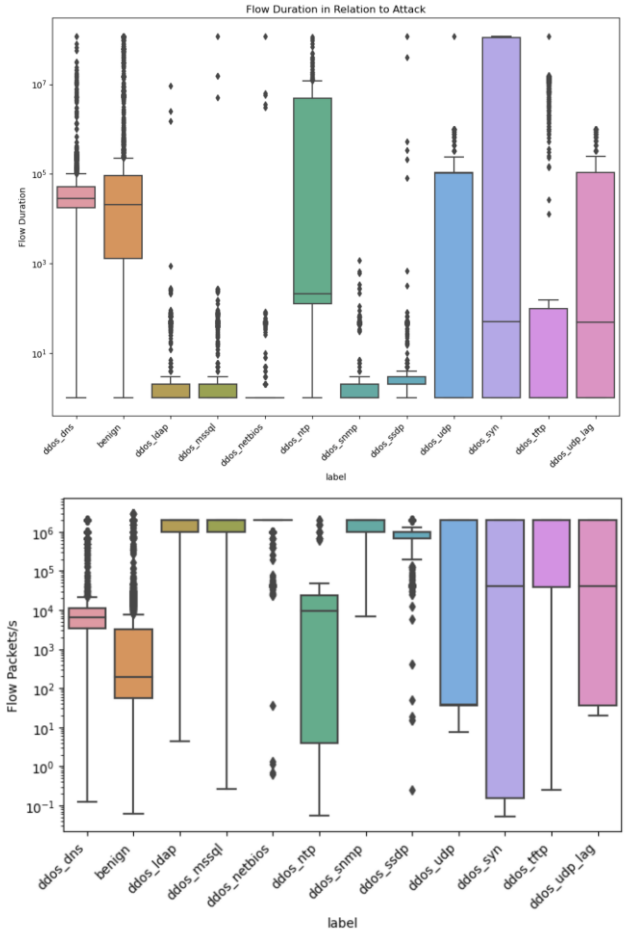
We find an interesting result: except for the ddos_ntp class, which we remember is underrepresented and could lead to misleading results, benign flows tend to use smaller ports, typically associated with basic protocols, while attack flows use larger and more unusual ports.



Regarding the ports used by the sources, we do not observe a linear trend as we do for the destinations. However, there are some common characteristics among certain attack types such as ddos_ldap, ddos_mssl, ddos_snmp,ddos_ssdp.

Flow duration distribution

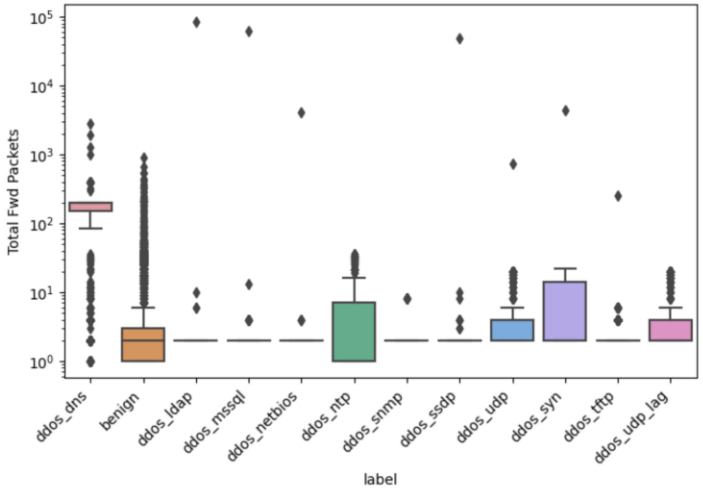
It is interesting to note the relationship between flow duration and packets per second; in fact, the two graphs appear to be mirror images of each other.



The flow duration provides an interesting perspective; not all labels assume the same value, although many share similar characteristics, such as ddos_ldap, ddos_mssl, ddos_snmp, and ddos_ssdp. However, it should be noted that the graph is on a logarithmic scale, and for certain labels like ddos_syn and ddos_ntp, there is a highly dispersed distribution.

