

# Rapport Projet MLSECU

## Dataset HITL

Arnaud Baradat - Tom Genlis - Théo Ripoll - Quentin Fisch

<b>I. Introduction</b>	<b>3</b>
<b>II. Exploration du Dataset</b>	<b>4</b>
1. Labels	4
a. Attaques	4
b. Types d'attaques	5
2. Features - Network	5
a. Features numériques	5
b. Features catégoriques	8
c. Corrélations globales	10
3. Features - Physical	11
4. Time series	14
<b>III. Détection d'anomalies - Network</b>	<b>16</b>
1. Non-supervised algorithms	16
a. Isolation Forest (IF)	16
b. Local Outlier Factor (LOF)	17
2. Deep Learning	17
a. Neural network	17
b. LSTM	18
3. Classifiers	20
a. Decision Tree	20
b. Random Forest	22
c. XGBoost	24
<b>IV. Détection d'anomalies - Physical</b>	<b>25</b>
1. Non-supervised algorithms	26
a. Isolation Forest (IF)	26
b. Local Outlier Factor (LOF)	26
2. Deep Learning	26
a. Neural Network	26
b. LSTM	29
3. Classifiers	30
a. Decision Tree	31
b. Random Forest	32
c. XGBoost	34
<b>V. Adversarial attack</b>	<b>35</b>
<b>VI. Conclusion</b>	<b>35</b>

# I. Introduction

Pour la sécurisation des infrastructures vitales telles que les systèmes de distribution d'eau, une gestion rigoureuse est essentielle. Le document présenté introduit un ensemble de données élaboré à partir d'une plateforme expérimentale dénommée « Hardware-in-the-Loop » (HIL). Cette plateforme vise à simuler avec précision les réseaux de distribution d'eau urbains dans un environnement contrôlé et reproductible.

La plateforme HIL, conçue avec minutie, combine des éléments physiques, tels que des conduites d'eau, des pompes et des réservoirs, avec des éléments informatiques comme des capteurs, des actionneurs et des algorithmes de contrôle. Cette combinaison permet de générer un ensemble de données détaillé et riche, incluant des informations sur les débits d'eau, les niveaux de pression, l'état des actionneurs, et bien d'autres paramètres pertinents. Les données ont été collectées à intervalles réguliers et couvrent une période spécifique, offrant ainsi une perspective complète du comportement du système dans diverses situations, y compris face à des menaces.

Des scénarios d'attaques, tant physiques que cybernétiques, ont également été intégrés et documentés dans l'ensemble de données. Les manipulations des systèmes de contrôle et les attaques sur les infrastructures physiques sont représentées de manière à faciliter l'analyse des vulnérabilités. La flexibilité et la scalabilité du testbed HIL sont aussi des atouts, permettant d'ajuster et d'étendre les scénarios d'attaque selon les besoins de la recherche. L'accessibilité et les conditions d'utilisation du dataset sont également détaillées pour faciliter son exploitation.

Ainsi, cet ensemble de données se révèle être une ressource précieuse pour un data scientist désireux d'explorer la résilience et les vulnérabilités des systèmes de distribution d'eau. En reflétant fidèlement les conditions réelles et les perturbations potentielles, cet ensemble offre une base solide pour des analyses approfondies. Les données, présentées de manière brute, sont prêtes à être utilisées dans des modèles et des analyses, contribuant ainsi à l'amélioration de la sécurité et de l'efficacité des réseaux de distribution d'eau.

Ce rapport vise à analyser et classifier les données issues d'un barrage hydroélectrique pour renforcer la cybersécurité et la sûreté des opérations grâce à la détection d'anomalies techniques au niveau du réseau, ou du barrage, de manière physique.

## II. Exploration du Dataset

Nous commençons notre rapport par la section exploration du dataset. Celle-ci comporte les observations qui ont définies la suite de notre approche ainsi que les constats préliminaires nécessaires à la bonne compréhension de notre analyse.

Il y a eu besoin d'un prétraitement de la donnée avant toute chose pour permettre une analyse cohérente et lissée de la donnée. En effet, il y a certains problèmes de typologies dans certains noms de colonnes ("lable\_n" au lieu de "label\_n" pour le dataset physique par exemple) ou encore dans les données.

De plus chaque dataset, que ce soit les données physiques ou les données réseau, est composé de 3 jeux de données en scénario "attaque" et un jeu de données en scénario normal.

Nous avons fait le choix de concaténer ces 4 datasets et d'ajouter une colonne "attack" pour différencier de quelle situation vient une ligne. Nous avons bien entendu pris soin de ne pas inclure ces colonnes dans nos algorithmes car elles n'ont qu'un but informatif.

Enfin nous avons ajusté certains types de colonnes ainsi que converti le champ date d'un format humainement lisible en timestamp.

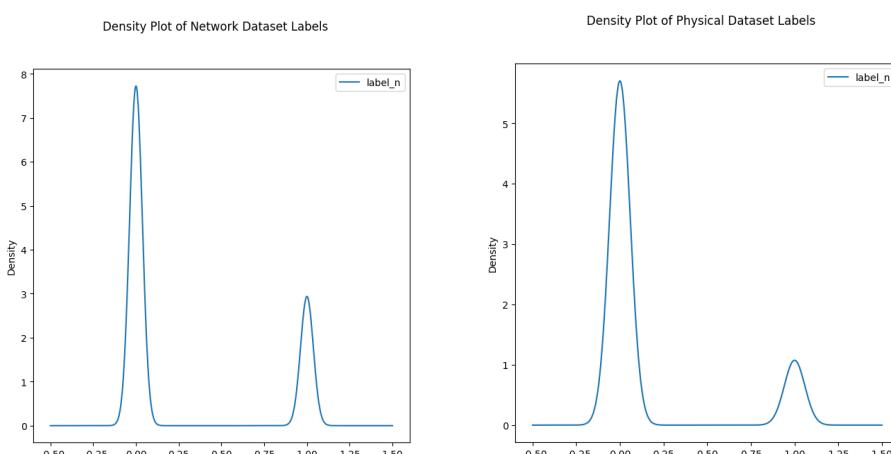
Il est intéressant de noter que le nombre d'échantillons de données que nous avons entre les deux datasets est immense, plusieurs millions pour le dataset réseau contre ~10 000 pour le dataset physique.

### 1. Labels

Commençons par analyser les labels des datasets. Les deux ont le même format se composant de 5 catégories (la colonne **labels**) rassemblées en 2 catégories (attaque ou non, la colonne **label\_n**). Deux approches de classification sont envisageables : une binaire avec **label\_n** et une multi-label avec **label**.

#### a. Attaques

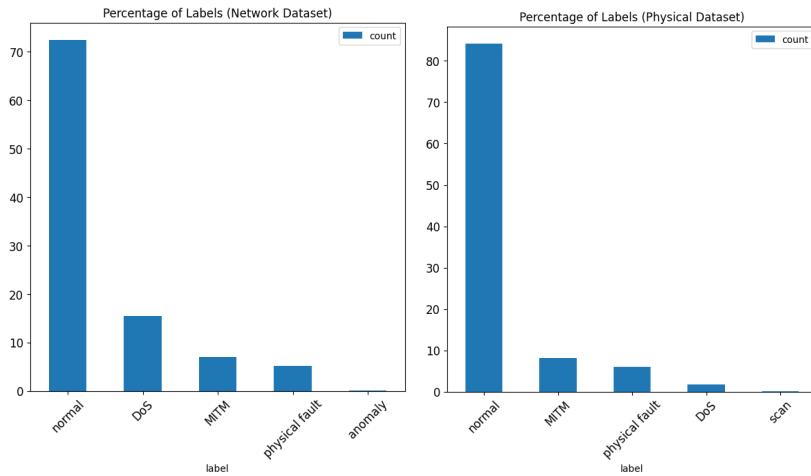
Regardons dans un premier temps la répartition entre les échantillons dans une situation normale et les échantillons dans une situation d'attaque.



Le dataset réseau à une répartition inégale d'échantillons avec seulement 1/3 étant considérés comme une attaque. Lorsque nous jetons un oeil au dataset physique, le constat est pire, seulement 1% des échantillons le sont.

## b. Types d'attaques

Qu'en est-il du type de ces attaques ? Pour cela, nous avons fait des histogrammes de la colonne **label**.



Ici encore, le nombre de labels normaux comparé aux attaques est incomparable. Cependant il y a plusieurs choses intéressantes à noter.

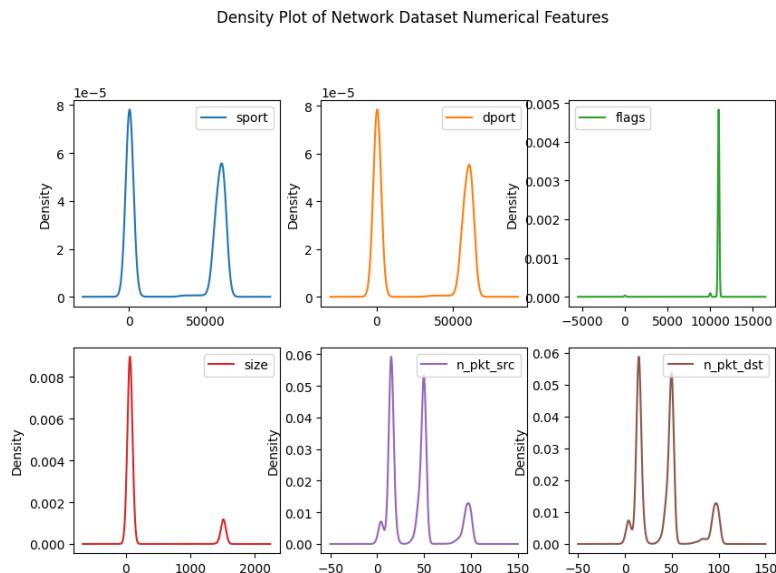
D'abord les attaques **Man in the Middle** et les **Physical Faults** sont en proportion égales sur les deux sets. Ensuite le ratio de **Denial of Service** est de magnitude 10 entre les deux data sets, quoique ceci semble logique de part la nature des attaques DoS. Enfin, il est intéressant de noter qu'il y a deux attaques en proportions infimes - **anomaly** et **scan** - qui, respectivement, n'apparaissent que 3 et 7 fois dans l'ensemble des données.

## 2. Features - Network

Passons maintenant aux features du dataset réseau. Elles sont au nombre de 14 où 6 sont numériques, 7 sont catégoriques et 1 est au format timestamp.

### a. Features numériques

Commençons par les 6 features numériques. Elles sont: **sport**, **dport**, **flag**, **size**, **n\_pkt\_src** et **n\_pkt\_dst**. Jetons un œil à leurs densités respectives.



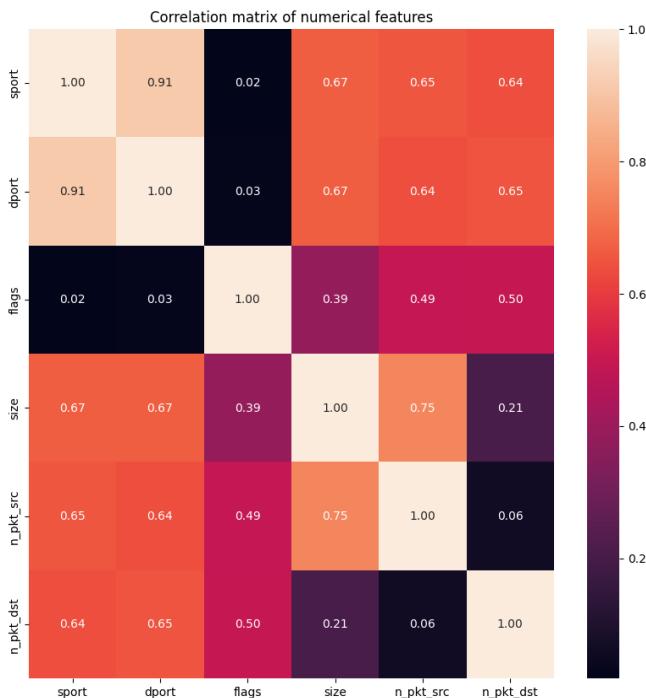
L'analyse de la densité des caractéristiques numériques révèle les tendances suivantes :

Les colonnes **sport** et **dport** présentent deux pics de densité notables, situés respectivement autour des valeurs 0 et 60000.

Pour la colonne **flags**, un pic conséquent est observable autour de la valeur 11000, tandis que très peu de données se situent autour des valeurs 10000 et 0.

En ce qui concerne la colonne **size**, deux pics principaux se distinguent : le premier se situe autour de la valeur 60, tandis que le second, bien que 10 fois moins dense, se trouve autour de la valeur 1500. Enfin, les colonnes **n\_pkt\_src** et **n\_pkt\_dst** montrent des données plus rapprochées avec des pics de densité plus marqués, les deux principaux étant situés autour des valeurs 15 et 50. Ces observations fournissent un aperçu initial utile de la distribution des données dans ces colonnes spécifiques.

En plus de leur répartition, nous avons été curieux de voir la corrélation entre les différentes colonnes numériques:



L'analyse de la matrice de corrélation révèle plusieurs relations significatives entre les différentes variables étudiées.

Une corrélation positive forte est observée entre **sport** et **dport** (0.91), suggérant une tendance commune à l'augmentation ou à la diminution de ces valeurs. De plus, une corrélation positive modérée est identifiée entre **sport**, **dport** et **size**, avec des coefficients respectifs de (0.66) et (0.66), indiquant que la taille des paquets est susceptible d'augmenter avec les valeurs de **sport** et **dport**.

