CYBERML – Project 1 : Classification, détection d'anomalies et attaques adversariales

CIC IoT-DIAD 2024 (Flow Based Features)

Cours : CYBERML
Année : 2025–2026
Enseignant : Pierre Parrend
Auteurs : *à compléter*

25 janvier 2026

Résumé

Ce rapport présente une chaîne de traitement *batch* pour l'analyse de données de flux réseau issues du dataset CIC IoT-DIAD 2024 (Flow Based Features) [1], avec deux objectifs : (i) détecter des attaques via détection d'anomalies non supervisée, et (ii) classifier les attaques (binaire puis multi-classes) afin de *tracker* des familles et sous-types. Trois méthodes non supervisées sont comparées (Isolation Forest [2], One-Class SVM [3], score de reconstruction PCA [4]) et trois méthodes supervisées (régression logistique, forêt aléatoire, XGBoost [5]). Sur la tâche binaire *Benign vs Attack*, XGBoost obtient une AUPRC de 0.982 tandis que la PCA reconstruction fournit la meilleure balanced accuracy (0.884) avec un très faible taux de faux positifs. En multi-classes, XGBoost atteint une balanced accuracy de 0.784 au niveau *familles*, et la régression logistique une balanced accuracy de 0.720 au niveau *sous-types* (subset filtré). Enfin, une étude d'attaque adversariale FGSM [6] sur la régression logistique met en évidence une forte dégradation : l'accuracy robuste chute de 0.728 à 0.172 pour $\varepsilon = 0.100$.

Abstract

This report describes a batch ML pipeline for network-flow cybersecurity analytics on the CIC IoT-DIAD 2024 (Flow Based Features) dataset [1]. We address (i) unsupervised anomaly detection for attack detection and (ii) supervised classification for attack tracking (binary, then multiclass at family and subtype levels). We benchmark three unsupervised methods (Isolation Forest [2], One-Class SVM [3], PCA reconstruction error [4]) and three supervised classifiers (logistic regression, random forest, XGBoost [5]). On *Benign vs Attack*, XGBoost reaches 0.982 AUPRC, while PCA reconstruction provides the best balanced accuracy (0.884) with very few false positives. For multiclass, XGBoost yields 0.784 balanced accuracy at the family level, and logistic regression 0.720 at the (filtered) subtype level. Finally, an FGSM adversarial study [6] on logistic regression shows a sharp robustness drop : robust accuracy falls from 0.728 to 0.172 at $\varepsilon = 0.100$.

Table des matières

Table des figures

Liste des tableaux

1 Introduction et contexte cybersécurité

1.1 Détection vs suivi d'attaques

Dans un SOC, l'objectif n'est pas seulement de détecter un événement malveillant, mais aussi de qualifier l'incident et d'en suivre l'évolution : une alerte *binaire* (attaque/normal) soutient la détection précoce, tandis qu'une classification *multi-classes* permet d'orienter l'analyse (famille d'attaque, puis sous-type). Ces deux niveaux correspondent à des exigences opérationnelles distinctes : (i) minimiser les faux négatifs (attaques manquées) en détection, tout en maîtrisant les faux positifs (alert fatigue), et (ii) fournir un *signal de suivi* suffisamment stable pour déclencher des playbooks de réponse.

1.2 Modèles ML sur des caractéristiques de flux

Le dataset étudié fournit des caractéristiques de flux réseau (flow-based) extraites de PCAP : volumes, durées, statistiques de paquets, etc. Ces représentations sont adaptées au traitement *batch* et à l'apprentissage supervisé, mais posent des défis typiques : forte dimension, distributions hétérogènes, et risque de fuite de données lorsque des flows quasi-identiques (issus du même fichier) se retrouvent en train et test.

1.3 Enjeux méthodologiques : éviter la fuite de données

Un split aléatoire par lignes peut sur-estimer la performance si des flows d'un même fichier (donc très corrélés) sont mélangés entre train/test. Nous adoptons un split *par fichier* via `GroupShuffleSplit` (group split) pour obtenir une évaluation plus crédible : un fichier de flows est soit dans le train, soit dans le test.

2 Présentation et caractérisation du dataset

2.1 Dataset CIC IoT-DIAD 2024 (Flow Based Features)

Le dataset CIC IoT-DIAD 2024 (Flow Based Features) [1] contient des fichiers CSV de caractéristiques de flux, organisés par familles et sous-types d'attaques. Dans ce projet, les familles ciblées sont : Benign, BruteForce, DDoS, DoS, Mirai, Recon, Spoofing, Web-Based (8 familles).

2.2 Chargement RAM-safe et subset expérimental

Pour rendre l'analyse exécutable sur une machine limitée, le notebook implémente un chargement *RAM-safe* par *budgets* : (i) 10 000.000 lignes max par fichier CSV, (ii) 60 000.000 lignes max par famille, (iii) 300 000.000 lignes max au total. Après nettoyage (typage numérique, suppression des NaN et des valeurs infinies), le subset chargé contient 274 287.000 flows et 77.000 variables numériques (hors labels). Le split *par fichier* est réalisé sur 34.000 groupes (fichiers présents dans le subset).

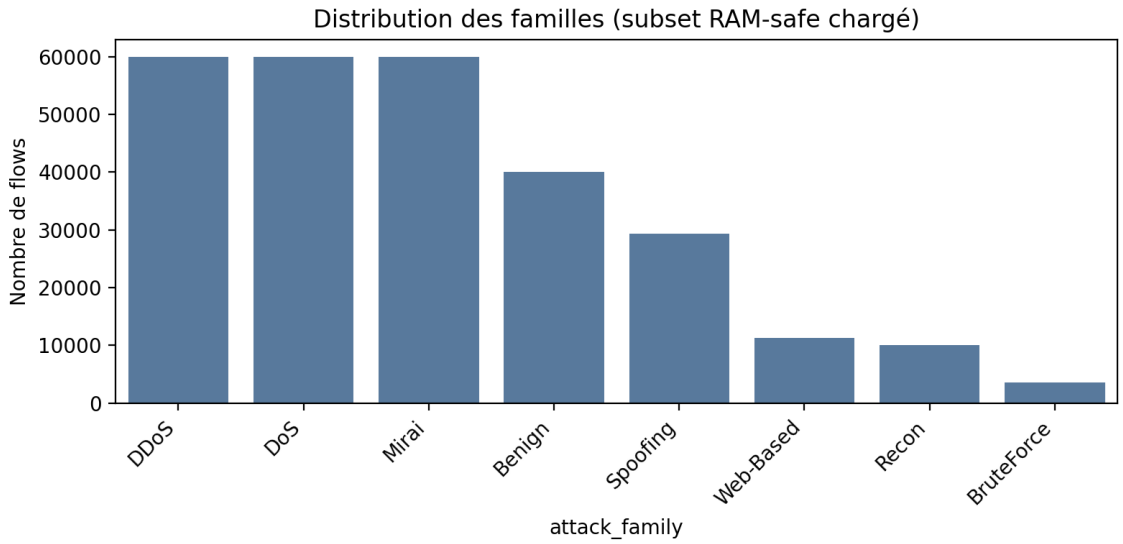


FIGURE 1 – Répartition des familles dans le subset chargé (budgets RAM-safe). Le cap par famille (60 000.000) conduit à une répartition non représentative du dataset complet.

