Université Cadi Ayyad
Faculté des Sciences et Techniques, Marrakech
Master : Science des Données et Aide à la Décision
Module : Data Mining

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

Réalisé par :
- ASAKOUR Ihsane
- ELMOURABIT Zohair
- NAIM Kawtar

Encadré par :
- Madame h.BOUZAACHANE

année universitaire 2022-2023

Table des matières :

Liste des abréviations :

IDS	Intrusion Detection System
INPT	Institut national des postes et télécommunications
NIDS	Network Based Intrusion Detection System
HIDS	Host Based Intrusion Detection System
MIDS	Mixed IDS
NBA	Network Behavior Analysis
IA	Intelligence Artificielle
SVM	Support Vector Machine ou Machine
DT	Decision Tree

Listre des figures :

Résumé :

Ce rapport décrit les étapes de notre travail sur un mini-projet dans le cadre du module de datamining, dans cette période nous avons la chance de travailler sur un sujet qui est inclus dans le domaine de la sécurité informatique. Notre sujet est Détection des intrusions dans le domaine de la Cybersécurité Approche avec Machine Learning. Il consiste à détecter et à signaler les activités suspectes ou malveillantes sur un réseau ou un système informatique avec une approche couramment utilisée c'est l'utilisation de l'apprentissage automatique (Machine Learning). Cette approche consiste à entraîner un modèle en utilisant des données de trafic normal et anormal, puis à utiliser ce modèle pour détecter de nouvelles activités suspectes. Il existe plusieurs algorithmes d'apprentissage automatique qui peuvent être utilisés pour la détection des intrusions, tels que les arbres de décision, les réseaux de neurones et les machines à vecteurs de support. Chacun de ces algorithmes à ses propres avantages et inconvénients en termes de précision et de rapidité de détection.

Dans ce travail nous avons devant le choix du meilleur algorithme parmi eux en basant sur des métriques de comparaison. Ces modèles vont entrainer sur le même jeu donné dans les mêmes conditions (même dataset pour l'apprentissage et aussi pour le test et aussi le même ordinateur avec les mêmes ressources).

Introduction générale :

Avec le développement des réseaux et notamment des réseaux Internet, les technologies de l'information et de la communication nous offrent actuellement des facilités essentielles en matière d'enseignement à distance, d'achat et de paiement en ligne, de communication par messagerie instantanée, de visioconférence et de nombreuses autres technologies émergentes telles que les distributeurs automatiques, les véhicules autonomes.

Aujourd'hui, avec la multiplication des cybers-attaques et le lourd tribut qui en découle, les mesures préventives, telles que les antivirus et les pare-feu, se sont révélées insuffisantes et ont conduit à la création de systèmes de détection. Celles basées sur l'intelligence artificielle.

Le système de détection d'intrusion (IDS) est une invention qui répond à ces exigences.

Il a été approuvé dans le but de prévenir toutes les menaces imminentes de violations des politiques de sécurité, des réseaux et des systèmes informatiques.

Avec le développement des cybers attaques en matière de complexité, d'échelle et de fréquence, l'utilisation des méthodes de détection traditionnelles n'était plus pratiquée et de nouvelles technologies de protection avancées étaient nécessaires. En conséquence, les entreprises de cybersécurité se tournent vers l'utilisation de l'apprentissage automatique, qui a été utilisé avec succès dans la reconnaissance d'images, la recherche et la prise de décision, pour améliorer l'efficacité de leurs produits contre de nombreux problèmes de détection de cyber attaques telles que la détection d'intrusion et la détection de logiciels malveillants.

Afin de produire des systèmes de détection modernes tirent profit des atouts de l’intelligence artificielle, qui sont capable de détecter même les attaques les plus sophistiquées en alertant de tout écart par rapport au comportement nominal du système ou du réseau.

Ce projet visé a amélioré les résultats obtenus dans le projet de fin d’études réalisé par Tariq MOUATASSIM étudiant en INPT, nous nous intéressons pour notre problème de classification binaire, aux différentes techniques utilisées en détection d'intrusion, qui sont généralement des techniques d'apprentissage automatique.

Notre rapport est organisé comme suit :

Le premier chapitre est consacré à la présentation des différents aspects de la sécurité informatique et les systèmes de détection d'intrusion.

Le deuxième chapitre est axé à la présentation des modéles de la classification, le dataset, ainsi que les approches proposées pour améliorer les résultats.

Le troisième chapitre consiste à présenter les résultats obtenus par les 7 méthodes de classification utilisées et établir une comparaison entre ces méthodes ainsi que donner des explications.

Finalement, dans le quatrième chapitre, une conclusion est présentée sur le projet en générale, et aussi nos perspectives.

