Rapport de Projet CYBERML

Chaîne de Traitement pour l'Analyse de Données de Cybersécurité

Jeu de Données Analysé : CIC IoT-DIAD 2024

Baptiste Villeneuve
Max Nagaishi
Alexis Petignat
Matthew Banawa

Numéro de Groupe : 9

28 Janvier 2026

Table des matières

1 Introduction **4**

2 Objectifs du Projet **4**

3 Caractérisation du Jeu de Données **5**
3.1 Sélection : CIC IoT-DIAD 2024 5
3.2 Structure des Données 5
3.3 Hiérarchie des Étiquettes 5
3.4 Chaîne de Pré-traitement 5

4 Méthodologie **7**
4.1 Métriques de Performance 7
4.2 Algorithmes Sélectionnés 7
4.2.1 Apprentissage Supervisé 7
4.2.2 Apprentissage Non Supervisé (Détection d'Anomalies) 7

5 Résultats Expérimentaux : Durée 1 Seconde **8**
5.1 Résultats Apprentissage Supervisé 8
5.1.1 XGBoost 8
5.1.2 Régression Logistique 8
5.1.3 Linear SVC 8
5.2 Résultats Apprentissage Non Supervisé 9
5.2.1 Isolation Forest 9
5.2.2 One-Class SVM & LOF 9
5.3 Importance des Caractéristiques (SHAP) 9

6 Résultats Expérimentaux : Durée 3 Secondes **10**
6.1 Résultats Apprentissage Supervisé 10
6.1.1 XGBoost 10
6.1.2 Régression Logistique & SVC 10
6.2 Résultats Apprentissage Non Supervisé 10

7 Résultats Expérimentaux : Durée 5 Secondes **11**
7.1 Résultats Apprentissage Supervisé 11
7.1.1 XGBoost 11
7.1.2 Régression Logistique & SVC 11
7.2 Résultats Apprentissage Non Supervisé 11

8 Résultats Expérimentaux : Durée 7 Secondes **12**
8.1 Résultats Apprentissage Supervisé 12
8.1.1 XGBoost - Performance Maximale 12
8.1.2 Régression Logistique & SVC 12
8.2 Résultats Apprentissage Non Supervisé 12
8.3 Importance des Caractéristiques (7s) 12

9 Analyse des Attaques Multiclasses **13**
9.1 Types d'Attaques Évalués 13
9.2 Performance Multiclasse XGBoost 13
9.3 Analyse de la Matrice de Confusion 13

10 Analyse Comparative et Discussion **14**
10.1 Impact de la Durée de la Fenêtre Temporelle 14
10.2 Supervisé vs. Non Supervisé 14
10.3 Stabilité des Caractéristiques 14

11 Conclusion et Perspectives **15**
11.1 Résumé des Réalisations 15
11.2 Recommandations Opérationnelles 15
11.3 Travaux Futurs : Robustesse Adversariale 15

A Annexe : Extraits de Code **16**
A.1 Chargement des Données 16
A.2 Visualisation SHAP 16

1 Introduction

La prolifération des appareils de l'Internet des Objets (IoT) dans les environnements industriels et grand public a créé une surface d'attaque massive pour les cybercriminels. Contrairement aux infrastructures informatiques traditionnelles, les appareils IoT souffrent souvent d'une puissance de calcul limitée, de mises à jour peu fréquentes et de protocoles hétérogènes, ce qui les rend vulnérables à un large éventail d'attaques réseau telles que le DoS, MitM et les injections de logiciels malveillants.

Le projet **CYBERML** vise à relever ces défis en concevant, déployant et évaluant une chaîne de traitement de données robuste dédiée à l'analyse de la cybersécurité. La philosophie centrale de ce projet est de tirer parti des techniques d'Apprentissage Automatique pour automatiser la détection des activités malveillantes au sein du trafic réseau.

