IA&DI - Final Report

Martin Bazire et Émilien André

Abstract

The objective was to create an Intrusion Detection System. The requirements were : the system should be backed with AI, the model should be trained with the ISCX dataset and selected from cross-validation with 5 folds. We were given two challenges, the first was to classify flows in two classes. The second challenge necessitated retraining a model with a heavy XML file and 3 classes. Each challenge requires a “blind submission” on an evaluation dataset.

Introduction

In a world where cyber-threats are growing, we had to realize a log-based intrusion detection system. Log-based intrusion detection cannot be firstly a human analysis because of the massive data. This task needs algorithms and machine learning to perform well. The objective is to train a classifier to predict if a log is malicious or not. It can only be “identify if there is a threat” or it can be “identify the type of threat” this log is attached to. At the beginning we setted up the system on a Virtual Private Server (VPS) to mutualize scripts and runs. It seemed like a smart move, a centralized spot for control and processing power. But, as we got deeper into the project, we realized that the VPS just couldn't keep up in computing power. We returned to our own computers working with the GIT tool. We will firstly understand the project structure, secondly the methodology and metrics used. To finish with a conclusion and perspectives for this project.

Project Structure

One should have elasticsearch up and running with security features disabled within a linux machine to run the scripts.

The project uses a python virtual environment to give a solution working with almost every linux distribution. Nowadays python packages are installed with a packet manager in most distributions, not all packages can be found and names can differ from distribution to distribution.

The scripts find the project directory and install data in the `data` sub-directory.

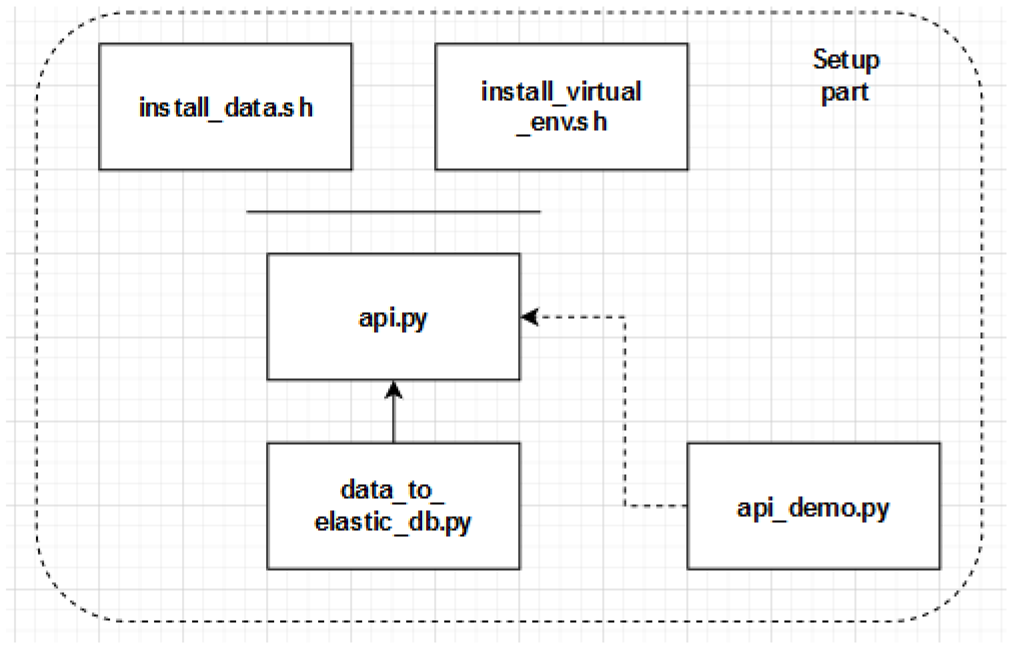
There are only two shell scripts: one is for installing the datasets, the other is for installing the python virtual environment, all other scripts are python.

