# Report ML SECU : Data Pipeline for Anomaly Detection

Bastien Pouessel
Gilles Recouvreux
Maxence Oden
Raphaël Mourot-Pelade
Vlad Argatu

Promotion 2024

November 2023

# Table Of Contents

# 1 Introduction

The primary objective at the heart of this endeavor is to harness the power of a comprehensive data chain, meticulously crafted to serve the noble purpose of anomaly detection. Our specific focus lies within the realm of tracking and identifying cyberattacks, a domain where vigilance and precision are paramount. In the pursuit of this overarching mission, our project shall capitalize on the invaluable "Hardware In The Loop" dataset. This dataset serves as a crucial linchpin, affording us the unique opportunity to delve into the multifaceted intricacies of cybersecurity data. What distinguishes our approach is our ability to conduct these explorations within a meticulously controlled and highly experimental environment, thus allowing us to unravel the subtle nuances of cyber threats and enhance our collective cyber defense capabilities.

# 2 Datasets

In this presentation, we will explore two datasets that are invaluable resources for researchers working in the domain of cybersecurity and the validation of solutions such as Intrusion Detection Systems (IDS) that leverage artificial intelligence and machine learning techniques. These datasets have been specifically curated to aid in the detection and categorization of threats in the context of Cyber Physical Systems (CPS). The data for these datasets was acquired from a hardware-in-the-loop Water Distribution Testbed (WDT), designed to emulate the flow of water between eight tanks through the use of solenoid valves, pumps, pressure and flow sensors.

## 2.1 Physical dataset

The physical dataset is a comprehensive collection of measurements obtained from sensors, solenoid valves, and pumps connected to Programmable Logic Controllers (PLCs) within the WDT. These measurements were recorded at a rate of one sample per second and were stored in a CSV file by a historical data recorder (Historian). This dataset consists of 41 features, each of which represents a specific aspect of the physical process, providing a detailed view of the system's behavior.

Features include but are not limited to :

— Sensor states ;
— Pump states ;
— Solenoid valves states ;
— Timestamp.

Each record in this dataset is labeled based on the presence of anomalies. A record is labeled '0' if it represents normal operation and '1' if it indicates an attack or anomaly. This labeling is determined based on attack logs, noting the start and end times of each attack, as well as the type of attack.

The physical dataset is a valuable resource for studying the impact of cyber and physical attacks on the physical process within a CPS. Researchers can use this dataset to develop and validate intrusion detection and anomaly detection algorithms that focus on the physical aspects of CPS.

## 2.2 Network dataset

The network dataset comprises data related to the traffic exchanged within the Supervisory Control and Data Acquisition (SCADA) network of the WDT. This data was captured using

Wireshark software and subsequently processed to extract relevant features.

Features in the network dataset include :

— Source and destination IP addresses ;
— Source and destination MAC addresses ;
— Source and destination ports ;
— Protocols ;
— TCP flags ;
— Packet payload sizes ;
— MODBUS function codes and values ;
— Number of packets from the same source or destination in the last 2 seconds.

Similar to the physical dataset, records in the network dataset are labeled based on the presence of anomalies. Records are labeled '0' for normal traffic and '1' for malicious or anomalous traffic. These labels are derived from analysis of attack logs.

The network dataset is an essential resource for studying the effects of cyberattacks on network traffic in a CPS. Researchers can utilize this dataset to develop and evaluate intrusion detection systems that focus on identifying network-based threats.

These two datasets, the physical dataset and the network dataset, provide a rich source of information for researchers and practitioners in the field of cybersecurity. By employing these datasets, researchers can develop, test, and validate innovative solutions for the detection and categorization of threats in Cyber Physical Systems (CPS). These resources are available for non-commercial research purposes and can be accessed freely via the provided link [1], with due credit to the authors.

## 3 Exploratory Data Analysis

In order to design, train and evaluate models, an exploration of the available dataset is required. This step is crucial as it allows us to identify the correlations in the dataset as well as the information that is the most crucial for the classification process. This, in turn, produces high quality preprocessing and significantly improves the performances of any model.

In this study, the exploratory data analysis is performed on both network and physical dataset separately. We produce similar plots for both the datasets. The data used is the concatenation of 'attack_1', 'attack_2' and 'attack_3' for the network part and 'phy_att_1', 'phy_att_2', 'phy_att_3' for the physical part. We intentionally put aside the 'normal' and 'phy_norm' csv files as the attack files already have enough normal labels in them and are thus sufficient for training any model.

1. ieee-dataport.org

## 3.1 EDA - Network

### 3.1.1 General Overview

The network dataset has tens of millions of samples. It is made of recorded traffic on the water treatment plant's network. Among all the recorded fields, we decided to drop the following columns :

— Source MAC, source port and source IP Address as those data can easily be controlled by an attacker and could thus compromise the resilience of all models trained on them as a model could easily learn to make its prediction only based on those fields.

— Destination MAC and destination IP Address as we wish to have a model resilient to the default network configuration of the power plant.

— The modbus_response column, as most of the data is missing from it.

The destination port column and the TCP flags column are considered as categorical instead of numerical, as it better represents their role in network communication. The destination port represents a service with which an external agent communicates, the TCP flags represent metadata for the communication protocol.

Attackers could use both to their advantages in an attack setting and we expect to detect this kind of behavior in our models.

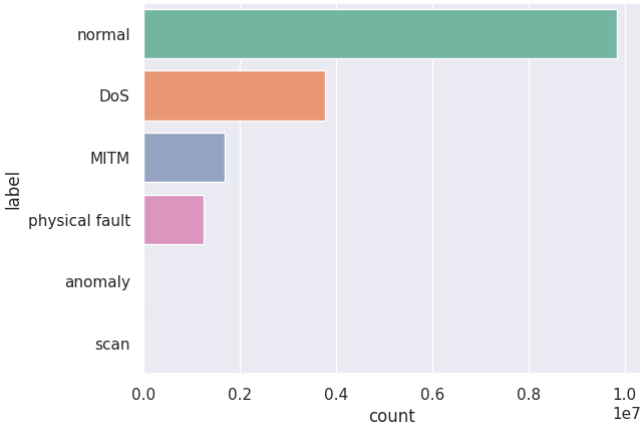



FIGURE 1 – Number of Labels for each class in our network dataset

Figure 1 shows the distribution of labels in the dataset. Six types of labels can be identified :

1. **Normal** : corresponding to normal traffic
2. **DoS** : corresponding to a Denial of Service attack
3. **MITM** : corresponding to a Man In The Middle attack
4. **Physical Fault** : corresponding to a physical attack
5. **Anomaly** : corresponding to an anomaly in the dataset
6. **Scan** : corresponding to a scanning attack

### 3.1.2 Numerical Analysis

Figure 2 shows the pair plot of the distribution of the numerical data of the network dataset. This plot shows the distribution of each variable according to the other variables and, on the diagonal, the distribution of the variable.