Chapitre 1 : Etat de l'art et contexte du projet

Introduction :

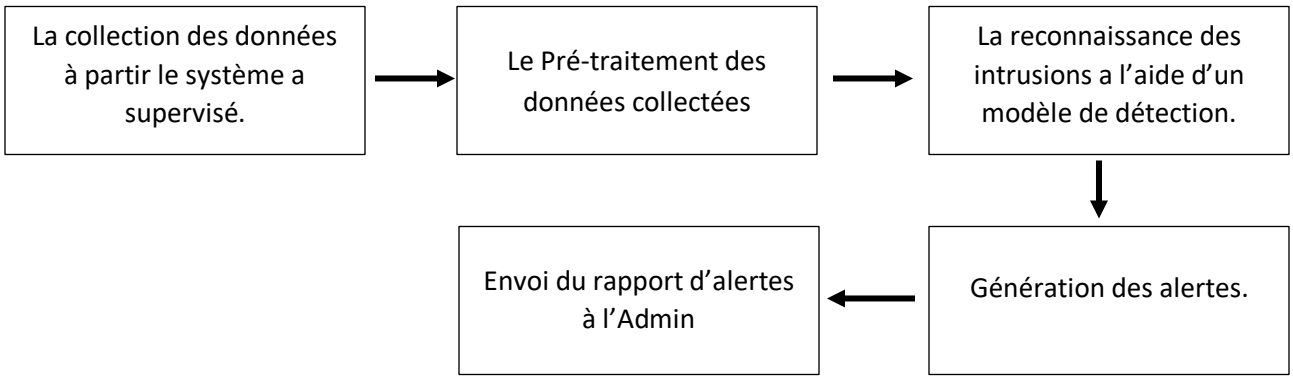
Dans ce chapitre on va parle sur le contexte de notre projet et aussi les notions de domaine de cybersecurity

1. Détection des intrusions

La détection d'intrusion est le processus de surveillance du trafic réseau et d'analyse de celui-ci pour détecter des signes d'éventuelles intrusions, telles que des tentatives d'exploitation et des incidents pouvant constituer des menaces imminentes pour le réseau. Pour sa part, la prévention des intrusions consiste à effectuer une détection des intrusions puis à arrêter les incidents détectés, généralement en supprimant des paquets ou en mettant fin à des sessions.

2.Processus de détection d'intrusion

La détection d'intrusion se fait en se basent sur les étapes suivantes :



3.Les types de système de détection d'intrusion (IDS)

On appelle **IDS** (*Intrusion Detection System*) un mécanisme écoutant le trafic réseau de manière furtive afin de repérer des activités anormales ou suspectes et permettant ainsi d'avoir une action de prévention sur les risques d'intrusion.

Il existe Il existe deux grandes familles distinctes d'IDS :

- Les N-IDS (Network Based Intrusion Detection System)

Le N-IDS assure la sécurité au niveau d'un réseau, il nécessite un matériel dédié et constitue un système capable de contrôler les paquets circulant sur un ou plusieurs lien(s) réseau dans le but de découvrir si un acte malveillant ou anormal a lieu.

- Les H-IDS (Host Based Intrusion Detection System):

Le H-IDS assure la sécurité au niveau des hôtes, réside sur un hôte particulier et la gamme de ces logiciels couvre donc une grande partie des systèmes d'exploitation.

Il existe deux autres types d'IDS Network Behavior Analysis (NBA) Based IDS et Mixed IDS (MIDS) ce dernier combine les avantages des différents IDS. En effet, il fournit une meilleure détection et prévention contre les intrusions et permet de s'affranchir des limitations des IDS prises individuellement.

4.Le Machine Learning au service de Cyber Security :

4.1. Définition du Machine Learning

Le machine learning, ou apprentissage automatique, est une forme d'intelligence artificielle. Il permet aux applications logicielles de prédire des résultats avec plus de précision, sans qu'elles ne soient programmées explicitement à cette fin.

Les algorithmes de machine learning vont pour cela se servir des données comme entrée pour prédire de nouvelles valeurs de sortie. Ils imitent ainsi la façon dont les êtres humains apprennent, en améliorant progressivement la précision de leur raisonnement.

Un excellent cas d'usage du machine learning est la définition de l'apprentissage automatique. Cette dernière nous vient des moteurs de recommandations, qui proposent du contenu à leurs utilisateurs en fonction de leurs préférences. Mais on trouve également des exemples de machine learning dans la détection des fraudes, le filtrage des spams, la définition et l'automatisation des processus métier ou encore la maintenance prédictive.

