



Université Cadi Ayyad
Faculté des Sciences et Techniques
Marrakech
Master SDAD
Module DataMining



كلية العلوم
والتقنيات - مراكش
FACULTE DES SCIENCES
ET TECHNIQUES - MARRAKECH

DÉTECTION DES INTRUSIONS DANS LE DOMAINE DE LA CYBERSÉCURITÉ APPROCHE AVEC MACHINE LEARNING

Réalisé par :

ASAKOUR Ihsane
ELMOURABIT Zohair
NAIM Kawtar

Encadré par :

M, K. BOUZAACHANE

Plan :

1

Introduction

2

Les Travaux Précédents

3

Dataset

4

Les Approches Proposées

5

Résultats et Discussions

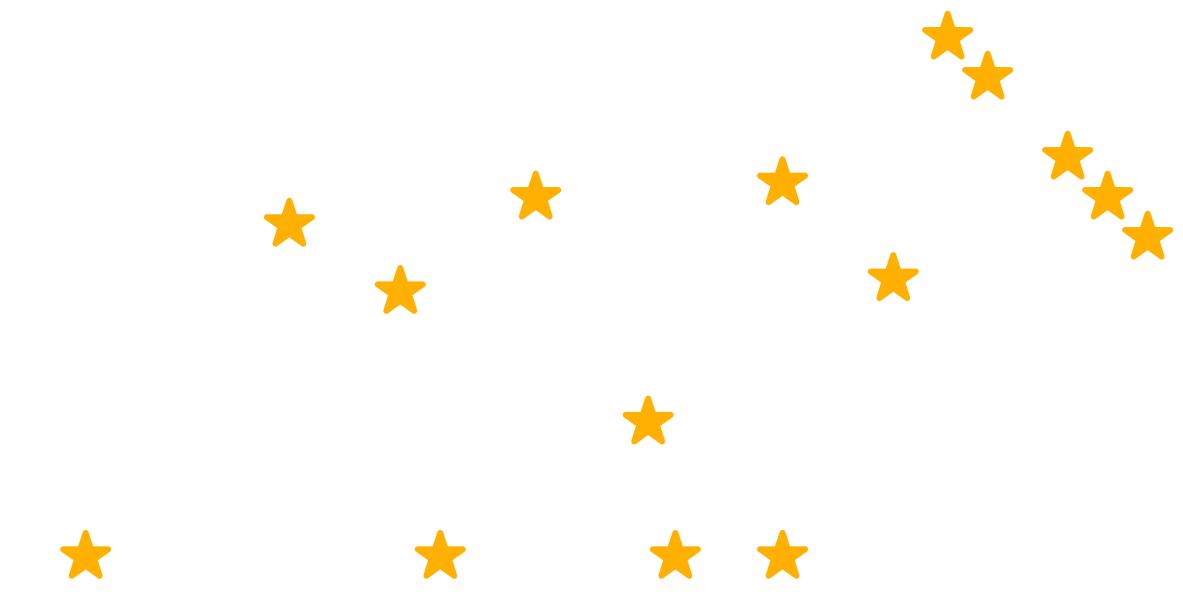
6

Conclusion et perspective

Introduction



Les travaux Précédents



Les Travaux Précédents

1

2

Partie1

Partie2

Partie1

Article	Dataset	Algorithmes	Model performant	Exactitude
MOUATASSIM, T, Détection des intrusions dans le domaine de la Cybersécurité. Approche avec Machine Learning.	NF-UQ - NIDS-v2	<ul style="list-style-type: none">• Adaboost• Random Forest• Decision Tree• XGboost	XGBoost	99,98%
Belavagi, M. C., & Muniyal, B. (2016). Performance evaluation of supervised machine learning algorithms for intrusion detection.	NSL-KDD	<ul style="list-style-type: none">• SVM• GMM• Random forest• logistics regression	Random Forest	99,00%