Scripts

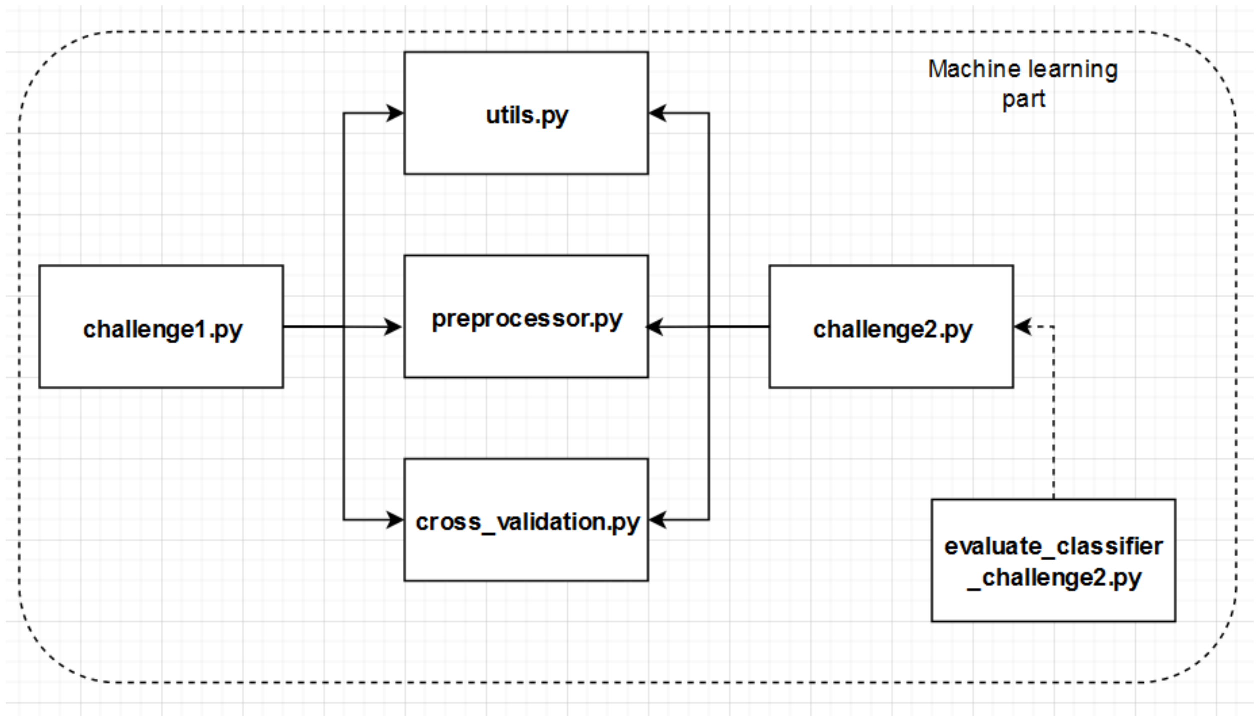
The `proj` is the project's directory.
Scripts are in proj/scripts :

- install_data.sh downloads and installs data for challenges 2 and ISDX dataset inside proj/data (access to challenge 1 data requires moodle credentials and has not been automated)
- install_virtual_env.sh installs a python virtual environment and downloads the necessary dependencies inside.
- data_to_elastic_db.py load the ISDX dataset into elasticsearch in the fly
- api.py is an API to extract ISDX data from elasticsearch
- api_demo.py is a demonstration of the API.
- preprocessor.py contains functions to get train data from the ISDX dataset for the challenge 1.
- cross_validation.py cross validates 4 models with a given train set and selects the one with the best average accuracy over 5 folds.
- challenge1.py uses api.py, preprocessor.py and cross_validation.py to solve the first challenge.
- challenge2.py uses a random forest classifier trained on the given train set and predicts the given test set. The bigxml library has been used to address the RAM issue caused by the dataset size.
- evaluate_classifier_challenge2.py evaluates 3 classifiers for the second challenge.

To understand clearly, we can separate the scripts in two groups. We call the first group the "setup group", this is all the scripts which are used to set up the environment and data.



The second group is all the scripts about machine learning with the 2 challenges.



We can see there that the challenge scripts are using utils to have useful/common functions. They also use the same preprocessing and cross validation.

Data extraction

For the second challenge we did parsing with the python library named bigxml not to run out of memory. Bigxml parses the input stream on pass and has an event-based programming approach with handlers.
To run without elastic we also used the module pandas, especially “pandas.read_xml”, this is really fast and extracts data in a dataframe format.

Preprocessing

In this step, we need to transform data in numerical format to be understandable by machine learning algorithms. We used the Scikit-learn module and a few “home-made” scripts for variable encoding.