4.2.Les types d'apprentissage automatique

- L'apprentissage supervisé :

L'apprentissage supervisé ou supervised learning est une méthode de machine learning s'appuyant sur des données ou exemples labellisés (étiquetés ou annotés) pour entraîner des modèles d'intelligence artificielle (IA) prédictifs.

En se basant sur cette base d'apprentissage, par exemple des photos assorties de la mention de ce qu'elles représentent, les paramètres du modèle s'ajustent en vue ensuite de réagir efficacement face à des situations similaires. Au fur et à mesure de l'enrichissement du modèle, le résultat gagne en pertinence, réduisant la marge d'erreur.

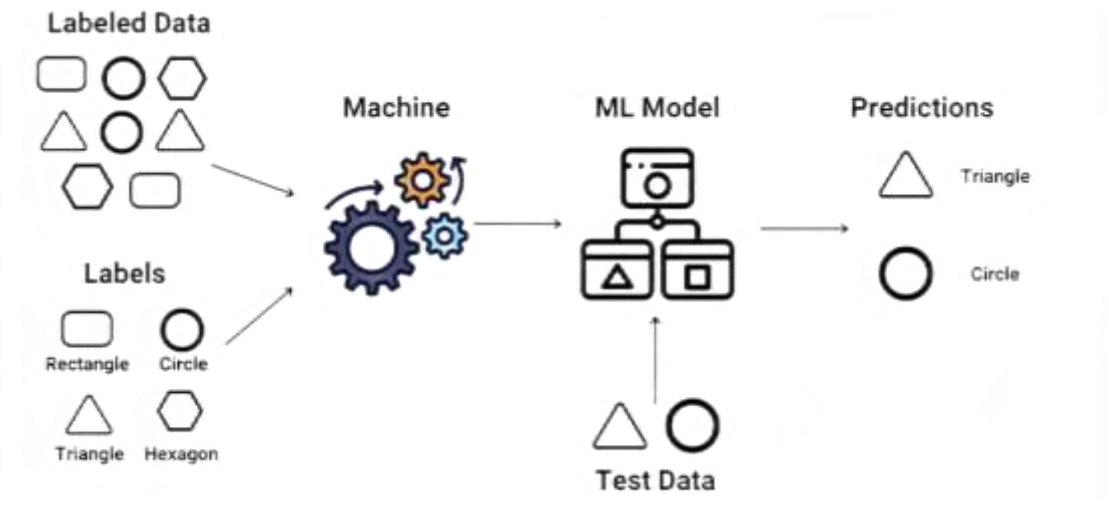


Figure 1: Processus d'apprentissage supervisé.

- L'apprentissage non-supervisé :

En machine Learning, la technique de l'apprentissage non supervisé (ou unsupervised learning) consiste à entraîner des modèles, sans réaliser d'étiquetage manuel ou automatique des données au préalable. Les algorithmes regroupent les données en fonction de leur similitude, sans aucune intervention humaine.

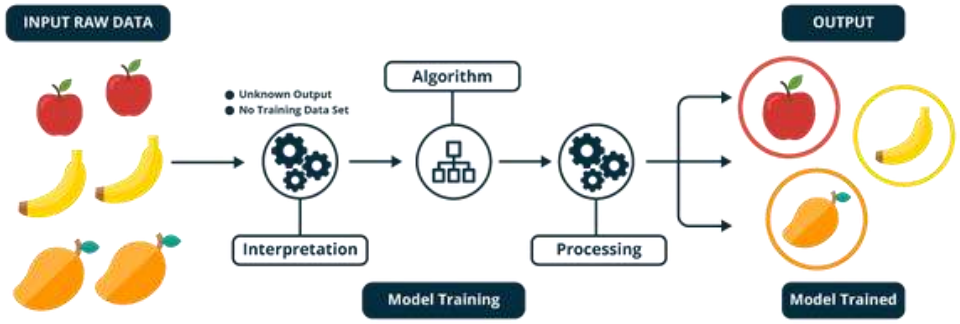


Figure 2:Processus d'apprentissage non- supervisé

- L'apprentissage par renforcement

L'apprentissage par renforcement est une méthode qui consiste à optimiser de manière itérative un algorithme uniquement à partir des actions qu'il entreprend et de la réponse associée de l'environnement dans lequel il évolue.

Cette méthode permet aux machines et aux agents de déterminer automatiquement le comportement idéal dans un contexte spécifique pour maximiser ses performances. Une simple rétroaction de récompense, connue sous le nom de signal de renforcement, est nécessaire pour que l'agent apprenne quelle action est la meilleure.