Ce rapport documente l'ensemble du cycle de vie de notre analyse, de l'ingestion et du pré-traitement des données à l'évaluation rigoureuse (benchmarking) des algorithmes de classification et de détection d'anomalies. Nous avons choisi le jeu de données **CIC IoT-DIAD 2024** comme vérité terrain, en l'analysant à travers plusieurs configurations de fenêtres temporelles (1s, 3s, 5s, 7s) pour comprendre la dynamique temporelle de la détection d'attaques.

2 Objectifs du Projet

L'objectif principal est la construction d'une **Chaîne de Traitement par Lots (Batch Processing)**. Les objectifs analytiques spécifiques sont doubles :

- **Objectif 1 : Classification et Détection d'Anomalies.**
  - *Classification Supervisée :* Entraîner des modèles pour distinguer le trafic 'Bénin' du trafic 'Attaque' en utilisant des données étiquetées.
  - *Détection d'Anomalies Non Supervisée :* Déployer des algorithmes capables d'identifier les valeurs aberrantes dans les données sans connaissance préalable des étiquettes d'attaque, simulant un scénario de détection d'attaque "Zero-Day".
  - *Caractérisation du Type d'Attaque :* Aller au-delà de la classification binaire pour identifier les vecteurs d'attaque précis (par ex., DDoS contre Bruteforce).
- **Objectif 2 : Benchmarking et Évaluation.**
  - Comparer les performances des modèles linéaires par rapport aux modèles non linéaires.
  - Évaluer l'impact des fenêtres d'agrégation du trafic (1 seconde contre 7 secondes).
  - Utiliser des métriques robustes incluant le Coefficient de Corrélation de Matthews (MCC), la Précision Équilibrée (Balanced Accuracy), la Précision et le Rappel (Recall).

3 Caractérisation du Jeu de Données

3.1 Sélection : CIC IoT-DIAD 2024

Nous avons sélectionné le jeu de données CIC IoT-DIAD 2024, une référence moderne pour la sécurité IoT. Il comprend un trafic réaliste généré à partir d'un banc d'essai d'appareils IoT subissant divers scénarios d'attaque.

3.2 Structure des Données

Les données sont fournies dans des fichiers CSV, agrégées par durée. Cette "durée" représente la fenêtre temporelle sur laquelle les caractéristiques des paquets réseau ont été calculées (par ex., taille moyenne des paquets sur 1 seconde).

Durée	Échantillons Attaque	Échantillons Bénins	Total
1 Seconde	90 391	136 800	227 191
3 Secondes	29 627	45 600	75 227
5 Secondes	17 695	27 360	45 055
7 Secondes	12 050	19 532	31 582

Table 1: Distribution du jeu de données selon les différentes fenêtres temporelles.

3.3 Hiérarchie des Étiquettes

Le jeu de données contient plusieurs colonnes d'étiquettes utilisées pour différentes granularités d'analyse :

- `label1` : Classification Binaire (Bénin / Attaque).
- `label2` : Catégorie d'Attaque (par ex., DoS, DDoS, Recon, Web, Bruteforce).

3.4 Chaîne de Pré-traitement

Pour préparer les données pour nos modèles d'apprentissage automatique, nous avons mis en œuvre un pipeline de pré-traitement strict en Python :

1. **Chargement :** Les données sont chargées dynamiquement en fonction de la durée cible.
2. **Imputation :** Les valeurs manquantes (`NaN`) sont remplies avec 0. Dans le contexte des caractéristiques réseau (comme "nombre d'erreurs"), une valeur manquante indique généralement l'absence d'un événement.
3. **Encodage :** Les étiquettes catégorielles sont factorisées en entiers. Une logique a été implémentée pour mapper ces entiers vers leurs représentations textuelles d'origine pour les rapports.
4. **Filtrage :** Les colonnes de caractéristiques non numériques sont supprimées pour assurer la compatibilité avec Scikit-Learn et XGBoost.

5. **Séparation :** Une séparation standard 80/20 Entraînement/Test est appliquée avec une graine aléatoire fixe (42) pour la reproductibilité.