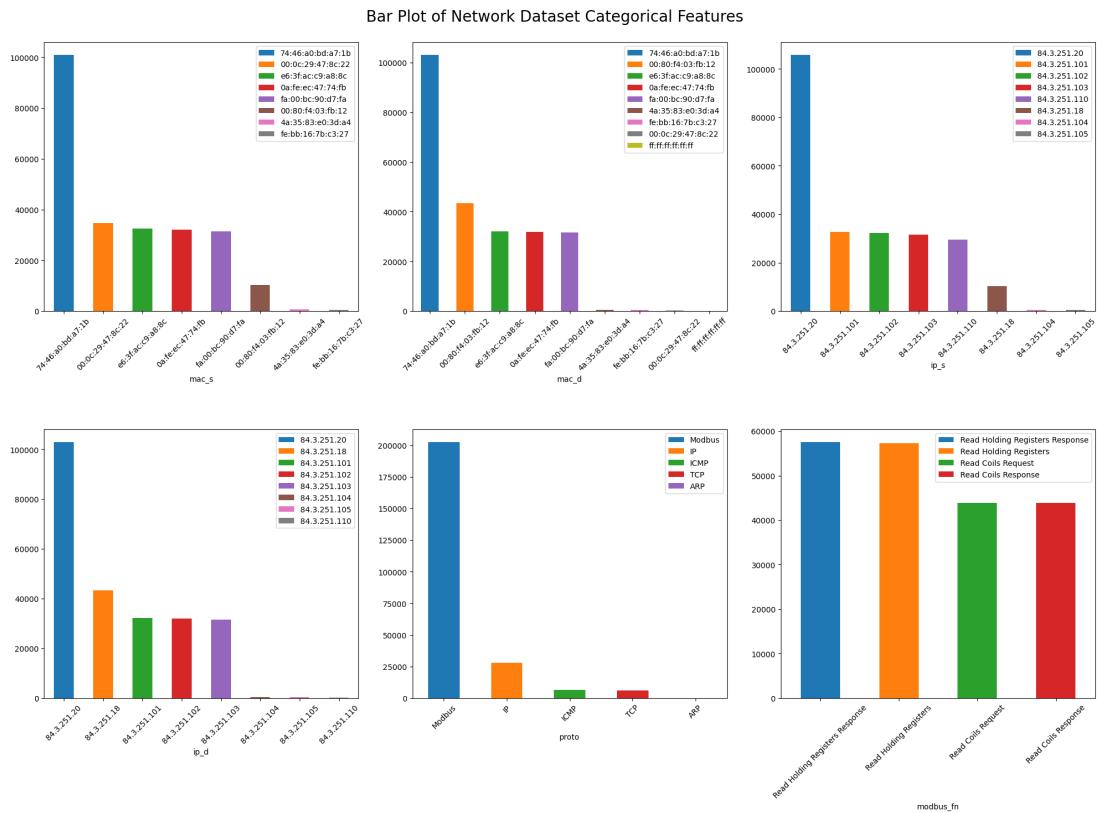
La variable **size** présente également une corrélation positive relativement forte avec **n\_pkt\_src** (0.75), soulignant une association probable entre la taille des paquets et le nombre de paquets provenant de la source.

En revanche, **n\_pkt\_src** et **n\_pkt\_dst** affichent une corrélation positive très faible (0.06), suggérant une faible liaison entre ces deux variables.

La variable **flags** montre des corrélations positives modérées avec **n\_pkt\_src** (0.49) et **n\_pkt\_dst** (0.49), et une corrélation plus faible avec **size** (0.38), indiquant des relations moins prononcées.

## b. Features catégoriques

Continuons avec les features catégoriques: **mac\_s**, **mac\_d**, **ip\_s**, **ip\_d**, **proto**, **modbus\_fn** et **modbus\_response**. Comme pour les features numériques, regardons leur répartition dans un premier temps.

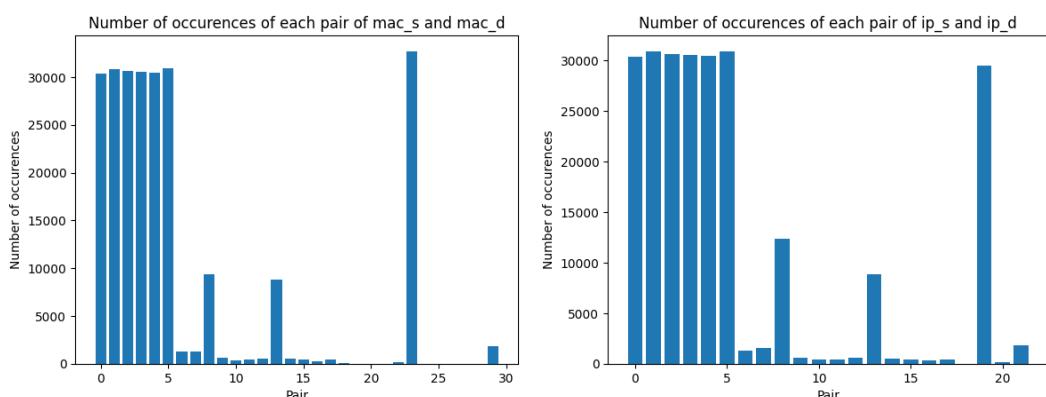


L'analyse des distributions présente certaines difficultés pour déduire des conclusions claires, cependant, quelques observations peuvent être soulignées.

Les distributions des variables **mac\_s**, **mac\_d**, **ip\_s** et **ip\_d** se révèlent très similaires, caractérisées par une adresse qui prédomine nettement sur les sept autres.

Par ailleurs, la distribution de **proto** affiche un déséquilibre notable, le protocole "Modbus" y étant dix fois plus représenté que les autres protocoles, ce qui suggère que cette colonne pourrait ne pas être pertinente pour la classification. Enfin, les données relatives à **modbus\_response** montrent un équilibre entre les quatre valeurs présentes, cependant c'est aussi le label avec le plus de valeurs inconnues.

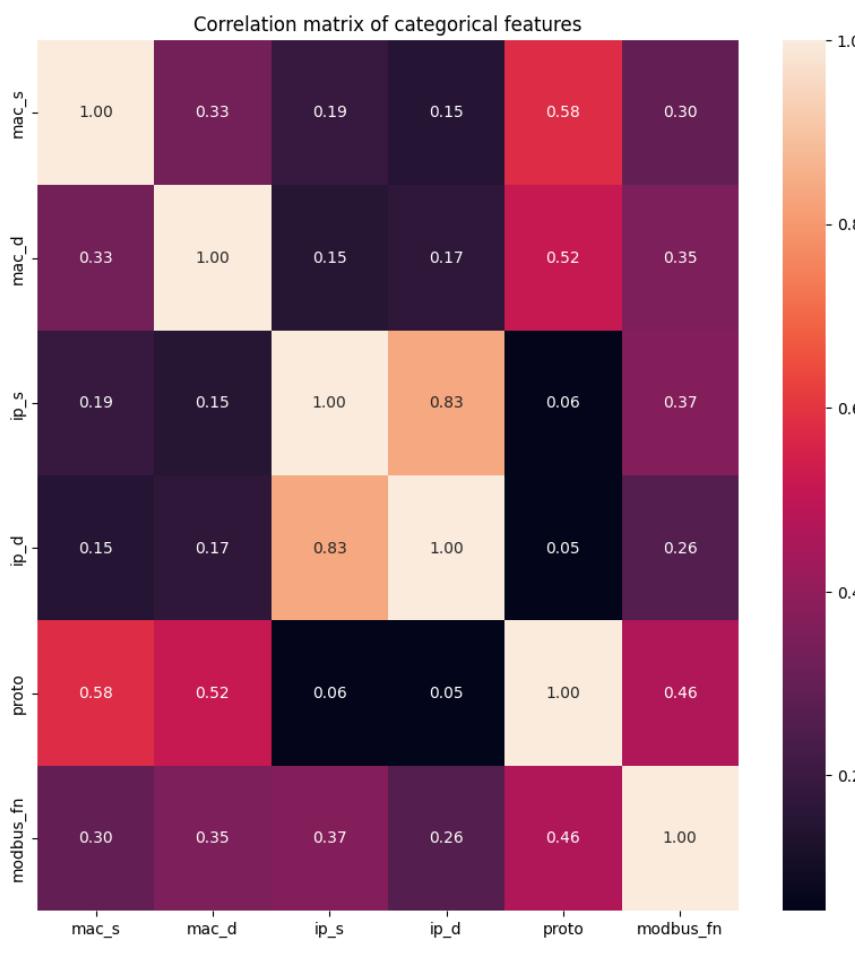
Nous avons aussi remarqué que le nombre de valeurs différentes pour les features **mac** et **ip** étaient faibles, nous avons donc essayé de regarder le nombre de paires entrée/sortie qui étaient utilisées:



L'analyse révèle qu'il existe environ 30 paires uniques d'adresses MAC source et destination (**mac\_s**, **mac\_d**), parmi lesquelles 7 à 9 paires sont majoritairement utilisées. De manière similaire, on compte environ 21 paires uniques d'adresses IP source et destination (**ip\_s**, **ip\_d**), la répartition étant également assez limitée avec 7 à 9 paires prédominantes. Il est à noter que la distribution des adresses IP reflète étroitement celle des adresses MAC.

Nous n'avons pas assez d'informations pour tirer des conclusions supplémentaires de ces informations.

Comme pour les features numériques, nous avons voulu avoir la possibilité de pouvoir analyser les corrélations en incluant les features catégoriques. Pour ce faire, nous avons utilisé un label encoder (scikit-learn) qui encode chaque unique label avec une valeur. La suite des résultats pour ce dataset sont à prendre avec précaution de part la méthode que nous avons choisie pour encoder ces labels.



L'analyse de la matrice de corrélation fournie met en lumière plusieurs relations intéressantes entre les variables étudiées.

Une forte corrélation est observée entre **ip\_s** et **ip\_d**, d'environ 0.83, indiquant une relation étroite entre les adresses IP source et destination. La variable **proto** montre des corrélations positives modérées avec **mac\_s** (environ 0.58), **mac\_d** (environ 0.52) et **modbus\_fn** (environ 0.46), suggérant que des variations dans **proto** sont associées à des changements dans les adresses MAC source et destination, ainsi que dans le code de fonction **Modbus**.

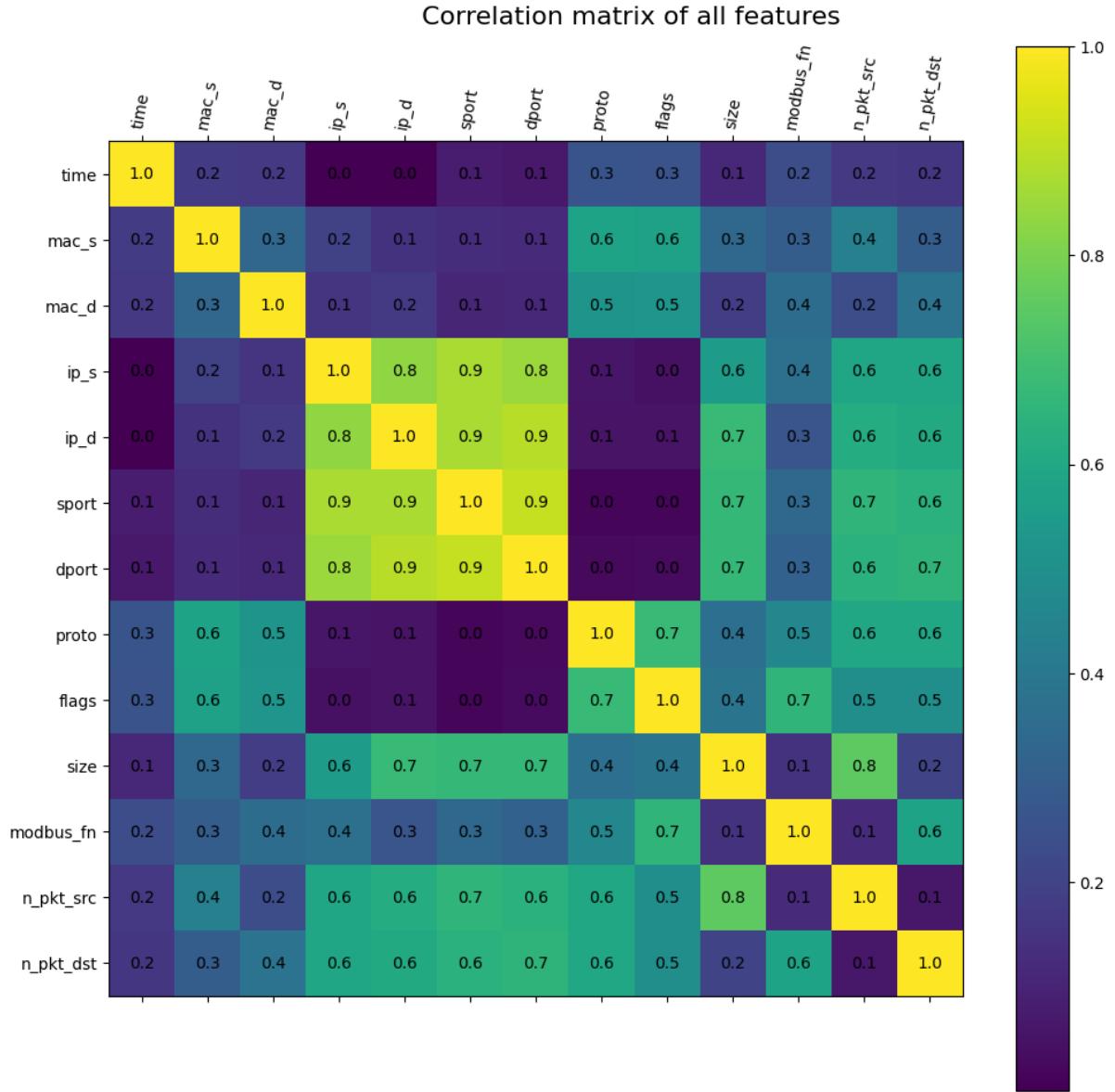
De plus, **modbus\_fn** affiche des corrélations positives modérées avec **ip\_s** (environ 0.37) et **mac\_d** (environ 0.35).

Les variables **mac\_s** et **mac\_d** présentent également une corrélation positive modérée, d'environ 0.33, révélant une certaine relation entre elles.

Les autres paires de variables présentent des corrélations faibles, indiquant l'absence de relations linéaires fortes entre elles. En résumé, bien que la relation la plus marquée soit entre **ip\_s** et **ip\_d**, les autres variables montrent des corrélations modérées ou faibles.

### c. Corrélations globales

Intéressons nous à des généralités qui ne se limitent pas aux deux types de features. Dans un premier temps nous regarderons la matrice de corrélation de l'entièreté des variables entre elles.



L'ensemble de la matrice de corrélation est complexe à interpréter, nous allons donc nous concentrer sur les aspects les plus pertinents. Nous observons les mêmes tendances qu'auparavant, issues des deux types de caractéristiques analysés séparément. Toutefois, il est également notable que l'algorithme peine à identifier des corrélations significatives entre ces deux types de caractéristiques. Cela est probablement dû à la manière dont nous avons encodé nos variables catégorielles. Bien que cela puisse sembler moins intéressant, cette observation pourrait néanmoins s'avérer précieuse, mettant en lumière des opportunités d'amélioration dans les méthodes d'encodage utilisées.

