DDoS Attacks
Detection and Characterization

Francesco Rosucci Giorgio Fardo Giuseppe Fracassi
Dario Pinto

February 18, 2024

Distributed denial-of-service (DDoS) attacks are a serious threat to the security of computer networks. These attacks can overwhelm a network with traffic or requests, making services unavailable to legitimate users.

1 Objectives

The objective of this project is to develop a machine learning model for the detection and classification of distributed denial-of-service (DDoS) attacks. The model will be trained on a dataset containing samples of real-world packet flows, with some representing benign traffic while others representing various types of DDoS attacks. So this model has to:

- Accurately and automatically detect DDoS attacks
- Classify DDoS attacks into different types

To achieve these objectives, the following steps have been performed:

- **Data analysis and Data preprocessing:** The data have been analyzed using various visualization techniques. Features have been normalized and highly correlated ones have been removed to improve performances and reduce the curse of dimensionality.
- **Model selection:** The dataset have been used to train different machine learning models in order to find the one that best suits our goal.
- **Model evaluation:** The model that performed the best has been evaluated on a portion of the dataset not used for training. The evaluation results have been used to determine the accuracy and performance of the model.
- **Clustering:** Clustering techniques have been used to assess similarities between different types of attacks and to determine whether or not benign traffic has similar characteristics to malicious traffic.

2 Data exploration and visualization

The dataset used in this study comprises 64,239 records and 86 features. It includes 12 labels representing various attack types and one benign label. The distribution of records among these labels is as shown in Table 1. This dataset was generated from packet captures of traffic between 257 unique IP addresses. Of these, 216 IP addresses generate flows, while the remaining act as receivers. Some source IPs also serve as destinations.

label	data points
benign	5658
ddos_dns	5369
ddos_lap	5928
ddos_mssql	5911
ddos_netbios	5830
ddos_ntp	986
ddos_snmp	5984
ddos_ssdp	5970
ddos_syn	5480
ddos_tftp	5261
ddos_udp	5876
ddos_udp_lag	5986

Table 1: Number of data points for each label

Firstly, the entire dataset has been visualised using various types of plots to highlights patterns, outliers, characteristics and to understand how the data have been collected and organized in the dataset.
Each kind of plot were used to highlight specific properties of the data:

- **Histograms** were used to visualize the distribution of some features.
- **Bar charts** were used to compare the distribution of features between different attack types. This helped to identify differences between the various types of attacks and the benign traffic.
- **Box plots** were used to visualize and compare the distribution of features like inter-arrival times or packets length for each type of attack and for benign traffic in order to accentuate potential differences
- **Heatmaps** were used to visualize the correlation between different features.
- **ECDF** were used to visualize and compare the distribution of some features from a different point of view with respect to histograms.

This analysis brought up some interesting information:

- A bar chart of flow duration and the total number of packets revealed that for DDoS DNS attacks, even though the duration of the flows, on average, is smaller than that of benign flows, the total number of packets is an order of magnitude higher. Moreover, DDoS NetBIOS and DDoS SNMP are characterized by a very low number of total packets, and the latter also has a flow duration several orders of magnitude lower than other types of traffic.

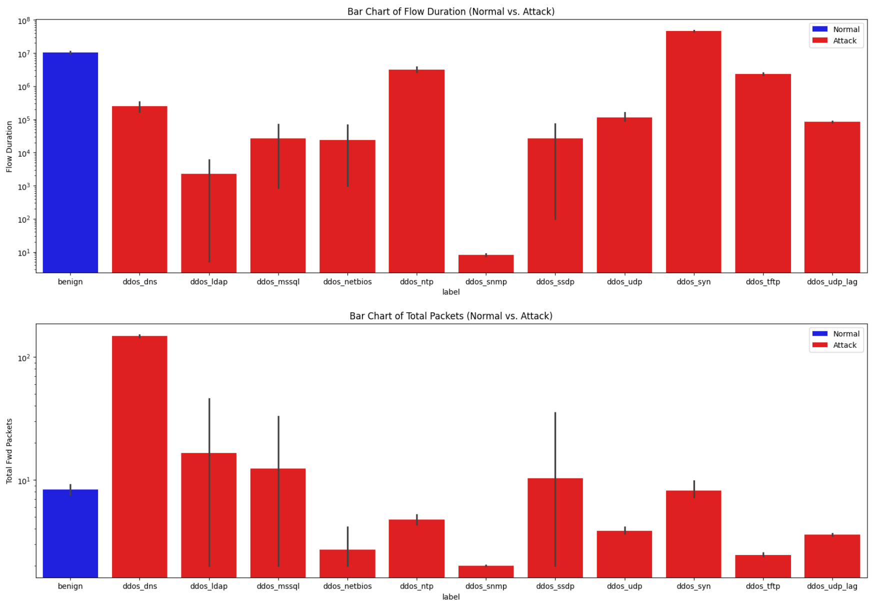

Figure 1

- A bar chart of the average inter-arrival times highlighted that IAT may vary a lot and are not necessarily an indicator of ddos attack since sometimes packets have an higher IAT than the ones belonging to benign flows. There are some exceptions as in ddos_snmp where we can see how this type of attack is characterized by an inter-arrival time of the packets several order of magnitude lower than benign traffic.

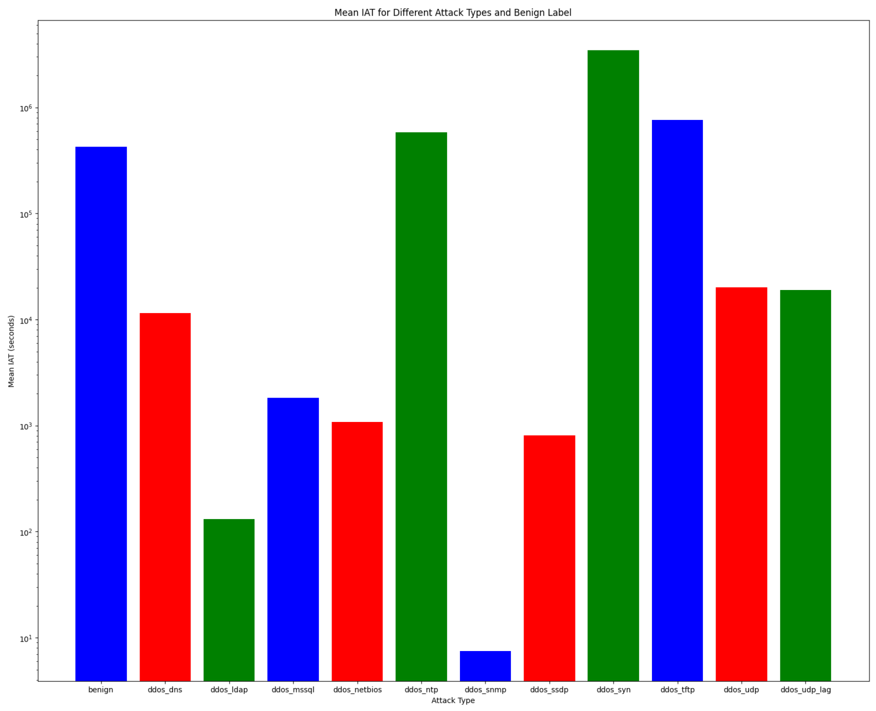

Figure 2

2.1 Possible dataset error: mislabeled attack

While analyzing the dataset, we identified a set of fields indicating the flags set in various TCP flows. Upon comparing these flags with the malicious labels in the dataset, we discovered that the flows categorized as ”ddos_syn” lacked the expected SYN flag. Instead, they possessed the ACK flag. This indicates ACK floods or attacks utilizing the ACK flag rather than ”ddos_syn”. Proceeding with the analysis we did not change the labels, as asked by the professor. This will impact the analysis only logically in the outcome of the supervised part of the analysis. Meaning that the resulting attacks detected as ddos_syn should be ddos_ack. Here follows the graph representing the flags set for the labeled attacks.

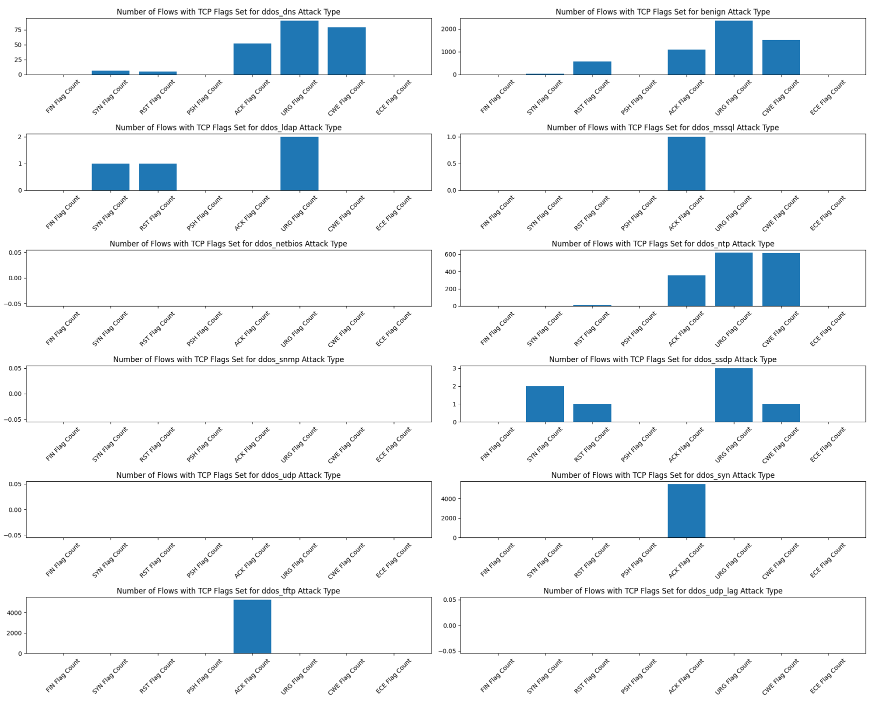

Figure 3: Flags data for each kind of label

3 Data preprocessing

Before using the dataset to train the machine learning models, we applied some preprocessing techniques.