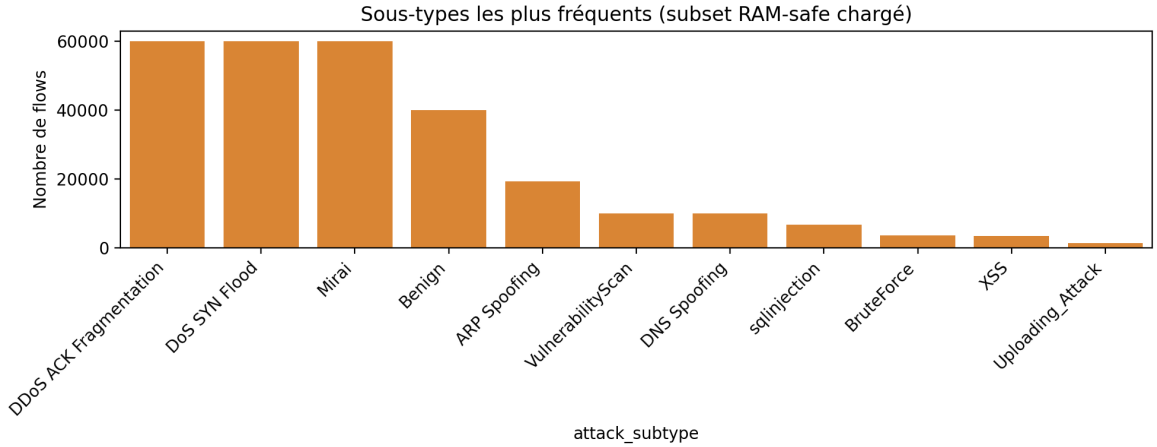


FIGURE 2 – Sous-types les plus fréquents dans le subset chargé (budgets RAM-safe). Dans ce subset, certaines familles sont dominées par un seul sous-type (ordre de lecture des fichiers + cap par famille).

TABLE 1 – Synthèse du subset utilisé : nombre de flows chargés par famille (après nettoyage) et nombre de fichiers CSV disponibles dans le dataset local.

Famille	Flows chargés	Fichiers CSV (dataset)
DDoS	60000	61
DoS	60000	30
Mirai	60000	29
Benign	40000	4
Spoofing	29343	3
Web-Based	11325	3
Recon	10000	1
BruteForce	3619	1

2.3 Déséquilibre et implications métriques

Le subset chargé est déséquilibré au niveau *familles* (e.g. BruteForce $\ll$ DDoS/DoS/Mirai). La tâche binaire *Benign vs Attack* présente un déséquilibre inverse des scénarios usuels : la classe *Attack* est majoritaire dans le subset, conséquence du cap par famille (Fig. 1). Dans ce contexte, l'accuracy peut être trompeuse ; nous reportons systématiquement : précision, rappel, AUPRC, balanced accuracy et MCC.

3 Chaîne complète de traitement des données

3.1 Prétraitement

Le chargement applique les étapes suivantes :

— **Sélection de variables** : exclusion des colonnes identifiantes (*Flow ID*, adresses IP, ports, timestamp) pour limiter les fuites et les artefacts de collection.

— **Typage et nettoyage** : conversion en numérique, suppression des valeurs infinies et des lignes contenant des valeurs manquantes.

— **Réduction mémoire** : conversion en `float32` et sous-échantillonnage contrôlé par budgets.

3.2 Feature engineering

Les caractéristiques sont utilisées telles quelles (features de flux), puis standardisées par un `StandardScaler` (centrage-réduction) afin d'homogénéiser l'échelle des variables, ce qui est particulièrement important pour les méthodes à base de distance ou de marge (SVM, PCA, régression logistique).

3.3 Séparation train/test (group split)

Pour chaque scénario, le split est réalisé par fichier (group split) avec un taux de test de 20%. Le split impose que toutes les classes soient présentes dans le *train* (contrainte nécessaire pour

les classifieurs multi-classes), mais il ne garantit pas que toutes les classes apparaissent en test ; cette limite est discutée en Section 7.

3.4 Définition des scénarios

Trois scénarios sont considérés :

— **Binaire** : $y = 1$ si `attack_family`≠Benign, sinon $y = 0$.
— **Familles** : y correspond à l'identifiant de famille (8 classes).
— **Sous-types** : pour éviter des classes trop rares et des splits instables, les sous-types sont filtrés (min 15 000.000 occurrences dans `df_all`), les 10.000 plus fréquents sont conservés, les autres sont regroupés sous *Other*, puis chaque classe est plafonnée à 10 000.000 flows.

4 Méthodes de détection d'anomalies

4.1 Isolation Forest

Isolation Forest [2] isole les points atypiques via des arbres aléatoires ; les anomalies sont isolées en peu de coupures et reçoivent un score élevé. Le modèle est entraîné uniquement sur du trafic Benign, puis évalué sur le test complet.

4.2 One-Class SVM

One-Class SVM [3] estime le support d'une distribution (ici, Benign) et détecte les points situés en dehors de la frontière. Cette méthode peut être efficace mais sensible au choix de noyau et à l'échelle des variables, d'où l'intérêt de la standardisation.

4.3 PCA + erreur de reconstruction

La PCA [4] apprend un sous-espace expliquant 95% de la variance du trafic Benign. Un flow est considéré comme anomalie si son erreur de reconstruction dépasse un seuil (ici le 95e percentile des erreurs de reconstruction sur le train Benign). Cette approche est simple, interprétable et robuste pour des anomalies qui dévient significativement du sous-espace normal.

5 Méthodes de classification

Les implémentations utilisées proviennent principalement de `scikit-learn` [7] et `xgboost` [5].

5.1 Régression logistique

La régression logistique (multinomiale en multi-classes) fournit un classifieur linéaire probabiliste. Elle est rapide, bien calibrée et sert de baseline solide ; elle est cependant sensible aux perturbations adversariales (Section 8).

5.2 Forêt aléatoire

Les forêts aléatoires agrègent des arbres de décision entraînés sur des sous-échantillons ; elles capturent des non-linéarités tout en restant relativement robustes au bruit. Le paramètre `class_weight` est activé pour limiter l'effet de déséquilibre.

5.3 XGBoost

XGBoost [5] réalise un boosting d'arbres de décision, souvent performant sur des données tabulaires. Nous utilisons `tree_method=hist` (CPU) pour l'efficacité, et des hyperparamètres adaptés à un compromis biais/variance.

6 Benchmark expérimental

6.1 Métriques

Pour la tâche binaire, on calcule :
— **Précision** et **rappel** sur la classe *Attack*.
— **AUPRC** (aire sous la courbe précision–rappel), adaptée aux déséquilibres et à l'optimisation de seuil.
— **Balanced accuracy** : $\mathrm{BA} = \frac{1}{2}(\mathrm{TPR} + \mathrm{TNR})$.
— **MCC** [8], corrélation entre prédictions et vérité terrain :

$$\mathrm{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}.$$

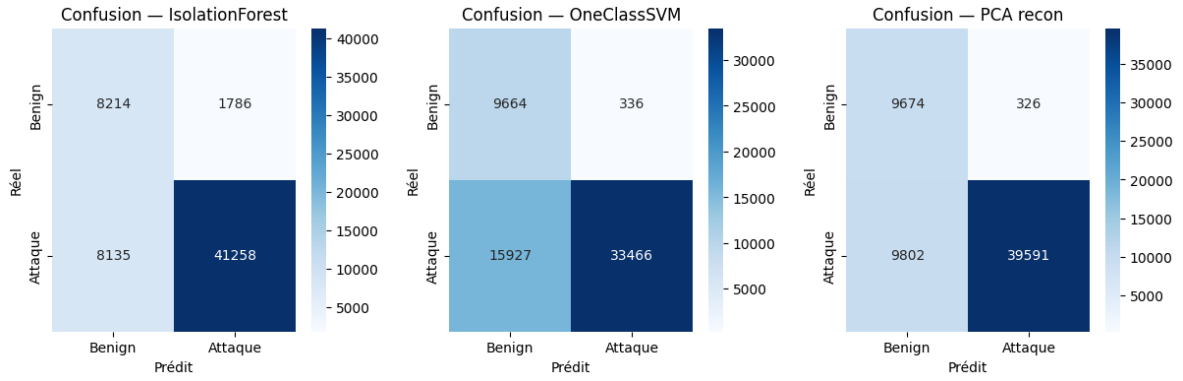
En multi-classes, on rapporte précision/rappel macro et pondérés, balanced accuracy, MCC et une AUPRC macro (one-vs-rest) sur les classes présentes.