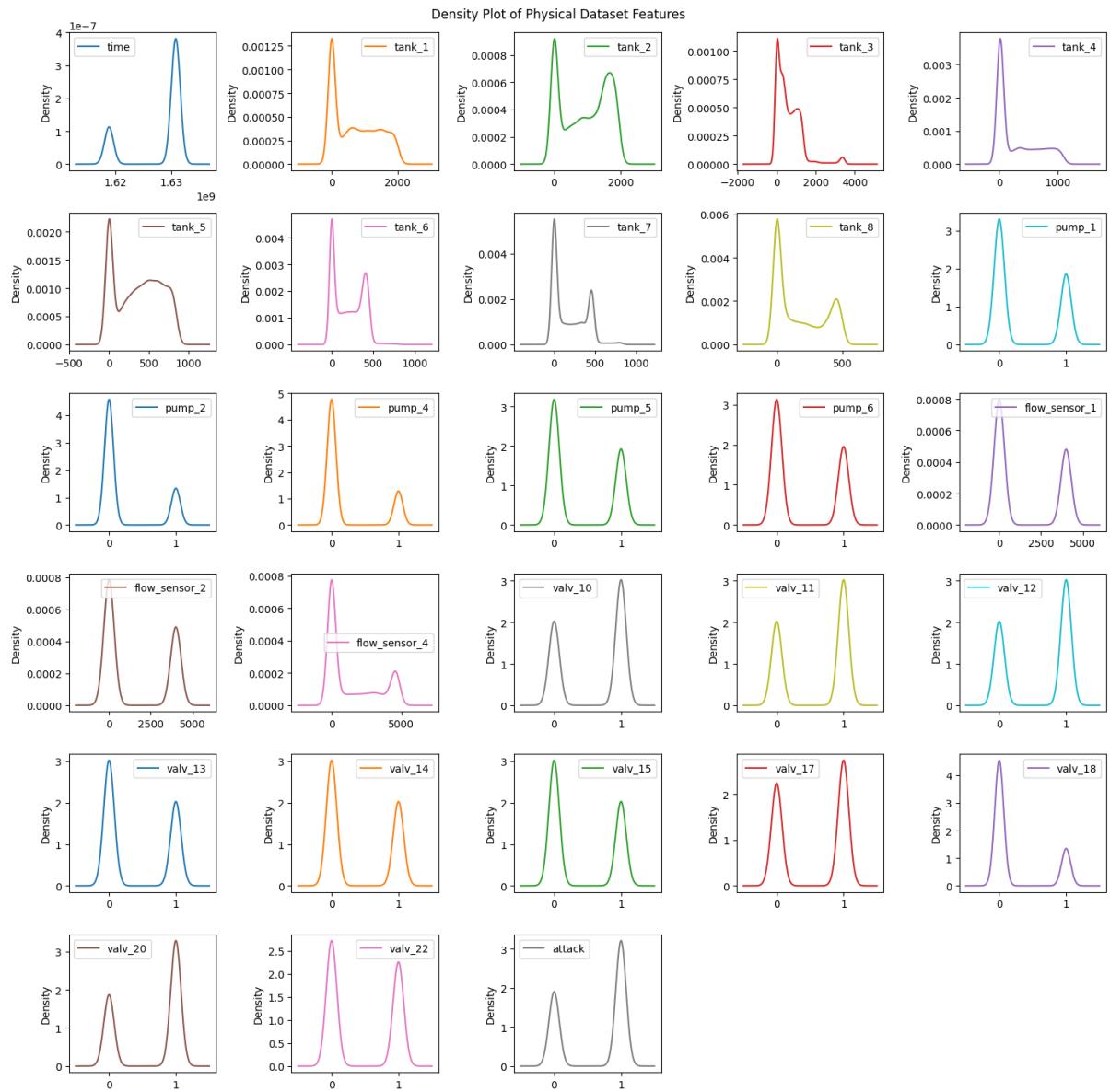
### 3. Features - Physical

Le dataset physique est composé de valeurs captées par 4 types de capteurs: les **tanks**, les **pumps**, les senseurs de **flow** et enfin l'ouverture ou non des **valves**. Il y a 8 colonnes de capteurs **tank**, 6 **pumps**, 4 **flow\_sensors** et 22 **valves**.

Pour avoir une idée des différents comportements des capteurs, nous avons fait des graphes des valeurs que les différents capteurs prennent en fonction du temps:



Nous pouvons voir que 14 capteurs ne varient pas tout du long de l'enregistrement. Nous allons donc les ignorer pour notre analyse et la suite de notre étude. Au-delà de ça, il est difficile de discerner d'autres informations. Pour ce faire, voyons la densité des valeurs des différents capteurs.



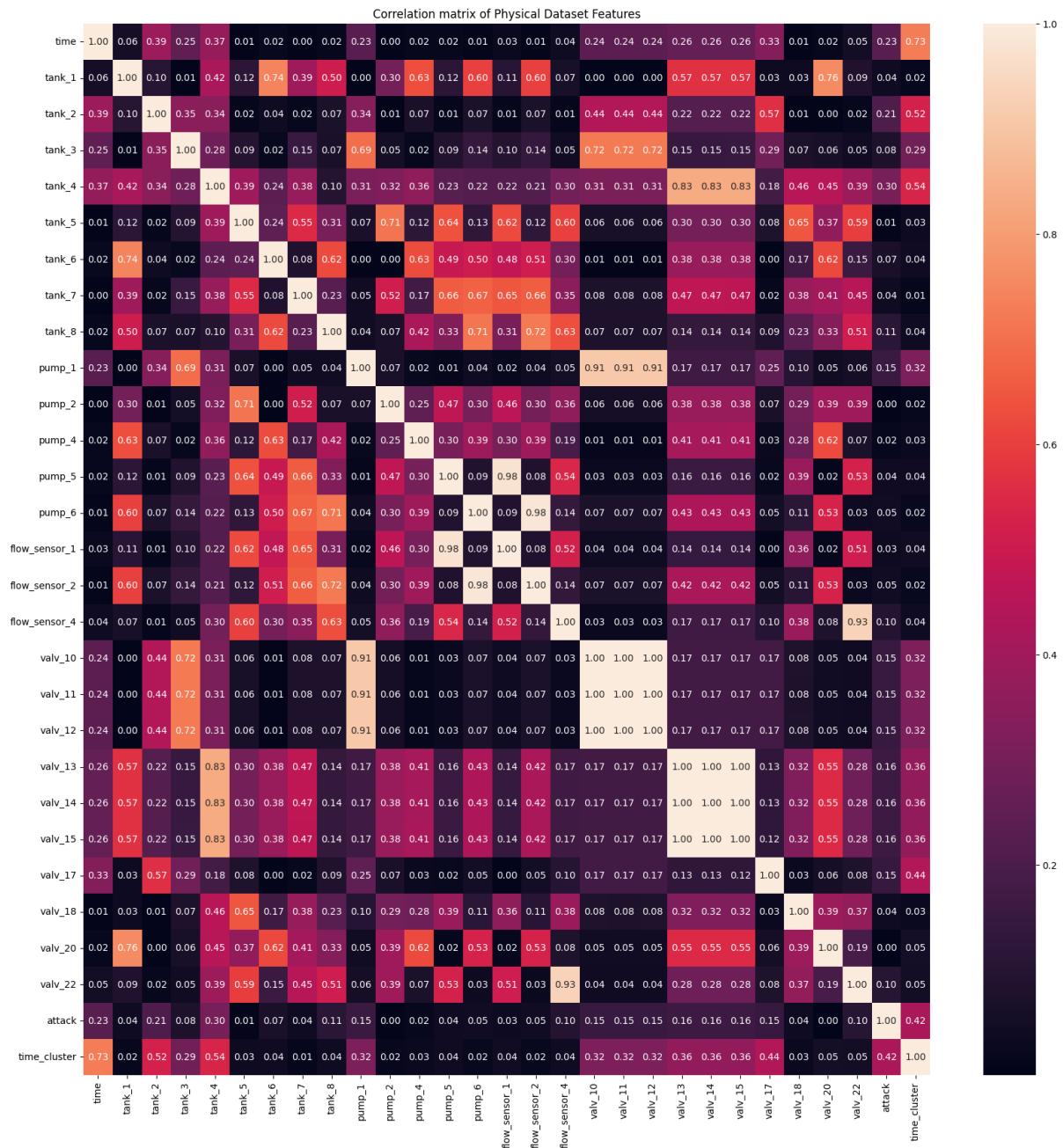
Ce graphique révèle plusieurs caractéristiques intéressantes. La majorité des données pour la colonne **tank** gravite autour de 0, bien que les capteurs 2, 5 et 6 affichent des valeurs non nulles significatives, proportionnellement plus grandes ou équivalentes aux valeurs 0.

Pour la colonne **pump**, qui est de nature booléenne avec des valeurs de 0 ou 1, on observe que la densité de 0 est deux à trois fois supérieure à celle de 1.

En ce qui concerne **flow\_sensor**, qui représente des valeurs entières entre 0 et 6000, la majorité des données est à 0, mais tous les capteurs présentent un pic notable autour de 4000.

Enfin, la colonne **valv**, également booléenne, montre une distribution variée entre les valeurs 0 et 1 selon les capteurs.

Jetons un coup d'oeil aux corrélations entre les différentes features:



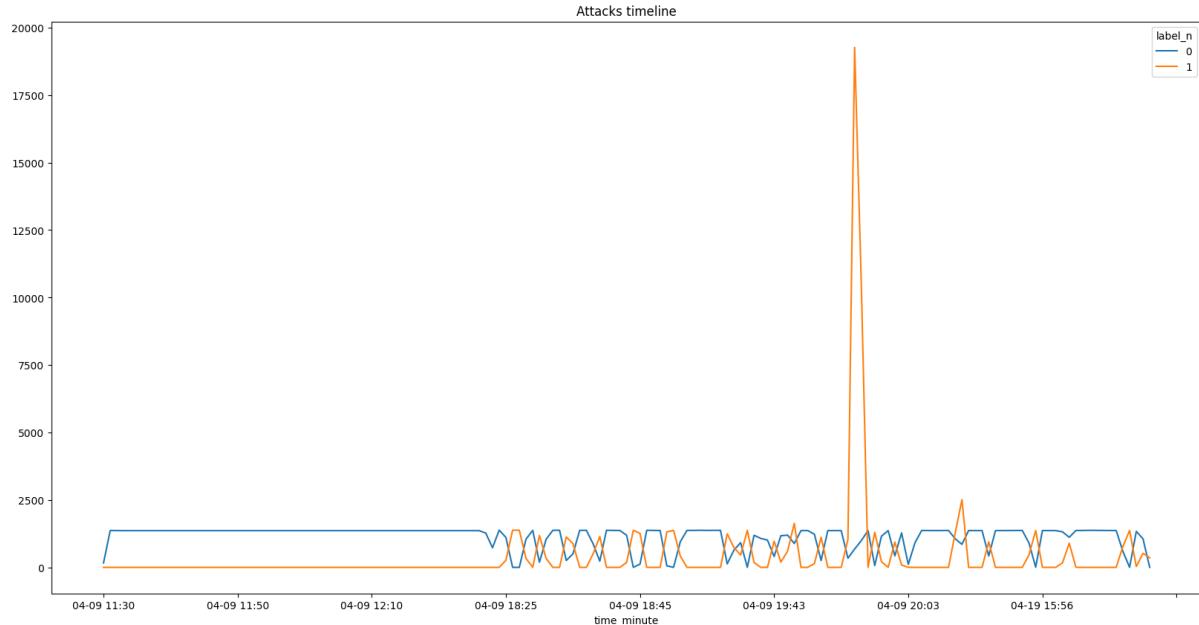
Quoique décousue, cette matrice de corrélation donne des informations sur les dépendances entre les modules d'équipement. Par exemple, quand le **tank\_1** augmente, **pump\_4** et **pump\_6** s'activent, **flow\_sensor\_2** semble être aussi dans la boucle et enfin, les **valves** 13, 14, 15 et 20 semblent être responsables du flow vers ce tank. On note aussi que les valves 10, 11 et 12 sont équivalentes, tout comme les valves 13, 14 et 15. On pourra donc en garder une seule pour chacun des deux groupes.

Une analyse très poussée permettrait, en théorie, d'avoir une idée globale de comment le système est construit. Cependant, de part la nature de notre étude, nous n'avons pas choisi d'approfondir cet aspect car il ne semblait pas s'inscrire dans la lignée de ce qui était recherché.

## 4. Time series

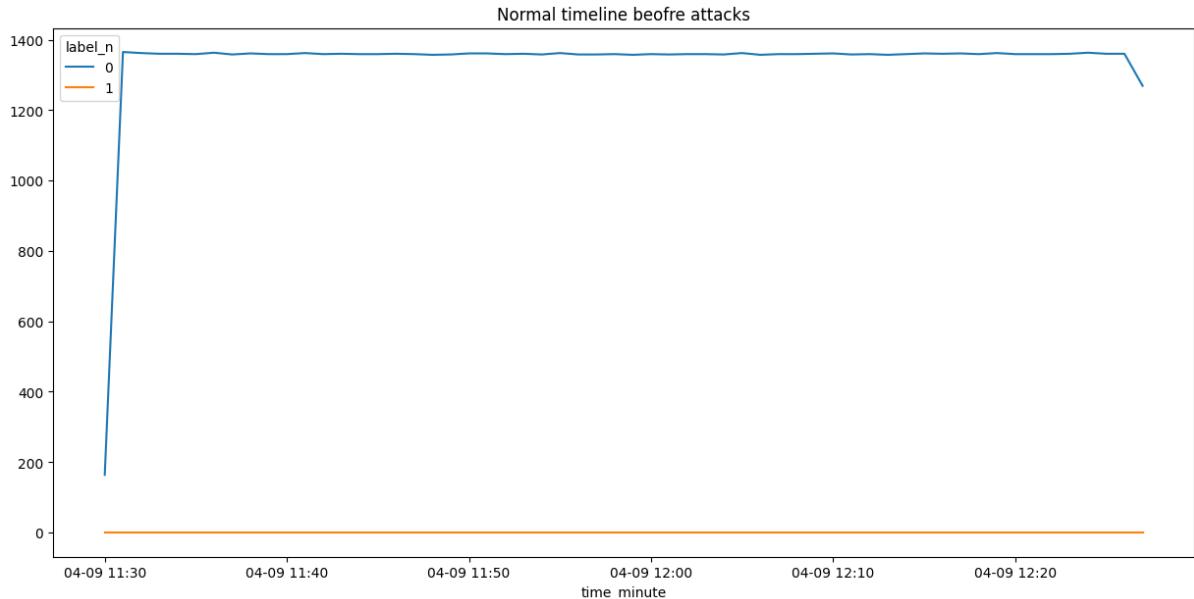
Nous finissons notre exploration par des remarques trouvées sur l'aspect temporel des données. Ces informations ne sont pas forcément nécessaires pour la suite de nos recherches cependant elles permettent d'avoir une meilleure compréhension des données capturées.