3.1 Data transformation and correlation analysis

The first step was to add new features that could help us achieve our goal. The new features included aggregating existing features such as performing min, max, mean, and standard deviation of source and destination ports and flow durations.

Secondly, we transformed the categorical features, such as timestamp, source and destination IP, into numerical ones in order to perform correlation analysis. This will help us especially in the clustering as we only have to work with numerical data, greatly diminishing the difficulty for the application of some clustering techniques.

Finally, we dropped the labels and the Flow-ID feature since it was composed by already present features. Subsequently, we conducted correlation analysis using a correlation matrix (with Pearson correlation) and heatmap.

In Figure 4 we can see the correlation matrix before any preprocessing. In Figure 5 we have the correlation matrix after the manual feature selection and in Figure 6 we have the correlation matrix after the PCA.

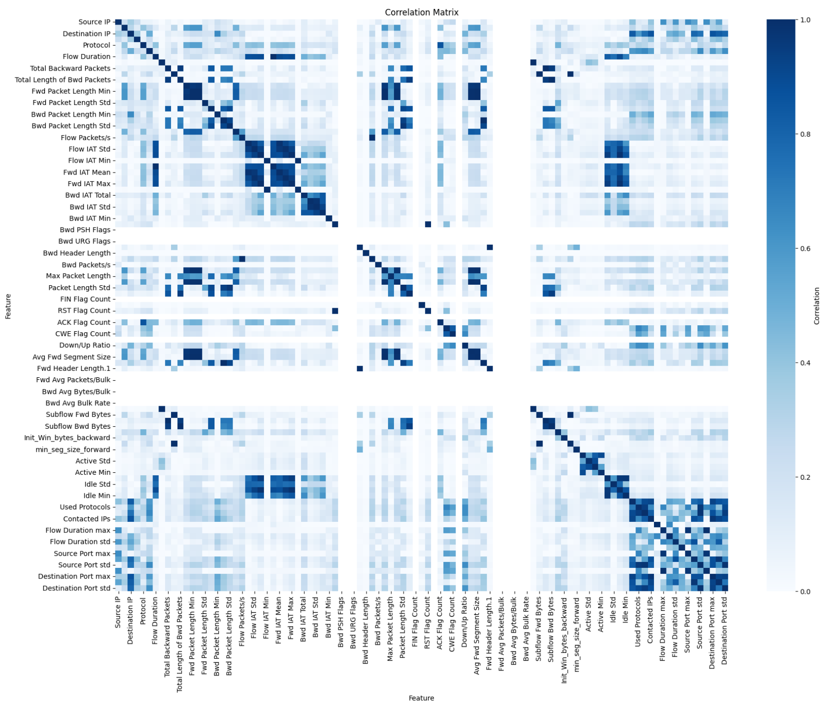

Figure 4: Correlation before dimensionality reduction

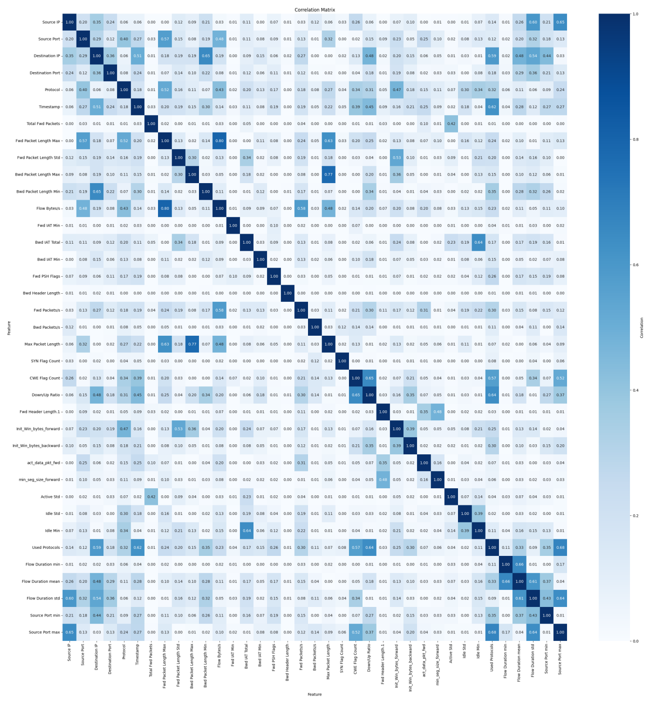
Figure 5: Correlation after manual dimensionality reduction

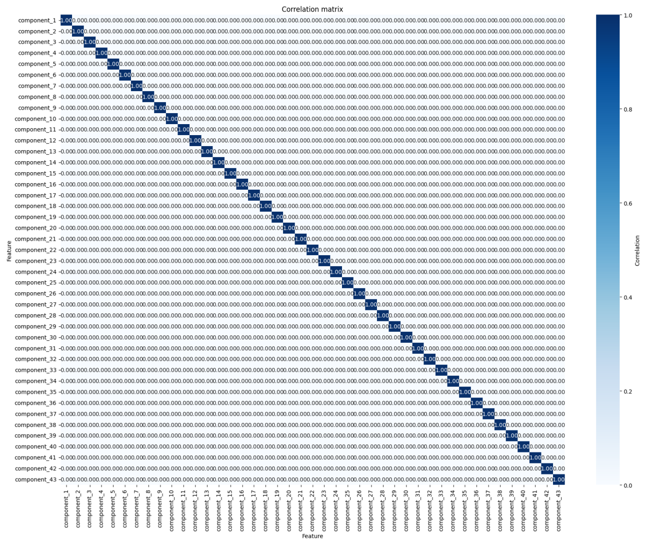
Figure 6: Correaltion after PCA

3.2 Features normalization

After conducting the correlation analysis, we proceeded with the normalization of the features through standardization. This was done to ensure that all features had the same weight by having a consistent scale before applying PCA to reduce the number of features.

3.3 Dimensionality reduction

To simplify the dataset and minimize correlations between features, we employed principal component analysis (PCA). PCA is a statistical technique that can be used to reduce the number of features in a dataset while retaining the most relevant information. Our PCA analysis revealed that the first 43 principal components captured 99.71% of the dataset's variance. This implies that these 43 principal components encapsulate all the necessary information for identifying and classifying DDoS attacks.