6.2 Détection binaire : non supervisé

Les modèles non supervisés sont entraînés uniquement sur Benign puis évalués sur le test complet. Le tableau 2 montre que la PCA reconstruction obtient la meilleure balanced accuracy (0.884) et un très faible nombre de faux positifs (FP=326.000), ce qui est intéressant du point de vue opérationnel.

TABLE 2 – Résultats binaire (non supervisé) sur le test : métriques et matrice de confusion agrégée.

Modèle	Précision	Rappel	AUPRC	Balanced Acc.	MCC	TN	FP	FN	TP
Non supervisé - IsolationForest	0.959	0.835	0.967	0.828	0.550	8214	1786	8135	41258
Non supervisé - OneClassSVM	0.990	0.678	0.970	0.822	0.487	9664	336	15927	33466
Non supervisé - Erreur de reconstruction PCA	0.992	0.802	0.974	0.884	0.613	9674	326	9802	39591



(a) Isolation Forest (b) One-Class SVM (c) PCA (reconstruction)

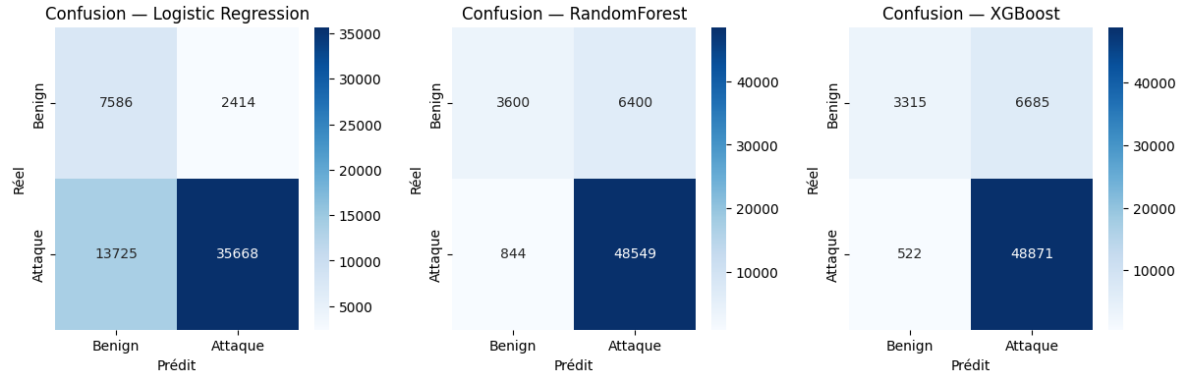
FIGURE 3 – Matrices de confusion – détection binaire non supervisée (test). Les faux positifs (Benign→Attaque) se traduisent directement en alertes inutiles.

6.3 Détection binaire : supervisé

Le tableau 3 compare les trois classifieurs supervisés. XGBoost maximise l'AUPRC (0.982) et le rappel (0.989), mais avec un coût important en faux positifs (FP=6685.000 sur 10 000.000 Benign), ce qui pénalise la balanced accuracy (0.660). À l'inverse, la régression logistique réduit fortement les faux positifs (FP=2414.000) au prix d'un rappel plus faible (0.722).

TABLE 3 – Résultats binaire (supervisé) sur le test : métriques et matrice de confusion agrégée.

Modèle	Précision	Rappel	AUPRC	Balanced Acc.	MCC	TN	FP	FN	TP
Supervisé - Régression logistique	0.937	0.722	0.960	0.740	0.375	7586	2414	13725	35668
Supervisé - Forêt aléatoire	0.884	0.983	0.930	0.671	0.488	3600	6400	844	48549
Supervisé - XGBoost (CPU hist)	0.880	0.989	0.982	0.660	0.489	3315	6685	522	48871



(a) Régression logistique (b) Forêt aléatoire (c) XGBoost

FIGURE 4 – Matrices de confusion – détection binaire supervisée (test). Un modèle à fort rappel peut rester difficilement exploitable si le taux de faux positifs est trop élevé (alert fatigue).

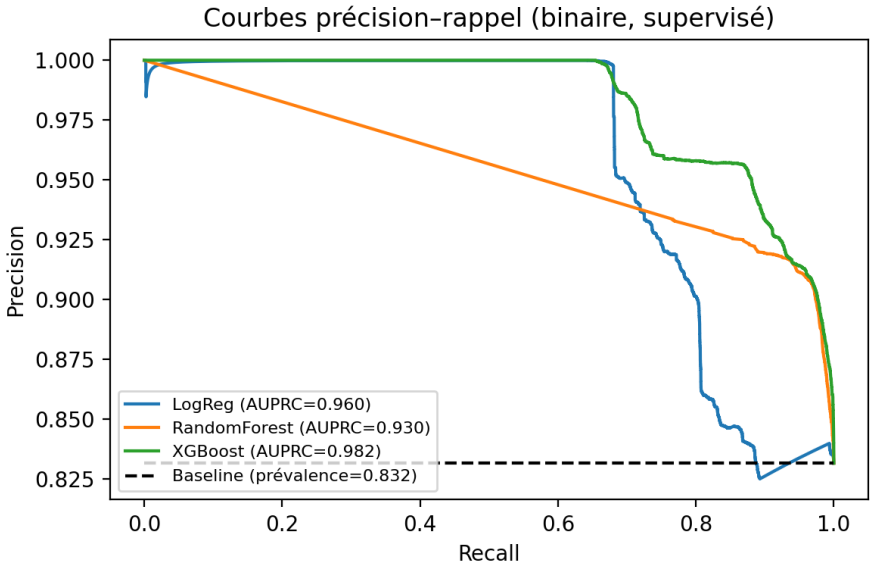


FIGURE 5 – Courbes précision–rappel en binaire supervisé. Elles permettent de choisir un seuil en fonction du compromis SOC (réduire FP vs réduire FN). La ligne pointillée correspond à la prévalence de la classe Attack dans le test.

6.4 Classification multi-classes : familles

La tâche *familles* vise à *tracker* le type d'attaque à un niveau exploitable. Le tableau 4 montre que XGBoost est meilleur en moyenne (balanced accuracy 0.784, MCC 0.752), tandis que la régression logistique souffre davantage des non-linéarités et du déséquilibre. Les métriques *pondérées* (dominées par les classes majoritaires) sont très élevées, mais les métriques *macro*

restent basses ; cela reflète le fait que les classes minoritaires sont difficilement apprises et/ou absentes du test.

TABLE 4 – Résultats multi-classes (familles) sur le test.