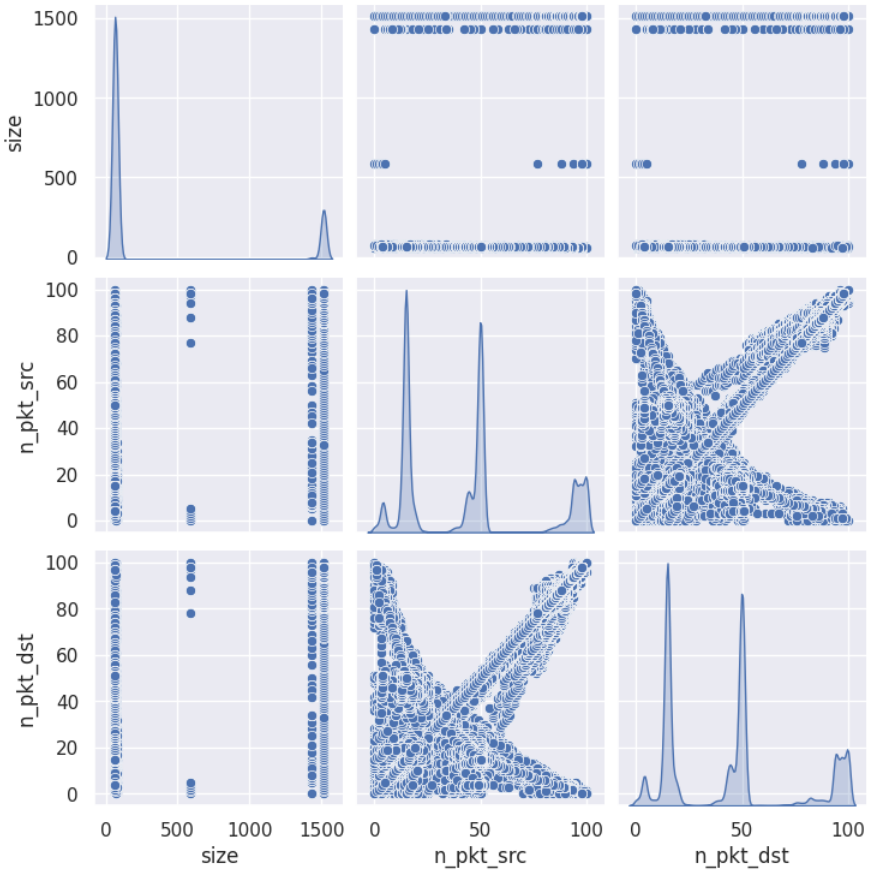



FIGURE 2 – Pair plot of the numerical data of the network dataset

From these graphs, we can deduce that 'n_pkt_str' and 'n_pkt_dst' have similar distributions. Moreover, the data distribution have the same intensities, only one of them could be kept in order to train machine learning models without loss of generality.

Evaluating the correlation matrix (Figure 6) we can deduce that the numerical data of the network dataset are heavily correlated. We can also see that 'n_pkt_str' and 'n_pkt_dst', although following the same distribution, are not really correlated.

Computing the mean of these distributions we find out that, although similar, they are not exactly the same with 'n_pkt_str' having 44.00 and 'n_pkt_dst' 43.66.

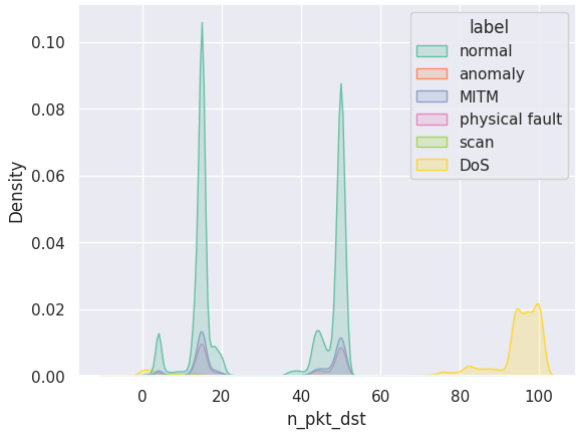



FIGURE 3 – Distribution of the labels as function of n_pkt_dst


FIGURE 4 – Distribution of the labels as function of n_pkt_src


FIGURE 5 – Distribution of the labels as function packet size

Figure 3 and Figure 4 show that the values of 'n_pkt_str' and 'n_pkt_dst' are crucial in detecting Dos type attack as it is manifested when those two values are very high, meaning a service is targeted by a single host.

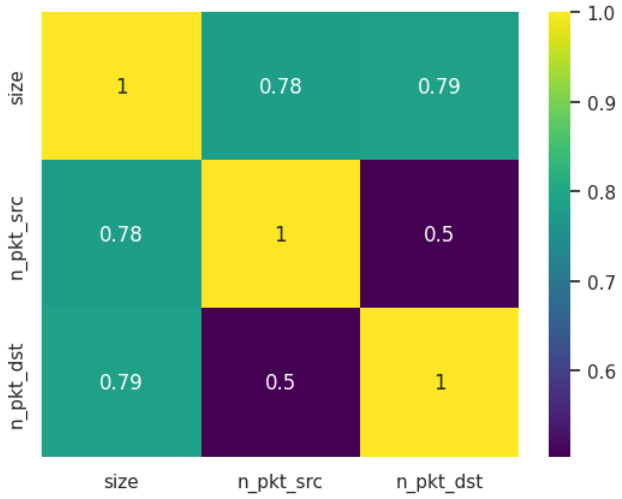



FIGURE 6 – Correlation matrix of the network dataset

Figure 5 shows us that the value of size does not play the most important role for the

discrimination of attack classes, as packet seems to have almost always a constant size.

### 3.1.3 Categorical Analysis

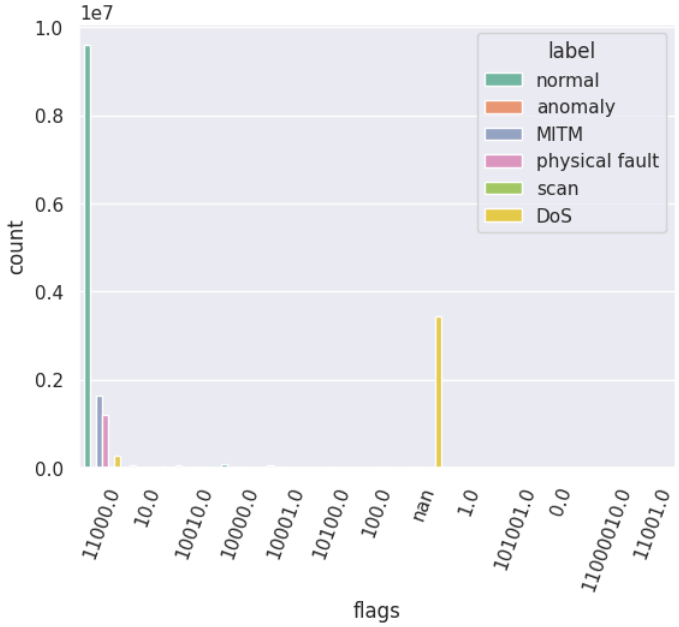



FIGURE 7 – Repartition of classes by network flags

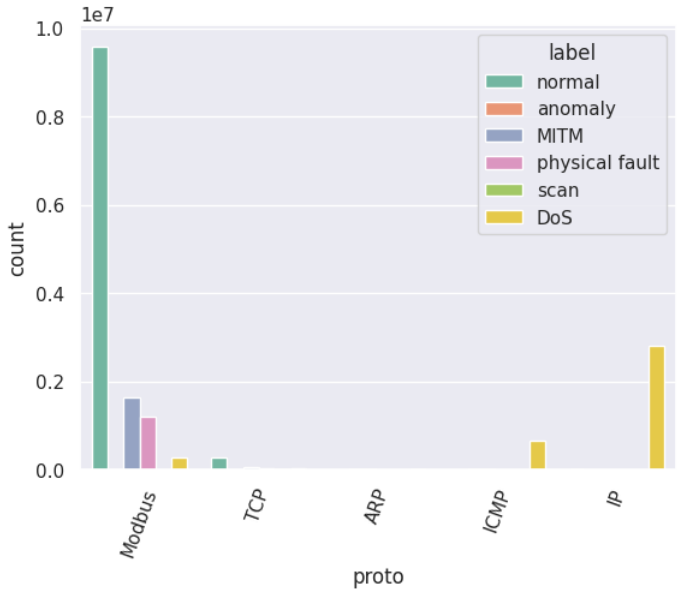



FIGURE 8 – Repartition of classes by network protocol

An analysis of the categorical data shows us that the two significant fields for the protocol classification are the network flags and the network protocol. Both represent a good discrimination factor for the DoS attack in the network dataset.

Figure 8 shows that most of DoS attacks are based on ICMP and IP protocols while the other attacks and the normal traffic use the modbus protocol. Figure 7 highlights the fact that TCP flags are missing for most of the DoS attacks. This can be explained by the fact that most of DoS attacks are based on IP and ICMP (Figure 8) which are protocols operating at a lower level than TCP.