4 Méthodologie

4.1 Métriques de Performance

Compte tenu du déséquilibre potentiel des classes dans les jeux de données de cybersécurité, la simple précision (accuracy) est souvent trompeuse. Nous nous appuyons sur les métriques suivantes :

- **Matrice de Confusion :** TP, TN, FP, FN.
- **Précision (Precision) :** $\frac{TP}{TP+FP}$ (Faible taux de fausses alarmes).
- **Rappel (Recall) :** $\frac{TP}{TP+FN}$ (Taux de détection élevé).
- **Précision Équilibrée (Balanced Accuracy) :** Moyenne arithmétique de la Sensibilité et de la Spécificité.
- **Coefficient de Corrélation de Matthews (MCC) :** Un coefficient de corrélation entre les classifications binaires observées et prédites. Il renvoie une valeur entre -1 et +1. Un coefficient de +1 représente une prédiction parfaite, 0 n'est pas meilleur qu'une prédiction aléatoire.

4.2 Algorithmes Sélectionnés

4.2.1 Apprentissage Supervisé

1. **XGBoost (Gradient Boosting) :** Un algorithme d'apprentissage automatique d'ensemble basé sur des arbres de décision qui utilise un cadre de boosting de gradient. C'est l'état de l'art pour les données tabulaires.
2. **Régression Logistique :** Un modèle statistique qui utilise une fonction logistique pour modéliser une variable dépendante binaire. Il sert de référence linéaire.
3. **Linear SVC :** Classification par Vecteurs de Support avec un noyau linéaire. Il tente de maximiser la marge entre les classes.

4.2.2 Apprentissage Non Supervisé (Détection d'Anomalies)

1. **Isolation Forest :** Un algorithme qui isole les observations en sélectionnant aléatoirement une caractéristique puis en sélectionnant aléatoirement une valeur de coupure. Les anomalies sont isolées plus rapidement (chemins plus courts) que les points normaux. Nous avons fixé la contamination ≈ 0.39, ce qui correspond à la proportion de données d'attaque.
2. **One-Class SVM :** Capture la forme des données d'entraînement "Bénines" et classe les valeurs aberrantes. Nous avons utilisé un noyau linéaire et un sous-échantillonnage en raison de l'intensité de calcul.
3. **Local Outlier Factor (LOF) :** Un algorithme qui calcule la déviation de densité locale d'un point donné par rapport à ses voisins.

5 Résultats Expérimentaux : Durée 1 Seconde

Le jeu de données de fenêtre 1 seconde est le plus grand ($N = 227191$), offrant une analyse à haute fréquence mais potentiellement plus de bruit.

5.1 Résultats Apprentissage Supervisé

5.1.1 XGBoost

XGBoost a démontré une performance supérieure, identifiant les attaques avec une haute précision.

Matrice de Confusion :

	Préd Bénin	**Préd Attaque**
Vrai Bénin	27 252 (TN)	55 (FP)
Vrai Attaque	2 405 (FN)	15 727 (TP)

Métriques Détaillées :

- **Classe Bénin :** Précision 0.92, Rappel 1.00, F1 0.96
- **Classe Attaque :** Précision 1.00, Rappel 0.87, F1 0.93
- **Précision Globale :** 0.95
- **Matthews Corrcoef (MCC) :** 0.8900
- **Précision Équilibrée :** 0.9327

5.1.2 Régression Logistique

La Régression Logistique a eu du mal par rapport à XGBoost, en particulier sur le Rappel (sensibilité).

Matrice de Confusion :

	Préd Bénin	**Préd Attaque**
Vrai Bénin	27 163	144
Vrai Attaque	4 903	13 229

Métriques Clés :

- **Rappel Attaque :** 0.73 (Significativement inférieur au 0.87 de XGBoost)
- **MCC :** 0.7783
- **Précision Équilibrée :** 0.8622

5.1.3 Linear SVC

Le classifieur à vecteurs de support a performé un petit peu mieux que la Régression Logistique.

