A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

Comparative analysis of the performance of supervised algorithms on the CSE-CIC-IDS2018 dataset.

Computer Engineering and Robotics – Academic Year 2024-2025

Salvatori Gianluca

0. Contents

List of Figures **2**

1 EDA **3**

2 Model evalutation **6**

3 Comparative analysis and final considerations **10**

0. List of Figures

1.1 Attack types distribution 4
1.2 Correlation matrix 5

2.1 Confiusion matrix for decision tree 7
2.2 Random forest confusion matrix 8
2.3 XGBoost confusion matrix 9

1. EDA

The dataset used in this project consists of network traffic instances classified into five categories of web attacks: Botnet, Brute-force, DDoS, DoS, and Web Attack.

The goal of the project is to develop three distinct classifiers to accurately distinguish between these attack types.To achieve this, three algorithms were employed: Decision Tree with pruning, Random Forest, and XGBoost.
These classifiers were then compared based on their performance scores on the dataset.

There are 5 different features, one for each attack type, that have been converted into a new single feature *attack* and then dropped.

```
dfToReplace = df[['Label_Botnet','Label_Brute-force','Label_DDoS
↪  attack','Label_DoS attack','Label_Web attack']]

df['attack'] = dfToReplace.idxmax(axis=1).str.replace('Label_', '')
attacks = df['attack']

mapping = {'Botnet': 0, 'Brute-force':1,'DDoS attack':2, 'DoS
↪  attack':3,'Web attack':4}

df['attack'] = df['attack'].replace(mapping)

df = df.drop(['Label_Botnet','Label_Brute-force','Label_DDoS
↪  attack','Label_DoS attack','Label_Web attack'], axis=1)
```

Furthermore, the dataset contains 66444 duplicated rows. After analyzing the scores with and without these samples, it has been noticed that all the models perform better after removing the duplicates, so they have been eliminated.
No null value is present in the dataset.

Plotting the distribution of the values on the label it is possible to see that unfortunately there are few examples of the web attacks. However this will not be a problem in the classification task as we will see later.

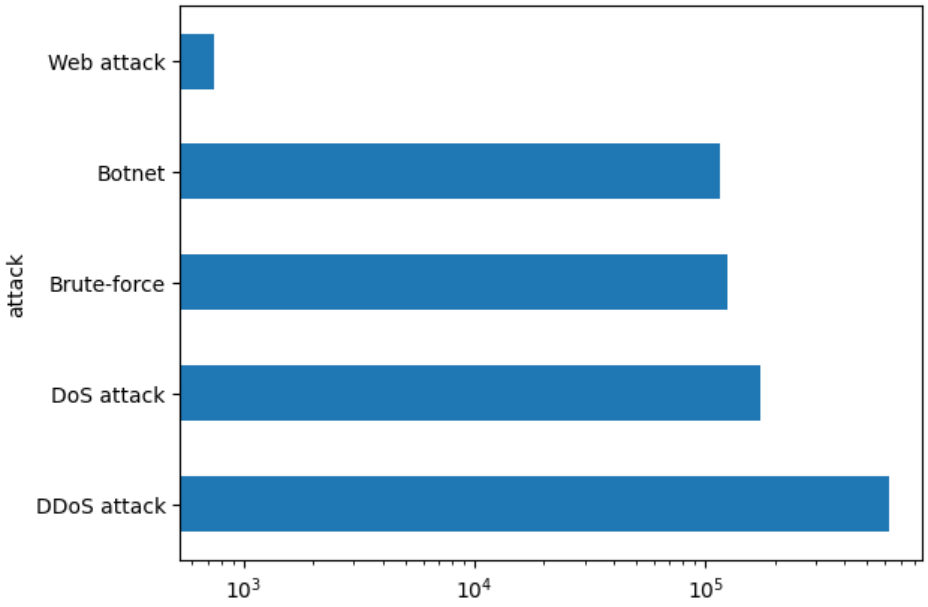

Figure 1.1: Attack types distribution

Finally, the correlation matrix was analyzed, revealing that certain features exhibit moderate to high levels of correlation. However, to preserve all available information, none of these features were removed.