## 3.2 EDA - Physical

### 3.2.1 General Overview

As far as the physical dataset is concerned, a split on the data type is once more performed. We use as numerical data the columns referring to the tank and the ones recording the flow sensors. That is 'Tank_1', 'Tank_2', 'Tank_3', 'Tank_4', 'Tank_5', 'Tank_6', 'Tank_7', 'Tank_8' and 'Flow_sensor_1', 'Flow_sensor_2', 'Flow_sensor_4'. Note that 'Flow_sensor_3' is discarded as all the data recorded by this sensor is 0 in our dataset.

The pump and valv columns, as they are only compose of 'true' and 'false' values, are considered as booleans and thus used as categorical data. The values of 'Valv_1', 'Valv_2', 'Valv_3', 'Valv_4', 'Valv_5', 'Valv_6', 'Valv_7', 'Valv_8', 'Valv_9', 'Valv_16', 'Valv_19', 'Valv_21' and 'Pump_3' are discarded as they are always false and thus do not give any additionnal information.

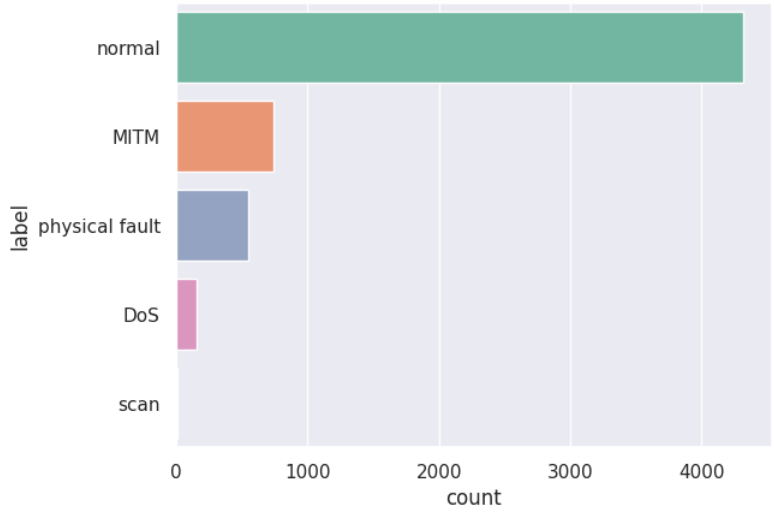



Figure 9 – Number of labels for each class in our physical dataset

We can identify the same labels as the previous dataset which makes sense as both datasets were recorded during the same period and have a record of the same events.

The order of magnitude is lower here, there are only thousands of labels. The dataset is, as the previous one, very unbalanced with a majority of labels associated to normal behavior of the water plant.

Having some labels only related to network attacks, such as MITM, in the dataset tends to indicate that the classification task using this dataset will be a lot harder. This is because, at first sight, a Man in The Middle attack cannot really be detected from a physical point of view

if the attacker is passive. Thus the data labeled MITM could have the same properties as the normal ones.

### 3.2.2 Numerical Analysis

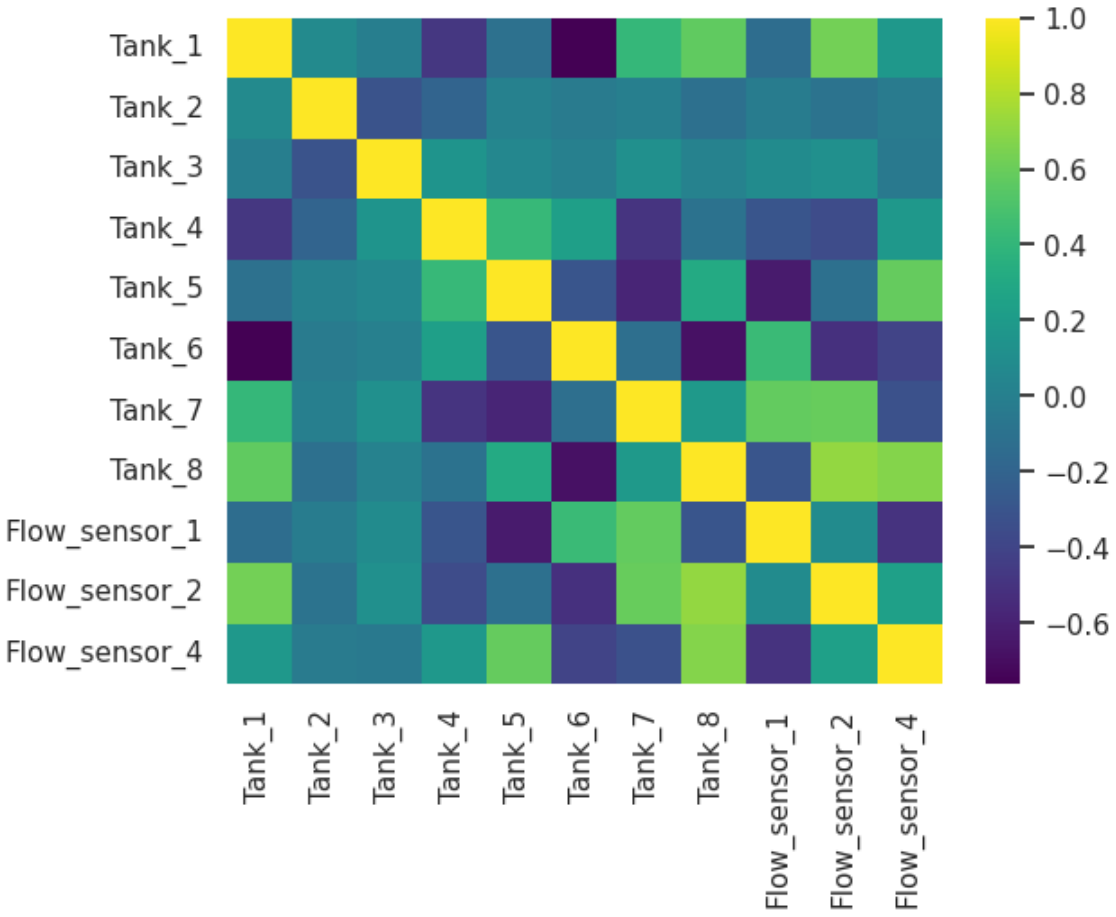



FIGURE 10 – Correlation matrix of the physical dataset

As opposed to the previous dataset, Figure 10 shows that most of the numerical data are not very correlated together.

Moreover, Figure 11 and 12 do not give any direct and obvious hints about the distribution of attacks according to one single variable. This is because, in all figures, the attacks distribution spans accross the same input range as the normal distribution. As a result, no value can act as a real outlier. This makes this part of the dataset much more complicated to exploit than the network.

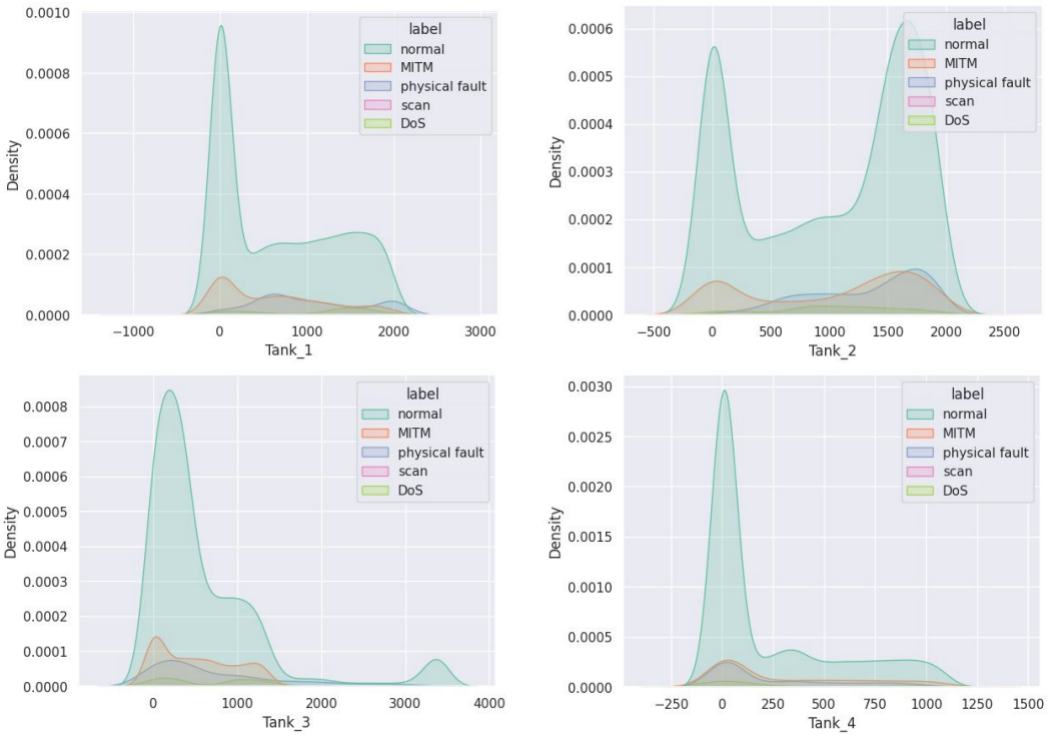



FIGURE 11 – Distribution of the labels as a function of the values of the first 4 tank data, the other tanks follow similar distributions