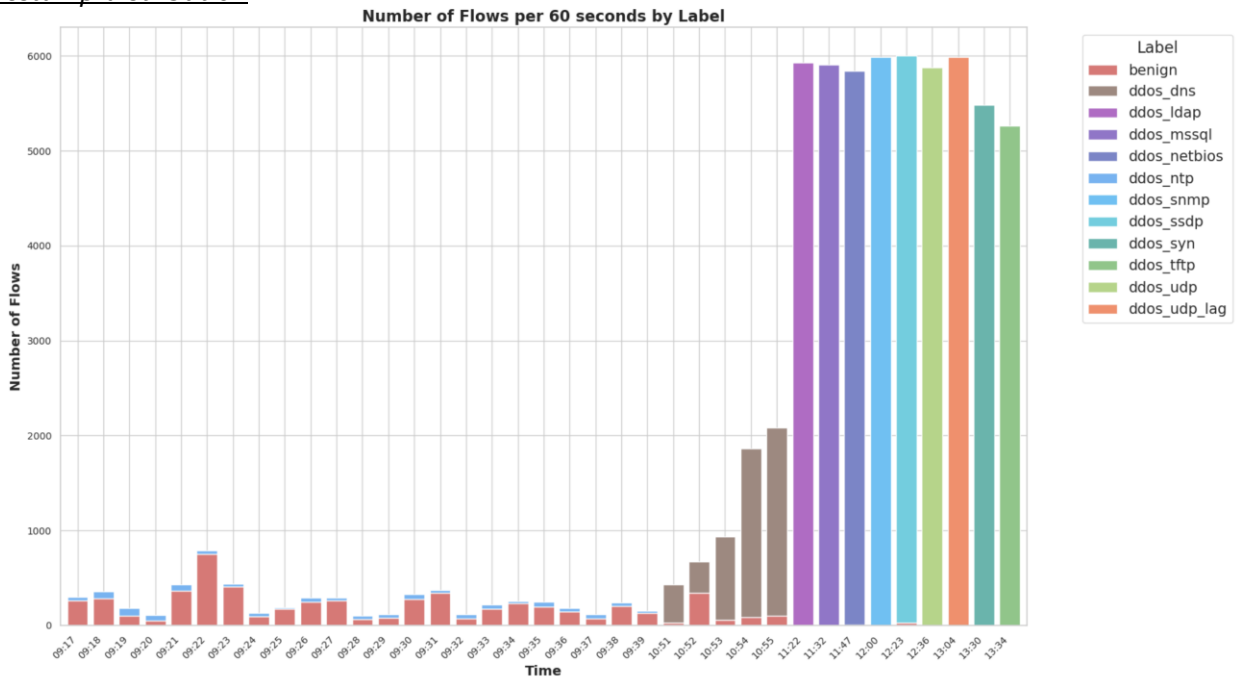
Distribution of Packets Forwarded per Flow

To overcome the limitations of the previous graphs, we will attempt to visualize the distribution of forwarded packets.



Here, we observe a very linear trend for all flows except for ddos_dns.

Timestamp distribution



This is a very important result. At first glance, it is evident that, with the exception of a few classes, the distribution of the others is concentrated in specific time slots.

Note: in the temporal distribution graph, we remember that for many time intervals there were no flows. In this latest graph with labels, these empty spaces have been removed because they hindered the readability of the graph itself. Indeed, various jumps between different dates can be noted.

At the beginning of our analysis, we were confident that the timestamp would provide us with crucial information, and it might have been so if not for the temporal distribution of our data.

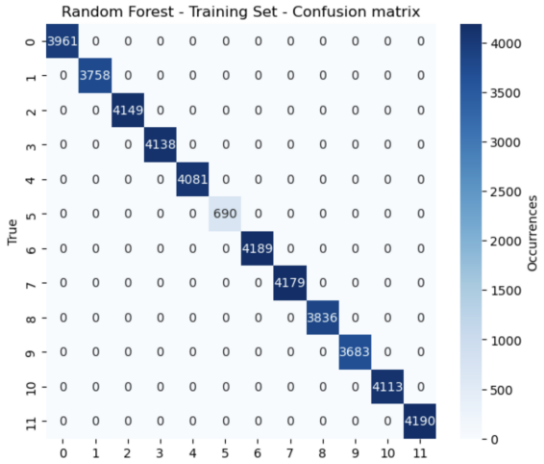
What we hoped to achieve with the timestamp was to calculate the intra-intervals between one flow and another, to understand how quickly the flows occurred.