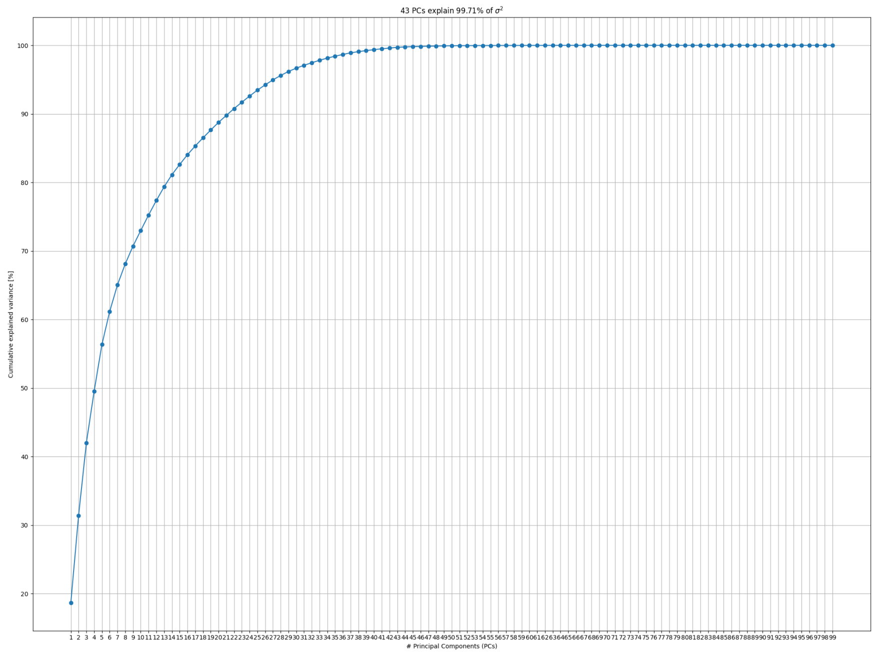

Figure 7: Cumulative variance

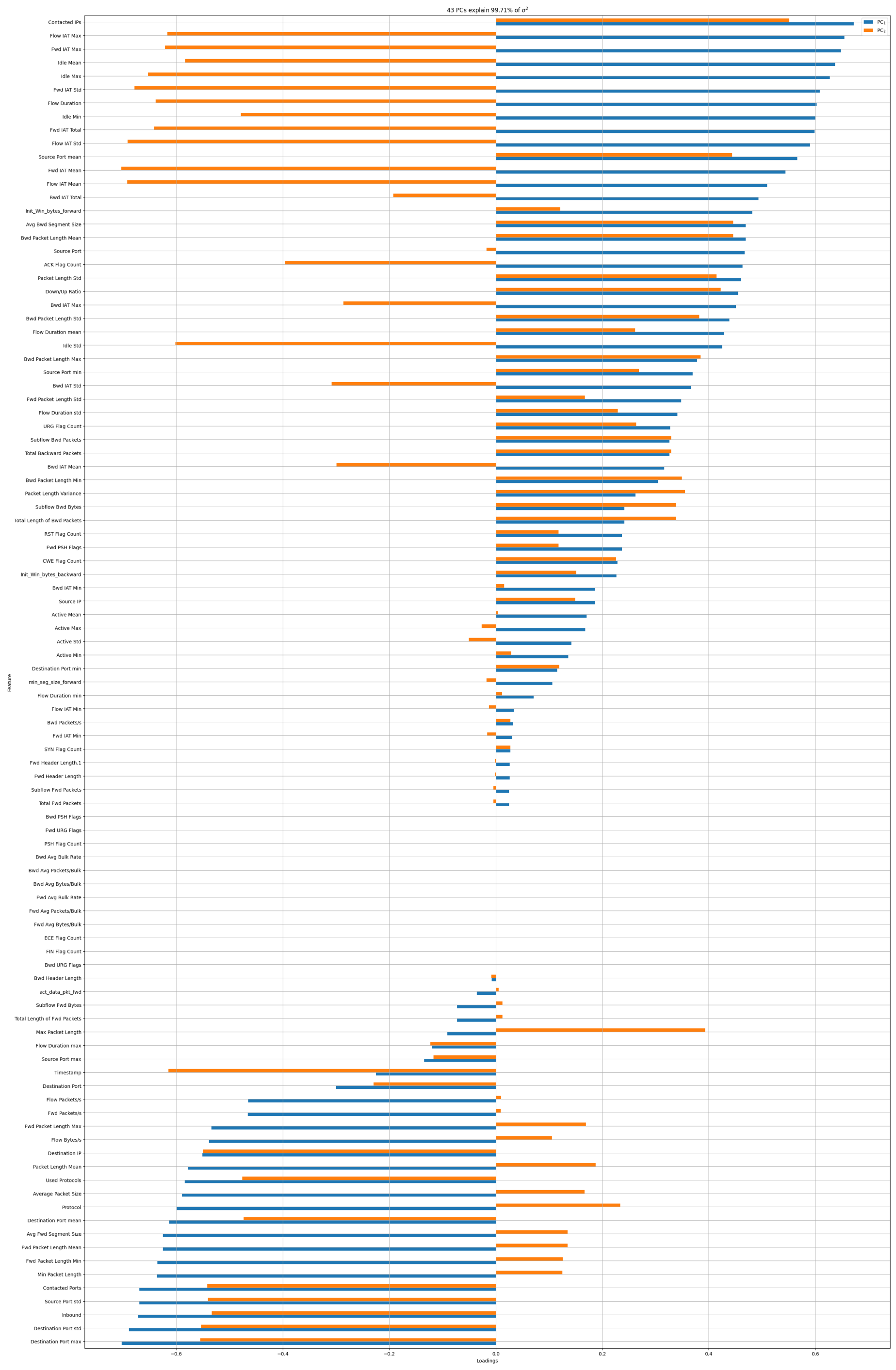

Figure 8: Loading scores

Figure 8 illustrates the contribution of each feature to the corresponding first component of PCA.

The PCA proved to be a valuable tool for reducing the dimensionality of the data and identifying the most important features for analysis and modeling. However, to provide an additional point of comparison, we also conducted dimensionality reduction while retaining the original features of the dataset. We iteratively identified and removed highly correlated features until we eliminated all those with a correlation coefficient greater than 0.8 .

4 Supervised learning

In the realm of machine learning, supervised learning stands as a pivotal approach that entails training a model on a labeled dataset, where the algorithm is guided by known outcomes. In this report, we present the results obtained from two different types of machine learning algorithms.

4.1 Binary

With the binary approach, our goal is to simply distinguish the data as either benign or malicious, without identifying the specific type of DDoS attack. To achieve this, we transformed the original labels, assigning '1' to malicious traffic and '0' to benign traffic.
We divided the dataset into training and validation sets and employed various models to compare their performances on this data split.

4.1.1 Logistic regression

Logistic regression is a relatively straightforward algorithm that employs a model consisting of linear hypothesis maps. The results revealed that logistic regression achieved an accuracy of 99.9% on the validation set, with an f1-score of 1.00 for each class.
The corresponding confusion matrix illustrated in figure 9.

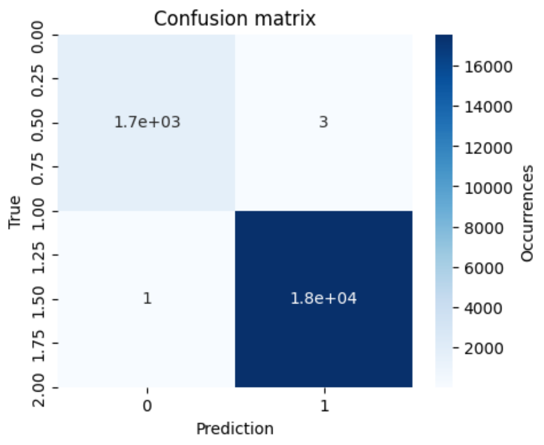

Figure 9: Logistic Regression confusion matrix

4.1.2 Binary KNN

K-nearest neighbors (KNN) is another relatively simple algorithm with the peculiarity of not using a loss function and not requiring training. This algorithm can be employed in either a multi-label classification setting or a binary one.
In this specific split, it achieved an accuracy of 99.97% on the validation set, with an f1-score of 1.00 for each class.
The corresponding confusion matrix is as follows:

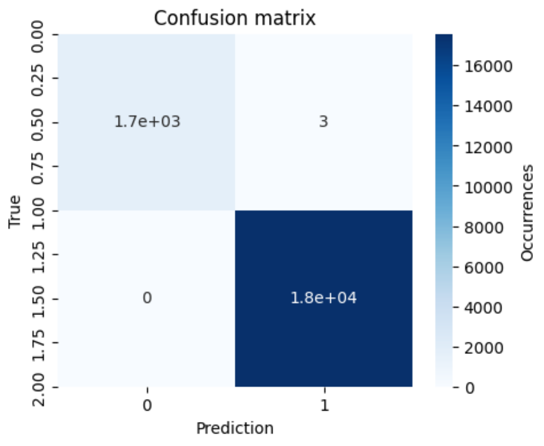

Figure 10: Binary KNN confusion matrix

4.1.3 Gaussian Naive Bayes (GNB)

Gaussian Naive Bayes is a probabilistic classifier that assumes features are independent. The results showed that GNB achieved an accuracy of 96.58% on the validation set. This is lower compared to the accuracy of the previous algorithm used, suggesting that the assumption of independence between features may not always be valid for DDoS attack detection. As depicted in Figure 11, another issue is the higher number of false benign instances in comparison with the other algorithms.

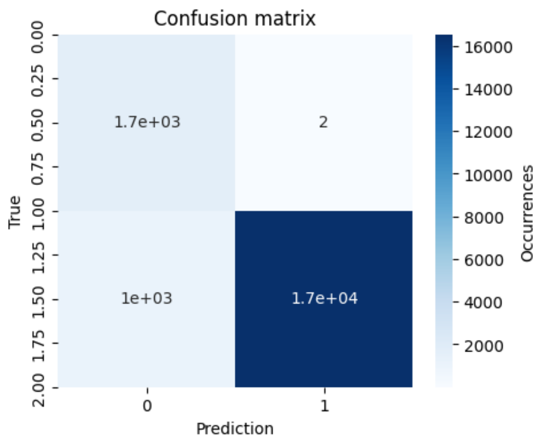

Figure 11: GNB confusion matrix

4.2 Multi-label

With the multi-label approach, our objective is to identify the various types of DDoS attacks. The dataset consists of 12 different labels, with one for benign traffic and eleven for different types of DDoS attacks. Once again, we divided the dataset into training and validation sets and utilized various models to compare their performances on this split of the data.