Modèle	Précision macro	Rappel macro	Précision pondérée	Rappel pondéré	AUPRC macro (OvR)	Balanced Acc.	MCC
Multiclass - LogisticRegression	0.392	0.306	0.857	0.723	0.751	0.611	0.626
Multiclass - RandomForest	0.429	0.380	0.904	0.807	0.826	0.760	0.731
Multiclass - XGBoost (hist)	0.435	0.392	0.912	0.822	0.860	0.783	0.752

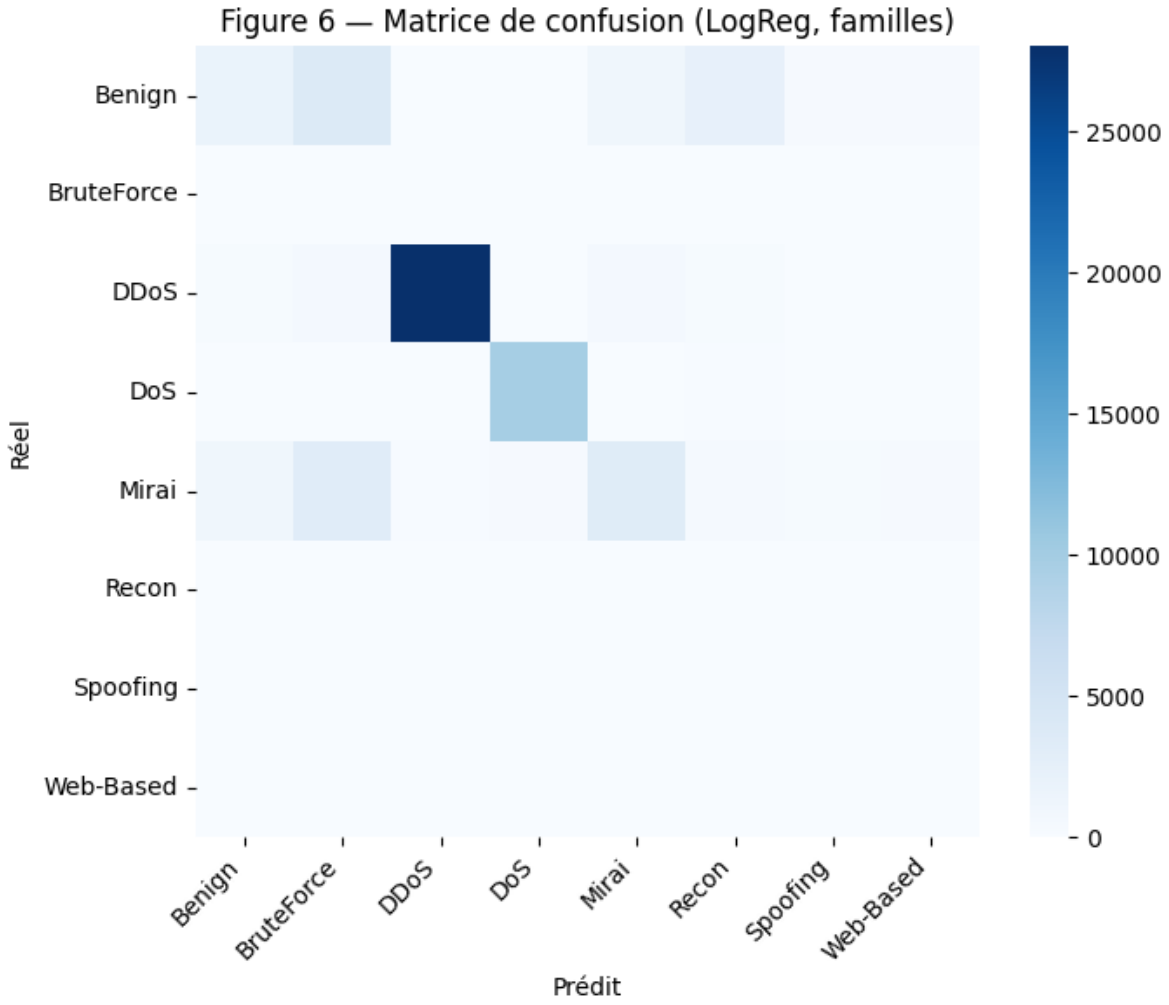


FIGURE 6 – Matrice de confusion (familles) – régression logistique. Les classes absentes du test apparaissent en lignes nulles (limite du group split sur un subset).

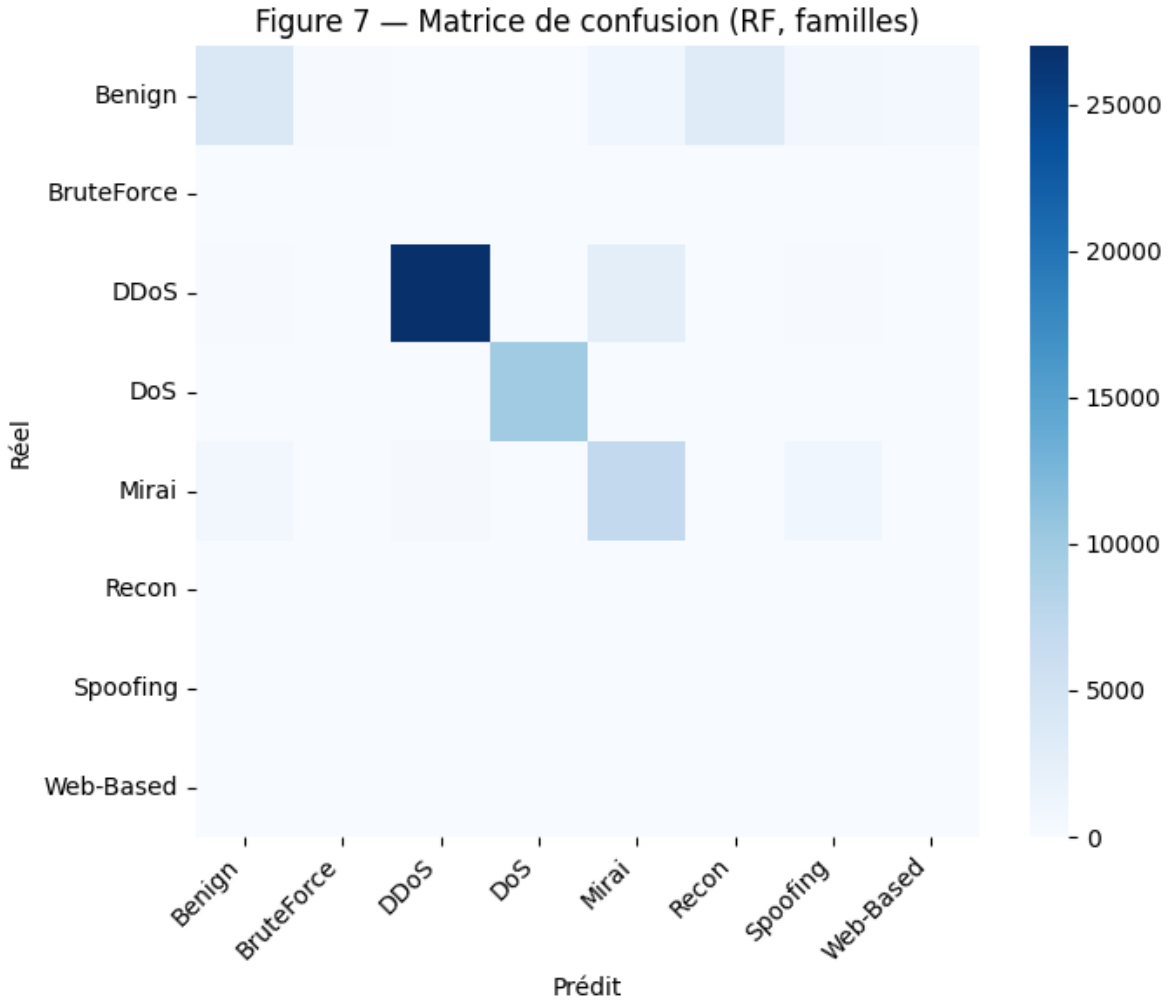


FIGURE 7 – Matrice de confusion (familles) – forêt aléatoire.