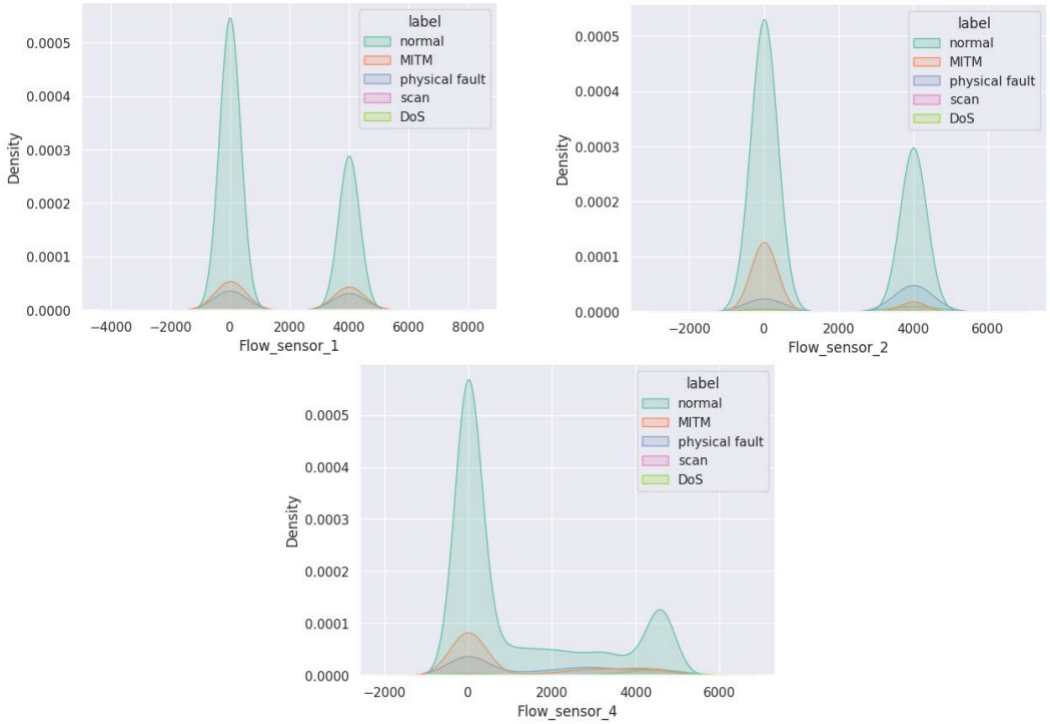



FIGURE 12 – Distribution of the labels as a function of the flow sensor values

### 3.2.3 Categorical Analysis

Figures 13 and 14 give similar information as the numerical data analysis. There is not a clear feature that allows us to easy discriminate an attack. It is also not clear how different categorical data are linked together, hence no obvious simplification can be made for model training.

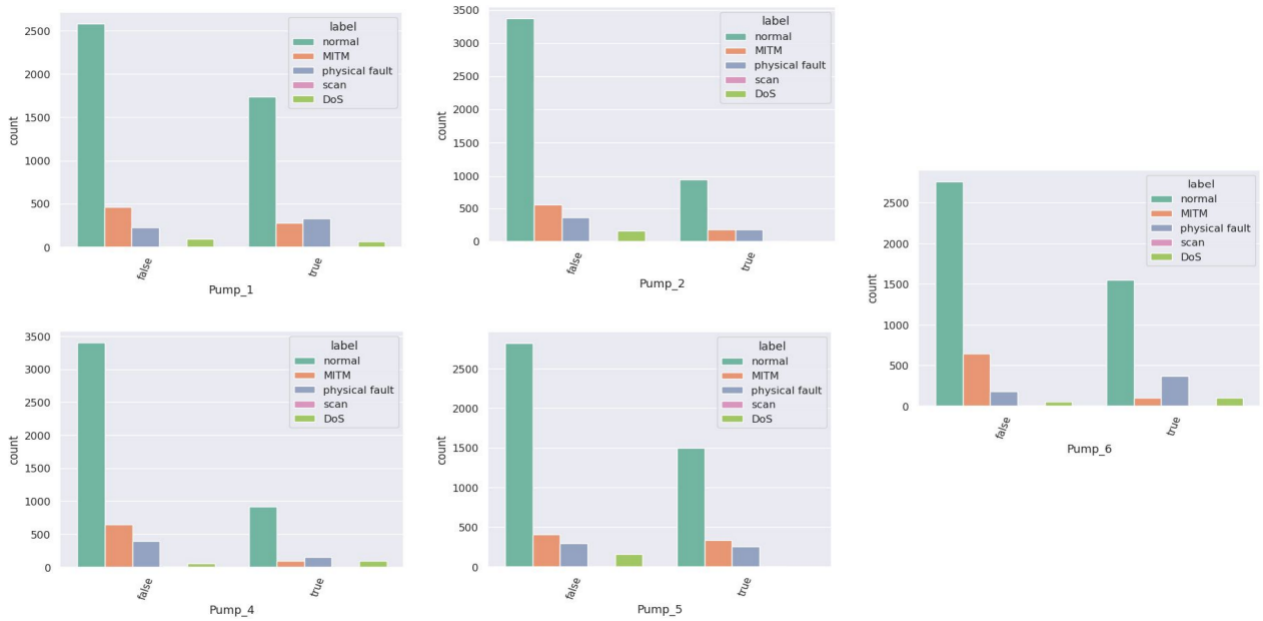



FIGURE 13 – Repartition of labels by pump values

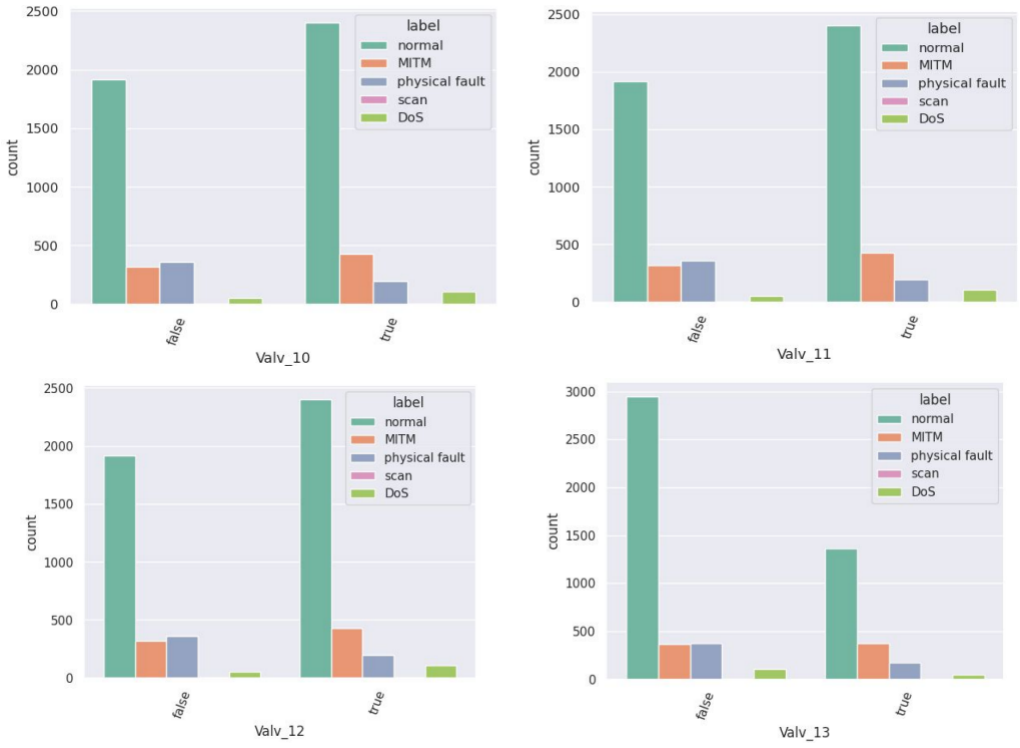



FIGURE 14 – Example of repartition of labels by valv, other valvs follow similar repartitions