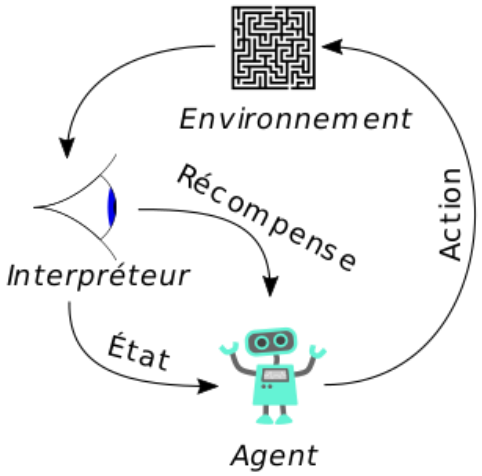


Figure 3:Processus d'apprentissage par renforcement

5. L'apprentissage automatique au service de Cyber Security :

Notre Projet se déroule autour de la thématique de l'intelligence artificielle, et plus précisément du Machine Learning. Avec la capacité de comprendre et d'analyser un contexte donné, l'IA peut aider à détecter des comportements anormaux, révéler des attaques et renforcer les outils de protection et de détection. L'étude concerne la détection des attaques et des intrusions dans le domaine de la Cybersécurité, et sera basée sur plusieurs méthodes d'apprentissage automatique. On effectue en premier lieu une défense prédictive. Cette approche prédictive, rendue possible par la data science, est un outil efficace pour lutter contre les attaques. Les organisations y trouvent un dispositif indispensable pour assurer la sécurité des systèmes et éviter d'être exposés à des risques importants.

L'idée de notre projet c'est de détecter les intrusions à l'aide de l'apprentissage automatique, a partir de l'entrainement d'un model machine Learning pour d'être capable de détecter les attaques, ce model va implémenté dans le NIDS, pour contrôler tous les trafics viennent via un réseau.

6. Rôle du NIDS au sein d'un réseau

Placé à l'intérieur du réseau local, comme illustré sur la figure 4, le NIDS permet de capturer tout le trafic qui circule à l'intérieur du réseau à la recherche des intrusions potentielles émanant aussi bien de l'intérieur du réseau que de l'extérieur. Par opposition au Pare-Feu qui se base sur des règles implémentées par l'administrateur pour interdire ou autoriser le trafic depuis et vers l'extérieur, le NIDS inspecte le trafic

circulant à l'intérieur du réseau, et déclenche des alertes en cas de présence d'une signature maligne dans les paquets, ou d'un comportement anormal. Parmi les fonctionnalités d'un NIDS, on peut citer :

- Enregistrement des événements de cybersécurité qui peuvent être acheminés et centralisés dans un collecteur de logs ou un Security Information and Event Manager (SIEM).
- Envoi des alertes par émail ou à travers des messages pop-up qui sont visualisés sur l'interface du NIDS.
- Génération des rapports qui peuvent être analysés à posteriori, comme la tendance de l'usage de la bande passante.

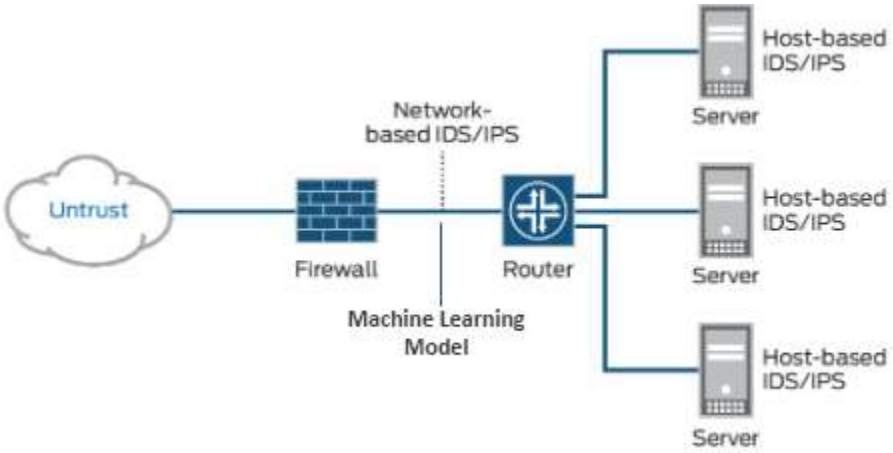


Figure 4:Implémentation de notre model machine Learning dans le NIDS .

7.Classification des intrusions par Machine Learning

Dans notre projet on va classifier les intrusions par Machine Learning suivant les étapes suivantes :

❖ Etape 1 : Collection des données et le pré-traitement :

La première étape consiste à rassembler les instances avant d'appliquer le pré-traitement. Chaque instance doit avoir une étiquette qui correspond à la catégorie de l'attaque s'il s'agit d'une intrusion, ou de la mention normale pour un trafic normal. Ces étiquettes doivent provenir d'un ensemble fini de catégories.