Unfortunately for us, this is not possible. Looking at the graph, it is evident that for certain periods of time there is a different type of attack. Using this information, the model would risk overfitting, learning that a certain type of attack occurs at a specific time slot, which is not what we want to achieve. To demonstrate this, we illustrate the obtained results (the code of which is visible at the end of section 2) including the timestamp information.

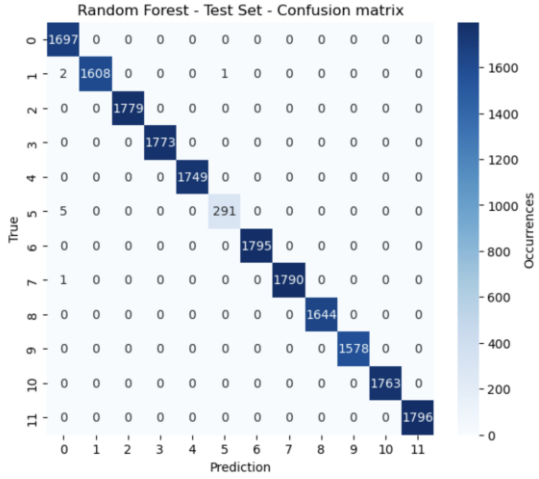
These are the lines used to add the features for the experiment:

```
ddos_data['Timestamp'] = pd.to_datetime(ddos_data['Timestamp'])
ddos_data['hours'] = ddos_data['Timestamp'].dt.hour
ddos_data['minutes'] = ddos_data['Timestamp'].dt.minute
ddos_data['seconds'] = ddos_data['Timestamp'].dt.second
ddos_data['milliseconds'] = ddos_data['Timestamp'].dt.microsecond // 1000
```

Below, we present the results obtained by training the database with a random forest after adding these temporal features.



	precision	recall	f1-score	support
0	1.00	1.00	1.00	3961
1	1.00	1.00	1.00	3758
2	1.00	1.00	1.00	4149
3	1.00	1.00	1.00	4138
4	1.00	1.00	1.00	4081
5	1.00	1.00	1.00	690
6	1.00	1.00	1.00	4189
7	1.00	1.00	1.00	4179
8	1.00	1.00	1.00	3836
9	1.00	1.00	1.00	3683
10	1.00	1.00	1.00	4113
11	1.00	1.00	1.00	4190
accuracy			1.00	44967
macro avg	1.00	1.00	1.00	44967
weighted avg	1.00	1.00	1.00	44967



	precision	recall	f1-score	support
0	1.00	1.00	1.00	1697
1	1.00	1.00	1.00	1611
2	1.00	1.00	1.00	1779
3	1.00	1.00	1.00	1773
4	1.00	1.00	1.00	1749
5	1.00	0.98	0.99	296
6	1.00	1.00	1.00	1795
7	1.00	1.00	1.00	1791
8	1.00	1.00	1.00	1644
9	1.00	1.00	1.00	1578
10	1.00	1.00	1.00	1763
11	1.00	1.00	1.00	1796
accuracy			1.00	19272
macro avg	1.00	1.00	1.00	19272
weighted avg	1.00	1.00	1.00	19272

As we anticipated, we encounter a situation of overfitting. As mentioned earlier, the reason for this lies in the fact that the attacks occur only within a specific time interval.

1.4 Feature Engineering

Following the exploration and visualization of the database conducted in the previous sections, we decided to expand some features within our database.

We have decided to remove the features of source IP and destination IP. This decision was based on the need to avoid overfitting. In fact, if a new IP were to start sending malicious traffic, it would likely be classified as benign because it is an IP that has never been seen before.