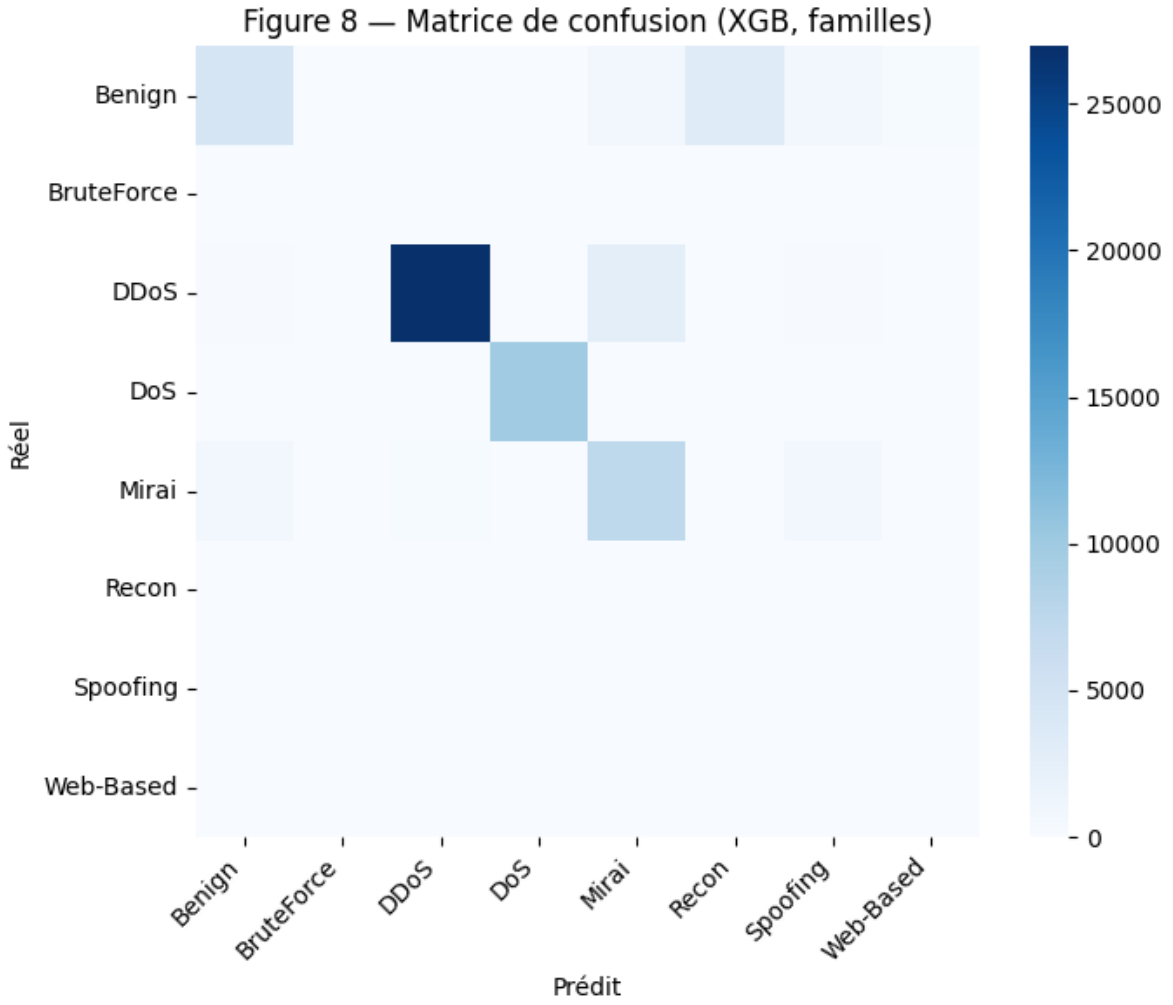


FIGURE 8 – Matrice de confusion (familles) – XGBoost.

6.5 Classification multi-classes : sous-types

Le niveau *sous-types* augmente la granularité mais exacerbe les difficultés : plus de classes, souvent plus rares, et risque de classes manquantes en train/test. Le notebook met en place un filtrage et un regroupement (*Other*) puis plafonne chaque classe à 10 000.000 flows pour stabiliser l'entraînement. Sur ce subset, la régression logistique obtient la meilleure balanced accuracy (0.720) tandis que XGBoost maximise l'AUPRC macro (0.869).

TABLE 5 – Résultats multi-classes (sous-types filtrés) sur le test.

Modèle	Précision macro	Rappel macro	Précision pondérée	Rappel pondéré	AUPRC macro (OvR)	Balanced Acc.	MCC
Subtype - LogisticRegression	0.521	0.480	0.787	0.654	0.838	0.720	0.587
Subtype - RandomForest	0.524	0.456	0.798	0.609	0.801	0.683	0.557
Subtype - XGBoost (hist)	0.527	0.457	0.803	0.611	0.869	0.685	0.561