Les Travaux Précédents

1

2

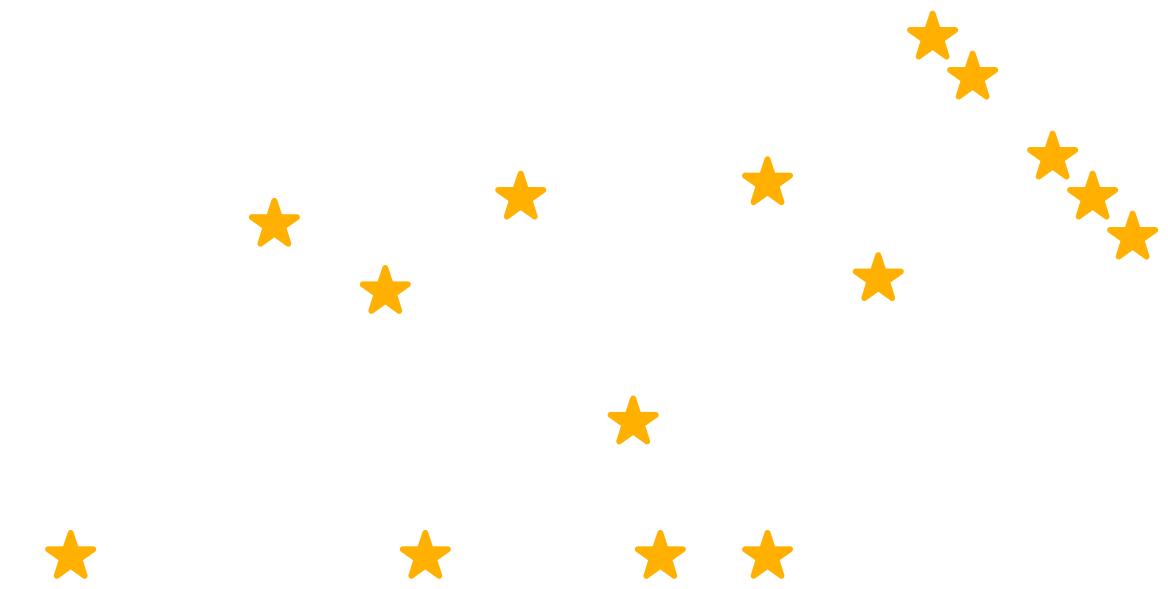
Partie1

Partie2

Partie2

Article	Dataset	Algorithmes	Model perform ant	Exactitude
El Mourabit, Y., Bouirden, A., Toumanari, A., & Moussaid, N. E. (2015). Intrusion detection techniques in wireless sensor network using data mining algorithms: comparative evaluation based on attacks detection. International Journal of Advanced Computer Science and Application	KDD'99	<ul style="list-style-type: none">• k-means• Random Forest• Naïve Bayes• SVM	Random Forest	99.0%
Mohammed M. Alani, Ali Miri, Towards an Explainable Universal Feature Set for IoT Intrusion Detection	TON_IoT	<ul style="list-style-type: none">• Random Forest• Logistic Regression• Decision Tree• Gaussian Naïve-Bayes	Random Forest	99,62%

Dataset



Dataset

1

L'origine

2

Description

l'origine : NF-UQ -NIDS-v2

Le Netflow University of Queensland Network Intrusion Detection System (NF-UQNIDS-v2) est un jeu de données utilisé pour la recherche et le développement de systèmes de détection d'intrusion (IDS). Il se compose de données de trafic réseau collectées à l'Université du Queensland en Australie et est conçu pour simuler un environnement réseau réel.

Le dataset comprend une variété de fonctionnalités, telles que la taille des paquets, le protocole et le port de destination, ainsi qu'une étiquette indiquant si chaque instance est un trafic normal ou d'attaque



On a réutilisé le dataset utilisé dans le projet de fin d'études de Tariq MOUATASSIM

Dataset

1

2

L'origine

Description

Description

	L4_SRC_PORT	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	IN_PKTS	OUT_BYTES	OUT_PKTS	TCP_FLAGS	CLIENT_TCP_FLAGS	...
0	0.582666	0.006790	0.023529	0.172269	0.000048	0.000264	0.000002	0.000246	0.139013	0.121076	...
1	0.583490	0.006790	0.023529	0.172269	0.000024	0.000122	0.000001	0.000123	0.139013	0.121076	...
2	0.672023	0.006790	0.023529	0.172269	0.000024	0.000122	0.000001	0.000123	0.139013	0.121076	...
3	0.523156	0.006790	0.023529	0.172269	0.000024	0.000122	0.000001	0.000123	0.139013	0.121076	...
4	0.578912	0.006790	0.023529	0.172269	0.000024	0.000122	0.000001	0.000123	0.139013	0.121076	...
...
81946	0.643442	0.001221	0.023529	0.028226	0.001525	0.000541	0.000018	0.000221	0.102804	0.030303	...
81947	0.675822	0.001221	0.023529	0.028226	0.001525	0.000541	0.000018	0.000221	0.102804	0.030303	...
81948	0.642741	0.001221	0.023529	0.028226	0.001525	0.000541	0.000018	0.000221	0.102804	0.030303	...
81949	0.624186	0.001221	0.023529	0.028226	0.001525	0.000541	0.000018	0.000221	0.102804	0.030303	...
81950	0.907301	0.768536	0.023529	0.000000	0.000193	0.000000	0.000016	0.000221	0.102804	0.010101	...