Dans un premier temps, regardons les labels de nos échantillons en fonction du temps sur le dataset network:

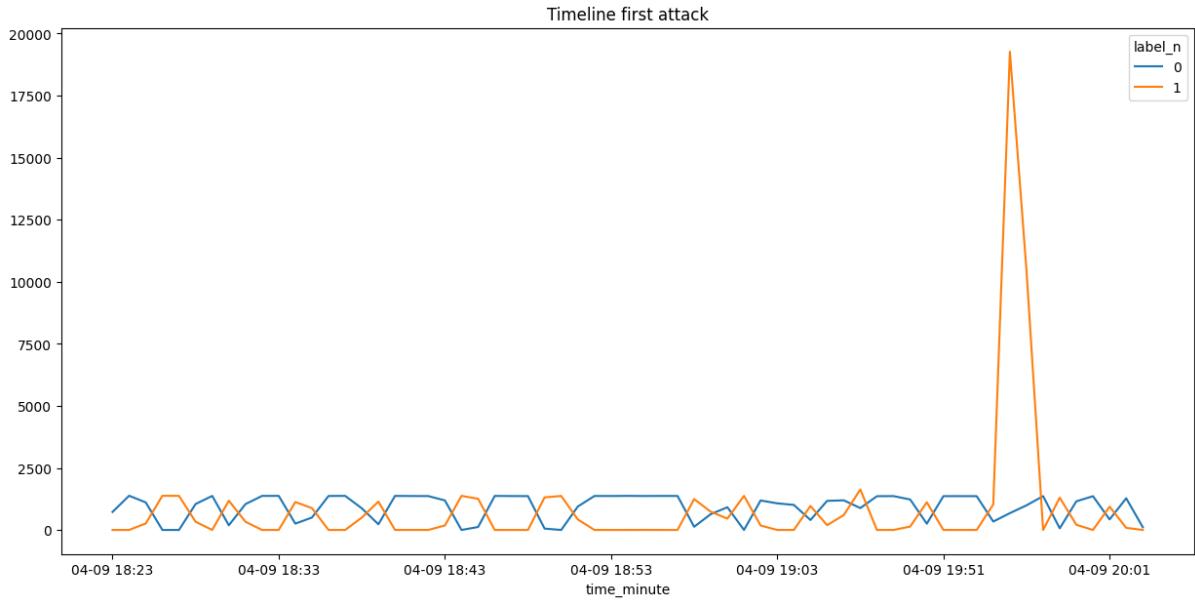


Comme prévu, de part le fait que les 4 datasets dont celui en situation normale, le début de nos données n'est uniquement composé de situation normale. Après une heure, des attaques commencent à arriver de manière semblable à de l'aléatoire avec un pic à la deuxième heure.

Regardons les deux situations d'un peu plus près:

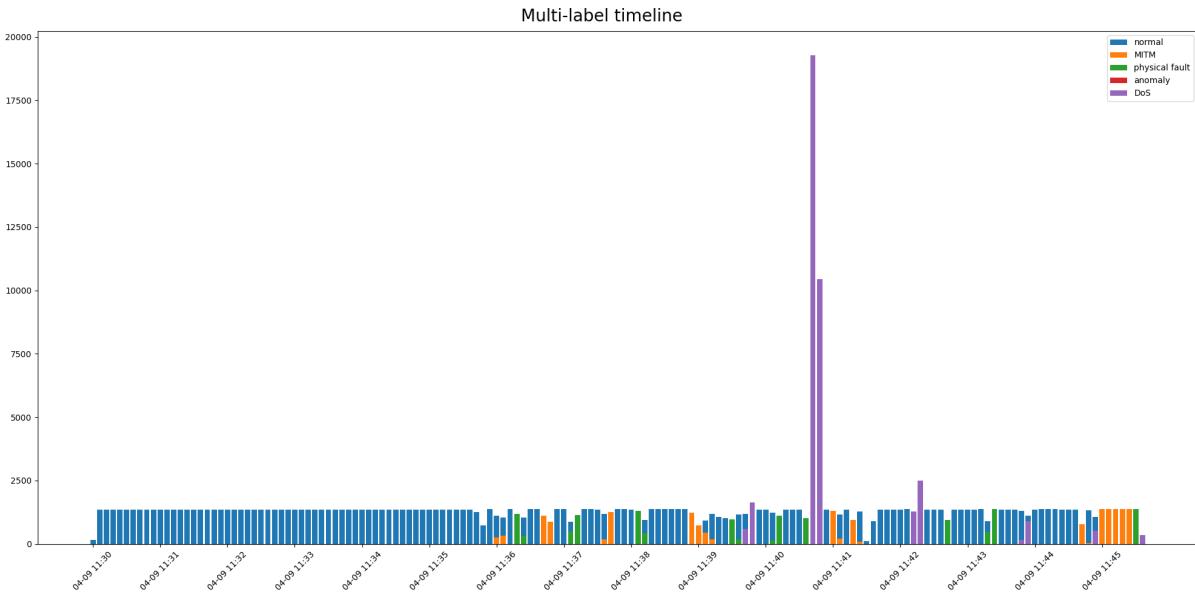


Dans le cas normal, il n'y a pas un seul échantillon qui correspond à une attaque.



Dans le cas d'une attaque, de nombreux échantillons sont considérés comme tels.  
Nous avons fait cette recherche pour comprendre un peu mieux les dynamiques générales du comportement de l'environnement.

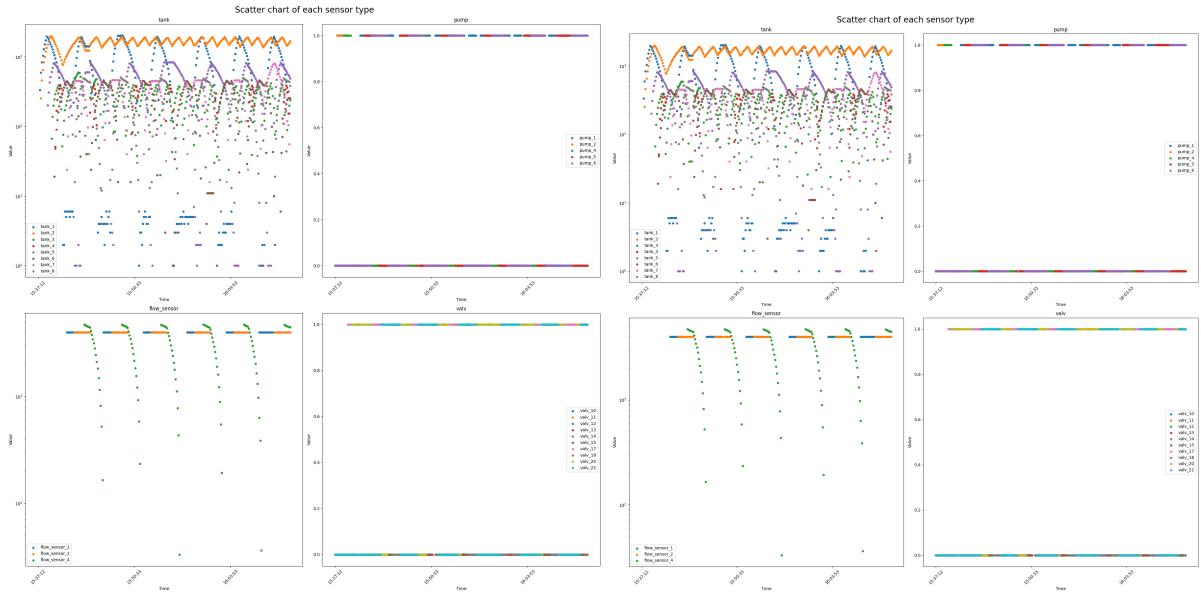
De plus, si l'on considère l'aspect multi label de la timeline d'attaques, voici ce que ça donne:



Les pics d'attaques correspondent à des attaques **DoS**, les attaques **Man in the Middle** se font par salves mais ne durent que peu de temps et enfin il y a souvent des **physical faults** après une attaque, peu importe son type.

Dans cette même optique, nous avons essayé de regarder les valeurs prises par les différents capteurs du dataset physique en fonction du temps. De part le fait que ce dataset a peu d'échantillons, nous avions espéré de pouvoir mettre toutes les données regroupées par type de capteur sur un même graph.

Cependant, nous avons clusterisé les données en différents groupes en fonction des horaires d'enregistrement des données. En effet le dataset physique est composé de deux séances d'enregistrement faites à des périodes différentes. De ce fait, nous voulions voir si les comportements globaux changeaient en fonction de la période regardée.



Les graphes sont très remplis, cependant nous pouvons distinguer certains motifs sur des capteurs. Les capteurs **tank** semblent varier selon un pattern qui est unique à chaque tank, cependant il semblent tous varient dans la même direction pour un instant donné. Les capteurs **flow\_sensor\_1** et **flow\_sensor\_2** semblent statiques par intermittence à ~10 000, alors que **flow\_sensor\_4** semble osciller entre ~10 000 et 0 dans un motif en cascade périodique. Enfin, il est visible qu'à plusieurs moments de la captation, aucun capteur **pump** n'est activé, ceci pourrait être un indicateur que quelque chose est anormal.

Pour conclure notre exploration, le jeu de données englobe à la fois des données physiques et des données réseau, indispensables pour comprendre l'impact des attaques sur le processus physique et le trafic réseau. Dans l'analyse du jeu de données réseau, nous avons examiné les distributions des caractéristiques, jetant ainsi les bases pour l'ingénierie des caractéristiques et la construction du modèle.

Par la suite, dans le jeu de données physique, nous avons été agréablement surpris de ne trouver aucune valeur manquante, bien que des problèmes de déséquilibre des labels soient apparus. Nos caractéristiques ont fait l'objet d'une analyse de densité, mettant en évidence des tendances dans les différents points de capture du système : réservoirs, pompes, capteurs de débit et vannes.

De plus, nous nous sommes penchés sur les aspects de séries temporelles du jeu de données, regroupant les données pour une analyse approfondie.

Dans l'ensemble, ces analyses fournissent des informations cruciales pour notre projet en cours, facilitant la sélection des caractéristiques, la préparation des données et la modélisation subséquente.

### III. Détection d'anomalies - Network

Dans cette section, nous allons comparer différentes familles d'algorithmes afin de détecter des anomalies. Nous nous attarderons ici sur le dataset réseau, qui contient des informations relatives au transfert de données comme des adresses Mac, des adresses IP, des ports ou des protocoles.

Notre objectif est donc de prédire, à chaque instant, si nous sommes face à un comportement normal ou d'attaque (DoS, Man In The Middle, physical fault, anomaly).

#### 1. Non-supervised algorithms

Les algorithmes non-supervisés ont pour avantage de pouvoir apprendre une distribution de données pour les classifier sans être informés de la nature des données d'entraînement. Dans le cas des deux méthodes que nous allons utiliser, elles permettent uniquement de faire de la classification binaire.

##### a. Isolation Forest (IF)

Dans un premier temps, nous avons laissé les paramètres par défaut, notamment le taux de contamination. Le taux de contamination permet d'indiquer le taux d'anomalies dans le dataset d'entraînement, afin que l'algorithme renvoie à peu près le bon nombre d'anomalies. Cette approche nous renvoie **12505** anomalies, parmi lesquelles **7695** sont bel et bien des anomalies, et **4810** sont normales. On obtient donc **61.5%** de précision, ce qui n'est pas brillant. De plus, le dataset donné à l'algorithme contient 66000 anomalies, ce qui rend le résultat encore moins impressionnant.

Nous avons donc fixé le taux de contamination, qui dans notre cas est à 27.5%. En utilisant ce taux ainsi qu'en fixant le nombre d'estimateurs à 100 (meilleure valeur testée empiriquement), l'algorithme nous renvoie **65687** anomalies dont **37940** en sont vraiment. Avec **57.7%** de précision, on peut conclure que cet algorithme n'est pas très efficace pour travailler sur notre dataset.

##### b. Local Outlier Factor (LOF)

En ce qui concerne LOF, nous avons eu beaucoup de mal à obtenir des résultats, car l'algorithme prend trop de temps sur notre dataset. De plus, dans les premières ébauches de notre analyse, nous avions testé l'algorithme sur un dataset plus petit mais bien moins représentatif. Nous avons obtenu de très mauvais résultats, avec seulement **11.6%** de précision et un très mauvais rappel également (30919 résultats).

Les algorithmes non-supervisés ne sont pas du tout performants pour notre problème, que ce soit d'un point de vue de la précision, du rappel ou même du temps d'exécution.

#### 2. Deep Learning

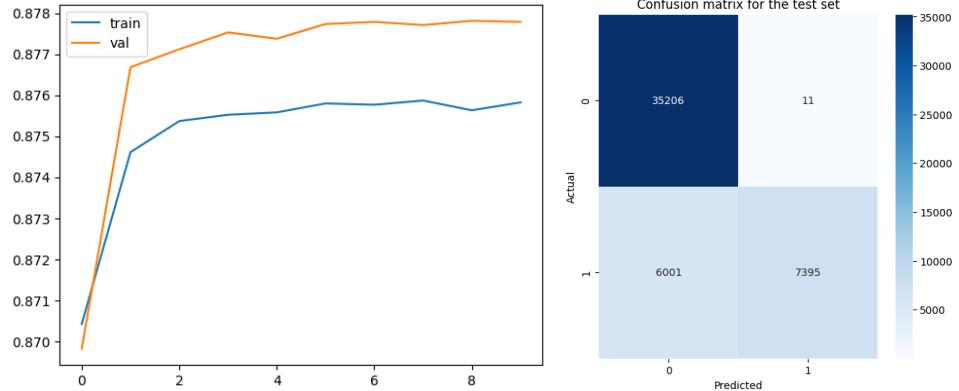
Passons maintenant aux méthodes de deep learning. Notre analyse s'est portée sur deux d'entre elles: un réseau de neurones séquentiel et un LSTM.

##### a. Neural network

Notre réseau de neurones a été initialement défini de façon arbitraire: trois couches denses cachées de respectivement 1024, 256 et 64 paramètres. Nous allons voir par la suite que les résultats ne sont pas liés à l'architecture du réseau. Pour l'entraînement, on supprime les données

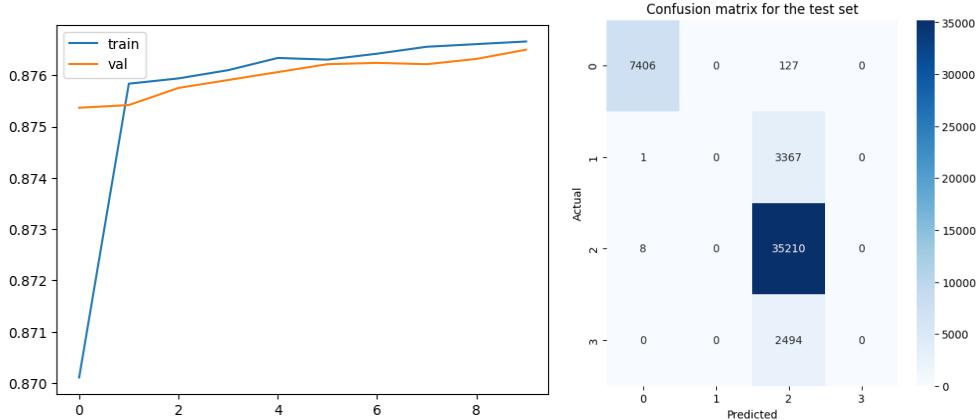
contextuelles telles que les adresses IP, MAC où le temps. Apprendre sur ces données impliquerait que des biais seraient pris en compte, ce qui pourrait tromper le modèle.

Tout d'abord, testons notre modèle sur un problème de classification binaire. On obtient la courbe d'entraînement et la matrice de confusion suivantes:



On note que dès la première epoch, l'accuracy du modèle atteint 87.5% pour l'entraînement, en ne varie presque pas jusqu'à la 10ème epoch. La matrice de confusion nous indique que le réseau identifie très bien les cas normaux, mais à plus de mal à caractériser les anomalies: environ 45% sont classées comme "normal".