# 4 Preprocessing

In our research, we conducted an exploratory data analysis on network and physical datasets. This initial phase provided valuable insights into potential improvements for data quality. We studied the distribution and investigated any missing values within the datasets to understand the unique characteristics of the features and the samples corresponding to each label. Emphasizing the importance of preprocessing, we recognized that it is an important step towards maximizing the efficacy of our models, as it significantly affects the outcome of the machine learning algorithms.

We have mutualized as much code as possible to ensure reproducibility and to prevent inconsistencies during preprocessing. To facilitate this, we decided to develop a library containing custom functions designed for loading both the network and physical datasets.

## 4.1 Data Cleaning

Data cleaning is an important step that simplifies the rest of the process and builds on the insights gained from the exploratory data analysis.

Data cleaning was tailored to each section of the dataset. We implemented different steps for the network and physical datasets.

The network dataset underwent processing through the following steps :
— We began by concatenating all CSV files into a single file to consolidate the data.
— We removed columns that contained exclusively invalid values or exhibited identical distributions, which were deemed unnecessary.
— We retained only those labels that were representative, specifically excluding physical, anomaly, and scan categories.

We opted to retain only 'attack1.csv', 'attack2.csv', and 'attack3.csv' as these files already included a substantial number of 'normal' labels, which was beneficial given the existing imbalance in the dataset. Subsequently, we concatenated these files to form a comprehensive dataset for analysis.

Some columns, such as n_pkt_dst and n_pkt_src, exhibited identical distributions ; therefore, we chose to remove one of these to streamline the dataset. Similarly, we eliminated the modbus_response column because it predominantly contained NaN values. This feature provided negligible variance, which is critical since we employed statistical machine learning models that rely on variability within features to make accurate predictions.

Some labels were significantly underrepresented, and despite attempts at oversampling and undersampling, gaining insights into these was challenging. For the network data, we chose to focus on the *DoS* attack, *MITM*, and *normal* labels. We narrowed our task to specifically classify types of network attacks, which led to our decision to exclude the *physical* label, as it was irrelevant to our network data context.

The physical dataset underwent processing through the following steps :
— We verified and corrected the encoding of the CSV files to ensure compatibility
— We began by concatenating all CSV files into a single file to consolidate the data.

— We removed columns that contained exclusively invalid values or exhibited identical distributions, which were deemed unnecessary.
— We removed unnecessary columns, corrected typographical errors in column names, and resolved inaccuracies in the labels
— We converted relevant columns into one-hot encoded features

The physical dataset consists mainly of categorical features, which necessitates less preprocessing. However, it required meticulous attention to correct typographical errors, such as nomal instead of normal in the label column, as well as misspellings in other column headers, such as Lable instead of Label.

To maintain consistency with the network dataset, we opted to retain only the *normal* and *physical* labels for the physical dataset. This decision was informed by the fact that physical data does not offer insights into Man in the Middle (MITM) and Denial of Service (DoS) attacks. Consequently, our analysis has been refocused towards a binary classification problem.

## 4.2 Transformation

The dataset comprises both categorical and numerical features. Tree-based and ensemble algorithms, such as RandomForest and XGBoost, are not sensitive to scaling. However, scaling is crucial for other machine learning algorithms that calculate distances between observations. Since machine learning models cannot interpret strings, it is necessary to convert categorical data into numerical format. There are multiple methods to encode these categorical features into numerical columns.

To streamline the transformation process with its various steps, we decided to utilize a sklearn pipeline. This approach offers significant advantages, including ease of reuse and compatibility with production environments. Pipelines are well-suited for deployment because they help ensure consistency in the processing of both training and test data. By integrating the pipeline just prior to model training, we can take advantage of applying the same transformation steps that were fitted to the training dataset directly to the test dataset, thus maintaining consistency and reducing the risk of data leakage.

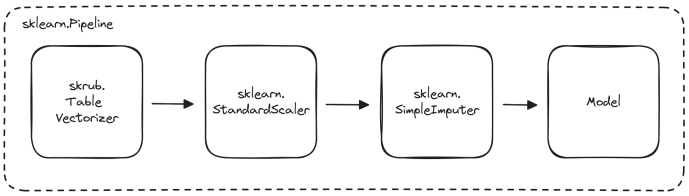



FIGURE 15 – Pipeline for data transformation

The Pipeline is composed of the following step :
— *TableVectorizer* : This component selects the appropriate encoding method for categorical features based on their cardinality, optimizing the encoding process.
— *StandardScaler* : This standardizes numerical features by removing the mean and scaling to unit variance.
— *SimpleImputer* : It handles missing values by imputing them with statistical measures

(mean in our case)

The TableVectorizer was introduced in *skrub*, a Python library developed by the scikit-learn team. *skrub* offers a suite of tools for processing tabular data, particularly for encoding. According to the *skrub* documentation, the TableVectorizer provides a turnkey solution by applying different data-specific encoders to various columns. It makes heuristic decisions that, while not necessarily optimal, typically serve as a very good baseline. It integrates well into pipelines and can adapt if the cardinality of the data changes.

The algorithm determines the encoding method based on the cardinality of the categorical columns and decide which algorithm to apply between the following algorithms :

— OneHotEncoder
— GapEncoder
— MinHashEncoder
— SimilarityEncoder

In our case, the low cardinality of the data meant that we utilized only the OneHotEncoder. Making the decision based on cardinality not only prevents excessive growth in the dataset's dimensionality but also avoids a significant increase in the time complexity of the algorithm.

Scaling the data is essential for algorithms influenced by distance metrics. For instance, SVM is sensitive to this parameter, unlike Decision Trees and Random Forests. However, applying scaling does not negatively impact these models, so we have decided to proceed with this approach on all the models.

The SimpleImputer is primarily used for testing purposes, as our training data does not contain NaN values in numerical features. However, to ensure our pipeline is robust, we have decided to include it.

## 4.3 Working with Time series

The dataset is constructed from time series data of consecutive attacks that occur sequentially. This inherent structure of the dataset reflects the chronological progression of events, which is crucial for any analysis or predictive modeling.

Time series data require specialized handling ; it is imperative that we do not apply random shuffling to the dataset. Shuffling would disrupt the time-dependent relationships within the data, and introduce a bias look-ahead bias.

It is essential to perform a *stratified* split of the dataset to ensure that both training and test sets contain a representative distribution of attack patterns. This stratification must be done with consideration to the temporal order, maintaining the sequence integrity.

The dataset should be split according to the following schema :

We introduce a function called *timesplit* that lets us divide the dataset into two parts : one for training and one for testing. We decided to use 80% of the data for training and 20% for testing. We have checked that the two splits are balanced and that no category is underrepre-

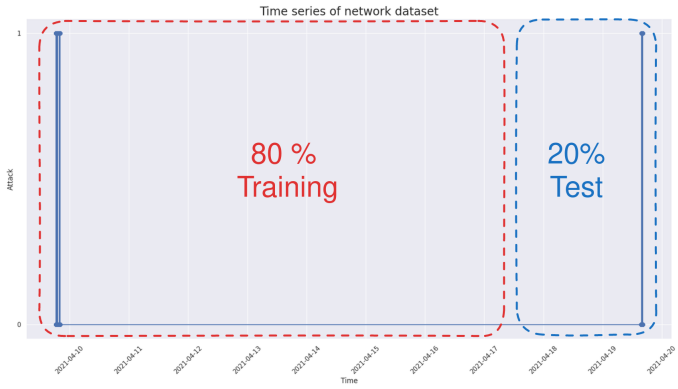



FIGURE 16 – Time series split

sented compared to the entire dataset. This method should be the best way to avoid any bias related to the order of events in our approach.