- **MCC :** 0.7950
- **Précision Équilibrée :** 0.8748

5.2 Résultats Apprentissage Non Supervisé

5.2.1 Isolation Forest

Isolation Forest a été le meilleur performeur non supervisé.

- **Matrice de Confusion :** TN: 22604, FP: 4703, FN: 5128, TP: 13004
- **MCC :** 0.5472
- **Analyse :** Bien que capturant une majorité d'attaques, le taux de Faux Positifs (prédisant le trafic bénin comme attaque) est élevé (∼17% du trafic bénin signalé).

5.2.2 One-Class SVM & LOF

Les deux méthodes basées sur la densité/frontière ont eu de mauvaises performances sur les données 1s à haute dimension.

- **One-Class SVM MCC :** 0.4519
- **LOF MCC :** 0.3112

5.3 Importance des Caractéristiques (SHAP)

Les principales caractéristiques pilotant la détection pour la fenêtre 1s étaient :

1. `network_packet-size_min` (Impact : 1.918)
2. `network_time-delta_min` (Impact : 1.442)
3. `network_window-size_min` (Impact : 1.312)

Cela indique que les attaques sont mieux détectées en regardant les tailles minimales de paquets et la vitesse des paquets entrants.

6 Résultats Expérimentaux : Durée 3 Secondes

La taille du jeu de données diminue ($N = 75227$), mais la stabilité des caractéristiques augmente probablement.

6.1 Résultats Apprentissage Supervisé

6.1.1 XGBoost

La performance s'est notablement améliorée sur la fenêtre 3s.

Matrice de Confusion :

	Préd Bénin	**Préd Attaque**
Vrai Bénin	9 121	26
Vrai Attaque	481	5 418

Métriques Clés :

- **Rappel Attaque :** 0.92 (En hausse par rapport à 0.87 pour la fenêtre 1s)
- **MCC :** 0.9303
- **Précision Équilibrée :** 0.9578

6.1.2 Régression Logistique & SVC

Les deux modèles linéaires ont montré une amélioration mais sont restés en retrait par rapport à XGBoost.

- **LogReg MCC :** 0.8083
- **SVC MCC :** 0.8190

6.2 Résultats Apprentissage Non Supervisé

La performance des modèles non supervisés a généralement diminué ou plafonné.

- **Isolation Forest MCC :** 0.5030 (Inférieur à 1s)
- **LOF MCC :** 0.2182 (Très faible)

7 Résultats Expérimentaux : Durée 5 Secondes

7.1 Résultats Apprentissage Supervisé

La tendance à l'amélioration de la performance supervisée se poursuit.

7.1.1 XGBoost

Matrice de Confusion :

	Préd Bénin	**Préd Attaque**
Vrai Bénin	5 391	20
Vrai Attaque	202	3 398

Métriques Clés :

- **Précision Globale :** 0.98
- **MCC :** 0.9491
- **Rappel Attaque :** 0.94

7.1.2 Régression Logistique & SVC

Les deux ont atteint un MCC d'$\approx$ 0.82, suggérant un "plafond" dans ce jeu de données.

7.2 Résultats Apprentissage Non Supervisé

- **Isolation Forest MCC :** 0.4610
- **One-Class SVM MCC :** 0.2925

8 Résultats Expérimentaux : Durée 7 Secondes

8.1 Résultats Apprentissage Supervisé

8.1.1 XGBoost - Performance Maximale

À 7 secondes, XGBoost atteint ses meilleurs résultats.

Matrice de Confusion :

	Préd Bénin	**Préd Attaque**
Vrai Bénin	3 868	14
Vrai Attaque	114	2 321

Métriques Détaillées :

- **Précision (Attaque) :** 0.99
- **Rappel (Attaque) :** 0.95
- **Précision Équilibrée :** 0.9748
- **MCC :** 0.9574

8.1.2 Régression Logistique & SVC

Stables à un MCC $\approx$ 0.81, confirmant que l'extension de la durée profite davantage aux modèles non linéaires complexes (XGBoost) qu'aux modèles linéaires.