Sur le problème multi-classes, on utilise la même architecture du réseau précédent et obtenons les résultats suivants:



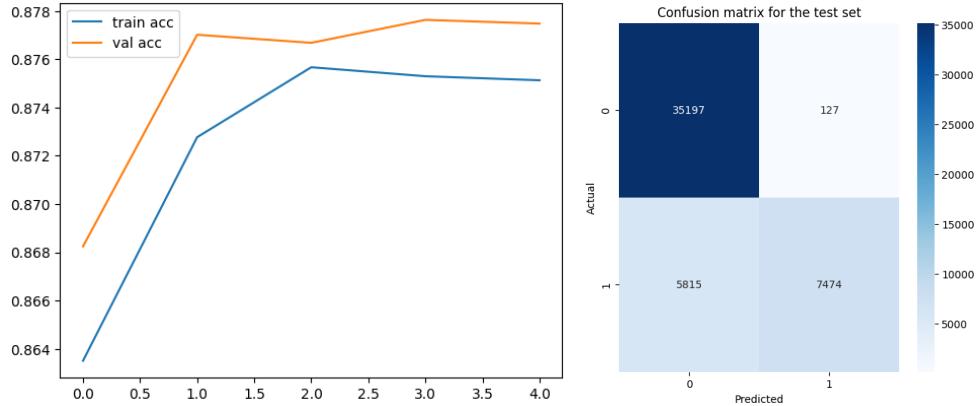
Les courbes d'entraînement sont similaires à celles de la classification binaire, et l'accuracy finale également autour de **87.7%**. Une fois arrivé à cette valeur, on semble avoir atteint un plafond de verre et ne plus pouvoir apprendre. Le résultat le plus surprenant s'observe sur la matrice de confusion. En effet, on s'aperçoit que le modèle n'est pas capable de prédire les classes 1 et 3, qui correspondent respectivement à "MITM" et "physical fault". On note que la classe 0 ("DoS") est presque parfaitement prédite, et que toutes les erreurs se retrouvent dans la classe 2 (normal). On constate donc que deux des trois classes d'anomalies ne sont pas prédites, et par défaut considérées comme normales.

Suite à ces observations, nous nous sommes directement tournés vers l'architecture du réseau. Nous avons expérimenté différentes configurations (une seule couche cachée, changement d'optimizer, modification du nombre de pods dans chaque couche...), mais cette barre à 87.7% était toujours présente. N'ayant pas réussi à avoir un impact sur les résultats en modifiant le réseau, nous avons essayé de modifier les features, c'est-à-dire le dataset d'entraînement. Encore une fois, en utilisant seulement les features "sport" et "dport", nous obtenons des résultats similaires. Nous avons du mal à expliquer ce phénomène, nous allons donc analyser les méthodes suivantes pour voir si on retrouve un comportement similaire.

## b. LSTM

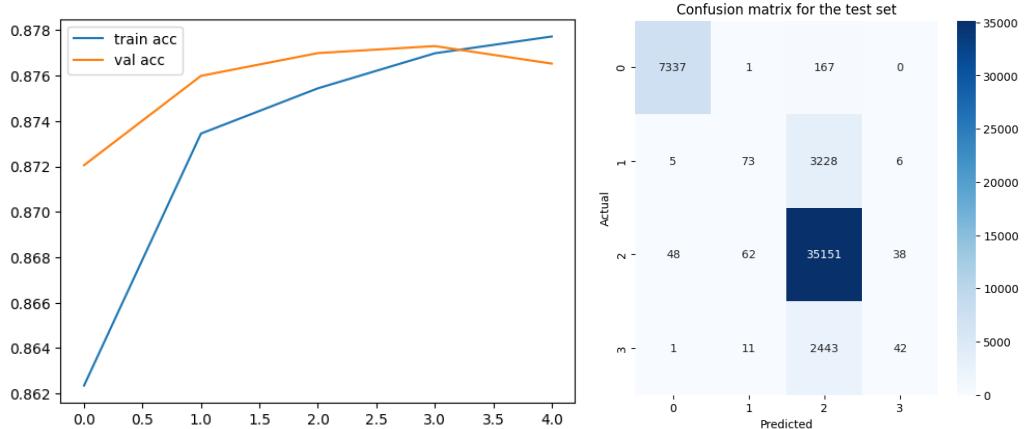
En utilisant des LSTM, nous espérons pouvoir capturer une complexité à plus long terme sur nos données. L'architecture du réseau entraîné est très simple et contient seulement un seul LSTM.

À nouveau, nous avons commencé notre analyse par la classification binaire. Les résultats que nous obtenons sont identiques à ceux du réseau de neurones séquentiel:



L'architecture LSTM n'a pas permis de dépasser le seuil de performance de 87.7% malgré des améliorations. La matrice de confusion est également similaire à celle de l'architecture précédente, avec des erreurs symptomatiques lors de la classification des erreurs.

De même pour le problème multi-classes:



Ici, les résultats sont encore très similaires à ceux du réseau de neurones précédent, et montrent toujours une difficulté à classer trois des anomalies. Nous avons ajouté une des classes d'anomalie supprimée pour le réseau de neurones séquentiel. Elle ne contient que 3 éléments (1 dans le test), et est donc peu pertinente. Cependant, l'ajouter permet de constater que le réseau ne semble pas capable de la classer correctement non plus.

Pour conclure sur la partie Deep Learning, nous avons été confronté à un plafond de verre lors de l'apprentissage que nous ne sommes pas en mesure d'expliquer. Plus précisément, il semble que les modèles font des erreurs sur trois classes d'anomalies en les classant comme normales. Néanmoins, la dernière classe d'anomalie est très bien gérée par les modèles. On note également que le comportement du réseau de neurones séquentiel et celui du LSTM sont très similaires. Dans la prochaine section sur les classifiers, nous espérons résoudre ce problème d'anomalies mal classées et atteindre des performances plus satisfaisantes.

### 3. Classifiers

Cette section porte sur les classifiers. Nous allons étudier trois d'entre eux en particulier: Decision Tree, Random Forest et XGBoost. Pour chacune de ces trois méthodes, nous n'allons pas nous attarder sur la classification binaire, car elle n'apporte pas plus d'information que la classification multi-classes, qui elle est bien plus intéressante.

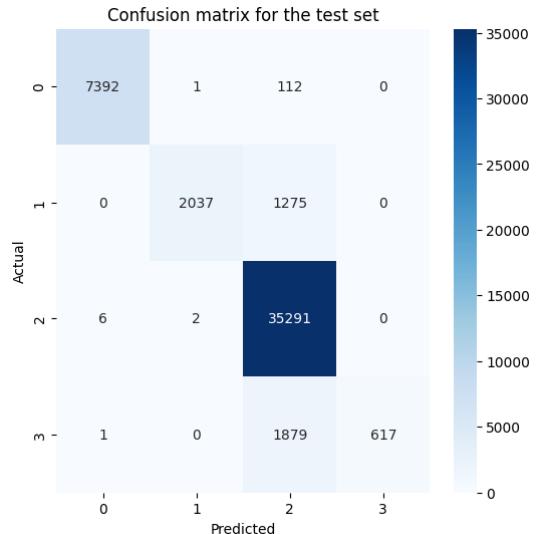
#### a. Decision Tree

La classe de Decision Tree comporte de nombreux hyperparamètres, sur lesquels nous avons effectué des tests empiriques pour déterminer ceux qui semblent les plus performants. Voici les paramètres retenus:

```
● ● ●  
1 params = {  
2     'max_depth': 8,  
3     'criterion': 'gini',  
4     'splitter': 'best',  
5     'random_state': random_state  
6 }
```

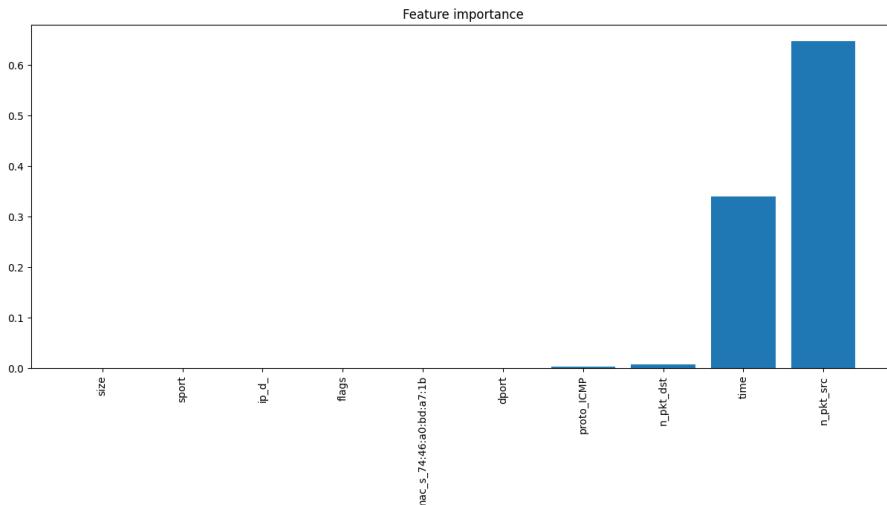
Les résultats obtenus sont les suivants:

```
● ● ●  
1 "Accuracy": 0.9326106185588218  
2 "Recall": 0.7117123708683838  
3 "F1": 0.776276153519922  
4 "MCC": 0.8435893918477205  
5 "Balanced accuracy": 0.7117123708683838
```



Comme on pouvait s'y attendre, l'accuracy est très bonne mais le recall beaucoup moins, ce qui donne un F1-score de **62.1%**, qui n'est pas très élevé. La balanced accuracy, qui dans notre cas est très importante au vu de l'inégalité des classes de notre dataset, est elle très moyenne aussi, ce qui souligne un problème qui peut être similaire à celui observé pour le deep learning: certaines classes moins présentent sont très mal prédites. Cependant, le MCC (Matthews CorrCoef) est à **84.4%**, ce qui indique une certaine efficacité de l'algorithme. La matrice de confusion est cette fois-ci plus cohérente avec les résultats attendus. L'algorithme arrive à classifier les classes 1 et 4, même si la précision des prédictions de ces classes reste mauvaise. On note une supériorité de Decision Tree par rapport aux méthodes de Deep Learning.

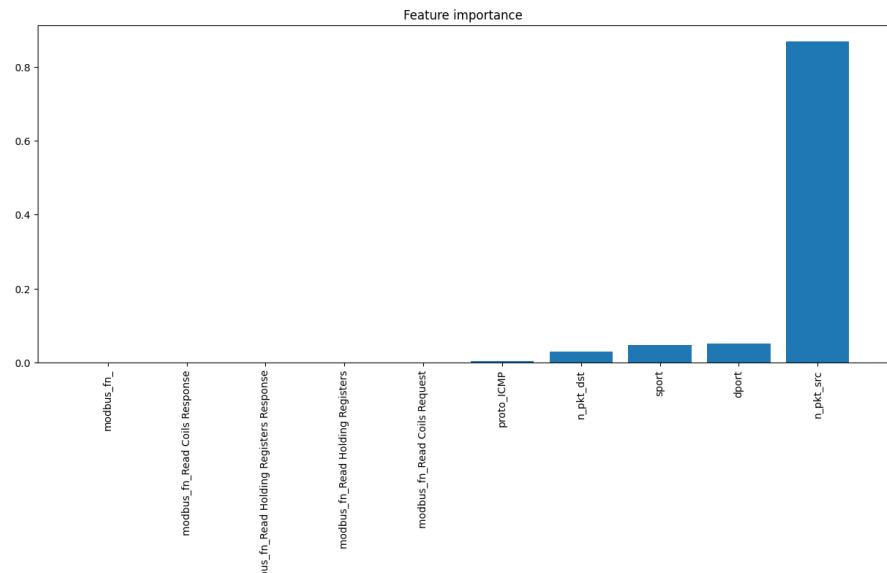
Arrêtons-nous maintenant sur l'importance de chaque feature. On constate que le timestamp joue énormément, ce qui n'est pas une très bonne nouvelle car nous voulons que notre modèle puisse détecter une anomalie peu importe l'heure ou le jour, et utiliser cette feature pourrait créer un biais.



Refaisons la même analyse sans les données contextuelles. Nous utilisons les mêmes paramètres pour le modèle, et les résultats sont les suivants:



Les résultats obtenus sont très intéressants. On retrouve une accuracy à **87.7%**, le plafond de verre obtenu avec le Deep Learning. Le MCC et la balanced accuracy sont bien plus faibles qu'avec les features contextuelles, ce qui est visible sur la matrice de confusion où à nouveau trois des quatre classes d'anomalie ne sont pas prédites. On en conclut donc que les features contextuelles sont la clé d'une bonne classification multi-classes sur ce dataset. Analysons, tout de même l'importance des features dans ce cas:



L'algorithme attribue une importance prédominante au nombre de paquets entrants. Globalement, une telle distribution de l'importance des features n'est pas optimale, ce qui peut être un des facteurs de si mauvais résultats.

## b. Random Forest

Tout comme Decision Tree, nous avons performé des tests empiriques pour déterminer les paramètres optimaux de l'algorithme dans notre cas:

```

● ● ●
1 params = {
2   'n_estimators': 300,
3   'max_depth': 8,
4   'random_state': random_state,
5   'n_jobs': -1,
6   'criterion': 'log_loss',
7 }

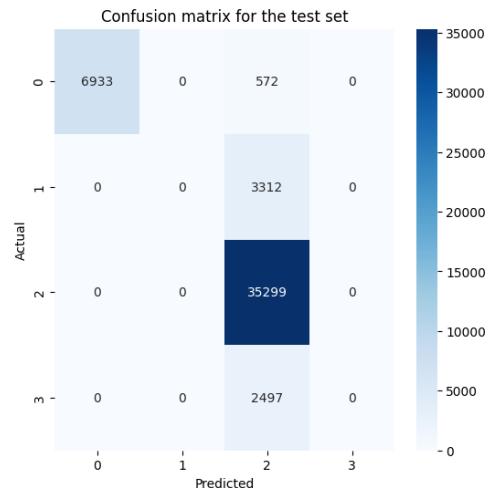
```

Random Forest étant un algorithme d'ensemble d'arbres relativement simple, nous ne nous attendons pas à des résultats excellents. Cependant, on espère quand même faire mieux que le Deep Learning, qui reste une baseline très modeste pour répondre à notre problème. Voici les résultats obtenus:

```

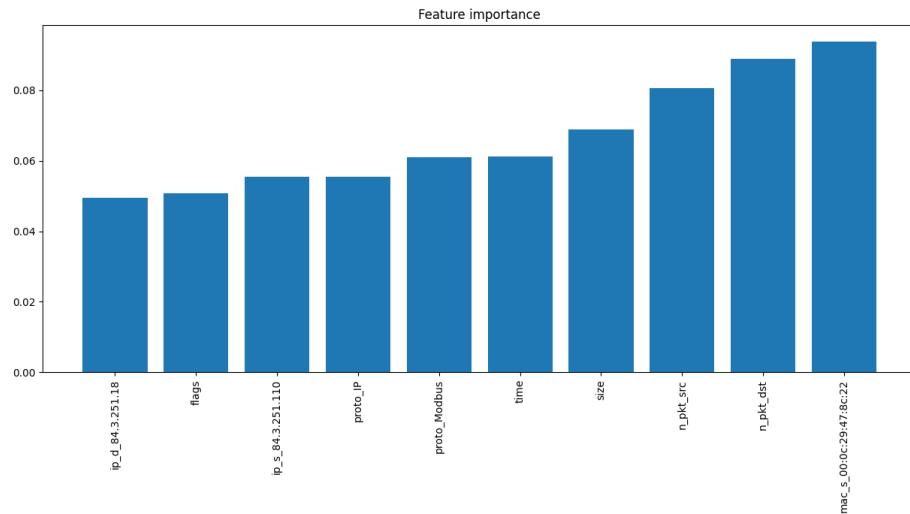
● ● ●
1 "Accuracy": 0.8687388147203423
2 "Recall": 0.480946035976016
3 "F1": 0.4693723968603624
4 "MCC": 0.6820751968761989
5 "Balanced accuracy": 0.480946035976016

```



On retrouve toujours ce même problème de classes non prédites, mais cette fois-ci avec un MCC à **67.8%** et une balanced accuracy à **38.2%**, qui sont des résultats très mauvais et inexploitables.