One of the major issues was the time complexity of some statistical classifiers, which was exacerbated by the fact that the dataset contains millions of samples. Additionally, the use of one-hot encoding introduced more columns, leading to a high-dimensionality problem for the network dataset. To avoid excessive time complexity, we sampled the dataset in a stratified manner to maintain the same class representation and to not eliminate the unbalanced nature of the dataset. We were also careful to preserve the temporal aspect, so we decided to sample every 150th step. This approach allowed us to reduce the dataset to only a few hundred thousand samples while keeping the critical time-based structure.

## 4.4 Balancing the dataset

One of the primary issues with the dataset is its apparent imbalance between the normal labels and others. Our exploratory data analysis revealed a ratio of 5 to 1 between normal instances and those labeled as 'Man in the Middle' attacks. To mitigate this issue, we have explored multiple methodologies tailored to balance the dataset effectively.

Firstly, we attempted to use random undersampling on the majority class utilizing the imbalanced-learn library. This method involves randomly reducing the number of samples in the majority class, which in our case is the 'normal' category. The objective was to decrease the count of 'normal' labeled samples in the dataset to favor the representation of the underrepresented 'Man in the Middle' samples for the network dataset.

The second approach we employed involved using an algorithm such as SMOTE to oversample the minority classes. This technique synthetically generates new examples of the minority class, thereby reducing the disparity between the minority and majority classes. The aim is to provide the classifier with more instances of the minority class to improve its ability to recognize these underrepresented samples.

Lastly, we attempted to compare conventional classifiers with a Balanced Random Forest, which incorporates undersampling within the model to address the imbalance in the data. The purpose of introducing this classifier was to make a comparison between the standard Random

Forest and the Balanced Random Forest to evaluate their performance in handling imbalanced datasets.

SMOTE demonstrated improved results in reducing imbalances ; therefore, we decided to retain it despite the cost of introducing synthetic values and increasing the time complexity. Furthermore, we conducted a comparison between the Balanced Random Forest Classifier and the traditional Random Forest Classifier. In order to stay consistent with a real use of our algorithm, we have applied only the SMOTE on the training dataset. If the model were put into production, we wouldn't be able to apply oversampling to the testing set because we wouldn't have access to the labels. Therefore, the testing would be done on an unseen and unmodified part of the dataset, which would not be balanced.

# 5 Outlier Detection

We began working with two unsupervised methods, Local Outlier Factor (LOF) and Isolation Forest (IF), to detect outliers. We compared the performance of these models on both the network and physical datasets. It was determined that the most appropriate application of these algorithms was to frame the task as binary classification, distinguishing between 'attack' and 'normal' data.

## 5.1 Local outlier Factor

We use the Local Outlier Factor, an unsupervised approach, to identify potential attacks. According to scikit-learn's documentation, the Local Outlier Factor measures the local deviation of density for a given sample with respect to its neighbors, determining outliers based on the surrounding neighborhood.

An important hyperparameter to consider is the contamination rate in the dataset. We opted to experiment with various values to determine which one yields the best results. To visualize the results, we performed Principal Component Analysis (PCA) with three components. The darker shades of blue indicate the presence of an attack.

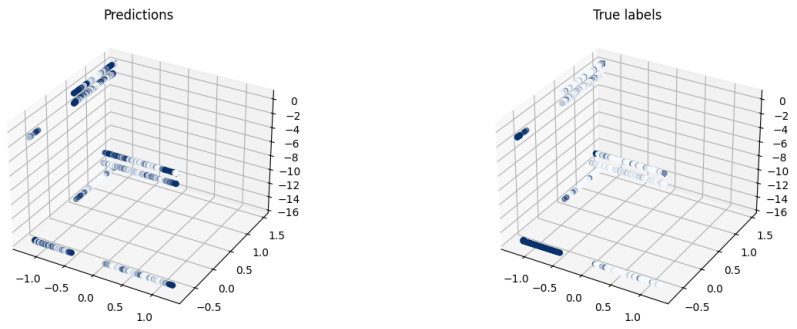



FIGURE 17 – Local Outlier Factor with contamination 0.03 on the network dataset

Through comparison of different performance metrics and visual analyses, we found that setting the contamination parameter to 0.03 delivers optimal results on our netwrok test data. A lower contamination rate fails to detect certain outliers, whereas a higher rate, like 0.05, results in an excessive number of false positives.

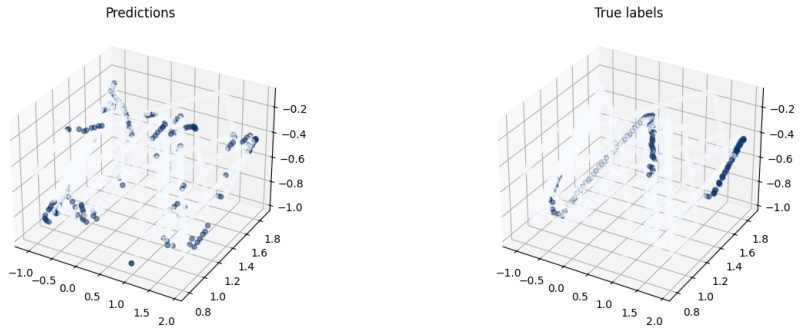



FIGURE 18 – Local Outlier Factor with contamination 0.1 on the physical dataset

## 5.2 IsolationForest

Isolation Forest is an unsupervised ensemble learning method used for identifying outliers. It employs recursive random partitioning of the data. Compared to the Local Outlier Factor (LoF), Isolation Forest offers a lower time complexity and provides a distinct approach by operating globally rather than locally.

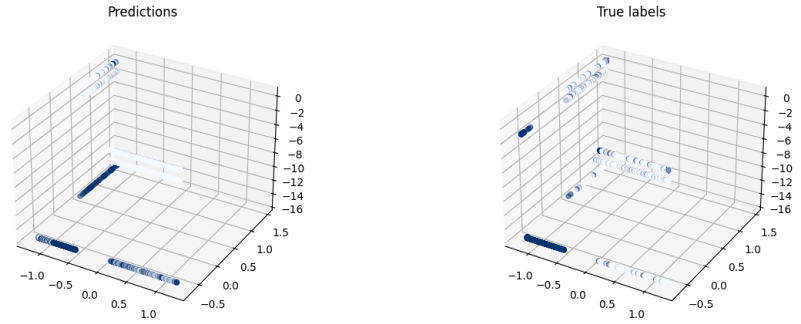
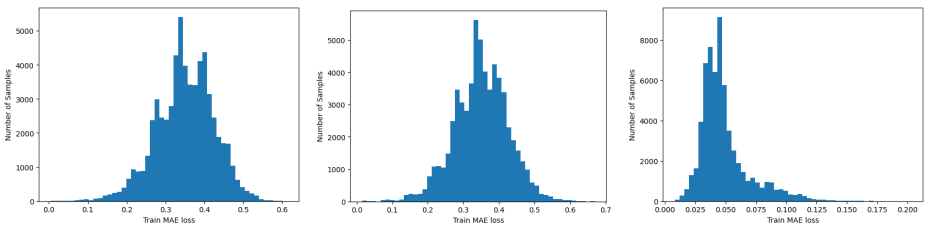



FIGURE 19 – Isolation Forest with contamination 0.01 on the network dataset

## 5.3 LSTM

We use LSTM to predict the next value of a features by giving a sequence. If the predicted value if to far from the reality, we can assume that value is an anomaly. We try to use LSTM on different features :

— n_pkt_src with a reconstruction error threshold of 0.67
— n_pkt_dst with a reconstruction error threshold of 0.61
— size with a reconstruction error threshold of 0.20




(a) Histogram of train MAE loss for n_pkt_dst (b) Histogram of train MAE loss for n_pkt_src (c) Histogram of train MAE loss for size