4.2.1 KNN

This time, the KNN algorithm is employed as a multi-label classifier and achieved an accuracy of 99.0%, with an f1-score close to 1 for all classes. The lowest f1-score is 0.95 for 'ddos_ntp,' which also has the lowest support (only 296 instances compared to more than 1600 for the other classes).
The corresponding confusion matrix is as follows:

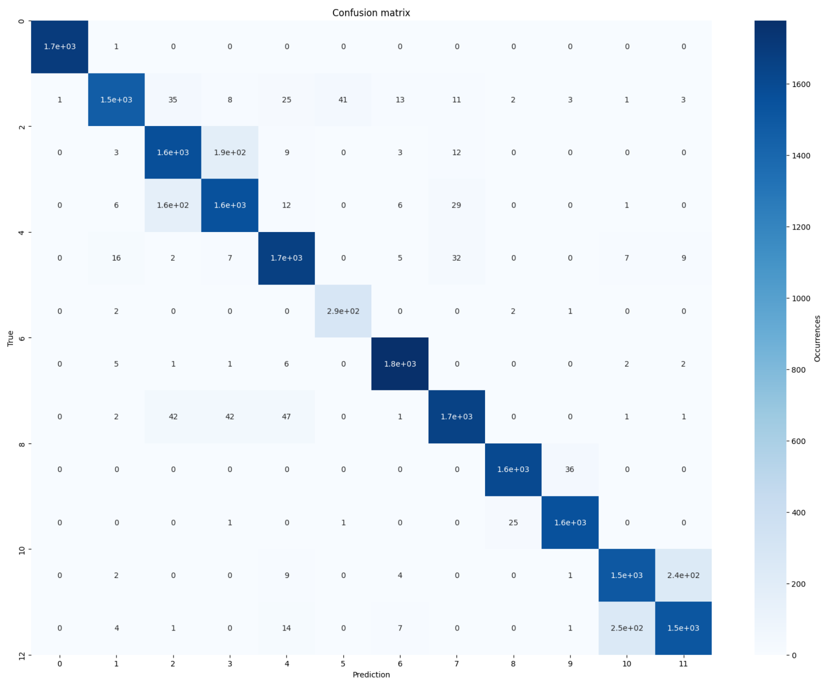

Figure 12: KNN confusion matrix

4.2.2 Decision tree

Decision trees are tree-based classifiers that classify data points by recursively splitting the feature space based on specific thresholds. The results showed that decision tree achieved an accuracy of 99.5% on the validation set.
The corresponding confusion matrix is as follows:

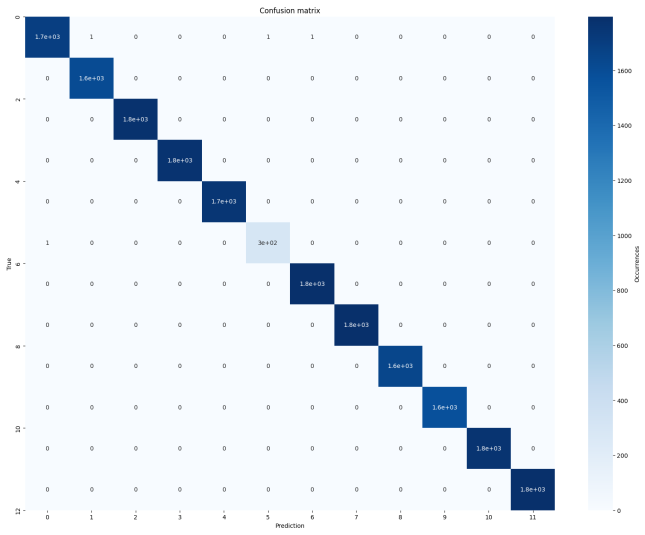

Figure 13: Decision tree confusion matrix

4.2.3 Random forest

Random forests are ensemble learning algorithms that combine multiple decision trees to enhance the overall predictive performance. The results demonstrated that random forests achieved an accuracy of 99.73% on this specific validation set. This high accuracy suggests that random forests are a promising approach for DDoS attack detection.
The corresponding confusion matrix is as follows:

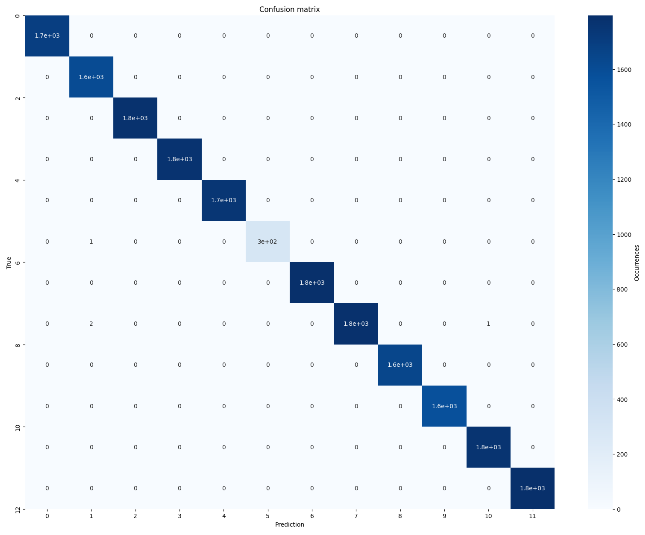

Figure 14: Random forest confusion matrix

4.3 Observations

This initial analysis was conducted without utilizing more advanced techniques, such as k-fold cross-validation. The objective was to gain a preliminary view of the performance through a random stratified split, providing a starting point for selecting the most suitable algorithm for our goal.

Possible observations on the possibility to do one shot detection only on as binary in the first place to further speed up inline detection, and then apply multi-label only after as offline classification.

5 Hyperparameters tuning

The performance of models is often contingent on various hyper-parameters, configurable settings that govern the learning process. Hyper-parameter tuning, therefore, emerges as a crucial step in refining and optimizing machine learning models. This process involves systematically exploring and adjusting hyperparameter values to enhance a model's predictive capabilities. In this section, we explore the grid-search method for hyperparameter tuning, with a specific focus on multi-class analysis. Our chosen models for refinement include Decision Trees, KNN, and Random Forest. In contrast to the first phase of supervised learning, we divided the dataset into training, validation, and test sets. We conducted tuning using the training and validation sets and tested the results on the test set.

5.0.1 Decision Tree

For the Decision Tree model we performed hyper-parameter tuning on:

- **max_depth:** the maximum depth of the decision tree
- **min_samples_split:** the minimum number of samples required to split a node
- **min_samples_leaf:** the minimum number of samples a leaf node must have

With the following values:

- **max_depth:** [4, 6, 8, None]
- **min_samples_split:** [2, 3, 4]
- **min_samples_leaf:** [1,2,3]

After conducting the grid search, the results are visualized in the following plot:

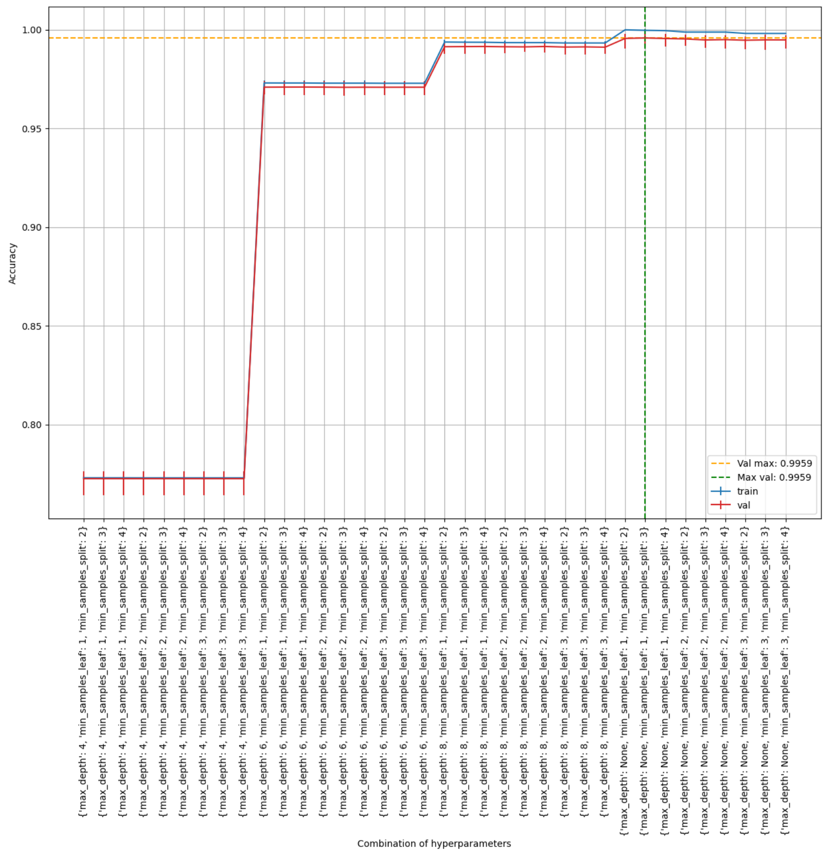

Figure 15: Decision Tree grid search

The grid search for the decision tree classifier was conducted using 10-fold cross-validation and the optimal parameters were determined to be:

- **max_depth:** None
- **min_samples_split:** 4
- **min_samples_leaf:** 1

With these parameters, we achieved an accuracy of approximately 99% with an f1-score close to 1 for each class.