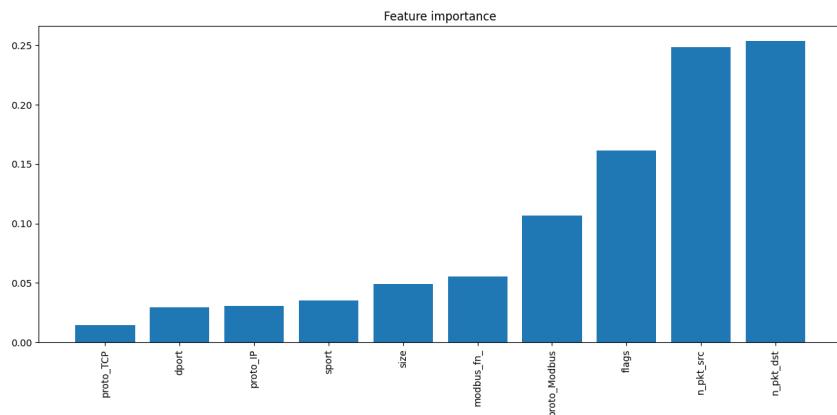
Concernant l'importance des features, on remarque que la distribution de leur importance est beaucoup plus homogène, ce qui est positif. Le timestamp reste une feature importante, et une des adresses MAC source est la plus utilisée, ce qui n'est vraiment pas le comportement attendu pour généraliser.



Lorsqu'on retire les features contextuelles, les résultats sont quasiment identiques, à un ou deux pourcents près:



Comme décrits plus haut, les résultats sont mauvais et ne permettent pas de classifier correctement nos données. L'importance des features est elle plus concentrée sur le nombre de paquets mais garde une distribution qui reste acceptable.



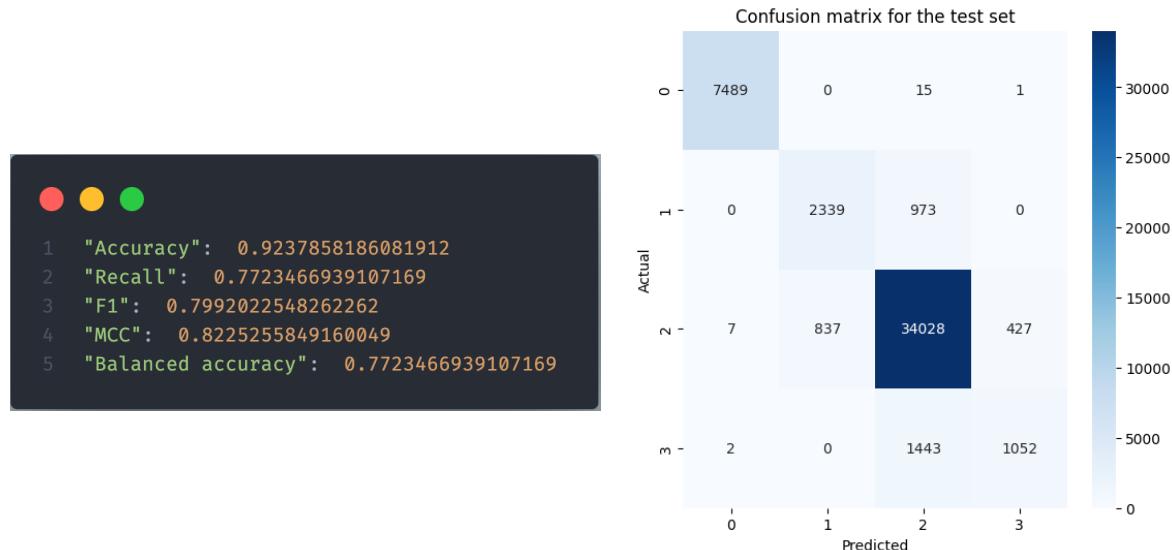
Random Forest n'est pas un algorithme brillant pour ce type de problème, bien que ce soit une simple classification. Les données sont complexes et implique que l'algorithme comprenne avec précision comment elles sont liées.

## c. XGBoost

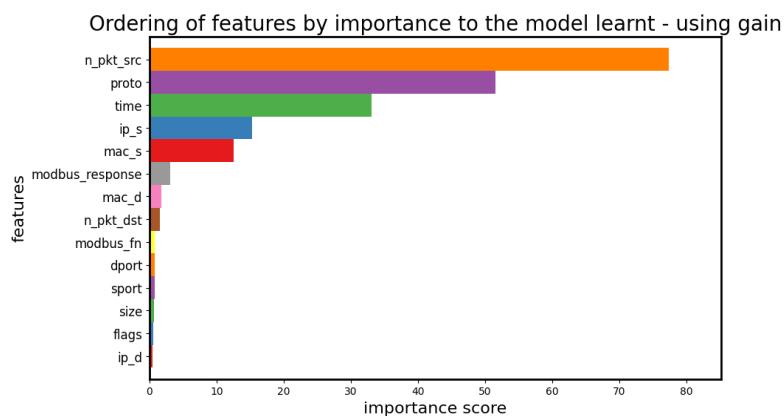
Le dernier algorithme de classification que nous avons étudié est XGBoost, l'un des meilleurs dans son domaine. Les paramètres que nous avons utilisé sont les suivants:

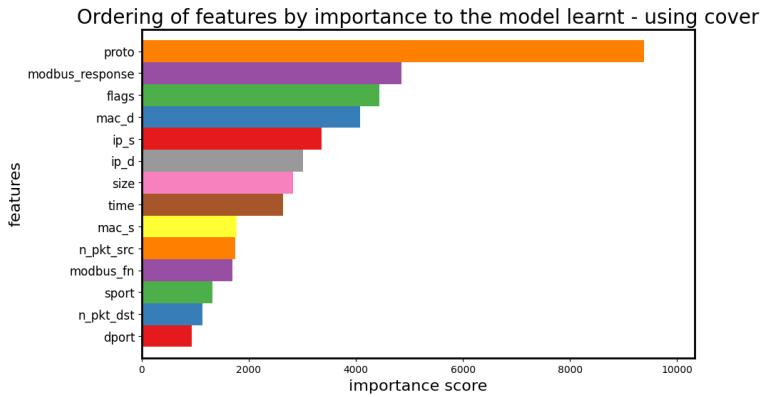
```
● ● ●
1 params = {
2     "objective": "multi:softmax",
3     "tree_method": "hist",
4     "seed": random_state,
5     "num_class": 5
6 }
7 n = 200
```

L'avantage de cet algorithme, contrairement à Random Forest ou Decision Tree, est qu'il n'est pas nécessaire de faire un OneHotEncoding sur les features catégoriques car elles sont gérées au même titre que les features numériques. Il est donc plus simple de manipuler les données d'entrées. Voici les résultats obtenus:

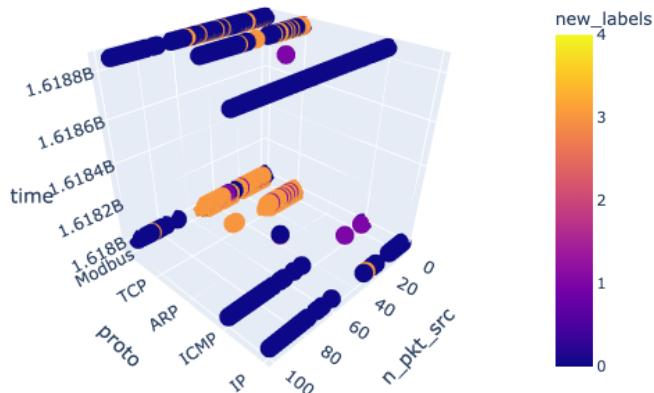


Nous obtenons enfin de bons résultats: toutes les métriques sont très bonnes, et la matrice de confusion est bien mieux répartie que précédemment. Bien sûr, ce n'est pas parfait, un nombre significatif d'attaques ne sont pas détectées et quelques comportements normaux sont considérés comme des anomalies. Analysons l'importance des features:





Il existe deux façons de regarder l'importance des features avec XGBoost: en utilisant le gain ou la cover. Le gain caractérise la contribution de chaque feature pour chaque arbre, tandis que la cover correspondant au nombre moyen d'observation de cette feature. Pour chacune des deux métriques, "proto" est présent dans le top 3, ce qui montre son importance. On note également que le temps est une feature importante d'un point de vue du gain. Ci-dessous une représentation 3D en utilisant les top 3 features du gain nous permet de clairement voir comment le modèle sépare les anomalies:



Pour conclure sur cette partie sur la détection d'anomalie sur le dataset réseau, on peut dire que c'est un dataset complexe de part ses features non triviales. Les différentes features catégoriques et contextuelles apportent un niveau de difficulté dans l'utilisation des modèles. D'un point de vue général, les algorithmes non-supervisés sont très peu performants, le Deep Learning n'arrive pas à comprendre la complexité des features catégoriques et bloque à 87.7% tout en étant incapable de prédire certaines anomalies. Les classifiers sont très disparates, à commencer par Random Forest qui est très mauvais. Decision Tree s'en sort plutôt bien car il arrive à dépasser le problème de non prédiction de certaines classes. Cependant, XGBoost est imbattable et produit des résultats très satisfaisants et exploitables, bien qu'améliorables.

## IV. Détection d'anomalies - Physical

Cette section se concentre sur l'identification d'anomalies dans les données provenant d'éléments physiques comme les capteurs, les pompes et les valves. L'accent sera mis sur l'identification des comportements atypiques qui pourraient indiquer des défaillances matérielles ou des actions malveillantes ciblant l'intégrité physique du système de distribution d'eau.

## 1. Non-supervised algorithms

Nous explorerons ici l'efficacité des algorithmes non supervisés tels que l'Isolation Forest et le Local Outlier Factor pour détecter des anomalies sans étiquettes préalables. L'objectif est de comprendre comment ces modèles peuvent identifier des instances aberrantes parmi les données de fonctionnement normal.

### a. Isolation Forest (IF)

L'approche initiale a impliqué l'application de l'Isolation Forest avec ses paramètres par défaut, y compris le taux de contamination. Ce taux est crucial car il indique à l'algorithme la proportion attendue d'anomalies dans le dataset d'entraînement, permettant ainsi de prédire un nombre d'anomalies plus ou moins exact. Les résultats obtenus avec cette configuration par défaut n'étaient pas à la hauteur de nos attentes : sur **6177** anomalies détectées, seulement **1180** étaient de réelles anomalies, soit un taux de précision de **19%**. Cela est particulièrement décevant si l'on considère que notre dataset ne contient que **1459** anomalies réelles.

Pour améliorer notre modèle, nous avons ajusté le **taux de contamination** à une valeur fixe de **0.16**. Cette valeur a été déterminée de manière empirique, afin de refléter plus fidèlement la distribution réelle des anomalies dans notre jeu de données. Après ajustement, l'Isolation Forest a identifié exactement **1459** outliers, parmi lesquels **519** étaient de véritables anomalies, portant ainsi la précision à **35.5%**. Bien que cette performance reste modeste, elle représente une amélioration notable par rapport aux paramètres par défaut, réduisant également le nombre de faux positifs. Ces résultats suggèrent que l'Isolation Forest, avec un réglage fin du taux de contamination, peut être plus efficace pour notre ensemble de données physiques, même s'il y a encore une marge de progression considérable pour atteindre un niveau de précision optimal.

### b. Local Outlier Factor (LOF)

Le Local Outlier Factor (LOF), un algorithme non supervisé, a été testé sur notre dataset physique. Il a identifié **71** outliers, mais seulement **24** étaient de véritables anomalies, ce qui est très inférieur aux **1459** outliers réels. Ces résultats démontrent que le LOF, avec sa faible précision et son incapacité à détecter efficacement les anomalies, n'est pas approprié pour notre ensemble de données physiques.

## 2. Deep Learning

La section suivante discute de l'utilisation de méthodes de Deep Learning pour appréhender la complexité des données physiques et détecter des anomalies en exploitant la capacité des modèles à apprendre de grandes quantités de données, malgré la petite taille du dataset.

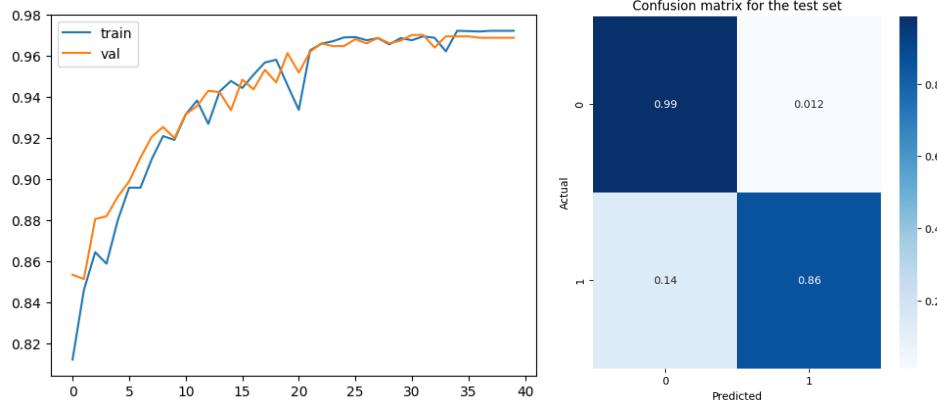
### a. Neural Network

Nous examinerons comment un réseau de neurones profond (DNN) peut être configuré et formé pour identifier des schémas non typiques dans les données des capteurs physiques.