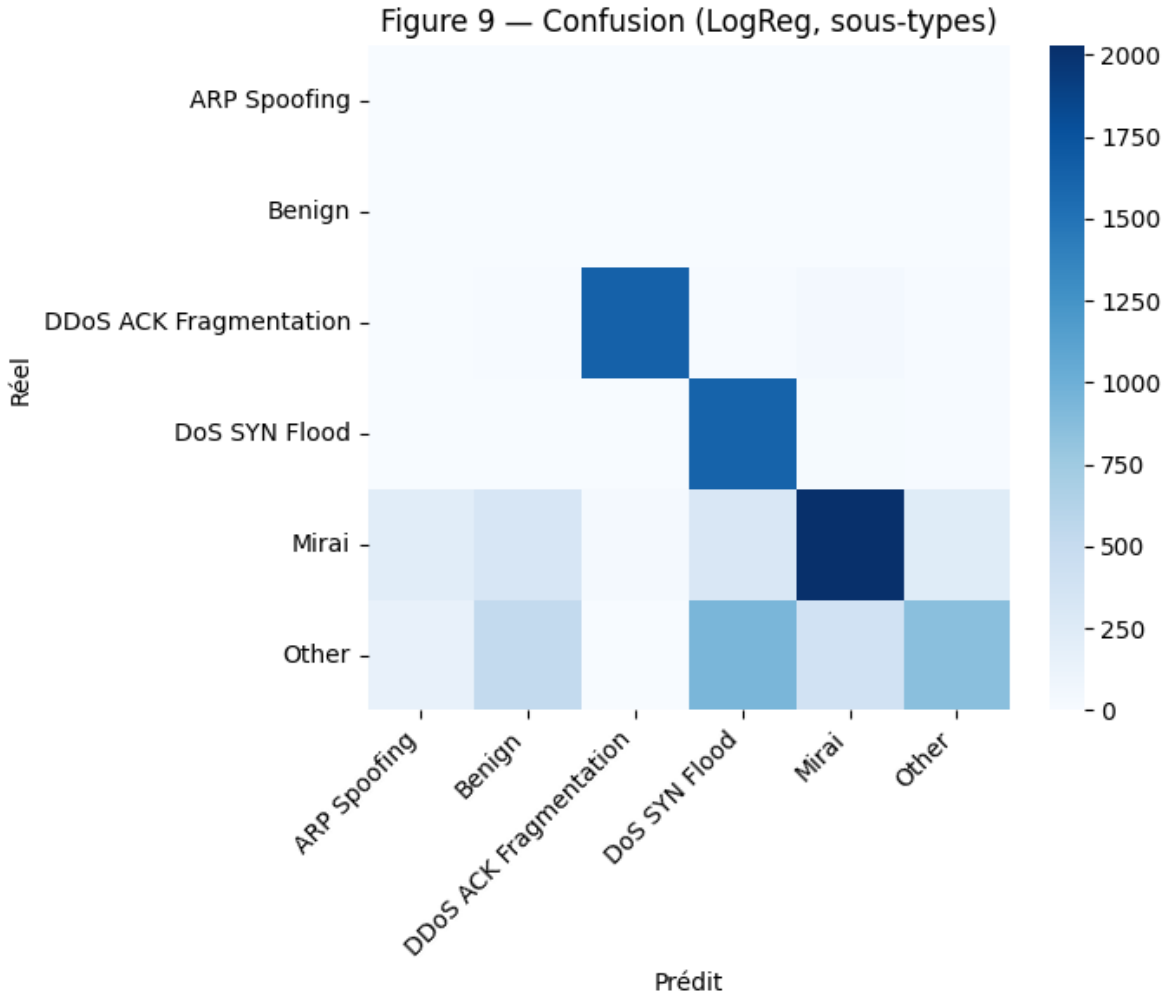


FIGURE 9 – Matrice de confusion (sous-types filtrés) – régression logistique.

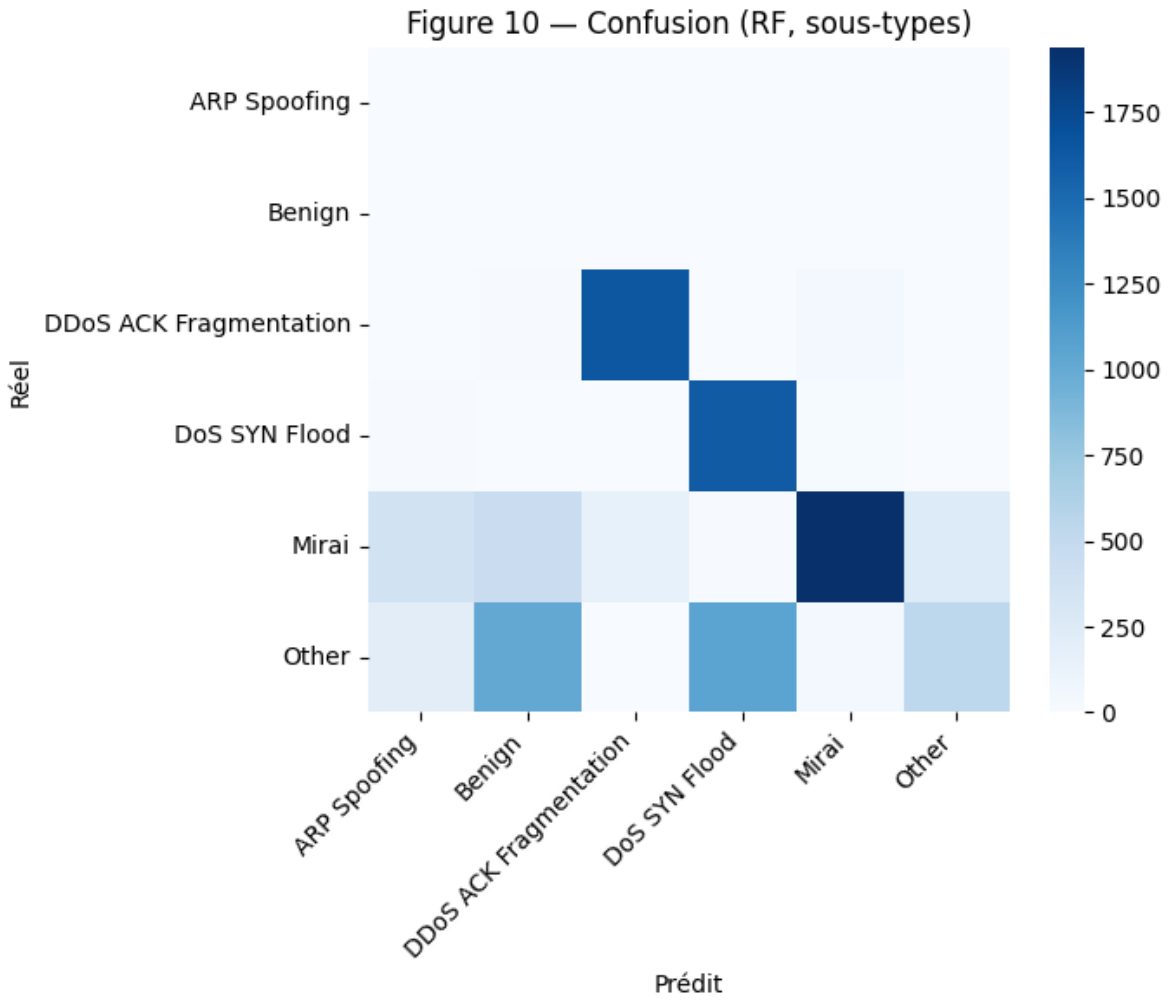


FIGURE 10 – Matrice de confusion (sous-types filtrés) – forêt aléatoire.