8.2 Résultats Apprentissage Non Supervisé

Isolation Forest reste constant autour d'un MCC de 0.46. Il semble que pour la détection d'anomalies, les caractéristiques sur 7s ne rend pas les valeurs aberrantes plus distinctes d'une manière qu'Isolation Forest puisse exploiter mieux qu'à 1s.

8.3 Importance des Caractéristiques (7s)

Changement intéressant dans les caractéristiques par rapport à 1s :

1. `network_packet-size_min` (Impact : 2.24)
2. `network_time-delta_min` (Impact : 1.79)
3. `network_packets_dst_count` (Impact : 1.24)

Le nombre de paquets de destination devient une des 3 meilleures caractéristiques à 7s, probablement parce que les anomalies basées sur le volume (comme les DoS) sont plus statistiquement significatives sur une fenêtre plus longue.

9 Analyse des Attaques Multiclasses

Au-delà de la détection binaire, nous avons évalué la capacité à classer des types d'attaques spécifiques. Nous avons utilisé les résultats de la fenêtre de 7 secondes pour cette analyse approfondie.

9.1 Types d'Attaques Évalués

Le jeu de données comprend : *Malware, Recon, DoS, Web, DDoS, Bruteforce, MitM.*

9.2 Performance Multiclasse XGBoost

Le modèle a atteint un score F1 moyen pondéré de **0.97**.

Scores F1 par Classe :

Type d'Attaque	Rappel	Score F1
Bénin	1.00	0.98
Malware	0.96	0.97
Recon	0.92	0.94
DoS	0.96	0.97
Web	0.81	0.90
DDoS	0.90	0.94
Bruteforce	0.94	0.97
MitM	0.96	0.96

Table 2: Performance de Classification Multiclasse (Fenêtre 7s)

9.3 Analyse de la Matrice de Confusion

La matrice de confusion révèle des erreurs de classification spécifiques :

- **Reconnaissance :** Souvent confondu avec le trafic bénin ou d'autres attaques (Rappel de 0.92, inférieur aux DoS). C'est attendu car les scans de reconnaissance peuvent ressembler à des tentatives de connexion normales.
- **Attaques Web :** Ont eu le rappel le plus faible (0.81). Les attaques Web ressemblent souvent à du trafic HTTP valide, ce qui les rend les plus difficiles à distinguer sur la base de simples statistiques réseau sans inspection approfondie des paquets.
- **DDoS vs DoS :** Très haute précision pour distinguer ces deux types, probablement grâce aux caractéristiques `network_macs_all_count` ou `ips_src_count` qui sont distinctes (Distribué vs Source Unique).

10 Analyse Comparative et Discussion

10.1 Impact de la Durée de la Fenêtre Temporelle

L'une des principales découvertes de cette étude est la corrélation positive entre la durée de la fenêtre temporelle et la performance de la classification pour les modèles supervisés.

Méthode	1s	3s	5s	7s
XGBoost	0.89	0.93	0.95	0.96
LogReg	0.78	0.81	0.82	0.82
IsoForest	0.55	0.50	0.46	0.47

Figure 1: Tableau de Comparaison du MCC selon les Durées

Interprétation :

- **Supervisé (XGBoost) :** Des fenêtres plus longues permettent aux caractéristiques statistiques (moyennes, écarts-types) de se stabiliser, formant des signatures plus claires pour les attaques. La variance du bruit dans les fenêtres de 1s crée plus de "zones grises" pour les classifieurs.
- **Non Supervisé (IsoForest) :** La performance s'est en fait légèrement dégradée avec des fenêtres plus longues. Nous faisons l'hypothèse que de courtes rafales d'anomalies sont distinctes dans les fenêtres de 1s, mais sont "noyées dans la moyenne" et cachées à l'intérieur du trafic valide sur 7s, les rendant plus difficiles à isoler comme valeurs aberrantes.

10.2 Supervisé vs. Non Supervisé

Les méthodes non supervisées ont significativement sous-performé par rapport aux méthodes supervisées ($MCC \approx 0.5$ vs $MCC > 0.9$).