5.0.2 KNN

For the KNN model we performed hyper-parameter tuning on:

- **n_neighbors:** the number of neighbors considered when making predictions for a new data point.

- **weights:** determine the weight assigned to each neighbor when making predictions
- **algorithm:** specifies the algorithm used to compute nearest neighbors

With the following values:

- **n_neighbors:** [3,5,8]
- **weights:** ['uniform','distance']
- **algorithm:** ['ball_tree','kd_tree','brute']

After conducting the grid search, the results are visualized in the following plot:

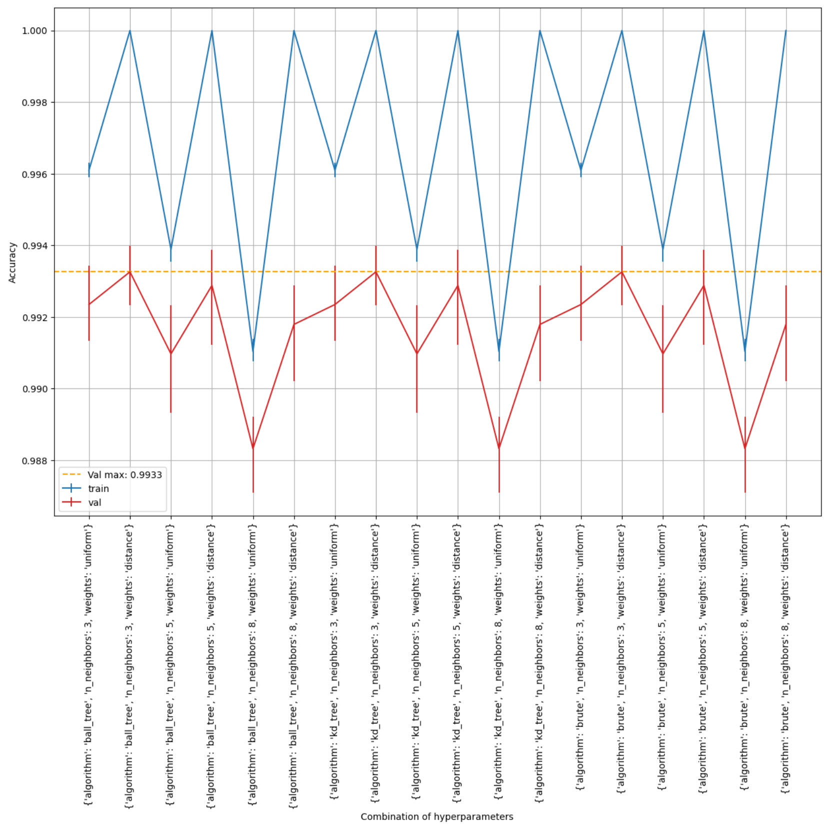

Figure 16: KNN grid search

The grid search for K-NN was conducted using 5-fold cross-validation and the optimal parameters were determined to be:

- **n_neighbors:** 3

- **weights:** distance
- **algorithm:** This parameter didn't affect performances in combination with n_neighbors=3 and weights=distance

With these parameters, we achieved an accuracy of approximately 99% with an f1-score close to 1 for each class.

5.0.3 Random Forest

For the Random Forest classifier we performed hyper-parameter tuning on:

- **max_depth:** the maximum depth of the decision trees in the forest
- **min_samples_split:** the minimum number of samples required to split a node in a decision tree
- **criterion:** the criterion adopted to measure the quality of a split in a decision tree
- **n_estimators:** the number of decision trees in the forest

With the following values:

- **max_depth:** [4, 6, 8, None]
- **min_samples_split:** [2, 3, 4]
- **criterion:** ['gini','entropy']
- **n_estimators:** [50, 100, 150, 200]

After conducting the grid search, the results are visualized in the following plot:

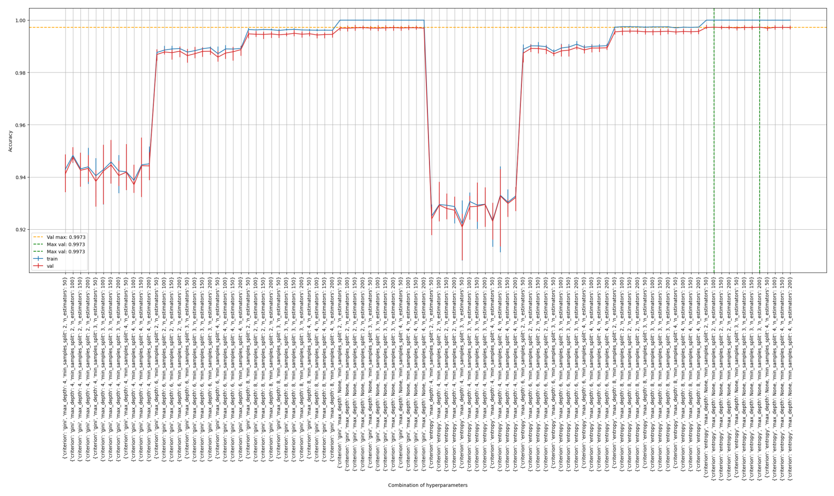

Figure 17: Random Forest grid search

The grid search for Random Forest was conducted using 5-fold cross-validation and the optimal parameters were determined to be:

- **max_depth:** None
- **min_samples_split:** 3
- **criterion:** Entropy
- **n_estimators:** 200

With these parameters, we achieved an accuracy of approximately 99.9% on average with an f1-score close to 1 for each class.

5.1 Observations

In the assessment of performance metrics encompassing accuracy, F-score, and training and prediction times, our observations led to the following conclusions: KNN exhibits minimal training time because it simply stores the entire training dataset. However, the prediction time for KNN can be relatively high, particularly with increasing dataset size, attributed to the necessity of calculating distances between the query point and all points in the training set during each prediction.
On the other hand Decision Trees demonstrate efficiency in both training and prediction phases, but their performance may be susceptible to dataset characteristics, posing a risk of overfitting, especially with larger datasets.
In contrast, Random Forests require more time for training compared to individual Decision Trees, given the construction of multiple trees. Nevertheless, their prediction time is faster than KNN and can be competitive with individual Decision Trees.
At the culmination of our evaluation, the Random Forest model demonstrated superior performance, and we deem it more scalable due to its enhanced resilience to overfitting compared to Decision Trees, particularly as the dataset size increases. Consequently, we believe that the trade-off between prediction time and robustness is justified, especially given that the prediction time remains notably swift.

6 Clustering

Clustering is an unsupervised machine learning technique that groups similar data points together. It has emerged as a powerful tool for analyzing cyberattacks, identifying different attack types and their underlying patterns, profiling attacker behavior, and detecting anomalies.

Clustering-based DDoS detection offers several advantages over traditional signature-based methods. It does not require prior knowledge of attack signatures or rules, making it more adaptable to evolving attack techniques. Second, clustering algorithms can handle high-dimensional data, which is often the case in network traffic analysis. Third, clustering can detect anomalies and identify attack patterns that may not be captured by signature-based systems. Some literature proposed method for using clustering algorithms for real time DDoS attack detection can be found explained in the work of Y. Gu et al. [2]

6.1 Clustering choices for DDoS attacks analysis

First and foremost, within the scope of our analysis, we employed both the normalized and cleaned dataset for clustering, as well as the dataset that underwent the PCA process, and compared the results.
A normalized dataset is preferred for clustering due to its inherent benefits over the raw dataset:

- **Consistency:** normalization ensures that all features have a consistent scale, preventing features with larger magnitudes from dominating the clustering process. This allows the clustering algorithm to focus on the underlying patterns rather than being influenced by the arbitrary scales of individual features.
- **Robustness:** normalization makes the clustering process more robust to variations in feature scales, potentially improving the overall clustering performance.
- **Fairness:** normalization promotes fairness in the clustering process by ensuring that all features have equal weight, regardless of their original scales.
- **Scalability:** normalization can enhance the scalability of clustering algorithms by reducing the computational burden associated with processing data on different scales.

Secondly, we performed a stratified sampling of the dataset during the parameter tuning phase. This approach allowed us to retain similar information compared to using the full dataset, while significantly reducing computational burden and timing for completing the tuning process.
We initiated the clustering analysis with the cleaned dataset that had the original features and subsequently compared the results by performing the same operations on the PCA-transformed dataset

6.2 Function definition to plot the clusters

In order to plot the clusters obtained from clustering algorithms, we have used an utility function called plot_clustering.
The **plot_clustering** function serves as a general-purpose tool for plotting clusters derived from various clustering algorithms. The function assigns different colors to each cluster based on the corresponding labels, providing a visual representation of the separation between clusters. Additionally, it optionally plots

the cluster centers as red circles, facilitating the identification of the cluster centroids.