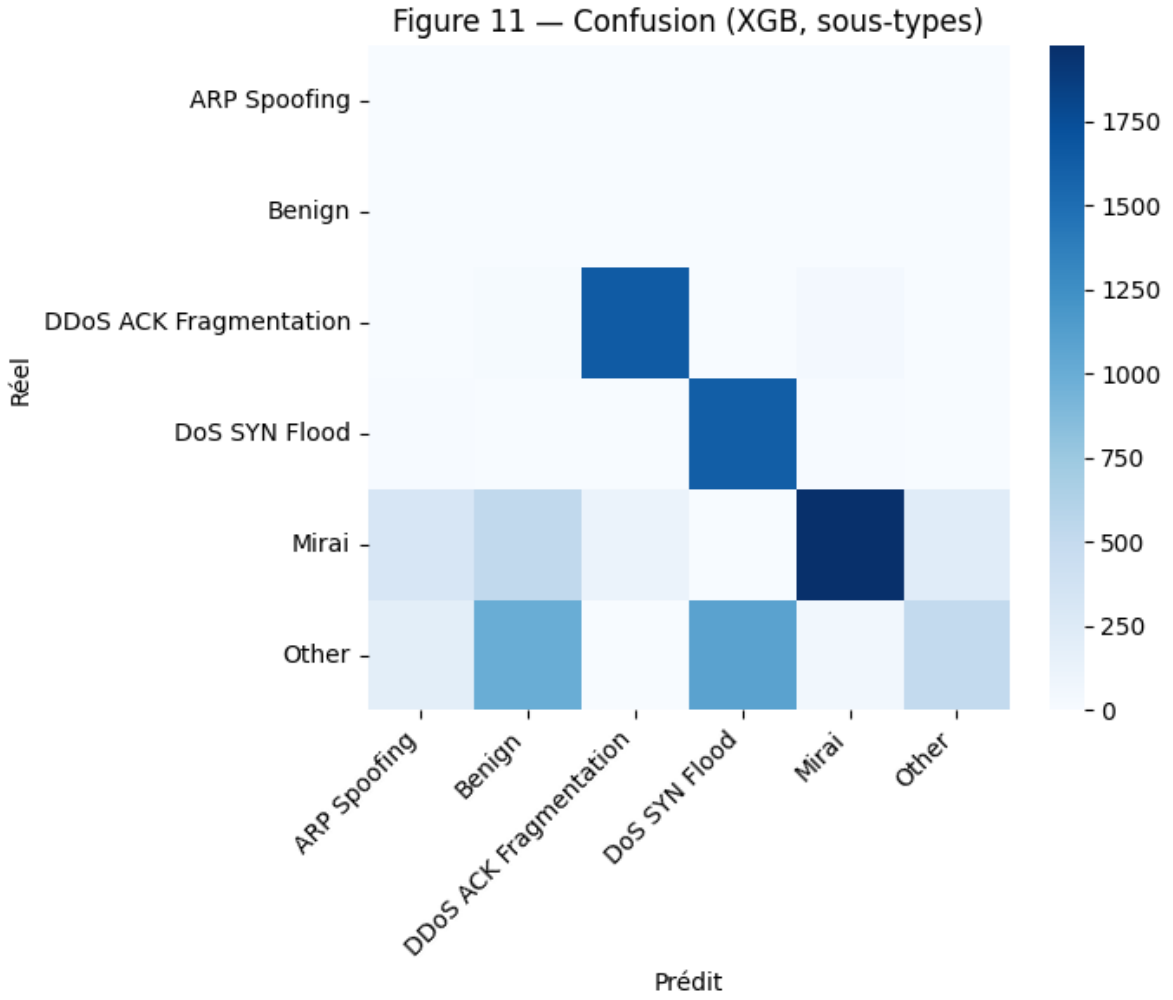


FIGURE 11 – Matrice de confusion (sous-types filtrés) – XGBoost.

TABLE 6 – Synthèse : meilleurs modèles selon la métrique principale de la tâche (issue du notebook).

Tâche	Meilleur modèle	Métrique	Valeur
Binary (Benign vs Attack)	Supervisé - XGBoost (CPU hist)	AUPRC	0.982
Multiclass (Families)	Multiclass - XGBoost (hist)	Balanced Acc	0.783
Multiclass (Subtypes)	Subtype - LogisticRegression	Balanced Acc	0.720

7 Analyse et discussion des résultats

7.1 Détection binaire : compromis FP/FN

Le tableau 3 illustre un point clé SOC : XGBoost a un rappel très élevé (FN=522.000), mais génère beaucoup de faux positifs (FP=6685.000). À seuil fixe (0.5), il peut être difficilement déployable sans mécanisme de triage, de corrélation ou d'ajustement de seuil. À l'inverse, la PCA reconstruction (Tableau 2) combine un bon rappel et un nombre faible de faux positifs, ce qui peut en faire un détecteur de première ligne lorsque les labels sont partiels.

7.2 Multi-classes et effet du group split

Le group split est indispensable pour éviter la fuite ; en contrepartie, il peut conduire à des classes absentes en test lorsque le subset est limité. Dans notre exécution, le test *familles* ne contient que 4 familles (Benign, DDoS, DoS, Mirai), et le test *sous-types* ne couvre qu'un sous-ensemble des classes filtrées. Cela explique en partie l'écart entre métriques macro et pondérées, et limite l'interprétation des performances sur les classes rares. Une extension naturelle consiste à répéter l'expérience sur plusieurs splits (ou une validation croisée *par groupe*) pour stabiliser les métriques.

7.3 Confusions dominantes

Les tableaux 7 et 8 listent les principales confusions (XGBoost pour familles, régression logistique pour sous-types). On observe notamment une confusion Benign→Recon et DDoS↔Mirai au niveau familles, suggérant des profils de flux partiellement similaires sur le subset chargé. Au niveau sous-types, la classe agrégée *Other* est la source principale d'erreurs : elle regroupe des attaques hétérogènes et devient naturellement difficile à séparer des classes spécifiques (DoS SYN Flood, Benign, Mirai).

TABLE 7 – Top confusions (familles) pour le meilleur modèle XGBoost sur le test.

Vrai	Prédit	Count
Benign	Recon	3350
DDoS	Mirai	2744
Benign	Mirai	942
Mirai	Benign	819
Mirai	Spoofing	777
Benign	Spoofing	717
Benign	Web-Based	315
Benign	DoS	164
Mirai	DDoS	160
DDoS	Spoofing	153

TABLE 8 – Top confusions (sous-types filtrés) pour le meilleur modèle (régression logistique) sur le test.

Vrai	Prédit	Count
Other	DoS SYN Flood	942
Other	Benign	526
Other	Mirai	388
Mirai	Benign	331
Mirai	DoS SYN Flood	309
Mirai	Other	245
Mirai	ARP Spoofing	229
Other	ARP Spoofing	153
DDoS ACK Fragmentation	Mirai	43
Mirai	DDoS ACK Fragmentation	35

7.4 Interprétabilité : importance des features (XGBoost binaire)

En cybersécurité, l'interprétabilité aide à valider qu'un modèle exploite des signaux plausibles, et à construire des règles de *sanity-check* (monitoring, détection de dérive). La figure 12 montre les 20 variables les plus importantes (importance *gain*) pour le modèle XGBoost binaire. On observe que des statistiques de longueurs de paquets, d'inter-arrivées (*IAT*) et des compteurs de drapeaux TCP (SYN/RST/CWR) dominent, ce qui est cohérent avec des attaques générant des patterns temporels et des signatures de handshake atypiques.