TimeStamp	biased (removed)
Duration	float
Protocol	integer
Src_IP_Add	Biased (removed)
Src_Pt	integer
Dst_IP_Add	Biased (removed)
Dst_Pt	integer
Packets	integer
Tos	integer
Flags	integer

Sheet: challenge 2 flows attributes

List of preprocessing choices on the variables. This is purely subjective, we only kept the variables which carried relevant information for our use case. For example the “TimeStamp” is probably not relevant because attacks may not be at the same time.

Challenge 1

The cross validation is used to evaluate the 4 given models and select the one with the best average accuracy.

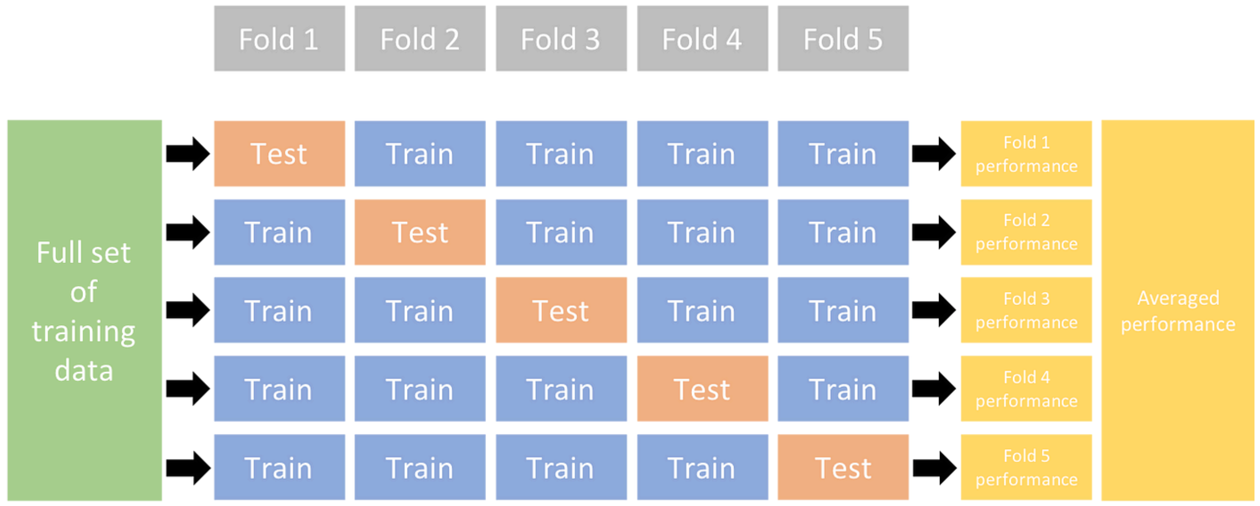


Figure: the cross validation technique we used

model	Fit time	Score time	Accuracy	Precision	Recall	F1
KNN	0.0089	0.1241	0.9965	0.99706	0.998169	0.99761
GaussianNB	0.00395	0.009905	0.984619	0.981489	0.997896	0.98962
RandomForest	0.34586	0.0229715	0.999587	0.999578	0.999860	0.99972
MLP	16.015	0.017808	0.9937028	0.992727	0.998742	0.99572