81951 rows × 44 columns

81951 observations, 44 caractéristiques

Label	Attack_Category	Attack
1	RANSOMWARE	ransomware
...
1	RECON	Reconnaissance

3 étiquettes

Les Approches Proposées



Les Approches Proposées

1

2

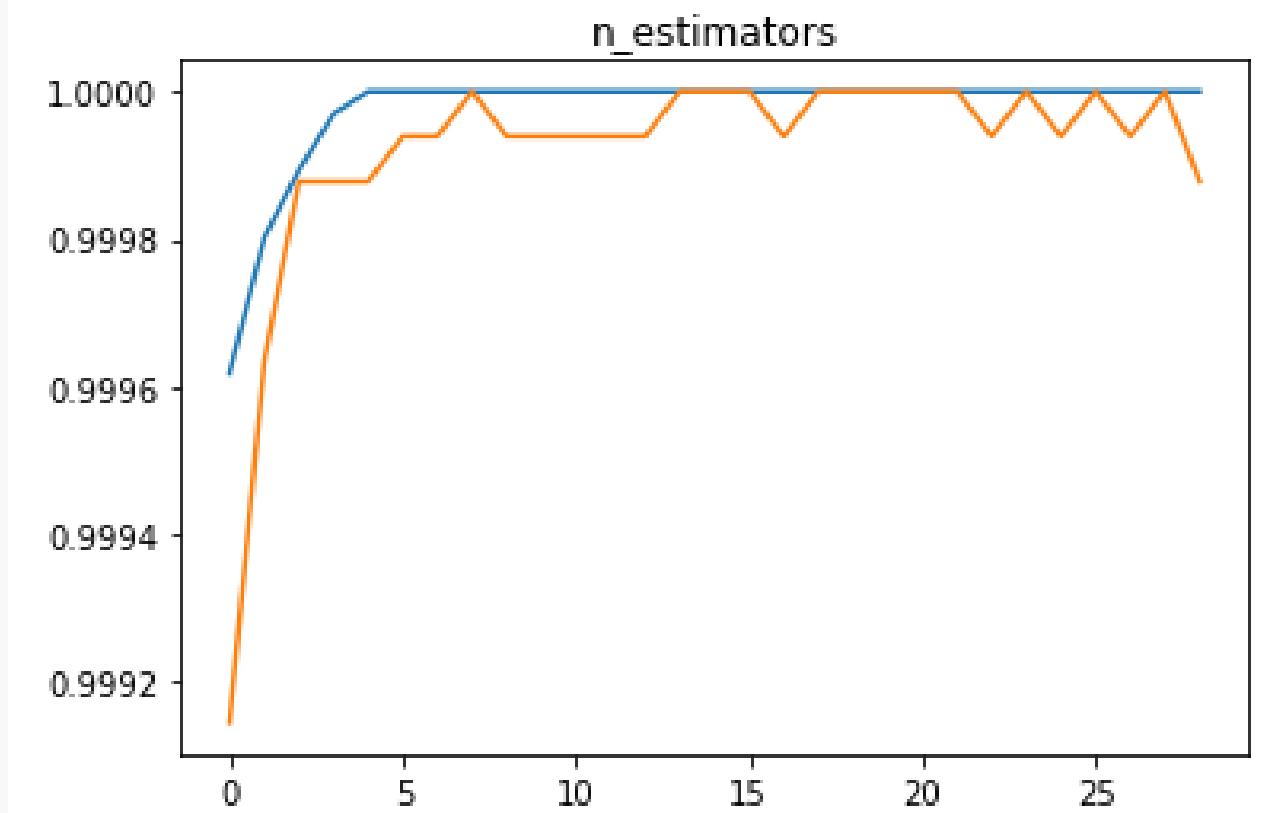
Approche 1

Approche 2

Hyperparameters Tuning / L'estimation des hyperparamètres

L'estimation des hyperparamètres d'un modèle de machine Learning consiste à trouver les valeurs optimales de ces hyperparamètres afin que le modèle soit le plus performant possible sur un jeu de données donné.

```
val_score_train = []
val_score_test = []
k_range = range(1,30)
for k in k_range:
    model = RandomForestClassifier(n_estimators=k)
    model.fit(X_train, y_train)
    score_train=model.score(X_train,y_train)
    score_test=model.score(X_test, y_test)
    print(k," score train : ",score_train," score test : ",score_test)
    val_score_train.append(score_train)
    val_score_test.append(score_test)
```



L'estimation d'hyperparametre "n_estimators" du modèle Random Forest

Les Approches Proposées

1

2

Approche 1

Approche 2

Training Loop / Boucle d'Apprentissage

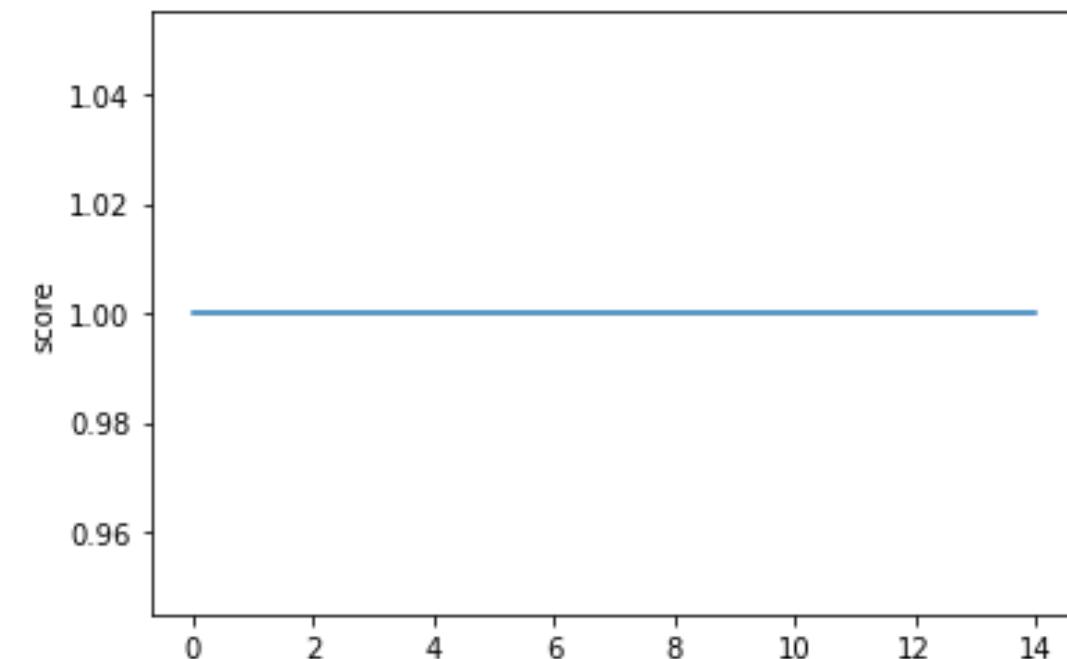
Le training loop est une étape cruciale dans la création d'un modèle de machine Learning. C'est la boucle qui permet de former le modèle en utilisant un ensemble de données d'entraînement.

Le principe de training loop est de mettre le modèle entraîné sur l'ensemble d'entraînement plusieurs fois afin de bien prédit sur l'ensemble de test.