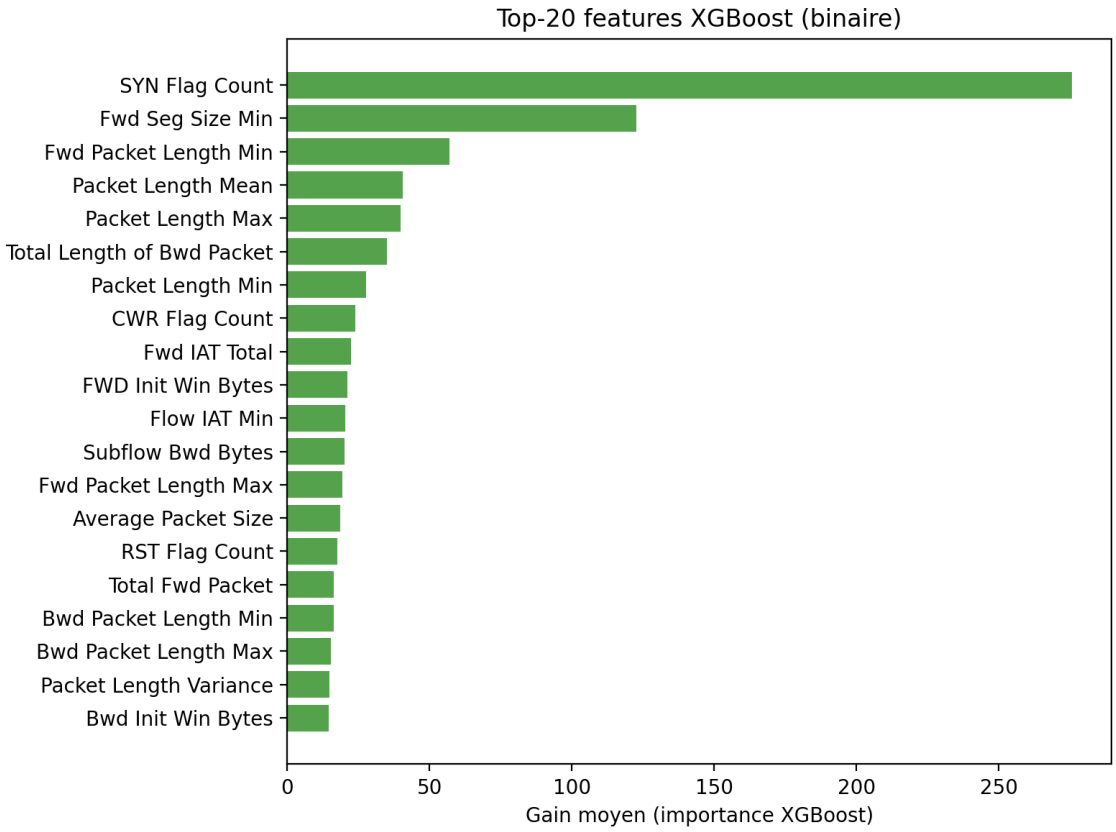


FIGURE 12 – Top-20 features du modèle XGBoost (binaire) selon l'importance *gain*. Cette lecture reste dépendante du subset chargé et de la configuration du modèle.

7.5 Lecture cyber par niveau

Niveau 1 – Détecter (binaire). L'objectif prioritaire est d'éviter des attaques manquées (FN) tout en réduisant l'alert fatigue (FP). Les résultats montrent qu'un modèle *supervisé* très performant en AUPRC peut générer trop d'alertes à seuil fixe, d'où l'importance de la courbe PR (Fig. 5) et d'un calibrage de seuil en production.

Niveau 2A – Tracker haut niveau (familles). La classification familles aide à prioriser et à orienter l'investigation. Sur le subset, XGBoost domine en balanced accuracy (Tableau 4), mais l'évaluation doit être renforcée (split couvrant toutes les classes, répétitions).

Niveau 2B – Tracker fin (sous-types). La granularité fine nécessite davantage de données par sous-type. Le filtrage et la classe *Other* stabilisent l'apprentissage mais introduisent une classe intrinsèquement hétérogène, ce qui se traduit par des confusions concentrées sur *Other* (Tableau 8).

8 Attaques adversariales et impact sur les modèles

8.1 Menace considérée

Nous étudions une attaque d'évasion de type FGSM [6] contre une régression logistique binaire. Dans le notebook, les entrées sont déjà standardisées (centrage-réduction) ; l'attaque applique une perturbation ℓ_∞ de norme ε dans cet espace normalisé. Cette hypothèse simplifie la génération d'exemples adversariaux, mais ne modélise pas directement une modification réaliste des paquets réseau : l'intérêt est de quantifier une *sensibilité* du classifieur.

8.2 Résultats

Le tableau 9 et la figure 13 montrent une dégradation extrêmement rapide de l'accuracy robuste : dès $\varepsilon = 0.100$, elle tombe à 0.172, et devient quasi nulle à partir de $\varepsilon = 0.200$. Ces résultats sont cohérents avec la vulnérabilité connue des modèles linéaires aux perturbations dirigées en grande dimension.

TABLE 9 – Accuracy robuste (binaire) en fonction de ε pour une attaque FGSM sur régression logistique.

ε	Robust accuracy
0.0	0.728
0.1	0.172
0.2	0.003
0.3	0.002
0.4	0.002
0.5	0.002

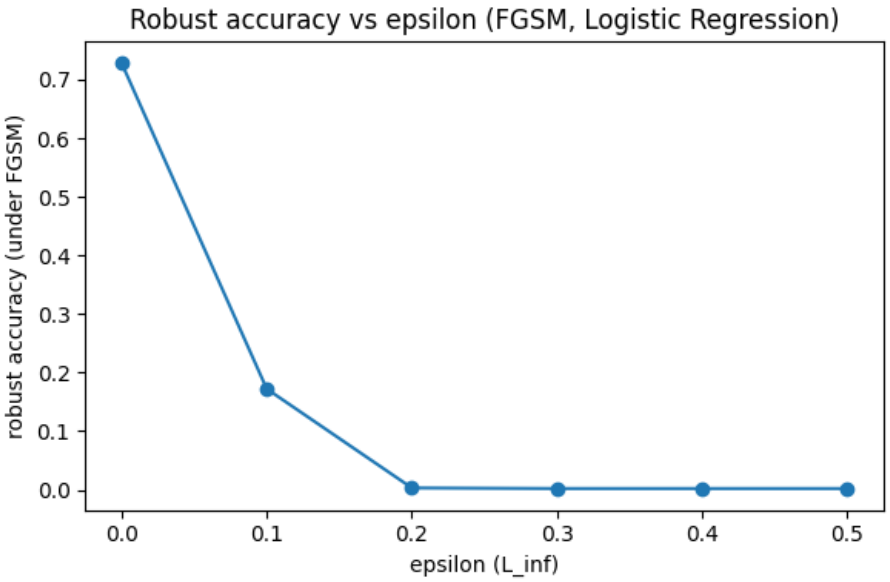


FIGURE 13 – Accuracy robuste vs ε (FGSM, régression logistique).

8.3 Pistes de mitigation

Sans prétendre à une défense complète, plusieurs pistes sont directement actionnables :
— **Adversarial training** sur une plage de ε compatible avec la sémantique des features.

— **Contraintes de validité** sur les features (bornes, invariants) et détection d'entrées hors distribution.
— **Choix de modèles** et ensembles : diversifier les inductive biases (linéaire + arbres) et surveiller la dérive.
— **Seuils adaptatifs** et corrélation multi-sources : réduire l'impact d'un contournement isolé.

9 Conclusion et perspectives cybersécurité

Ce projet met en place une chaîne *batch* complète (ingestion, nettoyage, split anti-fuite, standardisation, entraînement, évaluation) sur CIC IoT-DIAD 2024 (Flow Based Features). En détection binaire, les méthodes non supervisées (notamment PCA reconstruction) offrent un compromis FP/FN très compétitif, tandis que XGBoost maximise l'AUPRC mais nécessite un choix de seuil adapté au SOC. En suivi multi-classes, XGBoost est le meilleur au niveau familles sur le subset considéré, et la régression logistique atteint la meilleure balanced accuracy au niveau sous-types filtrés, avec des erreurs dominées par la classe *Other*. Enfin, l'expérience FGSM souligne l'importance de considérer la robustesse adversariale dès la conception : même un modèle simple peut être fortement contourné si l'attaquant peut perturber les features. En perspectives : (i) répéter l'évaluation sur plusieurs splits *par fichier*, (ii) élargir le subset (budgets) pour couvrir davantage de familles/sous-types en test, et (iii) étendre l'étude adversariale aux modèles d'arbres et au multiclass avec des attaques compatibles tabulaire.

Références

[1] Canadian Institute for Cybersecurity (CIC), "Cic iot-diad 2024 dataset." `https://www.unb.ca/cic/datasets/iot-diad-2024.html`, 2024. Accessed : 2026-01-25.

[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 2008 IEEE International Conference on Data Mining*, pp. 413–422, IEEE, 2008.

[3] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[4] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2 ed., 2002.

[5] T. Chen and C. Guestrin, "Xgboost : A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, 2016.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn : Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.