Sheet: cross validation averages for each model

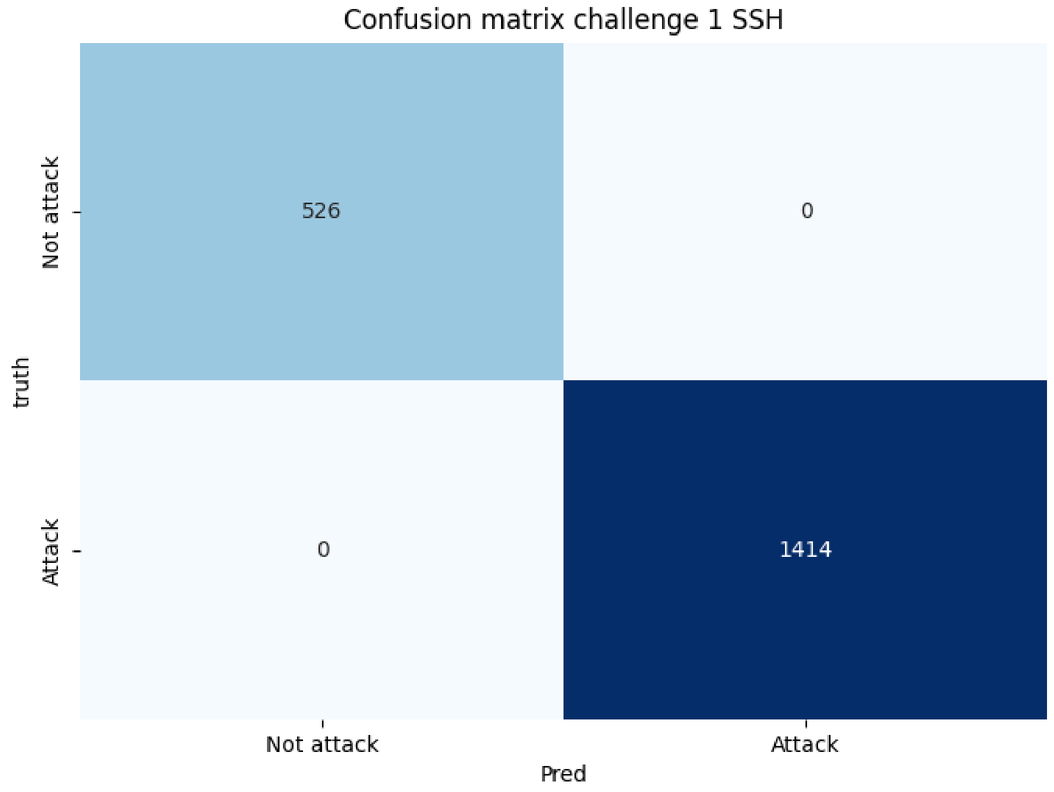
Challenge 2

As the data size was enormous for our personal computers we could not do cross validation which needed 5 training, one day was not enough to cross validate even one model. We chose a random forest classifier because it gives the best cross validation scores for the ISDX dataset which is similar.