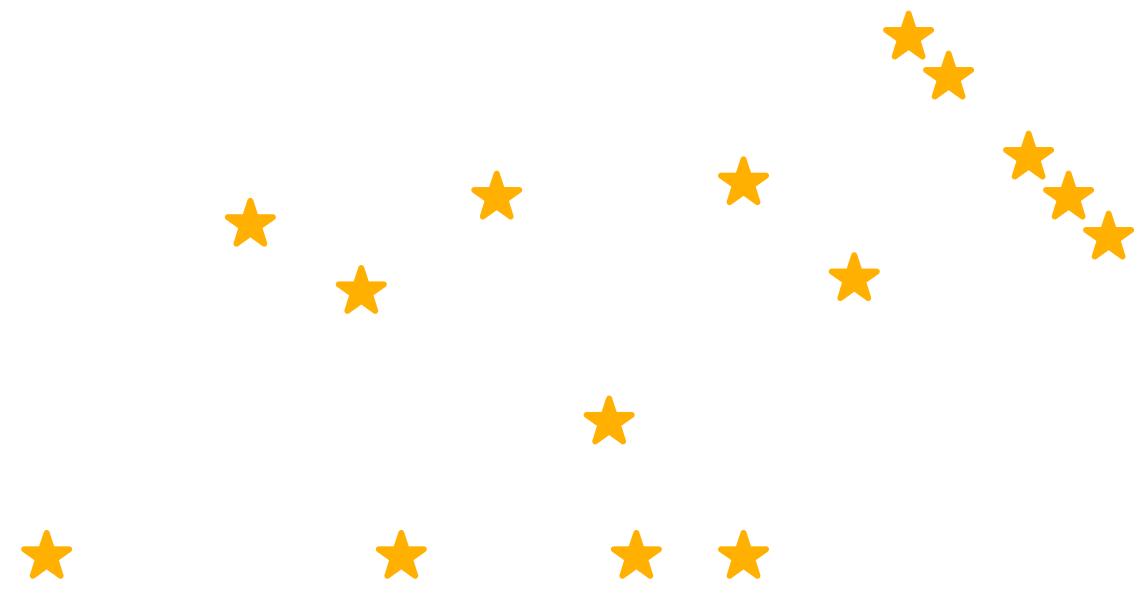
- **Epoch** : (ou époque en français) est une itération complète sur l'ensemble des données d'entraînement dans un modèle de machine learning.
- **Batch** : (ou lot en français) est un sous-ensemble des données d'entraînement utilisé pour mettre à jour les poids du modèle.

```
val_score=[]
model = RandomForestClassifier(n_estimators=11)
for i in range(1,16):
    data_train=shuffle(data_train)
    X=data_train.iloc[:,41]
    y=data_train.iloc[:,41]
    model.fit(X,y)
    score=model.score(X,y)
    print('EPOCH ',i,' \t ACCURACY= ',score)
    val_score.append(score)
```

Exemple de training loop sur le modèle RF



Résultats et Discussions



Résultats

1

2

Résultats

Matrice de confusion

Performance des méthodes machine learning sur la classification binaire

	Random Forest	Gradient Boosting	Logistique regression	Xgboost	Decision Tree	Adaboost	SVM
Epoch	15	10	1	1	20	20	10
Hyperparametre	N_estimators = 11	N_estimators = 100	solver='saga' max_iter =1000	n_estimators = 240		n_estimators = 100	C = 52
Temp entrainement	12.539312601089478	296.1378493309021	4.167687177658081	12.571461915969849	1496.9018936157227	237.46993827819824	1444.2798902988434
Temp prédiction	0.03103041648864746	0.08055949211120605	0.010747194290161133	0.045426130294799805	0.01096963882446289	0.5714704990386963	6.02584433555603
Précision	1.0	0.9999389983579534	0.858582281143786	0.9999389983579534	0.9999389983028945	0.9999593297497833	0.9985360497579499
Score Train	1.0	0.9999694935936546	0.8577333740085418	1.0	1.0	1.0	0.9984746796827334
Exactitude	1.0	0.9999389909096456	0.8585199194680008	0.9999389909096456	0.9999389909096456	0.9999593264459449	0.9985357818314929
Erreur score	0.0	6.1009090354446904e-05	0.14148008053199923	6.1009090354446904e-05	6.1009090354446904e-05	4.067355405512618e-05	0.0014642181685070588
F1 - SCORE	1.0	0.9999389909119165	0.858514510003111	0.9999389909119165	0.999938990884211	0.9999593264435898	0.9985357819949923
Recall	1.0	0.9999389909096456	0.8585199194680008	0.9999389909096456	0.9999389909096456	0.9999593264459449	0.9985357818314929