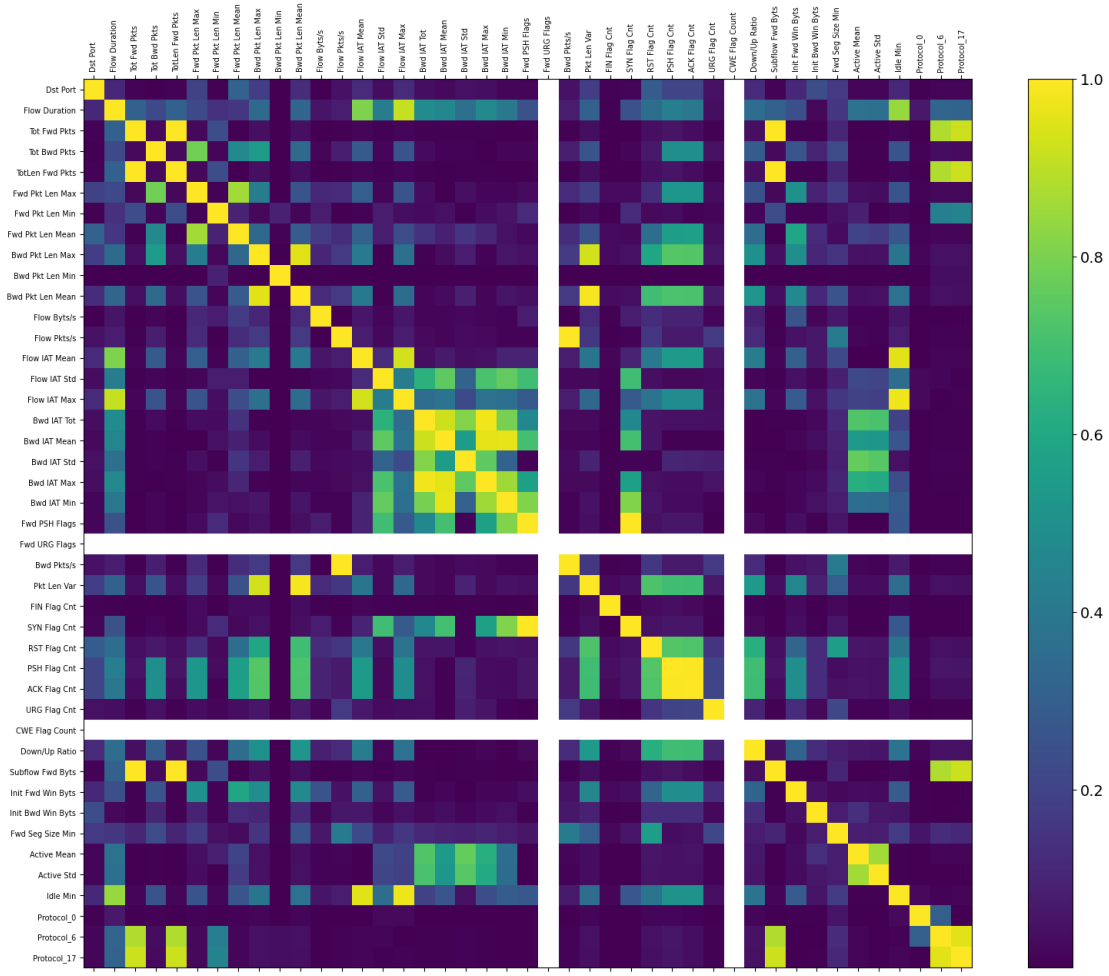

Figure 1.2: Correlation matrix

2. Model evalutation

As previously mentioned, three different algorithms were selected: Decision Tree, Random Forest, and XGBoost.All of them were fine-tuned by optimizing key parameters using the GridSearchCV method. (This part of the script can be found only on the .py file)

Decision Tree

The decision tree has been pruned first of all. So there have been selected several values for the parameter *ccp_alpha* and trained different trees in order to find the optimal value.

```
#PRUNING, SEARCHING FOR THE BEST CCP_ALPHA VALUE

t0 = time.monotonic()
ccp_values=[0, 0.001, 0.002, 0.005, 0.01, 0.02, 0.03, 0.05]
clfs = []
for ccp_alpha in ccp_values:
    clf = DecisionTreeClassifier(criterion='entropy',random_state=42,
    ↪  ccp_alpha=ccp_alpha)
    clf.fit(X_train,t_train)
    test_scores = clf.score(X_train,t_train)
    clfs.append(clf)
    print("Number of nodes in the tree with ccp_alpha {} is {} ,
    ↪  depth {} or {}, and score {}".format(ccp_alpha,
    ↪  clf.tree_.node_count,  clf.tree_.max_depth, clf.get_depth(),
    ↪  test_scores )  )
t1 = time.monotonic()
```

The best params for the decision tree are the following
'ccp_alpha': 0, 'criterion': 'entropy', 'max_depth': 12, 'splitter': 'best'

The table shows the times (from a specific code run, not averaged) for pruning the tree, constructing it with the chosen parameters, and the resulting scores (measured as mean accuracy) on both the training and test sets.

