# **DDoS Attack Detection and Classification**

## Abstract

This report presents a comprehensive project in the field of machine learning, centered around the analysis and mitigation of Distributed Denial of Service (DDoS) attacks. The project commences with the exploration and visualization of a provided DDoS dataset, aiming to gain insights into the characteristics and patterns of the network traffic associated with such attacks. The visualization and exploration phase leverages data visualization techniques to unveil key features and trends within the dataset.

Subsequently, the project delves into the application of supervised learning techniques for classification purposes. Multiple machine learning models are employed to classify network traffic instances: KNN, Support Vector Machines, Gaussian Naive Bayes and Random Forests. The evaluation of these models involves metrics such as accuracy, precision, recall, and F1 score, providing a comprehensive assessment of their performance in identifying DDoS attacks.

Furthermore, the project incorporates clustering techniques to uncover hidden structures within the dataset. By applying clustering algorithms such as K-means, DBSCAN and Gaussian Mixture, the project aims to group similar network traffic instances together. Evaluation of clustering results is performed using appropriate unsupervised and supervised metrics.

## Section 1

## Section 2

In this project, to solve the multilabel classification problem, four supervised ML classifiers have been evaluated, which are K-Nearest-Neighbours-Classifier(KNN), Random Forest Classifier (RF), Support Vector Machine (SVC) and Gaussian-Naive-Bayes (GNB). The following are the metric used for model evaluation:

* Accuracy
* Precision
* Recall

F1-scoreIn a first session, dataset is split in training and test set and performance are evaluated with models' default parameters.

Immagine che contiene testo, schermata, Parallelo, numero

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Parallelo, numero

Descrizione generata automaticamente

KNN, RFC (the best) and SVC perform very well even on test set. On the other hand, GNB it’s the worst model since it starts from the assumption of Gaussian distribution of data and related probability independence among features.

Considering the confusion matrix on test set, it is possible to display which class of attacks are misclassified.

Immagine che contiene testo, schermata, quadrato, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, quadrato, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, quadrato, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, quadrato, diagramma

Descrizione generata automaticamente

The best model is RFC for this task of multi-classification. GNN is the model that performs the worst. There are few features that are classified erroneously in all the three models:

* ddos\_dns (true) misclassified with ddos\_ldap, ddos\_netbios, ddos\_ntp
* ddos\_ldap (true) misclassified with ddos\_mssql (prediction) and vice versa with same percentage
* ddos\_ntp (true) misclassified with benign (prediction)
* ddos\_syn (true) misclassified with ddos\_tftp (prediction) 0.02% highest misclassified value for random forest classifier
* ddos\_tftp (true) misclassified with ddos\_syn (prediction)
* ddos\_udp\_lag (true) misclassified with ddos\_udp (prediction)

all misclassifications are equal to 0.01% for RFC, for the other models the percentage increases (due to their worse performance), but the classes involved in the misclassification are the same.

After this first model evaluation, we proceeded with hyperparameters tuning for each model to try to increase their performances on our DDoS dataset. The following are the hyperparameters tuned in the process which involved sklearn.GridSearchCV():

* KNN:
  + “n\_neighbors”: [3, 5, 7]
  + “weights”: [“uniform”, “distance”]
  + “p”:[1, 2]
* RFC:
  + “criterion”: [gini, entropy]
  + “n\_estimators”: [50, 100]
  + “max\_depth”: [None, 10]
  + “min\_samples\_split”: [2, 3]
* SVC:
  + “C”: [0.1, 1, 10]
  + “kernel”: [rbf, poly]
* GNN:
  + “var\_smoothing”: [1e-11, 1e-10, 1e-9, 1e-8, 1e-7]