Résultats

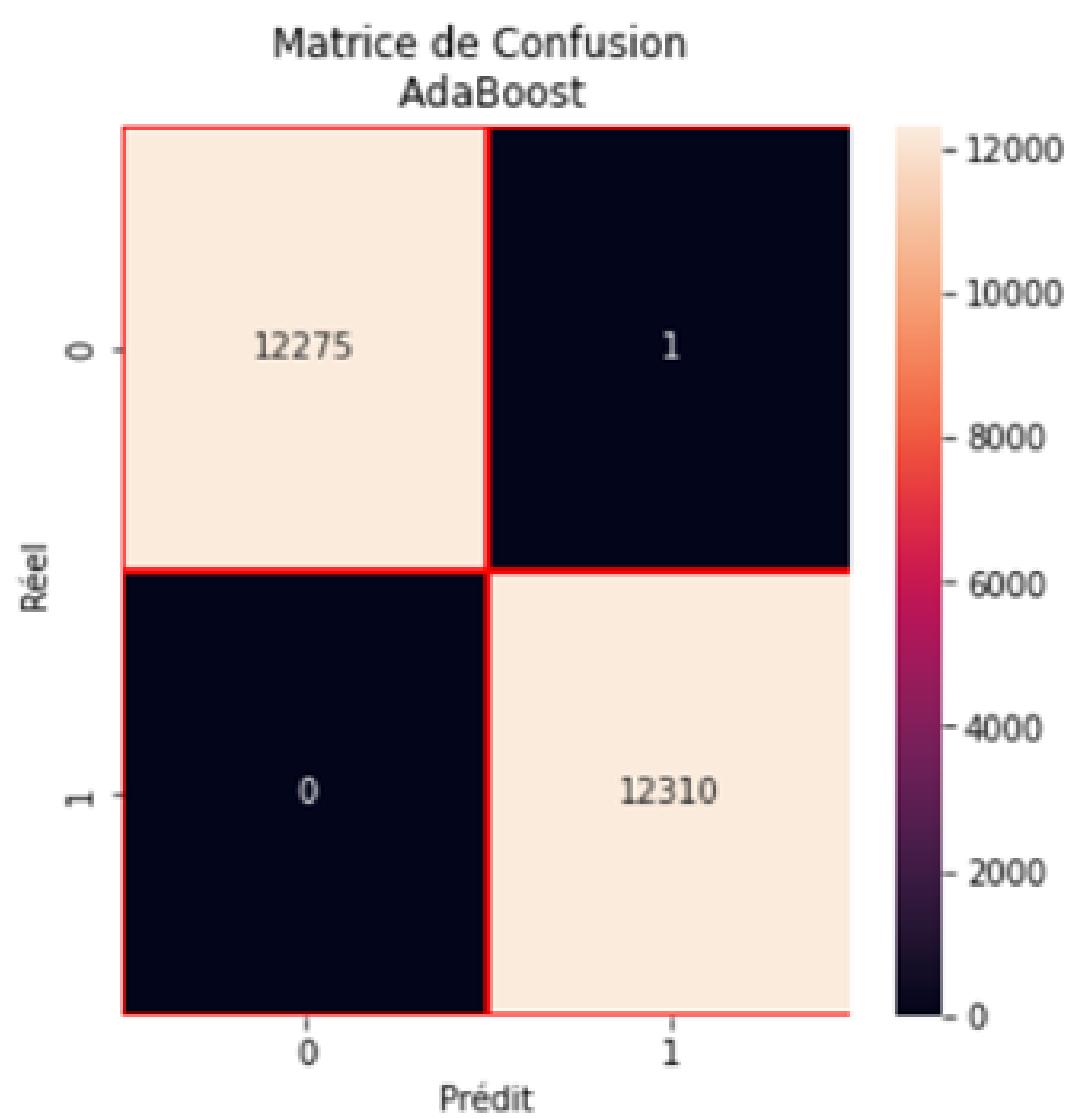
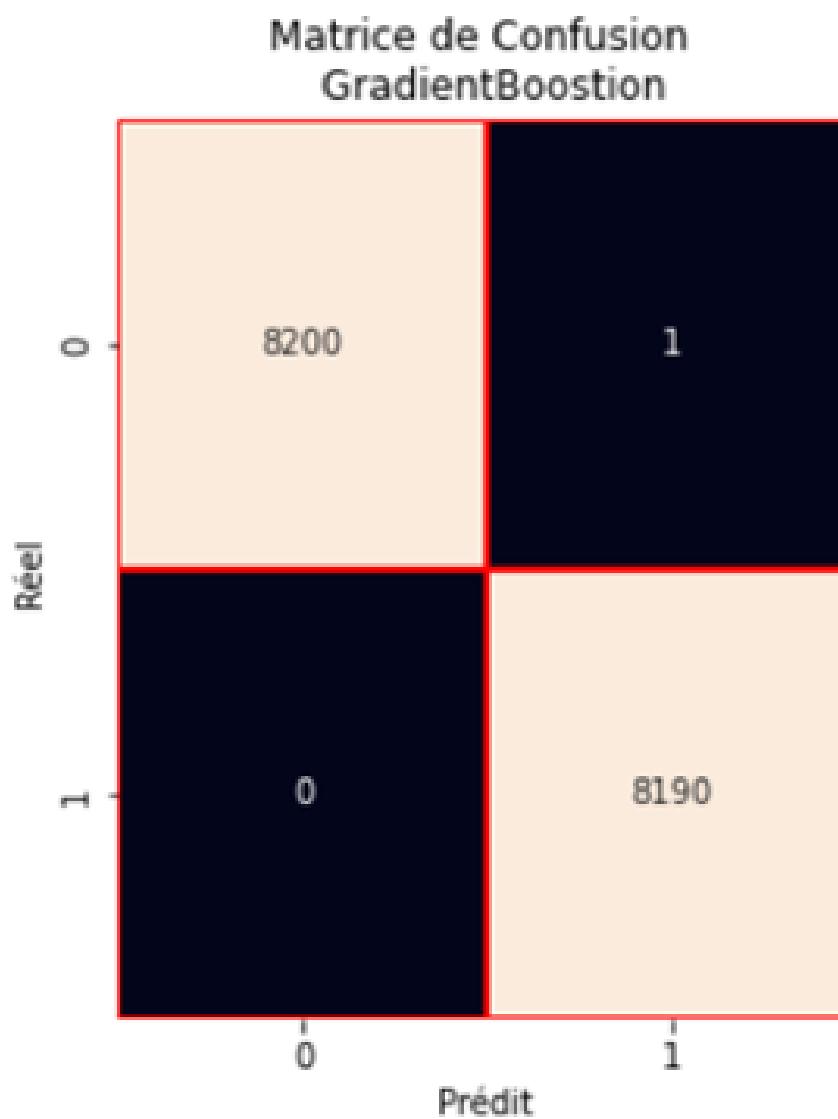
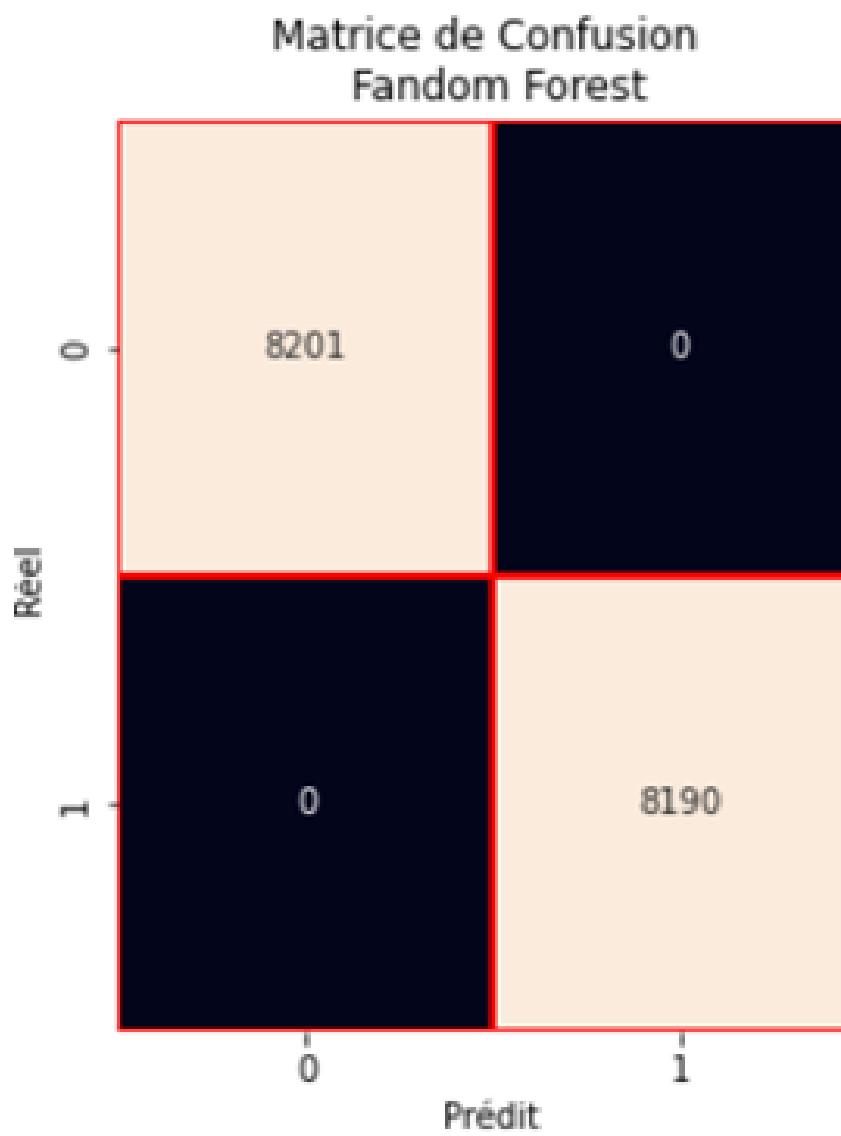
Performance des méthodes machine learning sur la classification binaire