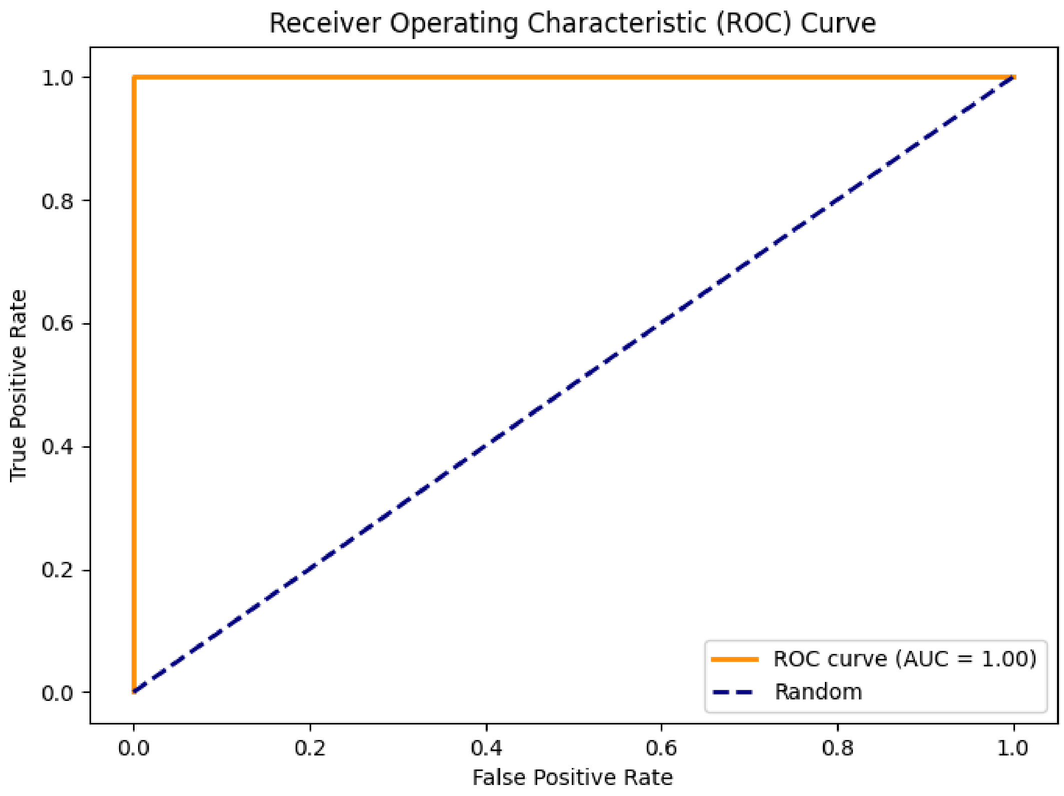
Evaluation

Challenge 1

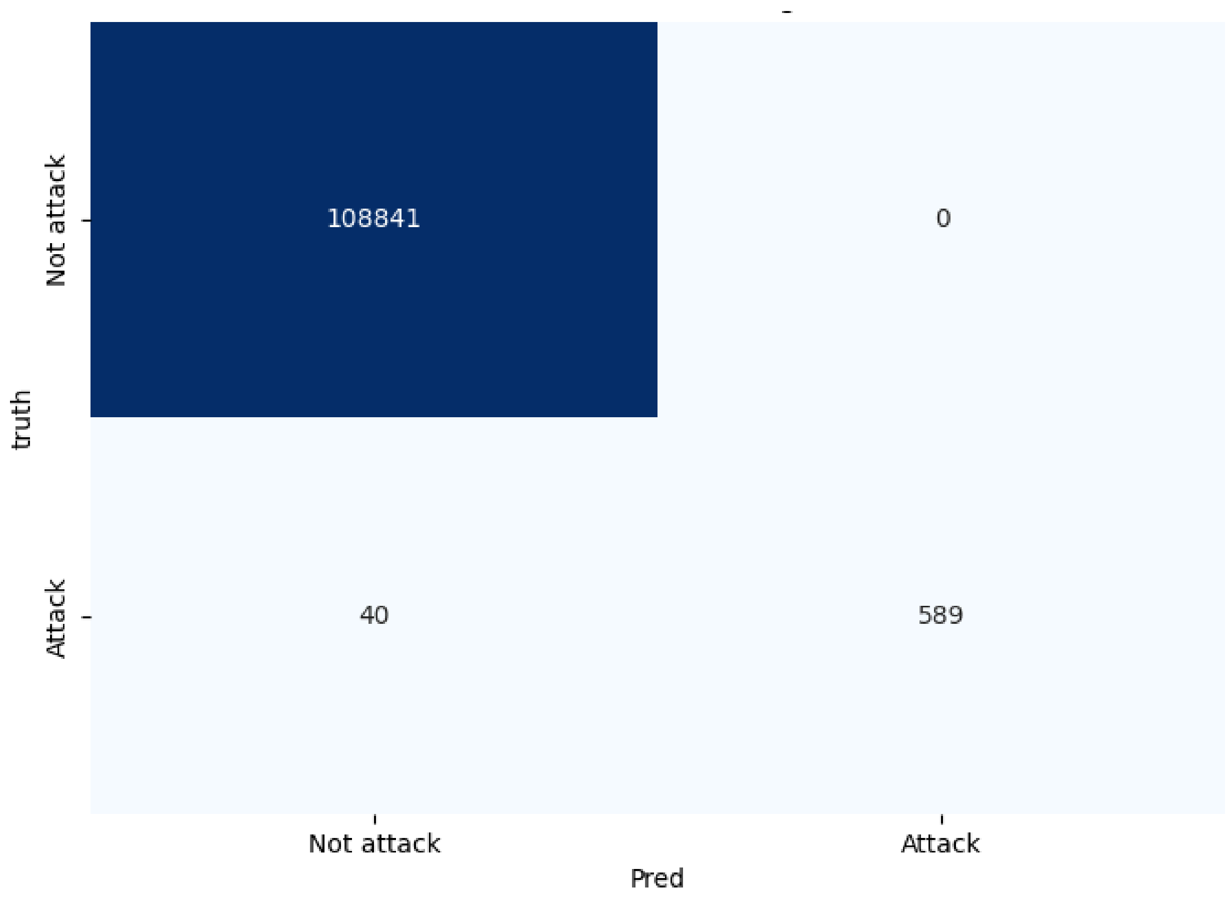
For the first challenge we did cross validation among 4 models and selected the model with the best average accuracy to avoid overfitted models. The random forest classifier has a mean accuracy of ≈ 1 on the ISDX dataset. We separate data in two groups, first only SSH data and second only HTTPWeb data. For the two groups, the chosen model is RandomForest.