De plus, les classes doivent être équilibrées. Le nombre d'instances pour chaque catégorie devrait être approximativement uniforme (c'est-à-dire le même nombre d'instances par catégorie) sinon notre classificateur sera naturellement biaisé pour s'adapter aux catégories fortement représentées.

❖ Etape 2 : Division de l'ensemble de données :

Dans cette partie on divise notre dataset en 2 parties :

Partie d'entraînement qui est utilisé par le classificateur pour "apprendre" à quoi ressemble chaque classe en faisant des prédictions sur les données d'entrée, puis se corrige lorsque les prédictions sont fausses. Une fois que le classificateur a été entraîné.

Partie de test qui est utilisé pour évaluer la performance de notre model.

Il est extrêmement important que l'ensemble d'entraînement et l'ensemble de test soient indépendants l'un de l'autre et ne se chevauchent pas. Si l'ensemble de test fait partie des données d'entraînement, le classificateur a un avantage injuste puisqu'il a déjà été formé sur ces données. Au contraire, il est primordial de garder l'ensemble de test entièrement séparé du processus d'entraînement, en utilisant une séparation du Dataset initial.

- Etape 3 : L'entraînement du modèle.

L'entraînement d'un modèle en machine Learning consiste à utiliser des données d'entraînement pour ajuster les paramètres du modèle afin qu'il puisse effectuer une tâche spécifique de manière précise. Dans notre cas on va entrainer le model pour bien connaitre et aussi de classifier les attaques.

- Etape 4 : Evaluation :

Au cours de cette étape, chacune des instances de l'ensemble de test sera présenté au modèle, qui se chargera de prédire ce qu'il pense être l'étiquette du trafic. Ensuite, ces prédictions du modèle sont comparées aux étiquettes de base de l'ensemble de test. Les étiquettes de base représentent la catégorie réelle de l'instance. Enfin, on peut calculer le nombre de prédictions correctes du classificateur et calculer les rapports agrégés tels que la précision, qui est utilisée pour quantifier la performance de notre modèle dans son ensemble.

Conclusion :

Ce chapitre a été consacré pour la présentation de notre projet aussi la définition de quelques notions dans le domaine de sécurité informatique, en donnant une définition d'une intrusion, après les systèmes de détection d'intrusion et l'utilisation des IDS. Enfin nous avons définir l'apprentissage automatique avec ces différents types et finalement la stratégie comment on applique le machine Learning dans la détection des intrusions.

Chapitre 2 : Modèles de classification, Dataset et les Approches Proposées

Introduction :

Dans le 2-eme chapitre, on présente les algorithme applicable sur notre dataset, aussi une description sur cette ensemble de données et finalement les approches proposées.

1. Modèles de classification

1.2. Decision Tree

Un arbre de décision est un modèle très simple. Etant donnée plusieurs caractéristiques, la décision se commence par un de ces caractéristiques; si ce n'ai pas suffisant, on utilise une autre, ainsi de suite. Il est largement connu et utilisé pour faciliter le processus de prise de décision et l'analyse des risques.

Afin d'avoir une décision il faut :

- Déterminer la meilleure caractéristique dans l'ensemble de données d'entrainement.
- Diviser les données d'entrainement en sous-ensembles contenant les valeurs possibles de la meilleure caractéristique.
- Générez de manière récursive de nouveaux arbres de décision en utilisant les sous-ensembles de données créés.
- Lorsqu'on ne peut plus classifier les données, on s'arrête.