1

2

Résultats

Matrice de Confusion



Résultats

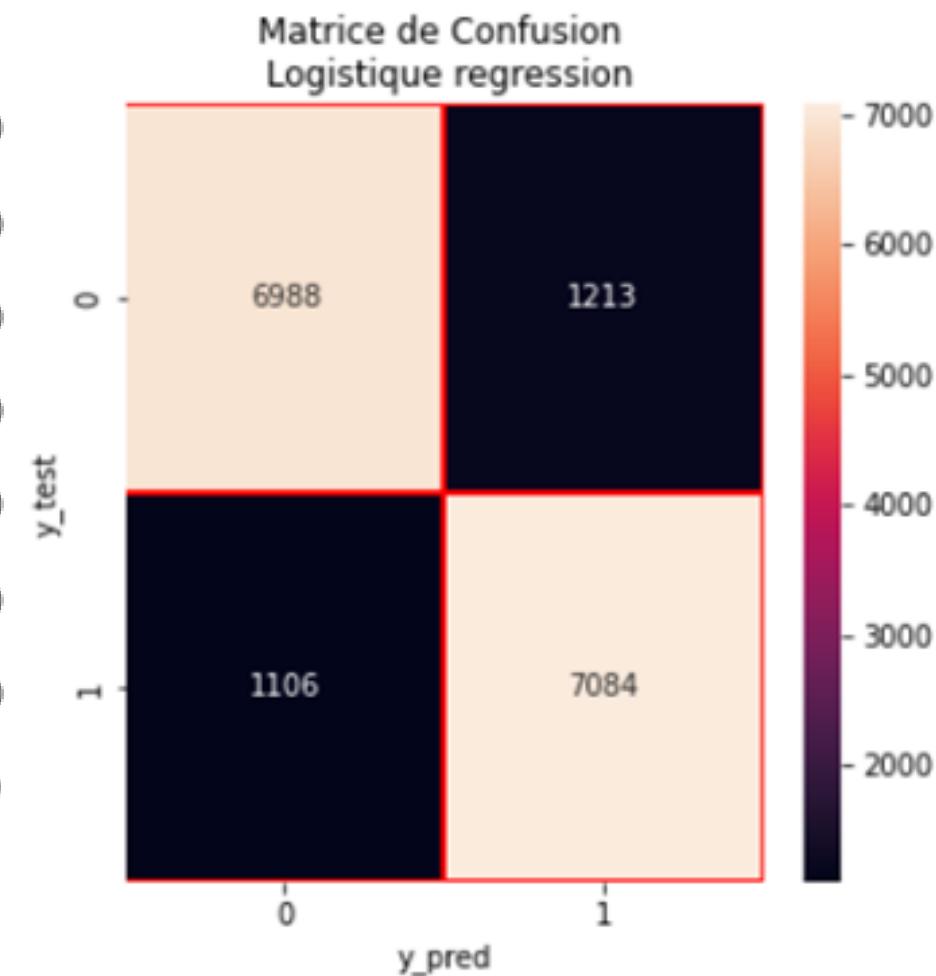
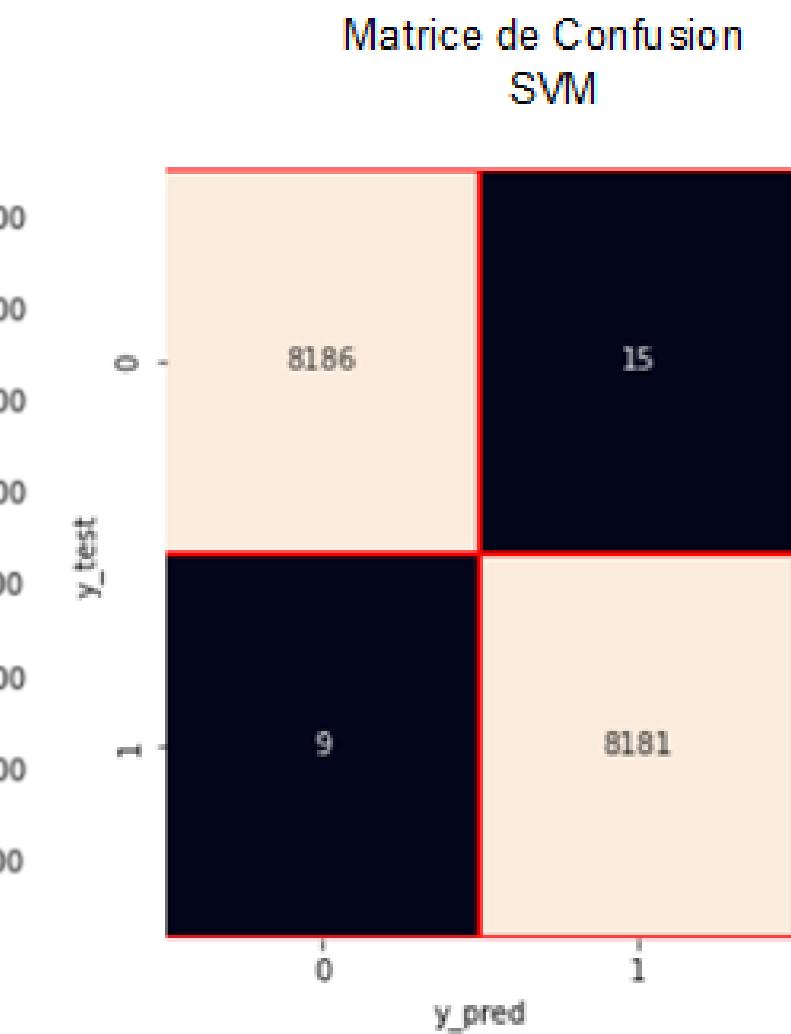