We have recognized the importance of ports, and since they are numerical features without an inherent order of magnitude, we need to decide on the appropriate technique to incorporate these features, considering that everything will need to be scaled. After careful analysis, we have decided to follow the frequency-based feature strategy, which involves using the frequency or distribution of the characteristics to transform categorical or numerical variables into a format that can be effectively used in machine learning models.

For all the reasons previously examined, we did not include the temporal features.

Additionally, we deemed it important to highlight the protocol used. Therefore, utilizing one-hot encoding, we introduced features named Protocol 0, Protocol 6, Protocol 16, which are assigned a value of 0 if absent, and a value of 1 if present.

1.5 Data Scaling or Standardization

Scaling or standardizing data is a fundamental preprocessing step that benefits various machine learning models by ensuring that each feature contributes equally to the analysis, regardless of their original scale. This not only helps in the application of PCA but also enhances the overall performance and accuracy of many machine learning algorithms by providing a uniform scale for all features, thus facilitating a more balanced and fair learning process.

1.6 Correlation Analysis and Dimensionality Reduction

In this section, we have decided to utilize correlation analysis, Principal Component Analysis (PCA), and loading scores to optimize the feature set of our dataset. These tools allow us to reduce the dimensionality of the data, eliminate redundancies, and retain the most relevant information, thereby enhancing the efficiency and interpretability of our model.

Correlation analysis was the first step in our process. This technique enabled us to identify the linear relationships between the different features of the dataset. We calculated the correlation matrix, which measures the strength and direction of the relationship between each pair of variables. This helped us identify highly correlated features. Variables with high correlation (greater than 0.9 or less than -0.9) are considered redundant, as they provide similar information. By eliminating these redundant features, we reduce the complexity of the model without losing significant information, thereby improving its ability to generalize to new data.

Subsequently, we applied Principal Component Analysis (PCA). PCA is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated variables called principal components. These components are ordered so that the first captures the maximum possible variance in the data, followed by subsequent components that capture the remaining variance in decreasing order. Before applying PCA, as mentioned in the previous paragraph, we standardized the features to ensure they all have zero mean and unit variance, as PCA is sensitive to the scale of the variables.

Once the Principal Component Analysis (PCA) was performed, we examined the loading scores of the first three principal components, which explain 50% of the cumulative variance. The loading scores represent the contribution of each original variable to the principal components, allowing us to assess the impact of individual features. In other words, they indicate how much each original variable influences a particular principal component.

By cross-referencing the data obtained from the correlation analysis with those derived from the PCA loading scores, we assessed all the most redundant features for elimination.