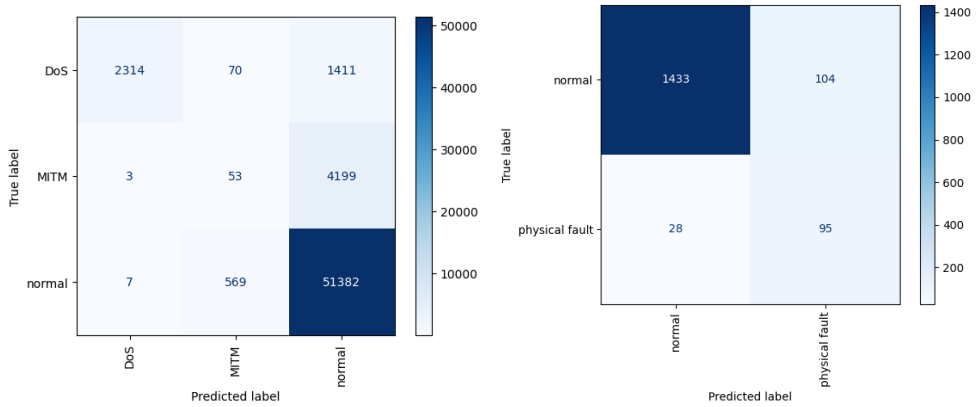
# 6 Attack - Classification

We compared five different models on both the network and physical datasets, following the previously mentioned preprocessing and data transformations. For the network dataset, we pursued a multiclass classification approach with three possible labels : 'normal,' 'MITM,' and 'DoS.' In contrast, we used a binary classification for the physical dataset, distinguishing between 'MITM' and 'DoS.'

The models we evaluated and compared on the same test split are :
— Decision Tree Classifier
— Random Forest Classifier
— Balanced Random Forest Classifier
— LinearSVC
— XGBoost Classifier

## 6.1 DecisionTree

The Decision Tree classifier, a fundamental classification algorithm, employs criteria such as Gini impurity to split nodes and categorize data. It effectively recognizes normal samples, but struggles to detect Man in the Middle (MITM) attacks, showing almost no recall for such incidents. In the case of Denial of Service (DoS) attacks, although it manages some detection, it does not successfully identify the majority of them.



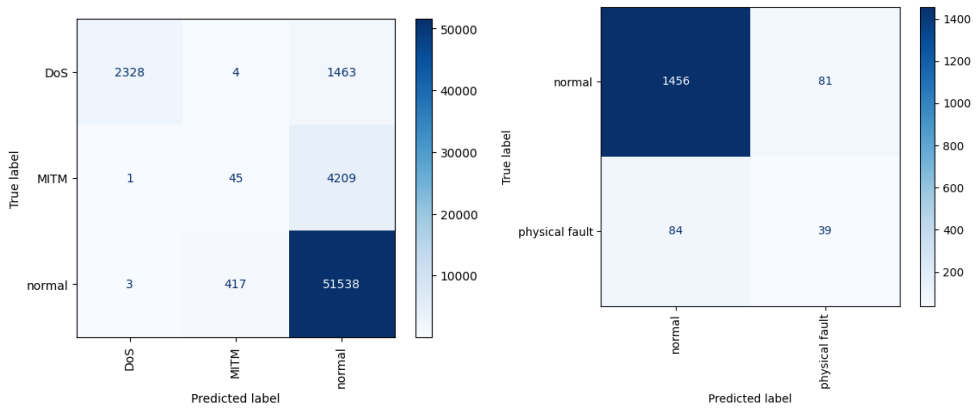

(a) Confusion Matrix on the physical dataset

(b) Confusion Matrix on the physical dataset

The strength of the Decision Tree classifier lies in its low time complexity compared to other algorithms, and in our case, it converges quite effectively.

## 6.2 RandomForest

Random Forest is an ensemble method that combines multiple decision trees. It typically outperforms simpler tree algorithms. However, in our case, the results did not meet our expectations. The Random Forest did not yield better outcomes than the DecisionTreeClassifier, and it also incurred higher time and memory complexity. The results were quite similar to those obtained with the DecisionTreeClassifier.



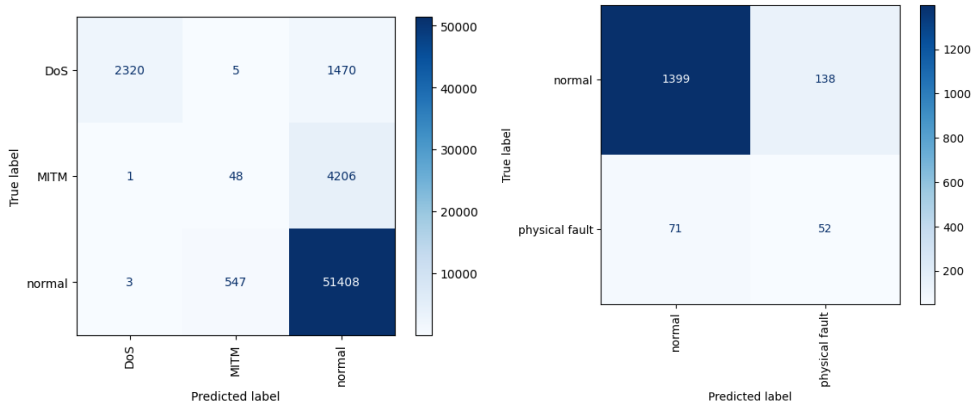

(a) Confusion Matrix on the physical dataset

(b) Confusion Matrix on the physical dataset

The results on the network dataset are comparable to those of the Decision Tree, many normal labels and DoS attacks are correctly classified, but very few MITM attacks are accurately identified. For the physical dataset, it detects significantly fewer attacks than the Decision Tree does.

## 6.3 BalancedRandomForest

The Balanced Random Forest operates on the same principles as the standard Random Forest but introduces random under-sampling of each bootstrap sample to achieve balance. Given the major challenge of underrepresentation of attacks in our dataset, we aimed to evaluate the benefits of using Balanced Random Forest and determine if it could offer improvements compared to the use of SMOTE alone.



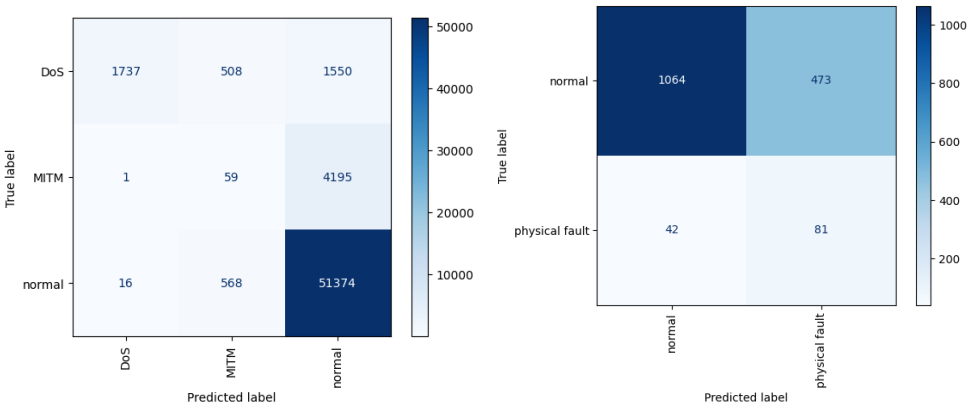

(a) Confusion Matrix on the physical dataset

(b) Confusion Matrix on the physical dataset

The Balanced Random Forest performed almost identically to the standard Random Forest, showing no significant advantage. In our scenario, SMOTE appears to be the better alternative, as it aids in enhancing the performance of all classifiers. Across both datasets, the balanced accuracy and the Matthews correlation coefficient remained consistent.

## 6.4 LinearSVC

Support Vector Machines (SVM) are a group of classifiers that have the advantage of being effective in high-dimensional spaces and can learn with various types of kernels. We chose to use LinearSVC because it converges faster, and the kernel choice did not result in significant performance changes.