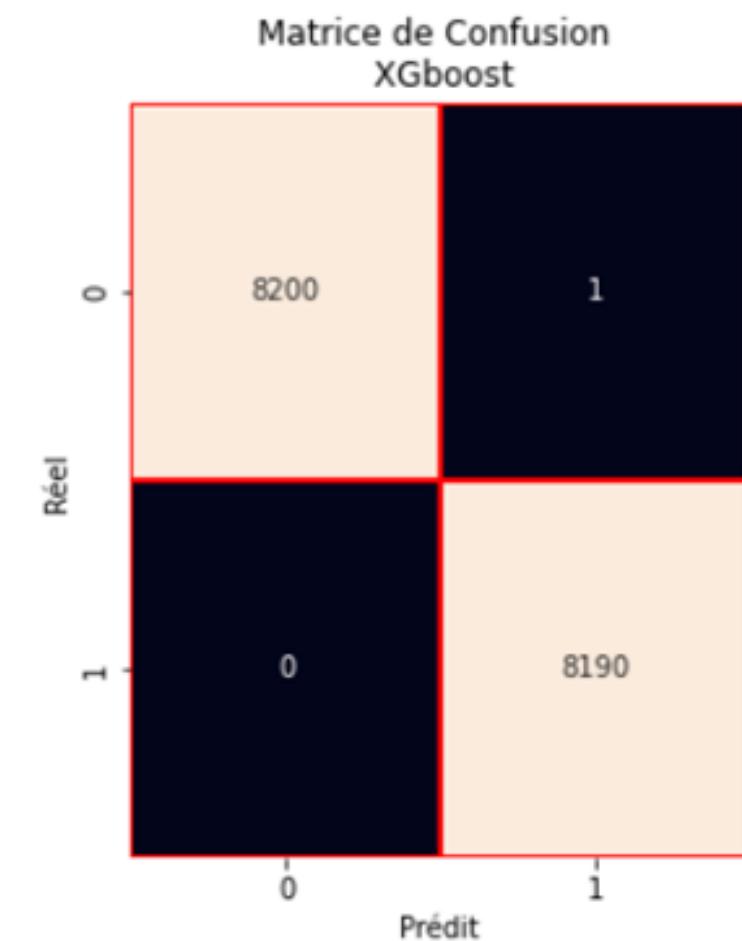
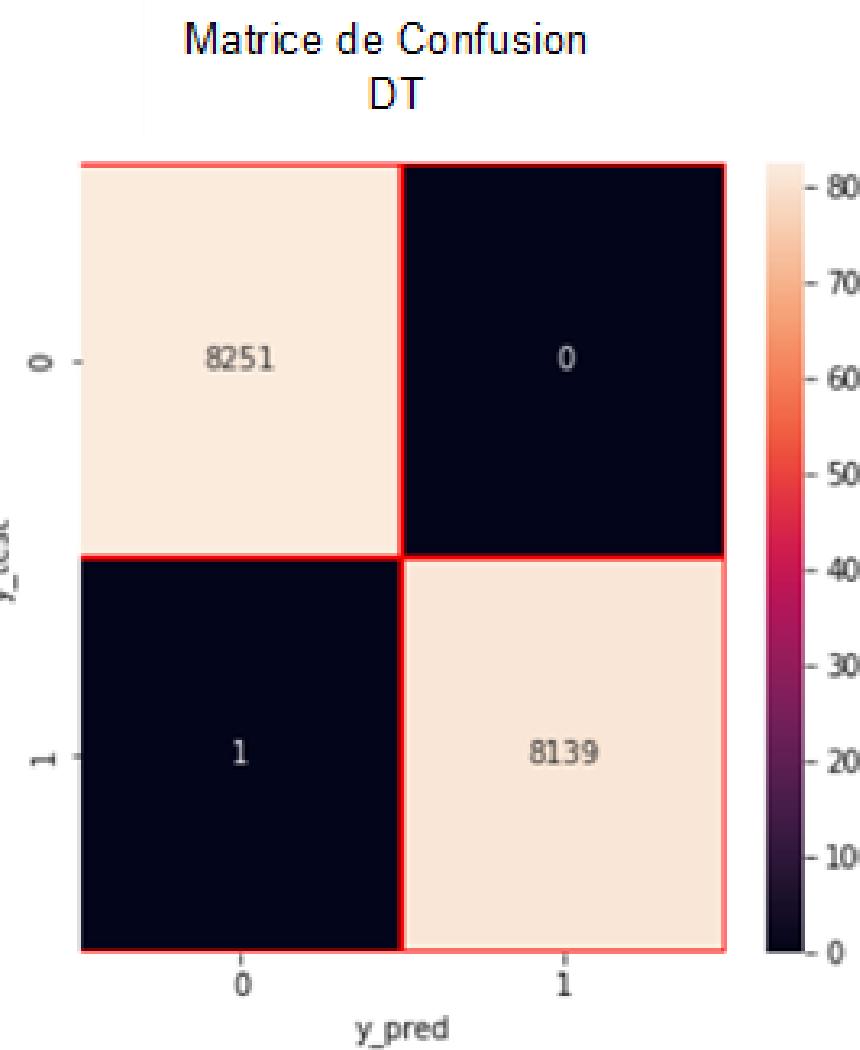
Performance des méthodes machine learning sur la classification binaire