- **Implication :** Bien qu'Isolation Forest puisse détecter $\approx 70 - 80\%$ des attaques, le taux de fausses alarmes est trop élevé pour un déploiement autonome dans une infrastructure critique. Cela nécessite une validation humaine ou un filtre secondaire.
- **Usage :** Les méthodes non supervisées peuvent rester précieuses pour des détection sur l'échelle 1s, ou comme solution de repli pour détecter de nouvelles attaques que le modèle supervisé (entraîné sur des attaques connues) pourrait manquer.

10.3 Stabilité des Caractéristiques

À travers toutes les durées, `network_time-delta_min` et `network_packet-size_min` sont restées les meilleures caractéristiques. Cela suggère que la *cadence* et la *taille de la charge utile* des paquets sont les empreintes invariantes de ces attaques IoT, quelle que soit la fenêtre d'agrégation.

11 Conclusion et Perspectives

Le projet **CYBERML** a réussi à déployer une chaîne de données pour analyser le jeu de données CIC IoT-DIAD 2024.

11.1 Résumé des Réalisations

1. **Déploiement de la Chaîne de Données :** Un pipeline Python entièrement fonctionnel a été construit pour ingérer, traiter et classer le trafic réseau IoT.
2. **Classification Haute Performance :** Nous avons atteint une Précision Équilibrée maximale de **97.5%** en utilisant XGBoost sur les données de fenêtre 7 secondes.
3. **Caractérisation des Attaques :** Nous avons différencié avec succès 7 types d'attaques distincts, avec des scores F1 dépassant 0.90 pour la plupart des catégories.
4. **Benchmarking :** Nous avons établi que le Gradient Boosting (non linéaire) surpasse largement les références linéaires (Régression Logistique, SVC) et que des fenêtres temporelles plus longues aident généralement la classification supervisée.

11.2 Recommandations Opérationnelles

Pour un déploiement réel dans un environnement IoT Industriel (IIoT) :

- Déployer des modèles **XGBoost** entraînés sur des fenêtres de 5s ou 7s pour la détection primaire.
- Exécuter **Isolation Forest** en parallèle sur des fenêtres de 1s pour capturer les anomalies soudaines et courtes qui pourraient être moyennées dans des fenêtres plus longues.
- Concentrer l'ingénierie des caractéristiques sur les **Deltas Temporels** et les **Tailles de Paquets**, car ce sont les signaux les plus importants.

11.3 Travaux Futurs : Robustesse Adversariale

Bien que nos modèles fonctionnent bien sur des jeux de données statiques, la prochaine étape (décrite dans l'Objectif 2) est d'évaluer les **Attaques Adverses**. Des attaquants intelligents pourraient manipuler le timing des paquets (modifiant le `time-delta`) ou faire du rembourrage de paquets (modifiant la `packet-size`) pour échapper à nos meilleures caractéristiques. Développer des modèles robustes à de telles perturbations — peut-être en les entraînant sur des exemples générés de manière adversariale — est la prochaine frontière critique pour ce projet.

A Annexe : Extraits de Code

A.1 Chargement des Données

Listing 1: Fonction de Chargement des Données

```
def load_and_merge_data(duration_sec):
    print(f"\n\nProcessing duration: {duration_sec} s")
    attack_file = f"attack_samples_{duration_sec}sec.csv"
    benign_file = f"benign_samples_{duration_sec}sec.csv"

    attack_data = pd.read_csv(DATA_FOLDER + ATTACK_FOLDER + attack_file)
    benign_data = pd.read_csv(DATA_FOLDER + BENIGN_FOLDER + benign_file)

    combined_data = pd.concat([attack_data, benign_data])
    return combined_data.sample(frac=1).reset_index(drop=True)
```

A.2 Visualisation SHAP

Listing 2: Traçage SHAP

```
def plot_shap(model, X_train):
    explainer = shap.TreeExplainer(model)
    explanation = explainer(X_train)
    shap.plots.beeswarm(explanation)
```