6.3 Hard clustering with K-means

Operating under the principles of unsupervised learning, K-means excels at partitioning datasets into distinct clusters based on similarity. The algorithm iteratively assigns data points to clusters while optimizing the centroids of these clusters to minimize the within-cluster sum of squares. In essence, K-means seeks to group data points in a way that minimizes the variance within each cluster, providing valuable insights into underlying patterns and structures.

6.3.1 Optimization metrics

In the context of k-means the **n_clusters** parameter specifies the number of clusters to be formed by the K-means algorithm. Determining an appropriate value for this parameter is crucial for achieving effective clustering. In order to do so we adopted two metrics:

- **Silhouette score:** measures the separation between clusters and the intra-cluster cohesion.
- **Clustering error:** measures the average distance between each data point and its assigned cluster centroid.

We assessed these metrics for values of **n_clusters** ranging from 2 to 50. The results are plotted in Figure 18 with two subplots:

- **Subplot 1:** The silhouette score is plotted against the number of clusters. A high silhouette score indicates good separation between clusters. The plot shows that the silhouette score peaks at **n_clusters = 25**, suggesting that 25 clusters are optimal for this dataset.
- **Subplot 2:** The clustering error is plotted against the number of clusters. A low clustering error indicates that the data points are well-grouped within the clusters. The plot illustrates that the clustering error decreases with an increasing number of clusters

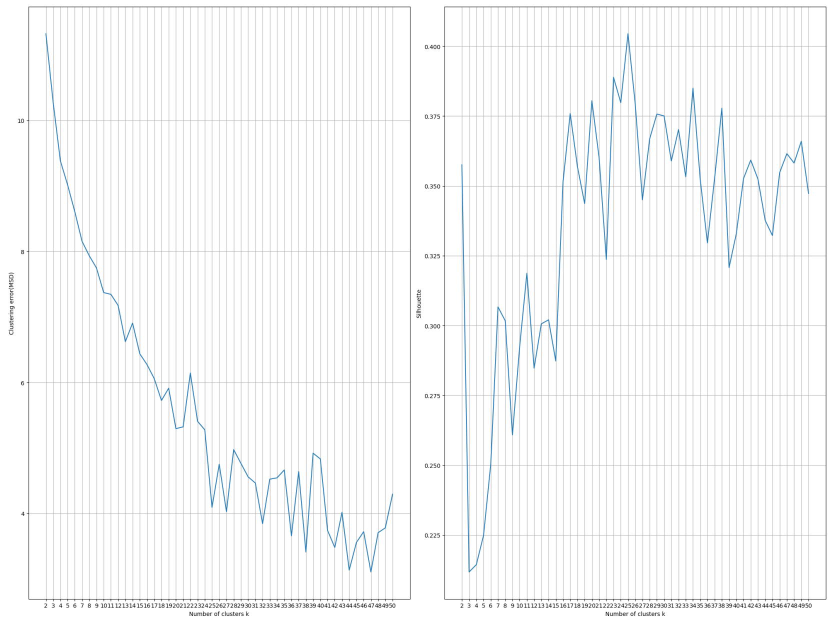

Figure 18: K-means metrics

Based on the evaluation of silhouette score and clustering error, we selected n_clusters = 25 as the optimal number of clusters.

6.3.2 K-means with the optimal n_clusters

After tuning the 'n_clusters' parameter on the stratified sample of the dataset, we applied the k-means algorithm to the entire dataset, evaluating metrics such as

- **Sum of sqared distances:** or SSD sum of squared distances of the samples from their centroid
- **Mean sqared distances:** or MSD mean squared distances of the samples from their centroid
- **Silhouette score:** ranges from -1 to 1, where a score closer to 1 indicates well-separated clusters with high intra-cluster cohesion and low inter-cluster separation.
- **Rand index:** or RI measures the similarity between the clustering labels and the ground truth labels. A higher RI index indicates better agreement between the two sets of labels.
- **Adjusted Rand index:** or ARI is a modification of the Rand index that adjusts for chance agreement. A higher ARI index indicates better agreement between the clustering labels and the ground truth labels, taking into account the possibility of random chance.

The SSD measures the overall compactness of the clusters, indicating how well the data points are grouped together. A lower SSD value indicates better cluster cohesion. If we divide the SSD by the number of data points we obtain the mean squared distances (MSD). MSD provides a more interpretable measure of cluster compactness, as it represents the average distance between each data point and its assigned cluster center. A lower MSD value indicates better cluster cohesion.

6.3.3 Results

The results for the mentioned metrics are:

- **SSD:** 953776
- **MSD:** 14.85
- **Silhouette score:** 0.4
- **RI:** 0.91
- **ARI:** 0.38

The silhouette score is not exceptionally high, suggesting that clusters may partially overlap.
Despite the Rand Index (RI) being very high, the Adjusted Rand Index (ARI) is relatively low, indicating that the clustering may not precisely align with the original labels. However, this is not necessarily a negative sign, as it could be attributed to the presence of sub-attacks not explicitly defined by the original labels.

6.4 Soft clustering with GMM

Hard clustering algorithms, such as K-means, assign each data point to a single cluster, resulting in hard boundaries between clusters. This can lead to overfitting, especially when the data is not well-separated. **Gaussian Mixture Models (GMMs)**, assign each data point to a probability distribution over the clusters. This means that a data point can have a high probability of belonging to one cluster and a lower probability of belonging to others.

6.4.1 Optimization metrics

As we did with K-means, we tuned the **n_components** parameter, which for GMM specifies the number of clusters to be formed. The optimal value for this parameter can be determined using various clustering metrics, such as the **silhouette score** and **log-likelihood**, which measures the overall fit of the model to the data. We evaluated the silhouette score and log-likelihood for different values of n_components, ranging from 2 to 50, as shown in Figure 19.

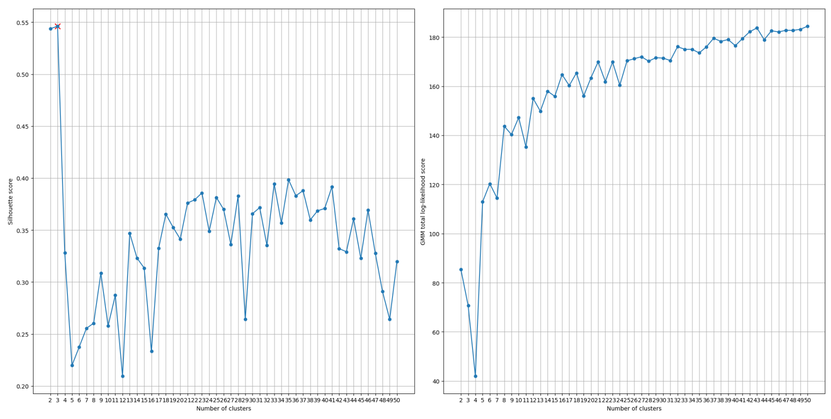

Figure 19: GMM metrics

In this case, the silhouette score peaks at **n_components = 35**, suggesting that 35 clusters are optimal for this dataset.

6.4.2 GMM with the optimal n_components

Similarly to the K-Means algorithm analysis, after tuning the 'n_clusters' parameter on the stratified sample of the dataset, we applied the gmm algorithm to the entire dataset, evaluating metrics such as:

- **Total log-likelihood score:** measures the overall fit of the GMM model to the data. A higher score indicates a better fit.
- **silhouette score**
- **RI**
- **ARI**

6.4.3 Results

- **Total log-likelihood score:** 179.17
- **silhouette score:** 0.34
- **RI:** 0.87
- **ARI:** 0.32

The results proved to be similar to those obtained with hard clustering, indicating no significant gain in performance despite the increased computational load.

6.5 Clustering with DBSCAN algorithm

DBSCAN is an effective clustering algorithm for identifying groups of DDoS attacks based on their network traffic characteristics. This is because DBSCAN can identify clusters with irregular shapes and variable sizes. Moreover, it can be used to identify outliers and is effective on non-Euclidean cluster structures where k-means and GMM may fail.

6.5.1 Optimization metrics

DBSCAN is a density-based clustering algorithm that requires two parameters: **eps** and **min_samples**. The eps parameter specifies the maximum distance between two points to be considered neighbors, while the min_samples parameter defines the minimum number of points required to form a cluster. After applying a 'trial and error' technique, we set min_samples at 40. We tuned the eps parameter by evaluating the silhouette score for different values ranging from 0.01 to 4, with a step size of 0.1.
The silhouette score peaks at eps = 1.11, indicating that 1.11 is the optimal value for this dataset, as depicted in the plot of the silhouette score (Figure 20).