1

2

Résultats

Matrice de Confusion



Discussions

1

2

Discussions

Conclusion et
perpective

Performance des méthodes machine learning sur la classification binaire

À l'exception de régression logistique qui a enregistré des performances en dessous de 86%, les métriques des autres algorithmes ont enregistré des résultats assez bons.

Il en ressort que le choix de l'algorithme le plus adéquat pour la classification binaire serait Random Forest avec 15 epoch et nombre d'estimateurs 11 qui a une 100% score dans train et 100% accuracy dans le test.

C'est normal que les méthodes prennent plusieurs temps d'entraînement car ils ont entraîné pour 15 à 20 epochs. Mais plus vite dans le temps de prédiction.

Les résultats des méthodes Xgboost , adaboost et Decision tree sont similaires, juste que Adaboost est plus long que les autres méthodes.

Et pour SVM à un inconvénient que la vitesse de convergence est faible, c'est à dire est lent à converger mais il donne une meilleure résultat par apport logistique régression.

- Nous avons trouvé que Random forest est le meilleur modèle pour la classification

Conclusion et perspective

1

2

Discussions

Conclusion et
perspectives

Dans ce travail on s'est intéressé aux techniques de détection d'intrusion, nous avons précisé sur la technique de machine Learning ainsi que on a fait la comparaison entre les méthodes de ce dernier. Cette comparaison est basée sur le taux de précisions dans l'entraînement et l'évaluation de la base de données du NF-UQ-NIDS-v2.

NEXT ➔

Classifier le dataset par des classes multiples c'est- à-dire le type d'attaque.

chercher à minimiser les attributs, par le 'Feature Selection' qui aide a choisir des fonctionnalités qui donneront une précision bonne ou meilleure tout en nécessitant moins de données pour améliorer davantage le temps de prédiction et celui d'entraînement.

Appliquer des algorithmes du Deep Learning

Merci !



Asakour & Elmourabit & Naim