Pruning time	79.23052429099994
Training time	8.343833604999418
Score on train set	0.9999587594853184
Score on test set	0.9999209556801936

Table 2.1: Decision tree time values

A visual representation of the results is provided through the confusion matrix:

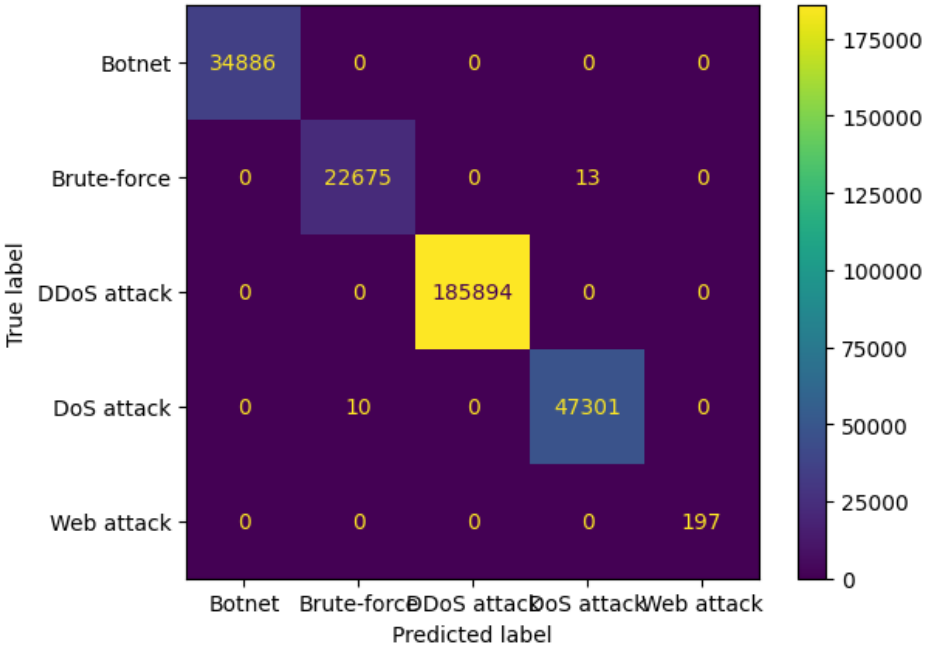

Figure 2.1: Confiusion matrix for decision tree

Random Forest

The best parameter obtained are the following:
max_depth: 14, n_estimators: 100, criterion: 'gini'

And so the times and the scores for this algorithms

Training time	140.96380029600004
Score on train set	0.9999602323608427
Score on test set	0.9999140822610799