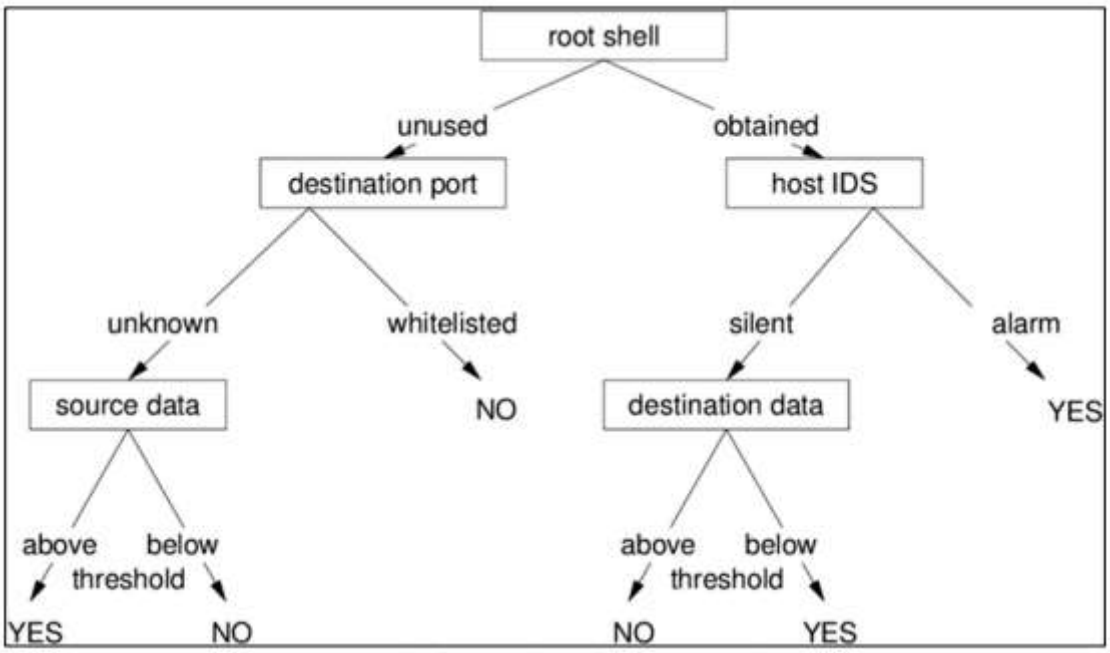


Figure 5: Illustration d'un arbre de décision

Le Decision Tree possède plusieurs hyperparamètres. Les plus basiques étant :

- **criterion** (« gini » ou « entropy ») – La fonction (« gini » ou « entropy ») a prendre en compte pour calculer l'incertitude sur la règle de discrimination utilisée.
- **random_state** (nombre entier) – permet de contrôler l'aléatoire dans l'algorithme.
- **max_features** (nombre entier) – Le nombre de feature à prendre en compte lors de la recherche du meilleur split
- **max_depth** (nombre entier) – la profondeur maximale de l'arbre, pour contrôler le overfitting

1.2. Random Forest

Random Forest est une technique de Machine Learning utilisée pour résoudre des problèmes de régression et de classification. Il utilise l'apprentissage d'ensemble, qui est une technique qui combine de nombreux classificateurs pour fournir des solutions à des problèmes complexes.

Un algorithme de forêt aléatoire (Random Forest) se compose de plusieurs Decision Tree. La "forêt" générée par l'algorithme de forêt aléatoire est entraînée par bagging ou bootstrap aggregating. Le bagging est un méta-algorithme d'ensemble qui améliore la précision des algorithmes d'apprentissage automatique.

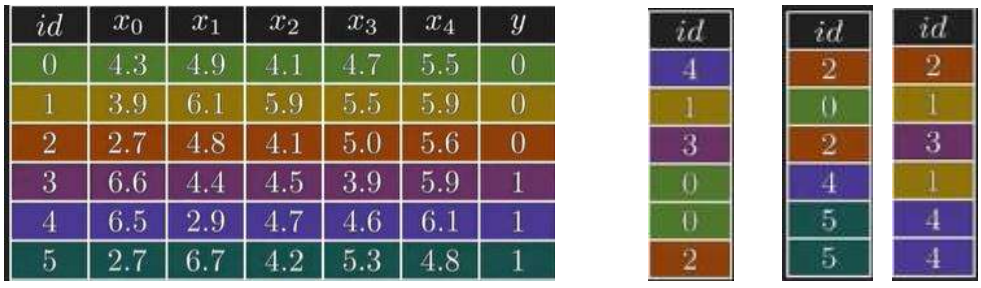


Figure 6:La génération du bootstapped data.

La classification finale est le résultat d'un vote majoritaire des arbres de décision constituant la forêt, comme illustré sur la figure 7