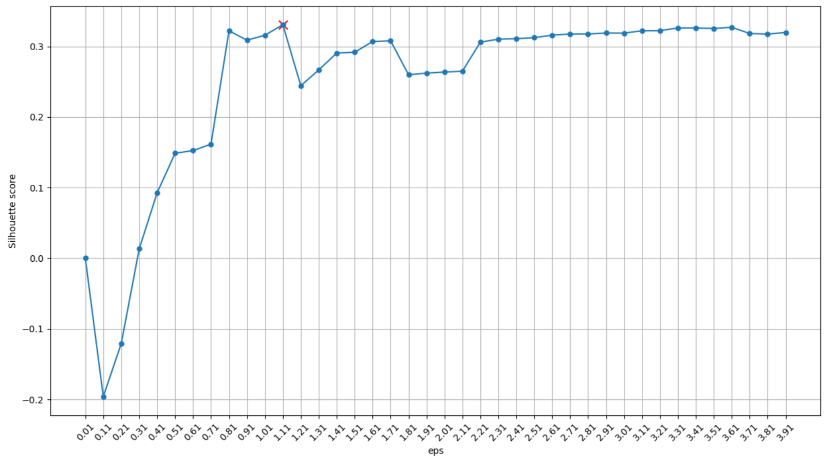

Figure 20: DBSCAN optimal parameters

6.5.2 DBSCAN with the optimal parameters

Similarly to K-Means and GMM, we used the DBSCAN algorithm to identify distinct groups of DDoS attacks using the previously determined optimal parameters, which are **eps = 1.11** and **min_samples = 40**. The code first creates a DBSCAN object with the specified parameters and then fits the model to the dataset. Firstly, we computed the **silhouette score** for the clustering results. Then, we counted the **number of clusters** and **noise points** identified by the DBSCAN algorithm.

6.5.3 Results

- **silhouette score:** 0.35
- **number of clusters:** 38
- **noise points:** 4122

6.6 K-means algorithm on PCA dataset

Principal Component Analysis (PCA) can be beneficial for clustering algorithms, as it can help to make the data more manageable and easier to cluster.

6.6.1 Optimization metrics

Evaluating the **clustering error** and the **silhouette score** in Figure the best performance are obtained with **n_clusters=25**.

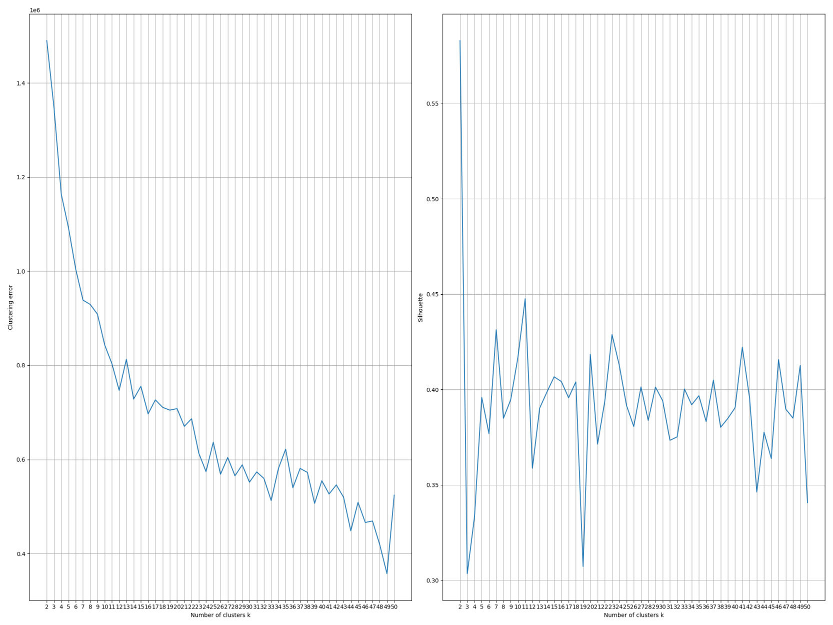

Figure 21: K-means optimal parameters

6.6.2 Results of k-means with optimal n_clusters

- **SSD:** 1937269
- **MSD:** 30.16
- **Silhouette score:** 0.37
- **RI:** 0.92

- **ARI:** 0.42

6.7 GMM algorithm with PCA dataset

6.7.1 Optimization metrics

From what we se in Figure 22, we have the best compromise between silhouette score, log-likelihood score with **n_components=39**.

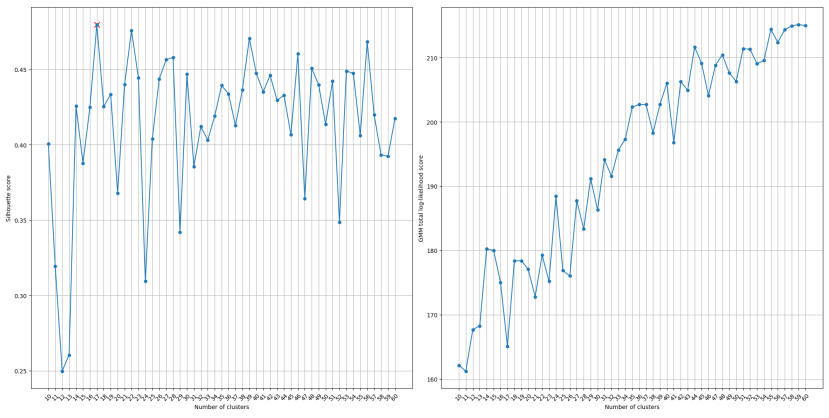

Figure 22: GMM optimal parameters

6.7.2 Results of GMM with the optimal n_components

- **Total log-likelihood score:** 203.8
- **Silhouette score:** 0.28
- **RI:** 0.84
- **ARI:** 0.25

6.8 DBSCAN algorithm with PCA dataset

6.8.1 Optimization metrics

Looking at the silhouette values in Figure 23, the best value of **eps** is **1.21**.

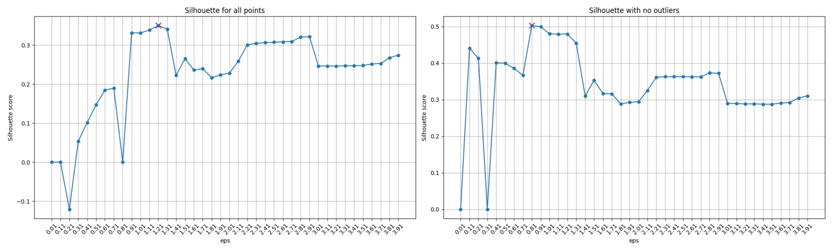

Figure 23: DBSCAN optimal parameters

6.8.2 Results of DBSCAN with the optimal parameters

- **Number of clusters:** 42
- **Number of noise points:** 4849
- **Silhouette:** 0.26
- **Silhouette without outliers:** 0.33

6.9 Observations

Although clustering on the PCA dataset didn't notably enhance performances (except for K-means), the difference becomes apparent in cluster visualizations. The clusters were challenging to visualize accurately when using the dataset with original features. However, performing clustering on the PCA dataset provided better visualization results from the plots of the first two features.

7 Clusters explanability

After applying hard, soft, and density-based clustering techniques, we analyzed the obtained clusters to compare them with ground truth labels and to identify patterns and characteristics.

7.1 Comparison with GT labels

7.1.1 K-means and GMM

The clusters generated by K-means and GMM exhibit striking similarity, and by merging some of these clusters, we can derive groups that closely resemble the ground truth (GT), as illustrated in Figure 24.

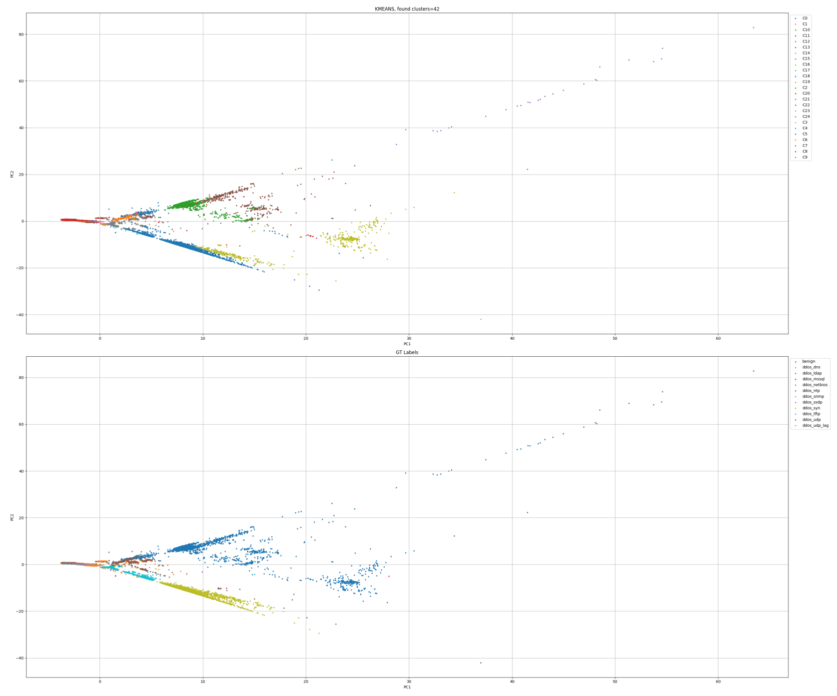

Figure 24: GT comparison

We evaluated the empirical cumulative distribution function (ECDF) of the number of clusters assigned to each class. As depicted in Figure 25, the ECDF illustrates that each class is distributed across multiple clusters. This may indicate that each class could be divided into sub-classes, and some of these might share common characteristics.