Table 2.2: Random forest time values

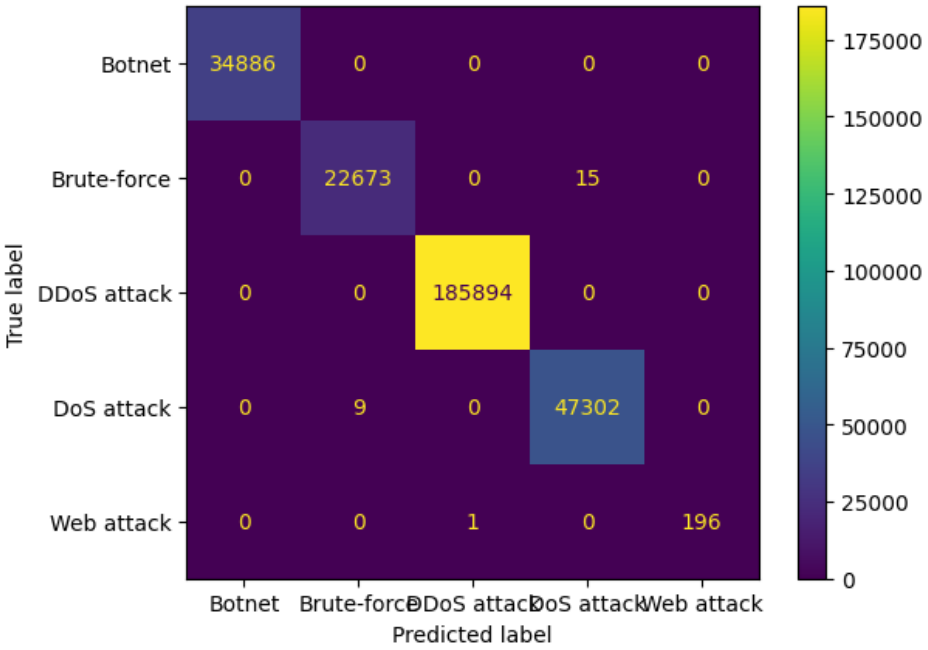

Figure 2.2: Random forest confusion matrix

XGBoost

Finally the values for XGBoost.

Optimal values:
objective: "multi:softprob", learning_rate: 0.1, max_depth: 10

Training time	47.77981827399981
Score on train set	0.9999499222321723
Score on test set	0.9999347025184208

Table 2.3: XGBoost time values

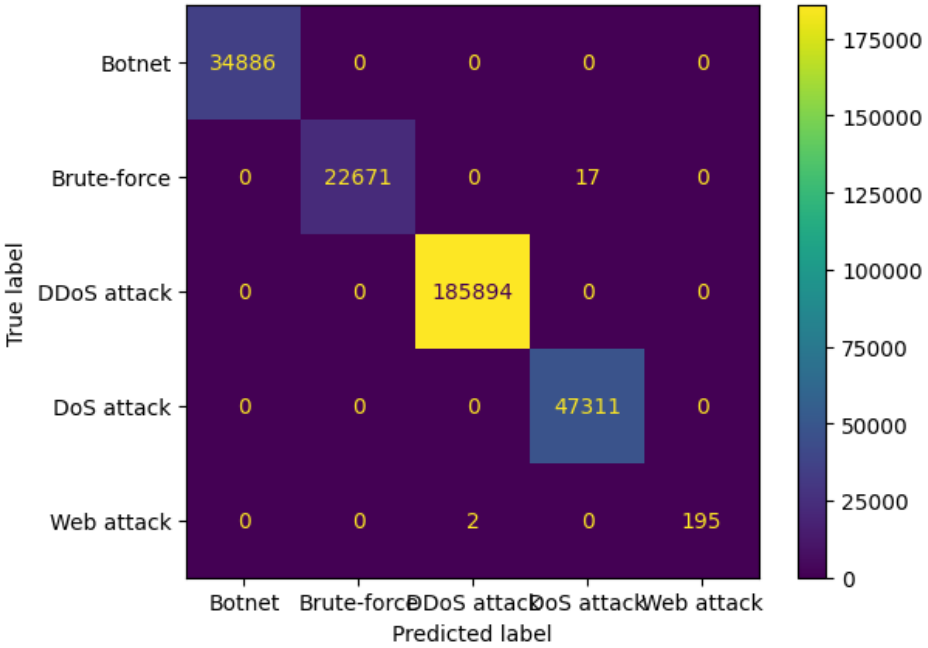

Figure 2.3: XGBoost confusion matrix

3. Comparative analysis and final considerations

All the algorithms perform incredibly well on the dataset.

It is evident that all the models face challenges in distinguishing certain brute force attacks from DoS attacks. On the other hand, Botnet and DDoS attacks are never misclassified.

Previously, we observed that the number of web attacks is significantly lower compared to other types. Nevertheless, nearly all of them are correctly classified, with a maximum error of only 2 out of 197 samples, accounting for just 1%.
While this represents a higher error rate compared to other attack classes, it may still be acceptable depending on the design choices of the project.

XGBoost seems to be the best algorithm of the group.
Although the performance difference can only be found from the fifth decimal place onwards, XGBoost also has an impressive improvement in build times.
Considering the computational times, Decision Tree requires approximately 80 seconds for pruning plus 10 seconds for training. In comparison, XGBoost takes half the time, and Random Forest requires nearly three times as much.
Despite this, XGBoost continues to achieve a higher mean accuracy compared to the other models.

Between Random Forest and the pruned Decision Tree, the score values on the test set are similar. However, even when factoring in the pruning time along with the construction time, the pruned Decision Tree is faster, making it a preferable choice.