Confusion matrix for the challenge 1 (SSH) with a test on a subset of the data. We can understand that the model is not making any mistake on this data. But the sample is really tiny, we cannot affirm that the model is perfect.

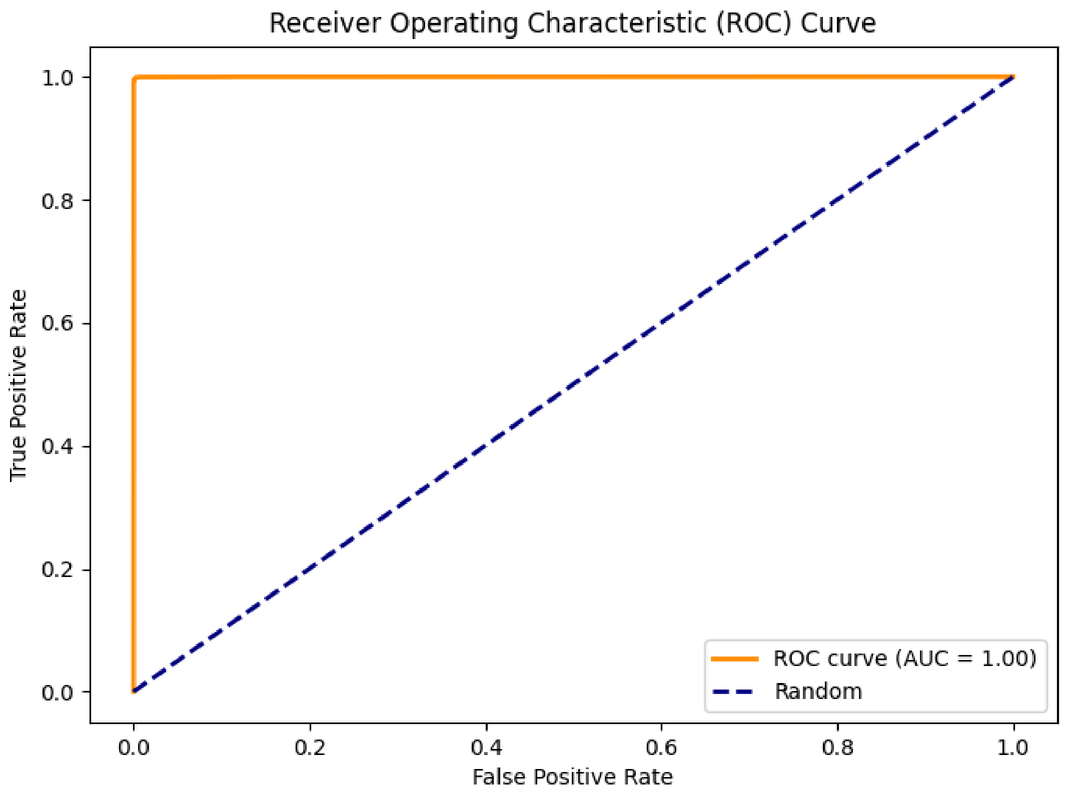


This is the ROC Curve for the model on SSH data. This result makes sense because we don't have any mistakes on the confusion matrix.



Confusion matrix for challenge 1 on the second data group. This includes all HTTPWeb data. This time there are some errors. We see that the main problem is “real attack” predicted as “no attack”, we call this a false negative. In the intrusion detection case, we prefer having false positives rather than false negatives.

Because we had to re-execute the script a few times to get the good result file and the training was taking a few hours, we made the script save and load the model when possible.



This is the ROC curve for the second data group. We have almost the same result as the first data group. It is because the percentage of mistakes is so small that we can't really see it on the curve.