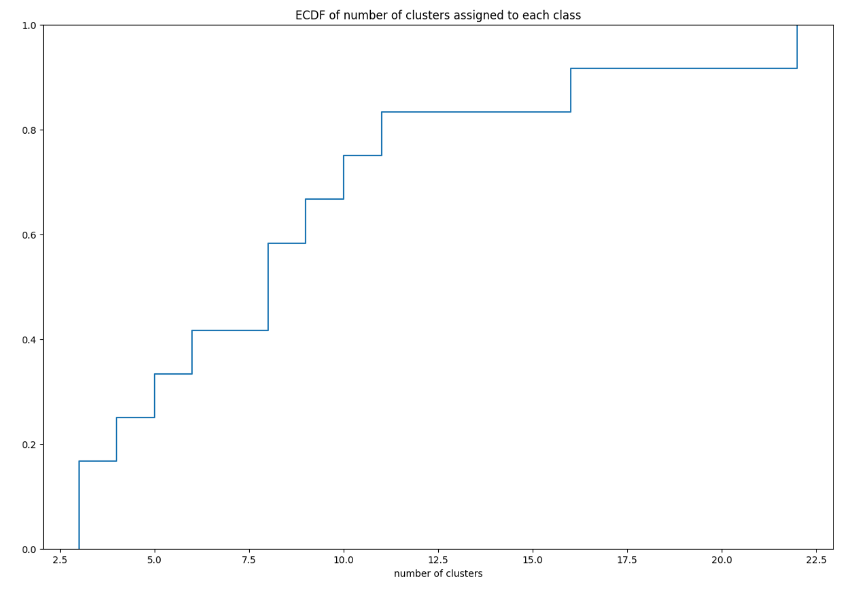

Figure 25: ECDF of clusters

On the other hand, according to our analysis, there are some clusters that are pure, where all elements belong to a single class. An example of such clusters can be observed in Figure 26, where the cluster is exclusively composed of elements belonging to the 'benign' label.

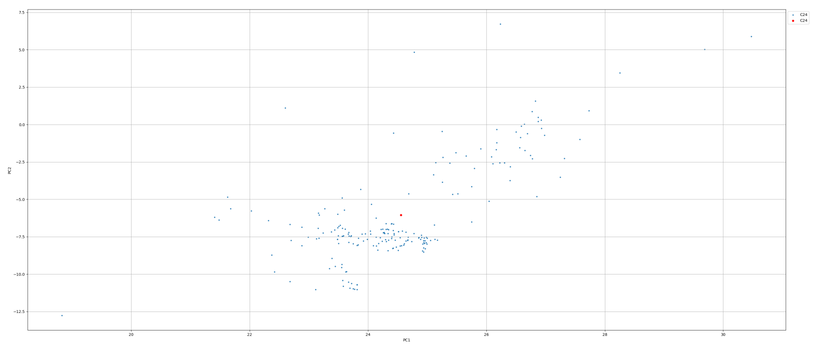

Figure 26: Cluster containing benign flows

7.1.2 DBSCAN

The peculiarity of DBSCAN is that it considers the majority of benign traffic as outliers or noise, thus isolating the attacks along with a small portion of benign traffic that shares characteristics with it, as shown in Figure 27.

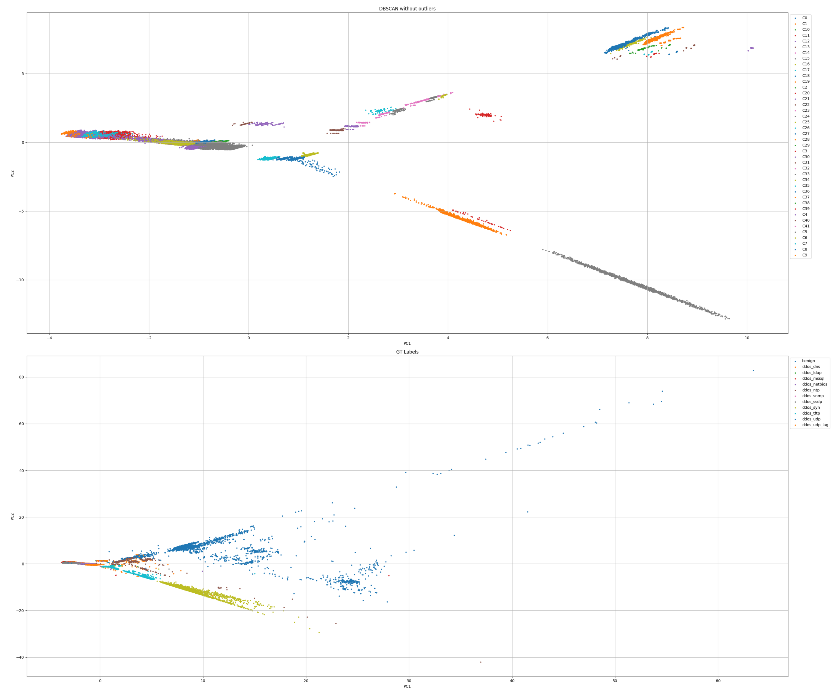

Figure 27: DBSCAN clusters without outliers

7.2 Important features

In order to visualize clustering results, it is pivotal to select the most important features. We utilized the first two principal components of the PCA to showcase the clusters. However, more sophisticated methods based on FSSEM, such as those employed by Jennifer G. Dy and Carla E. Brodley [1], offer further complexity and insight.

7.3 Sub-attacks

The various clustering methods have identified a larger number of clusters than the original labels, suggesting that each attack may be subdivided into sub-attacks. An illustrative example can be seen in Figure 28, where the 'ddos_syn' label is notably divided into two distinct clusters, potentially representing sub-attacks.

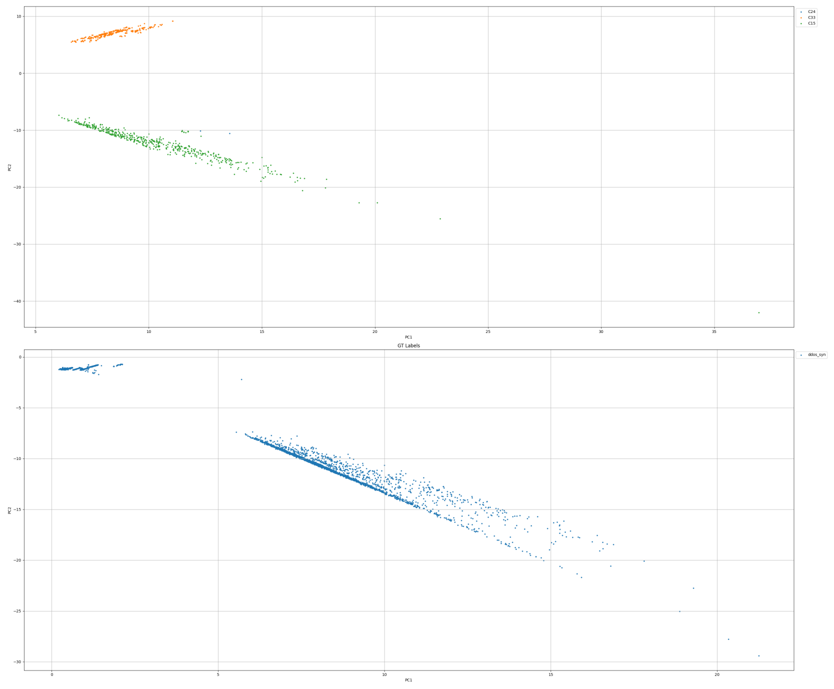

Figure 28: Clusters included in ddos_syn

8 Conclusions

In this exploration of DDoS attack detection, we delved into the application of both supervised learning and clustering techniques to tackle the complexity and diversity inherent in these attacks within the realm of computer networks. Results obtained through supervised learning, particularly employing algorithms like KNN, Decision Tree and Random Forest demonstrated significant accuracy in classifying benign and malicious traffic.

Simultaneously, the clustering approach unveiled intriguing hidden structures in the data. Leveraging clustering algorithms such as K-means, GMM, and DBSCAN allowed the identification of patterns and groupings that may indicate variations in attacker behaviors.

It is noteworthy that the integration of both approaches, supervised learning and clustering, contributed to a more comprehensive understanding of the DDoS detection challenge. The combination of these techniques provides a deeper insight into the complexity of attacks, enabling the identification of subgroups, anomaly detection, and enhancing the overall resilience of the defense system.

In conclusion, this research illustrates that a hybrid approach, combining supervised and unsupervised methods, can be pivotal in addressing the ever-growing challenge of DDoS attacks in modern network infrastructures. The ongoing evolution of these techniques is crucial to adapt to new threats and ensure the security of networks in an ever-changing digital landscape.

References

[1] Jennifer Dy and Carla Brodley. "Feature Selection for Unsupervised Learning". In: *Journal of Machine Learning Research* 5 (Aug. 2004), pp. 845–889.

[2] Yonghao Gu et al. "Semi-Supervised K-Means DDoS Detection Method Using Hybrid Feature Selection Algorithm". In: *IEEE Access* 7 (2019), pp. 64351–64365. DOI: `10.1109/ACCESS.2019.2917532`.