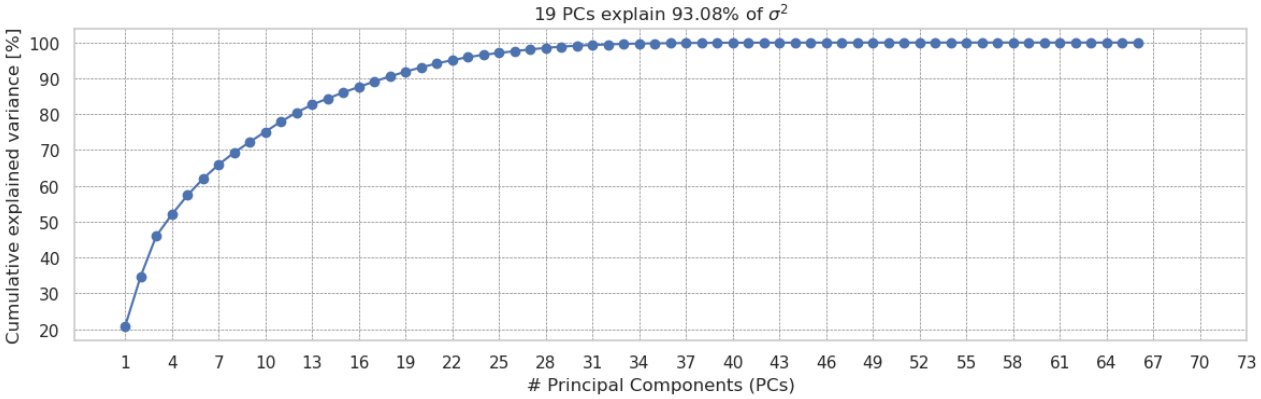
Correlation analysis

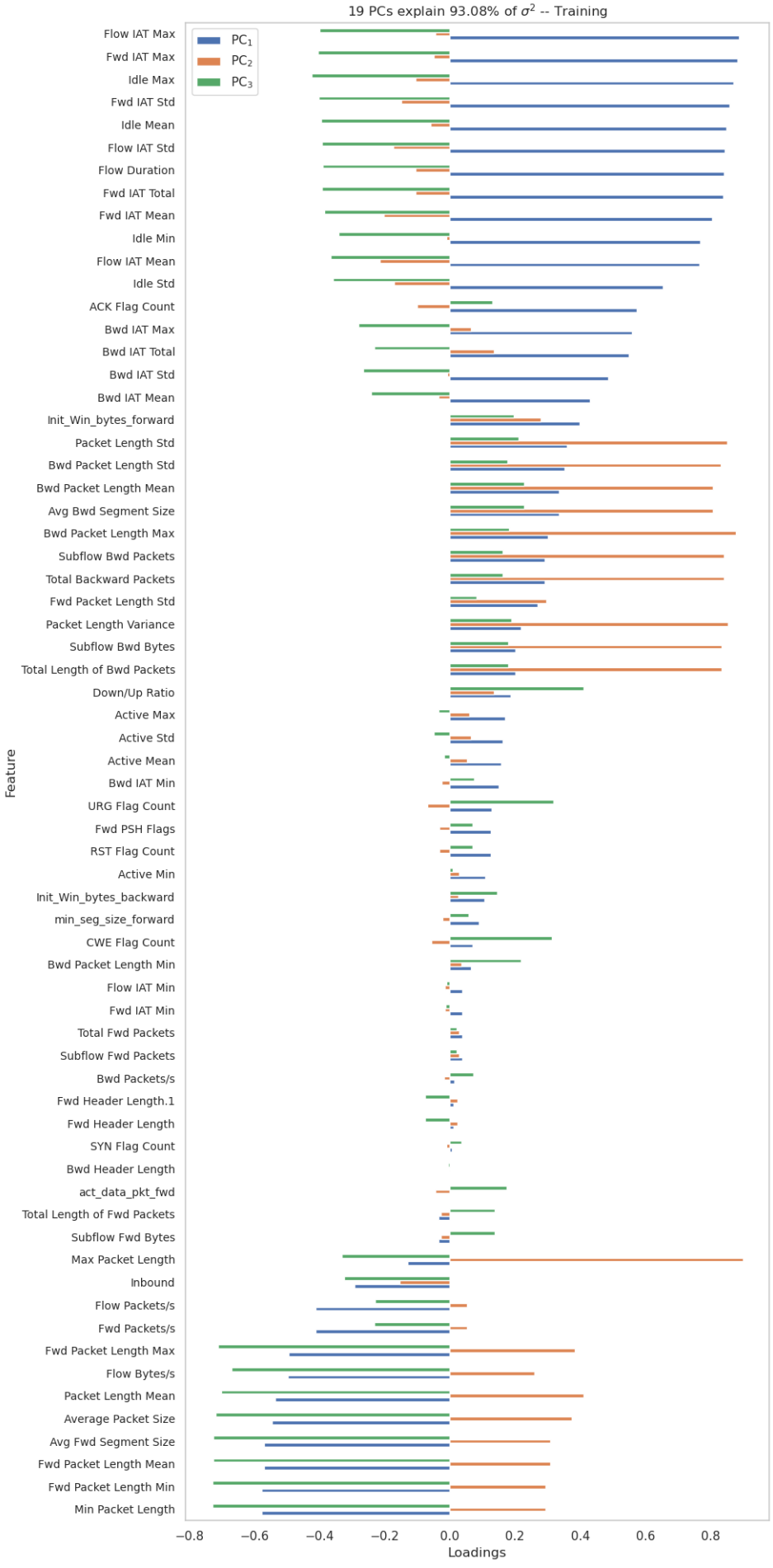
We selected and removed the most correlated features using a correlation threshold of 0.85: for each pair of features with a correlation higher than this value, we eliminated one of the two features, making sure not to remove the same feature multiple times in case it was highly correlated with multiple other features. This process resulted in the selection of a unique set of highly correlated features, while retaining those with more independent information in our dataset.