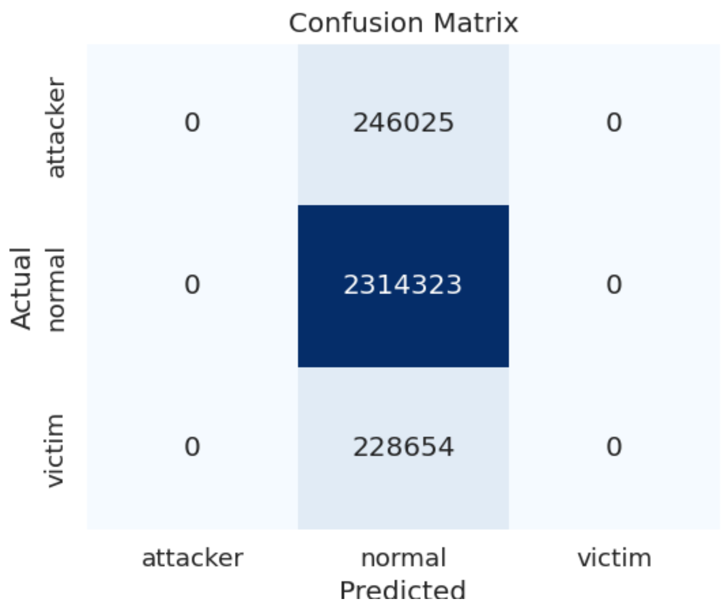
Challenge 2

For the second challenge we did not cross validate models because it was too time consuming. We split the train set into train and test sets with a ratio of 0.33 and measured the mean accuracy of the RandomForestClassifier, MLP and GaussianNB.

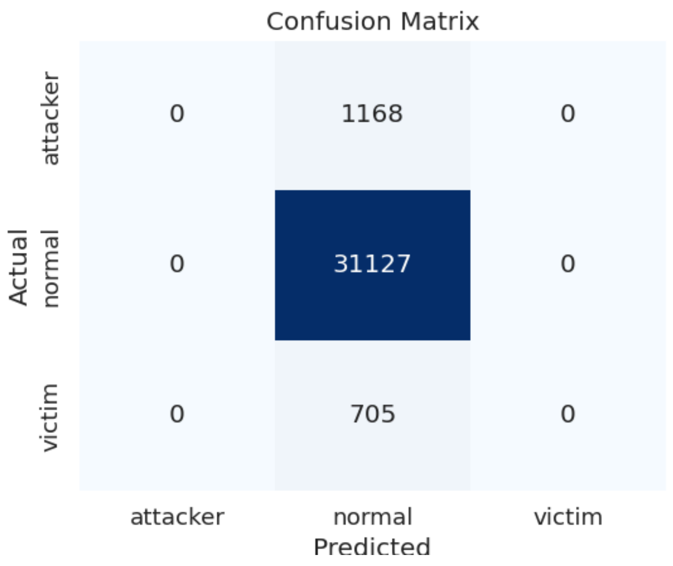
```
GaussianNB()
mean accuracy: 0.08210535524893851
RandomForestClassifier()
mean accuracy: 0.8293913019782704
MLPClassifier(max_iter=1000)
mean accuracy: 0.8293913019782704
```

We tried to evaluate a KNN classifier with k=5 then k=3 but we could not reach the end, it was too time consuming. The train dataset contains 8,451,519 flows. RandomForest and MLP having the same accuracy seems odd.

After doing a confusion matrix with the same data split we see the model always classifies data as normal



We did think the model was overfitted because of the data size and reduced it to 100,000 but got the same anomaly.



We think the error comes from the preprocessing of the data but we couldn't go further because time was lacking.

Conclusion & perspectives

To conclude, one of our main problems was the organization/communication. Solving all the big data related problems on our VPS was too time consuming. We were working at the same time on the VPS and on our computer. We should have used GIT from the beginning. Another problem was that we didn't capitalize on each "run/train" we started. We repeated close "train" multiple times.
We could have used a framework like "optuna" with cross validation to fine tune hyperparameters of the random forest classifier. Currently we use the default hyperparameters provided by scikit-learn.
We saw in the second challenge how the confusion matrix could help spot anomalies in the classifier when the accuracy does not.

User Manual

1. Install elasticsearch, deactivate all the security features in /etc/elasticsearch/elasticsearch.conf, run the service with `systemctl start elasticsearch`
2. Install the datasets by running the install_data.sh script.
3. Install the virtual environment by running the install_virtualenv.sh script.
4. Enter the python virtual environment by running `source env/bin/activate`
5. Load the ISDX dataset in elasticsearch by running data_to_elastic_db.py
6. You can see data has been successfully loaded by running api_demo.py
7. Download the challenge 1 data in moodle
8. Run challenge1.py to generate the result file.
9. Run challenge2.py to generate the challenge 2 result file (it can take a while).
10. You can always run cross_validation.py to cross validate models on the ISDX dataset for a specific protocol or application.