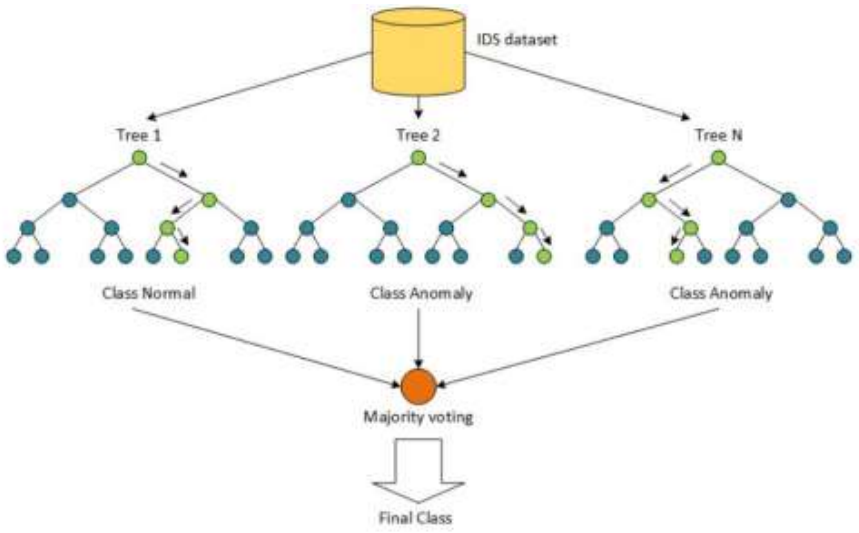


Figure 7:Illustration d'une forêt d'arbres de décision

1.3. Logistic Regression

L'algorithme de régression logistique 'Logistic Regression', également appelée modèle d'apprentissage prédictif, est une méthode statistique d'analyse des données qui s'alimente par une ou plusieurs variables indépendantes pour fournir une sortie.

L'objectif de cet algorithme est de trouver le meilleur modèle permettant d'étudier les relations entre un ensemble de variables qualitatives Xi et une variable qualitative Y.

Un modèle de régression logistique permet aussi de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce résultat varie toujours entre 0 et 1. Lorsque la valeur prédite est supérieure à un seuil, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure au même seuil, il ne l'est pas.

La régression logistique utilise une fonction logistique qui modélise une variable dépendante binaire. Elle transforme la sortie à l'aide de la fonction sigmoïde logistique pour renvoyer une valeur de probabilité qui peut ensuite être mappée à deux ou plusieurs classes discrètes, figure 8.

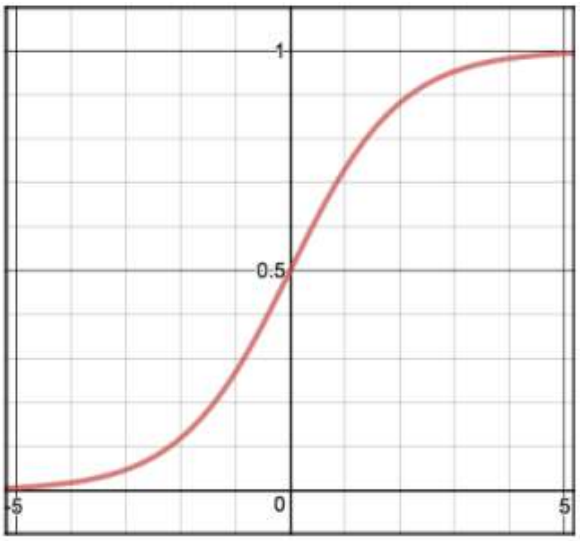


Figure 8: La fonction sigmoïde

1.4. AdaBoost

AdaBoost (Ada ptive **Boost** ing) est une technique de boosting très populaire qui vise à combiner plusieurs classificateurs faibles pour construire un classificateur fort. Le concept fondamental d'AdaBoost est de réaffecter les poids à chaque instance et de donner des poids plus élevés aux instances mal classées.

Un seul classificateur peut ne pas être en mesure de prédire avec précision la classe d'un objet, mais lorsque nous regroupons plusieurs classificateurs faibles, chacun apprenant progressivement des objets mal classés des autres, nous pouvons construire un tel modèle fort. Le classificateur mentionné ici peut être n'importe lequel de vos classificateurs de base, des arbres de décision (souvent la valeur par défaut) à la régression logistique, etc.

Afin de construire un classier AdaBoost, il faut suivre les étapes suivantes :

- Un classificateur faible (par exemple, une souche de décision) est créé au-dessus des données d'apprentissage sur la base des échantillons pondérés. Ici, les poids de chaque échantillon indiquent à quel point il est important d'être correctement classé. Initialement, pour la première souche, nous donnons à tous les échantillons des poids égaux.
- Nous créons une souche de décision pour chaque variable et voyons dans quelle mesure chaque souche classe les échantillons dans leurs classes cible. Par exemple, dans le diagramme ci-dessous, nous vérifions l'âge, la malbouffe et l'exercice. Nous examinerons le nombre d'échantillons correctement ou incorrectement classés comme aptes ou inaptes pour chaque souche individuelle.

- Plus de poids est attribué aux échantillons mal classés afin qu'ils soient correctement classés dans la souche de décision suivante. Le poids est également attribué à chaque classificateur en fonction de la précision du classificateur, ce qui signifie une grande précision = un poids élevé !
- Réitérez à partir de l'étape 2 jusqu'à ce que tous les points de données aient été correctement classés ou que le niveau d'itération maximal ait été atteint, figure 9.