34 features to be removed

['Active Max', 'Active Mean', 'Average Packet Size', 'Avg Bwd Segment Size', 'Avg Fwd Segment Size', 'Bwd IAT Std', 'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Duration', 'Flow IAT Max', 'Flow IAT Min', 'Flow IAT Std', 'Fwd Header Length', 'Fwd IAT Max', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Total', 'Fwd Packet Length Mean', 'Fwd Packet Length Min', 'Fwd Packets/s', 'Idle Max', 'Idle Mean', 'Min Packet Length', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance', 'RST Flag Count', 'Subflow Bwd Bytes', 'Subflow Bwd Packets', 'Subflow Fwd Bytes', 'Subflow Fwd Packets', 'Total Backward Packets', 'Total Length of Bwd Packets', 'Total Length of Fwd Packets']

Principal Component Analysis




19 PCs explain 93.08% of σ^2 -- Training
PC$_1$
PC$_2$
PC$_3$
Flow IAT Max
Fwd IAT Max
Idle Max
Fwd IAT Std
Idle Mean
Flow IAT Std
Flow Duration
Fwd IAT Total
Fwd IAT Mean
Idle Min
Flow IAT Mean
Idle Std
ACK Flag Count
Bwd IAT Max
Bwd IAT Total
Bwd IAT Std
Bwd IAT Mean
Init_Win_bytes_forward
Packet Length Std
Bwd Packet Length Std
Bwd Packet Length Mean
Avg Bwd Segment Size
Bwd Packet Length Max
Subflow Bwd Packets
Total Backward Packets
Fwd Packet Length Std
Packet Length Variance
Subflow Bwd Bytes
Total Length of Bwd Packets
Down/Up Ratio
Active Max
Active Std
Active Mean
Bwd IAT Min
URG Flag Count
Fwd PSH Flags
RST Flag Count
Active Min
Init_Win_bytes_backward
min_seg_size_forward
CWE Flag Count
Bwd Packet Length Min
Flow IAT Min
Fwd IAT Min
Total Fwd Packets
Subflow Fwd Packets
Bwd Packets/s
Fwd Header Length.1
Fwd Header Length
SYN Flag Count
Bwd Header Length
act_data_pkt_fwd
Total Length of Fwd Packets
Subflow Fwd Bytes
Max Packet Length
Inbound
Flow Packets/s
Fwd Packets/s
Fwd Packet Length Max
Flow Bytes/s
Packet Length Mean
Average Packet Size
Avg Fwd Segment Size
Fwd Packet Length Mean
Fwd Packet Length Min
Min Packet Length
Feature
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
Loadings

Incrociando manualmente I risultati della analisi delle correlazioni e i risultati della loading scores, abbiamo individuato le seguenti featuers da rimuovere:

36 features to be removed

['Active Max', 'Active Min', 'Average Packet Size', 'Avg Bwd Segment Size', 'Avg Fwd Segment Size', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Duration', 'Flow IAT Max', 'Flow IAT Min', 'Flow IAT Std', 'Flow Packets/s', 'Fwd Header Length', 'Fwd IAT Max', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Total', 'Fwd Packet Length Mean', 'Fwd Packet Length Min', 'Idle Max', 'Idle Mean', 'Min Packet Length', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance', 'Protocol 17', 'Protocol 6', 'Subflow Bwd Bytes', 'Subflow Bwd Packets', 'Subflow Fwd Bytes', 'Subflow Fwd Packets', 'Total Backward Packets', 'Total Length of Bwd Packets', 'Total Length of Fwd Packets']