Approche 1 - Classification binaire

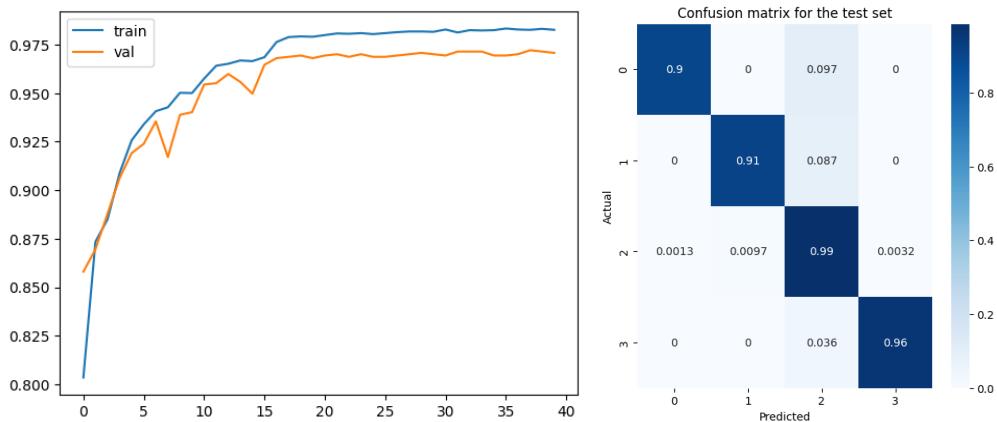
Pour notre modèle de réseau de neurones traitant des données physiques, nous avons décidé d'éliminer les informations contextuelles telles que le temps pour éviter les biais potentiels. Nous avons conçu un réseau dense simple avec trois couches cachées de 1024, 256 et 64 noeuds, totalisant **321 921** paramètres. Après avoir entraîné le modèle sur 40 époques avec une taille de lot

de 256, les résultats sur le jeu de test étaient prometteurs, atteignant une précision de **96,74%**. La matrice de confusion indiquait une bonne sensibilité et précision avec le modèle détectant 85% des anomalies tout en maintenant un faible taux de faux positifs. Cependant, le taux de faux négatifs s'élevait à 14%, suggérant une marge d'amélioration dans la reconnaissance des anomalies.

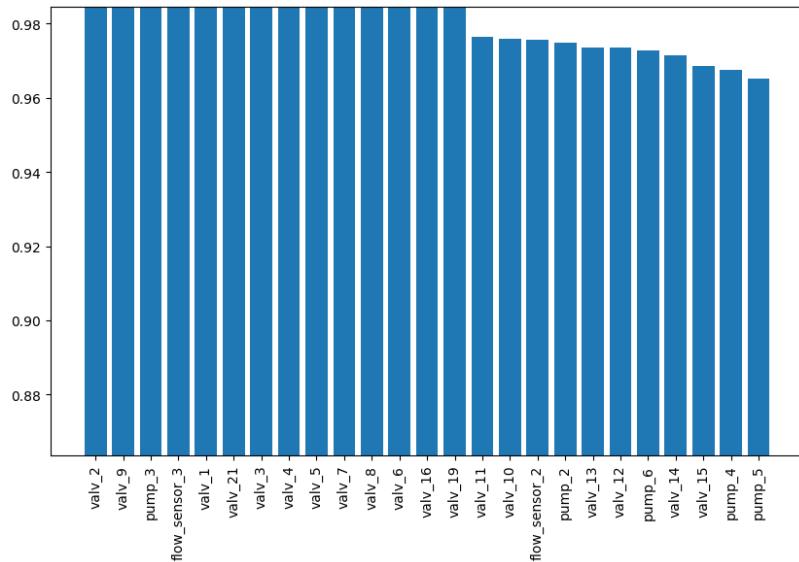


#### Approche 2 - Classification multiclasses

En passant à la classification multiclasses (plusieurs types d'attaques différentes), nous avons conservé la même architecture de réseau de neurones, à l'exception de la couche de sortie qui a été modifiée pour accueillir 5 neurones, correspondant à chaque type d'attaque. Nous avons dû exclure la cinquième classe du dataset en raison de son faible nombre de données. Les résultats de cette configuration, qui avait le même nombre de paramètres et était soumise au même processus d'entraînement, ont montré une précision de test de 97,72% avec une amélioration globale significative. La matrice de confusion révélait une réduction considérable du nombre de faux négatifs, bien que le modèle produisait toujours un nombre élevé de faux positifs. Cela dit, dans le contexte de la sécurité, il est souvent préférable d'avoir des faux positifs plutôt que des faux négatifs. Les métriques de performance, telles que le rappel, la précision et le score F1, ont toutes montré des améliorations, attestant de la capacité du modèle à différencier précisément les différents types d'attaques.



En termes d'importance des caractéristiques, il a été observé que de nombreuses caractéristiques semblaient aussi importantes les unes que les autres, ce qui suggère que le modèle ne surajuste pas et que toutes les caractéristiques introduites sont pertinentes pour la prédiction des classes d'anomalies. Cette distribution de l'importance des caractéristiques confirme la nécessité de toutes les intégrer dans le modèle pour capturer l'intégralité de la structure des données.

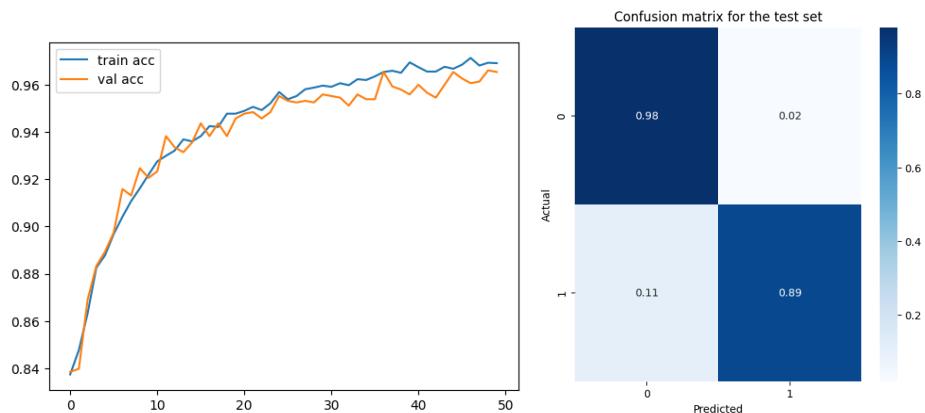


## b. LSTM

L'utilisation des réseaux LSTM sera envisagée, exploitant leur aptitude à identifier des dépendances temporelles étendues et à détecter des anomalies dans des séries chronologiques de données.

### Approche 1 - Classification binaire

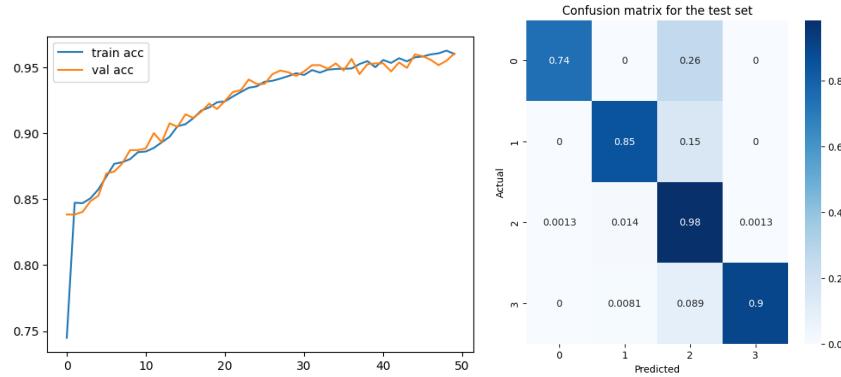
Dans notre étude des données physiques, nous avons déployé un réseau LSTM dans l'espérance de saisir des complexités temporelles plus profondes au sein des données. Avec une architecture simplifiée comprenant une seule couche LSTM, nous avons entraîné notre modèle pendant 50 époques, illustrées par une courbe d'apprentissage. Les résultats de la classification binaire ont été prometteurs avec une précision de test de 96.31% et un faible taux de perte de test de 0.0881. Néanmoins, la matrice de confusion a révélé un nombre significatif de faux négatifs, similaire à celui observé avec le réseau de neurones dense (DNN). Cette tendance indique que, bien que le modèle puisse identifier correctement la majorité des comportements normaux, il a tendance à mal classer un nombre non négligeable d'anomalies réelles comme étant normales.



### Approche 2 - Classification multiclasse

Pour la classification multiclasse, les performances sont restées élevées avec une précision de test de 96.20% et une perte de test de 0.1079. La matrice de confusion montre cependant que le modèle a du mal avec la troisième classe, la confondant fréquemment avec d'autres classes. Cela est

attribuable à la prédominance de cette classe dans l'ensemble de données, ce qui souligne une sensibilité du LSTM aux déséquilibres de classe. L'architecture LSTM, malgré sa capacité à capturer des dépendances à long terme, ne semble pas appropriée pour ce type de problème où l'équilibre des classes est crucial.



### 3. Classifiers

Cette partie évalue la performance des algorithmes de classification conventionnels pour la détection d'anomalies dans les données physiques, avec un accent sur leur capacité à discerner des comportements anormaux.

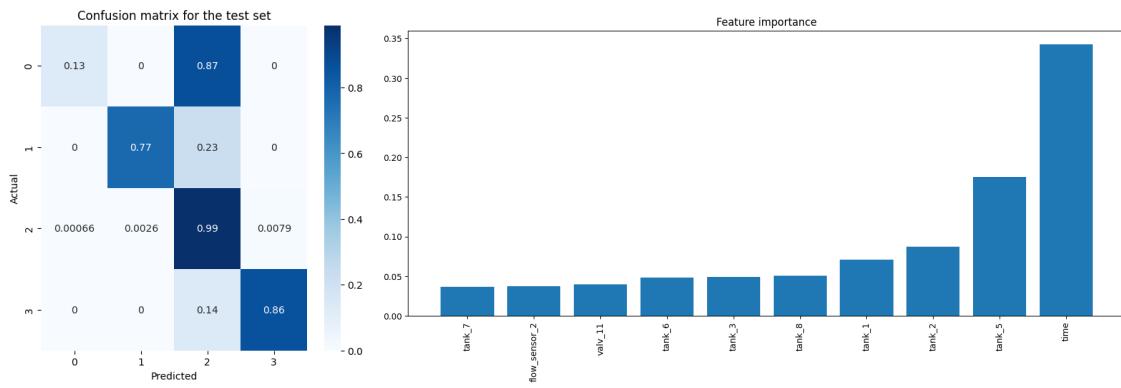
À présent, nous focaliserons exclusivement sur la classification multiclasse.

#### a. Decision Tree

Le modèle est défini avec des paramètres empiriques pour optimiser sa performance. Les paramètres sélectionnés sont une profondeur maximale de 8, l'utilisation du critère Gini pour la mesure de qualité des divisions, le choix de la meilleure division à chaque noeud, et un état aléatoire défini pour la reproductibilité. Le modèle est intégré dans un pipeline comprenant une étape de normalisation préalable via StandardScaler, essentielle pour équilibrer l'échelle des différentes caractéristiques.

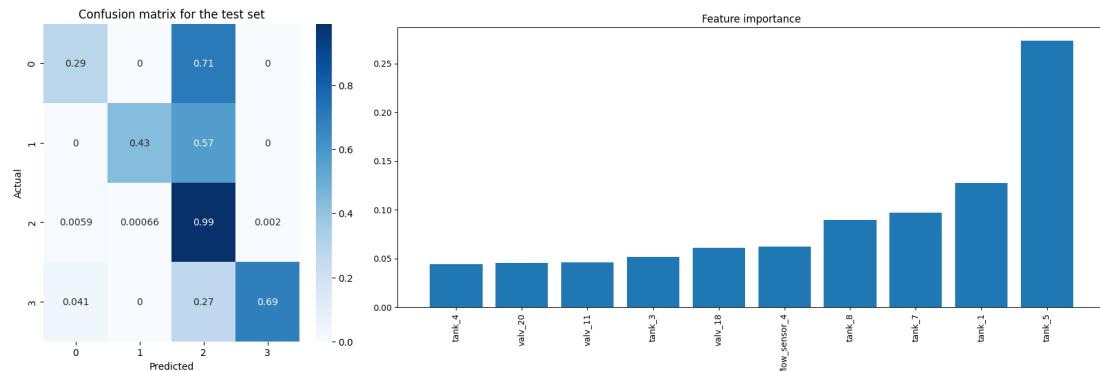
##### Approche 1 - Avec information contextuelle

Les résultats obtenus avec les données contextuelles, y compris la colonne temporelle, révèlent une précision remarquable (94.68%), une précision équilibrée et un rappel identiques (68.82%), ainsi qu'un score F1 solide (73.26%). Le coefficient de corrélation de Matthews (MCC), qui prend en compte le déséquilibre des classes, affiche une valeur élevée (81.34%), indiquant une performance globale de qualité. Cependant, l'arbre de décision montre une sensibilité aux déséquilibres des données, ce qui pourrait expliquer des résultats moins performants dans certaines classifications, comme le montre la matrice de confusion. Les caractéristiques les plus importantes identifiées sont le temps (0.35), le capteur de niveau du tank\_5 (0.30), et le tank\_2 (0.1).



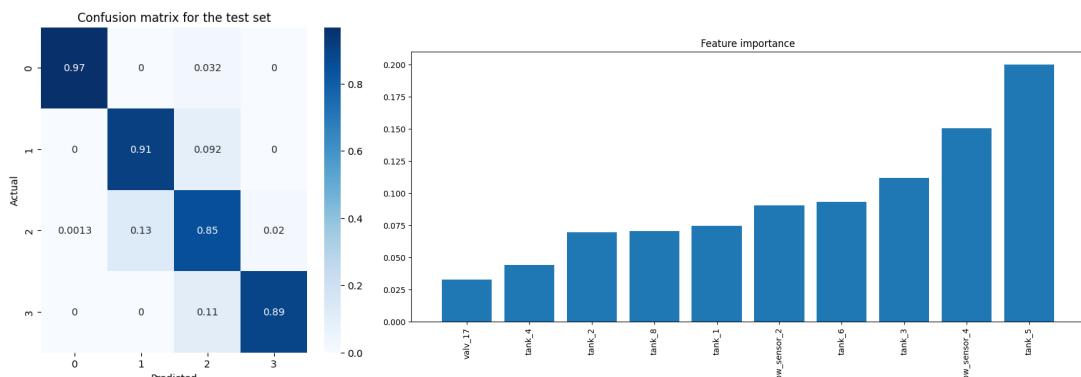
### Approche 2 - Sans information contextuelle

Lorsque l'information temporelle est retirée, la précision diminue légèrement (90.99%), tout comme le rappel (60.06%) et le score F1 (67.17%). Le MCC est également réduit (66.60%), indiquant une dégradation de la qualité de la prédiction. La matrice de confusion montre un changement dans la répartition des erreurs, confirmant l'impact du retrait de l'information temporelle sur la performance du modèle. Les caractéristiques importantes restent les mêmes que dans le scénario précédent, à l'exception de la valeur temporelle.