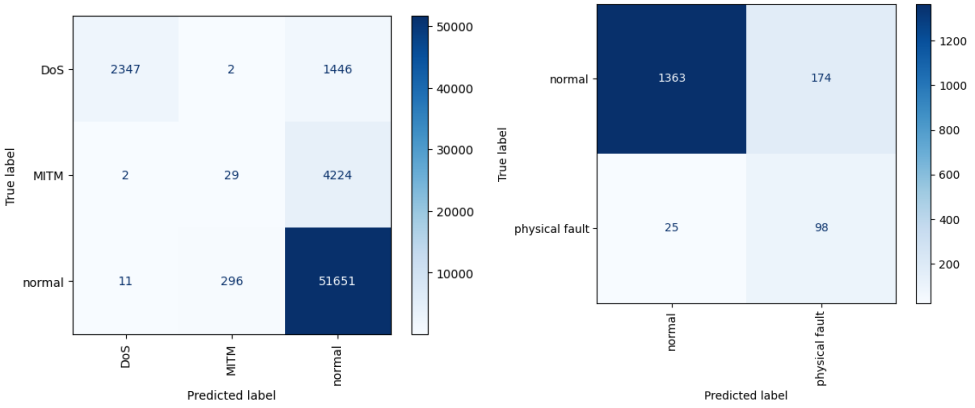



(a) Confusion Matrix on the physical dataset (b) Confusion Matrix on the physical dataset

LinearSVC did not yield favorable results compared to other models, particularly on the physical dataset, where it misclassified many normal samples as physical faults. In terms of security, this would result in a high number of false positives, potentially incurring significant costs. Moreover, its higher time complexity, when compared to models like the Decision Tree, makes it a less desirable choice for our task.

## 6.5 XGBoostClassifier

The XGBoost Classifier is frequently regarded as a state-of-the-art algorithm for tabular datasets. It typically outperforms other machine learning techniques, especially with large volumes of data and high-dimensional problems. In our case, it performed almost exactly as well as the Decision Tree Classifier, but required more time to converge..




(a) Confusion Matrix on the physical dataset (b) Confusion Matrix on the physical dataset

The model's performance is almost identical to that of the Decision Tree, although on certain metrics, it performs slightly worse than the Decision Tree. The Decision Tree's low time

complexity makes it one of the more suitable options. However, with a larger dataset, XGBoost could be the more appropriate choice due to its scalability.

# 7 Results analysis

We evaluated our model using various metrics and experimented with different preprocessing strategies. By balancing the dataset through various methods, we explored the impact of omitting certain features and considered the effects of removing some samples.

We assessed the algorithms based on the following metrics :

— *Accuracy* : The ratio of correctly predicted observations to the total number of observations.
— *Balanced Accuracy* : The average of recall obtained on each class, which is particularly useful for imbalanced datasets.
— *Precision* : The ratio of correctly predicted positive observations to the total predicted positive observations.
— *Recall* : The ratio of correctly predicted positive observations to all observations in actual class.
— *F1-Score* : The weighted average of Precision and Recall, taking both false positives and false negatives into account.
— *Matthew Correlation Coefficient (MCC)* : A coefficient ranging from -1 to +1 that provides a balanced measure of an algorithm's quality across all four quadrants of the confusion matrix

## 7.1 Result on the Network Dataset

The table below compares the various models using the metrics previously mentioned on the network dataset. An underlined value indicates the highest result for a specific metric within a column (for a metric).

| | **Accuracy** | **Balanced Accuracy** | **Precision (weighted)** | **Recall (weighted)** | **F1 Score (weighted)** | **MCC** |
|---|---|---|---|---|---|---|
| **Decision Tree** | 0.90 | 0.55 | 0.85 | 0.90 | 0.87 | 0.49 |
| **Random Forest** | 0.90 | 0.54 | 0.85 | 0.90 | 0.87 | 0.47 |
| **Balanced Random Forest** | 0.90 | 0.54 | 0.85 | 0.90 | 0.87 | 0.46 |
| **LinearSVC** | 0.89 | 0.49 | 0.84 | 0.89 | 0.86 | 0.39 |
| **XGBoost** | 0.90 | 0.54 | 0.85 | 0.90 | 0.87 | 0.48 |
| **Local Outlier Factor** | 0.86 | 0.56 | 0.82 | 0.86 | 0.82 | 0.19 |
| **Isolation Forest** | 0.86 | 0.65 | 0.85 | 0.87 | 0.86 | 0.35 |

The analysis of the results allows us to conclude the following :

— Model performance appears to be relatively uniform across the board ; however, metrics such as balanced accuracy and the Matthew correlation coefficient are critical in differentiating effectiveness. Certain classifiers are more sensitive to data imbalance, while others are well-suited to the existing dataset structure.
— The impact of model selection on the final outcomes appears to be secondary ; the primary factors influencing performance are the preprocessing steps and the management of data imbalance.
— The Decision Tree Classifier emerges as the most efficient model. It matches the performance of more complex models while benefiting from reduced computational and memory requirements.
— Unsupervised methods demonstrate satisfactory performance on this dataset. However, caution is advised given that the task was binary classification and not a multi-class scenario, which may affect the interpretability of the results.

## 7.2 Result on the Physical

The table below compares the various models using the metrics previously mentioned on the physical dataset. An underlined value indicates the highest result for a specific metric within a column (for a metric).

| | **Accuracy** | **Balanced Accuracy** | **Precision (weighted)** | **Recall (weighted)** | **F1 Score (weighted)** | **MCC** |
|---|---|---|---|---|---|---|
| **Decision Tree** | 0.92 | 0.85 | 0.94 | 0.92 | 0.93 | 0.56 |
| **Random Forest** | 0.90 | 0.63 | 0.90 | 0.90 | 0.90 | 0.26 |
| **Balanced Random Forest** | 0.87 | 0.66 | 0.90 | 0.87 | 0.89 | 0.27 |
| **LinearSVC** | 0.69 | 0.67 | 0.90 | 0.69 | 0.76 | 0.19 |
| **XGBoost** | 0.88 | 0.84 | 0.94 | 0.88 | 0.90 | 0.48 |
| **Local Outlier Factor** | 0.92 | 0.52 | 0.89 | 0.93 | 0.89 | 0.15 |
| **Isolation Forest** | 0.92 | 0.50 | 0.87 | 0.89 | 0.86 | 0.03 |

The analysis of the results allows us to conclude the following :
— Unsupervised methods like outlier detection have high accuracy but fall short in balanced accuracy, suggesting they may not perform well across all categories of data.
— Decision Trees show promising results. However, to rule out overfitting, we compared them with XGBoost using a high number of estimators. The comparison suggests there is no overfitting and highlights XGBoost as a generally more robust option.
— RandomForest and Support Vector Machine (SVC) methods do not seem well-suited for this particular problem.
— The use of Synthetic Minority Over-sampling Technique (SMOTE) made a minor difference to balanced accuracy, suggesting it might not be as effective for this dataset, particularly in the case of the network data which could be due to the sample size.

# 8 Conclusion

In conclusion, after conducting a thorough exploratory data analysis and benchmarking multiple algorithms on two datasets, we have established a robust pre-processing framework for data transformation.

Our exploratory data analysis and classifiers benchmarks have led us to conclude the following :

- Time dependency of attacks highlights the importance of constructing a reliable baseline for our training and test samples, taking care to avoid introducing any forecasting bias.
- Certain features, like IP and MAC addresses, are context-specific and have been omitted from our methodology to enhance its robustness.
- Simple classifiers, such as Decision Trees, exhibit strong performance on the dataset ; however, they are susceptible to overfitting. XGBoost emerges as a suitable alternative, offering comparable performance with added robustness to overfitting.
- Overall, model performance tends to be similar across the board, underscoring the significance of data preparation, transformation, and adoption of robust methods to handle imbalances.
- Our classifiers have demonstrated a high detection rate for Denial of Service attacks with a minimal false positive rate. However, the detection of Man in the Middle attacks may require additional features, as indicated by moderate balanced accuracy despite extensive preprocessing.
- The physical dataset posed fewer challenges, with similar classifiers having the best outcomes. The potential of unsupervised learning shows promising result. A possibility could be to combine an unsupervised classifier with a supervised classifier.