### Approche 3 - Sans information contextuelle et avec pondération des classes

En ajustant le poids des classes pour contrecarrer le déséquilibre des données, nous constatons une amélioration significative du rappel (90.53%) et de la précision équilibrée, bien que la précision globale diminue (86.10%). Le score F1 (82.10%) et le MCC (66.57%) montrent une capacité accrue du modèle à détecter les anomalies, confirmée par une matrice de confusion montrant une détection à 100% des anomalies. Cependant, des confusions persistent, notamment entre les classes 2 et 1, ainsi que 3 et 2. Les caractéristiques les plus importantes dans ce scénario sont le capteur de niveau du tank\_5 (0.2), le capteur de débit flow\_sensor\_4 (0.15), et le tank\_3 (0.1).

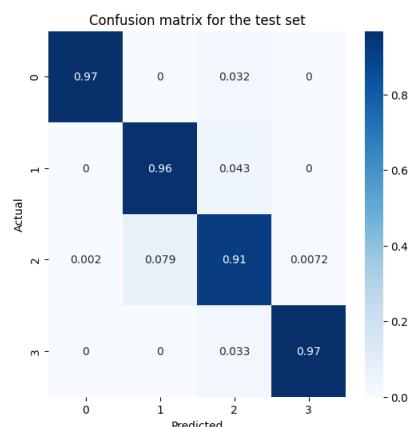


## b. Random Forest

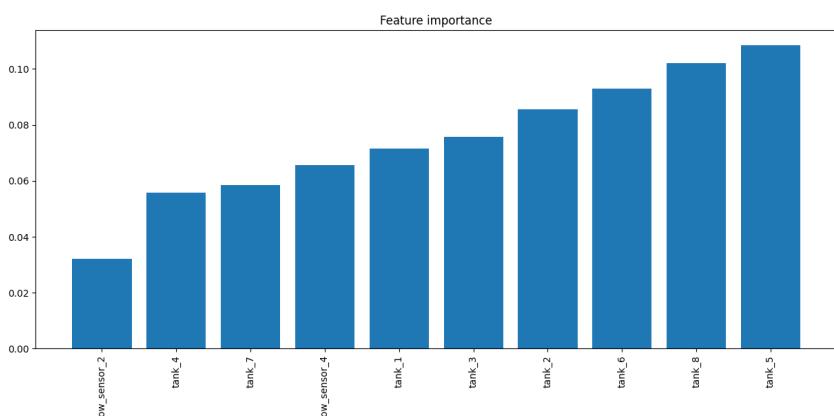
Notre étude de la détection d'anomalies au sein du dataset physique comprend l'application de Random Forest avec des poids de classe ajustés, pour aborder les problèmes potentiels liés au déséquilibre des classes. Avec les paramètres spécifiés - une profondeur maximale de 8, le critère gini, 100 estimateurs, un état aléatoire fixé à 42, et des poids de classe équilibrés - nous avons construit notre modèle à l'intérieur d'un pipeline avec une mise à l'échelle standard des caractéristiques.

Les résultats obtenus avec cette configuration ont été très prometteurs. Une précision de 92.02% montre une capacité considérable du modèle à identifier correctement les états normaux et anormaux des données physiques. Plus impressionnant encore, le rappel atteint 95.09%, indiquant que le modèle est extrêmement compétent pour détecter les véritables anomalies. Le score F1 de 88.43% et le MCC de 78.90% confirment l'excellence de la performance du modèle, réconciliant précision et rappel de manière efficace. La précision équilibrée, égale au rappel dans ce contexte, souligne la capacité du modèle à traiter de manière égale toutes les classes malgré leur déséquilibre.

La matrice de confusion affiche des valeurs diagnostiques de précision pour les différentes classes, avec un taux de classification correcte particulièrement élevé pour la plupart des états, signalant que le modèle distingue avec précision les différentes conditions d'anomalie par rapport aux états normaux.



Quant à l'importance des caractéristiques, le modèle met en évidence l'influence significative de certains capteurs, en particulier ceux liés aux réservoirs. Les capteurs tank\_5, tank\_8 et tank\_6 émergent comme les plus influents, avec des poids respectifs de 15%, 10% et 9%. Cela peut refléter la sensibilité du modèle aux variations dans les niveaux des réservoirs, qui sont cruciaux dans l'identification des anomalies physiques dans le système.



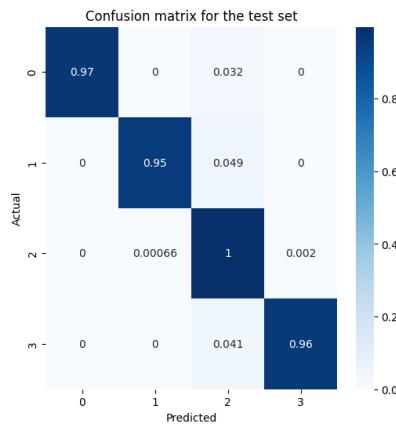
Cette analyse démontre que le Random Forest, lorsqu'il est correctement paramétré et appliqué avec des poids de classe équilibrés, surpassé les méthodes précédentes et fournit une approche robuste pour la détection d'anomalies dans les systèmes physiques. En comprenant précisément comment les caractéristiques sont reliées et en mettant en œuvre une stratégie de pondération équilibrée, nous sommes parvenus à un modèle à la fois précis et fiable pour identifier les comportements anormaux dans les données physiques.

### c. XGBoost

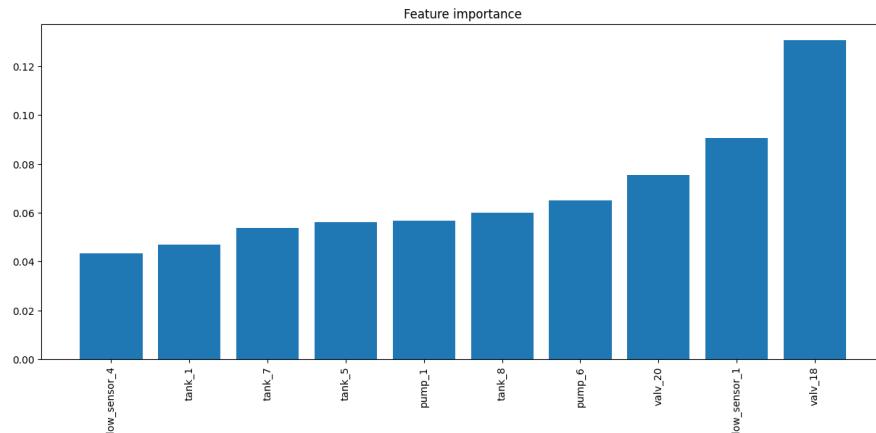
Enfin, nous avons mis en œuvre l'algorithme XGBoost, réputé pour sa performance en matière de classification. Les hyperparamètres sélectionnés pour cette tâche étaient les suivants : une profondeur maximale de 8, 100 estimateurs et un état aléatoire fixé à 42 pour garantir la reproductibilité des résultats. Comme avant, la pipeline a été créé en combinant un StandardScaler, pour normaliser les données, avec le classificateur XGBoost, optimisant ainsi le traitement des données d'entrée.

Les résultats obtenus avec XGBoost sur les données physiques ont été exceptionnels, avec une précision globale de 0.9902, un rappel de 0.9688, un score F1 de 0.9792, un coefficient de corrélation de Matthews (MCC) de 0.9674 et une précision équilibrée de 0.9688. Ces métriques indiquent une excellente performance du modèle à la fois en termes de précision globale et de capacité à détecter des anomalies de manière équilibrée à travers les différentes classes.

La matrice de confusion reflète cette performance avec des valeurs élevées de vrais positifs pour la classification normale et les divers types d'anomalies, et des erreurs limitées à une fraction minoritaire des prédictions. Plus spécifiquement, le modèle a correctement identifié les anomalies avec une précision remarquable de 97% pour les cas normaux et 96% pour les anomalies les plus fréquentes, avec quelques faux positifs et faux négatifs distribués de manière minoritaire.



L'analyse de l'importance des caractéristiques a révélé des perspectives intéressantes sur les facteurs qui influencent le plus le modèle. Contrairement aux autres modèles examinés, XGBoost a identifié 'valv\_18' comme la caractéristique la plus influente, avec un score d'importance de 0.15, suivie de 'flow\_sensor\_1' à 0.09 et 'valv\_20' à 0.087. Cela suggère que l'état de ces vannes et le débit enregistré par le capteur de flux sont des indicateurs clés pour déceler des anomalies dans les données physiques.



## V. Adversarial attack

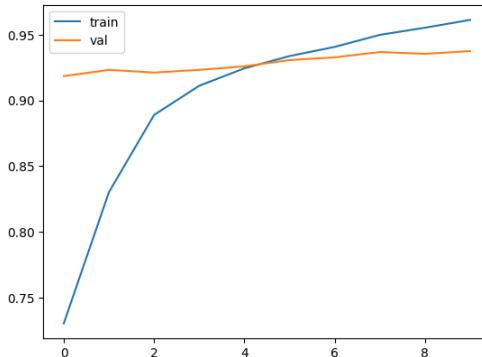
Nous avons fait une courte étude sur la robustesse de nos modèles face aux attaques adversaires. Nous nous sommes concentrés sur les modèles DNN (deep neural network) et sur les deux types de dataset (network et physical). Une des façons les plus simples de générer des points destinés à tromper le modèle et d'utiliser la Fast Gradient Sign Method (FGSM). Elle consiste à partir du gradient d'une entrée légitime, de calculer une perturbation d'une certaine ampleur (définie par l'hyper paramètre epsilon) et de l'ajouter à l'exemple initial. Cela impact en général fortement les performances du réseau et intégrer ces données lors de l'entraînement permet de rendre le modèle plus robuste face à ce genre d'attaques.

Sur le dataset physical, on remarque bien que notre réseau penne à classifier correctement les données qui ont été altérées passant d'une précision de 97% à 49%.

```
# let's verify we still have the same accuracy
predictions = classifier.predict(X_test)
accuracy = np.sum(np.argmax(predictions, axis=1) == y_test) / len(y_test)
print("Accuracy on benign test examples: {}%".format(accuracy * 100))
✓ 0.1s
Accuracy on benign test examples: 97.66304347826087%
```

```
predictions = classifier.predict(x_test_adv)
accuracy = np.sum(np.argmax(predictions, axis=1) == y_test) / len(y_test)
print("Accuracy on adversarial test examples: {}%".format(accuracy * 100))
✓ 0.1s
Accuracy on adversarial test examples: 49.07608695652174%
```

Pour contrer ce problème, on peut fine-tuner le modèle grâce à ces mêmes exemples et monitorer l'entraînement avec le même set de validation afin de s'assurer que l'on ne perd pas en qualité sur les prédictions initiales.

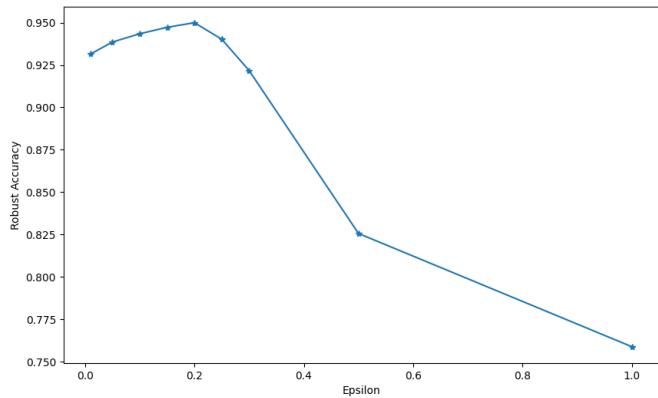


```
model_copy.evaluate(X_test, y_test)
✓ 0.3s
1/58 [=====] - ETA: 2s - loss: 0.2381 - accuracy: 0.9062
58/58 [= =====] - 0s 5ms/step - loss: 0.1286 - accuracy: 0.9489
[0.1286405771970749, 0.948913037776947]

model_copy.evaluate(x_test_adv, y_test)
✓ 0.2s
1/58 [=====] - ETA: 1s - loss: 0.1651 - accuracy: 0.9375
58/58 [= =====] - 0s 3ms/step - loss: 0.1358 - accuracy: 0.9440
[0.13581281900405884, 0.9440217614173889]
```

On obtient alors de très bonnes performances sur les exemples altérés passant de 49% à 95% au détriment d'une baisse de 2% sur le jeu de test.

Enfin, on peut s'intéresser à la robustesse du modèle en fonction de l'ampleur des perturbations.



On remarque que le modèle est maintenant robuste jusqu'à un epsilon de 0.2 puis devient fragile ensuite. Ce résultat est logique puisque le modèle a été entraîné à nouveau avec des exemples altérées en utilisant 0.2 comme valeur pour epsilon.

Cet exemple d'attaque adverse, aussi simple qu'il soit, permet de mettre en lumière la vulnérabilité de certains modèles qu'on ne pense pas être attaquables. Ici, si un attaquant parvenait bel et bien à générer des données erronées, cela créerait une faille très importante dans le protocole de sécurité du système.

## VI. Conclusion

À l'issue de notre analyse approfondie du dataset HITL, plusieurs conclusions émergent clairement. D'abord, nous avons mis en évidence l'importance d'un prétraitement minutieux, une étape souvent sous-estimée, mais vitale pour la qualité de la modélisation et de la détection d'anomalies. L'exploration des données a révélé des différences significatives entre les datasets physiques et réseau, notamment en termes de répartition des classes et de la nature des features. Cette distinction a guidé nos choix méthodologiques tout au long du projet.

L'analyse des algorithmes non-supervisés a révélé leurs limitations dans le contexte de nos datasets, notamment en raison de la complexité des patterns d'anomalies et de la grande dimensionnalité des données. En revanche, les techniques de Deep Learning, malgré l'atteinte d'un plateau de performance, ont montré une certaine promesse, en particulier dans la distinction entre les comportements normaux et anormaux. Nous avons pu également montrer la sensibilité des réseaux de Deep Learning face aux attaques adverses et la manière dont on pouvait renforcer la fiabilité du modèle en intégrant des entrées altérées dans l'entraînement.

L'approche par classification a démontré l'utilité des modèles traditionnels comme les arbres de décision et la forêt aléatoire, mais c'est XGBoost qui a émergé comme le plus performant, surpassant les autres méthodes dans presque tous les aspects de la détection d'anomalies. Notamment, XGBoost a brillé par sa capacité à traiter les features catégoriques sans transformation préalable et sa précision élevée dans la prédiction des différentes classes d'anomalies.

Ce rapport met en lumière la valeur indéniable des données de haute qualité et de l'ingénierie des features dans la construction de modèles de machine learning robustes. Il souligne également l'importance d'une sélection de modèles adaptée à la nature des données et aux spécificités des tâches de classification. Enfin, il confirme la position prédominante de XGBoost dans l'arsenal du data scientist pour les tâches complexes de classification, tout en rappelant que l'exploration de nouvelles

méthodes et l'amélioration continue des modèles existants restent essentielles pour progresser dans le domaine de la cybersécurité des infrastructures critiques.