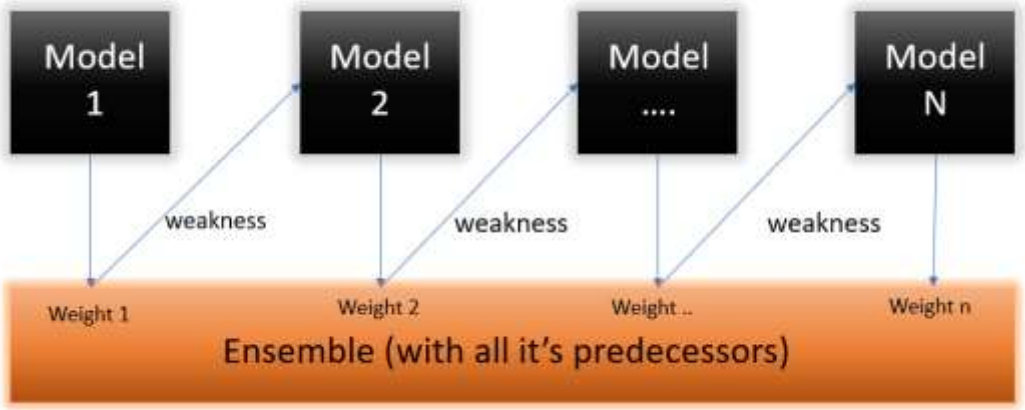


Figure 9:Le processus de l'algorithme Adaboost

1.5. Gradient Boosting

Le gradient boosting est une technique d' apprentissage automatique pour les problèmes de régression et de classification qui produit un modèle de prédiction sous la forme d'un ensemble de modèles de prédiction faibles. Cette technique construit un modèle par étapes et généralise le modèle en permettant l'optimisation d'une fonction de perte différentiable arbitraire. Le gradient boosting combine essentiellement les apprenants faibles en un seul apprenant fort de manière itérative. Au fur et à mesure que chaque apprenant faible est ajouté, un nouveau modèle est ajusté pour fournir une estimation plus précise de la variable de réponse. Les nouveaux apprenants faibles sont corrélés au maximum avec le gradient négatif de la fonction de perte, associée à l'ensemble entier. L'idée de l'amplification de gradient est que vous pouvez combiner un groupe de modèles de prédiction relativement faibles pour créer un modèle de prédiction plus fort, Figure 10.

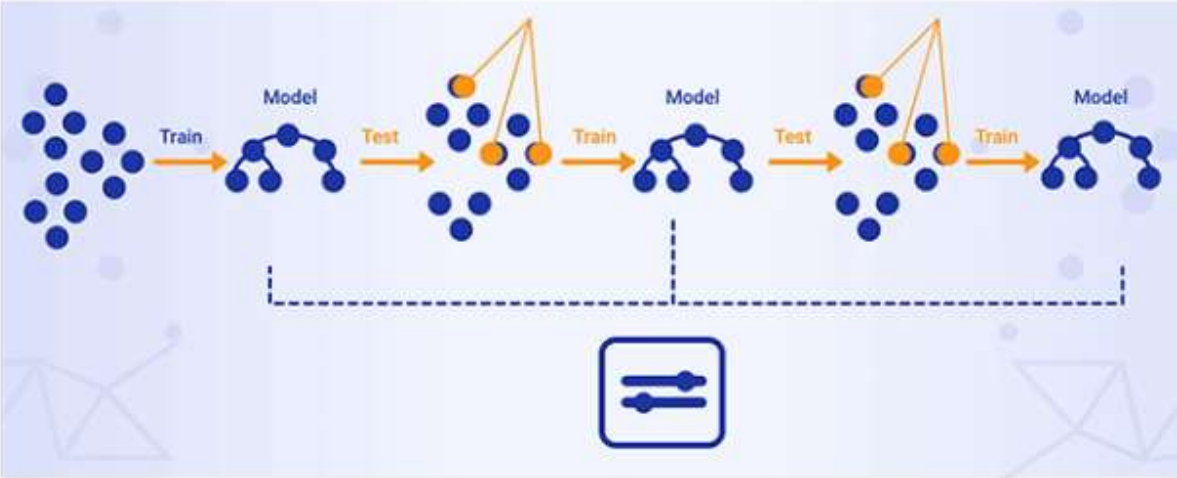


Figure 10:Le processus de l'algorithme Gradient Boosting .

1.6. XGBoost

Extreme Gradient Boost XGBoost est une implémentation de l'algorithme gradient Boosting, utilisée pour résoudre les problèmes de classification et de régression.

La bibliothèque XGBoost est conçue pour être efficace et évolutive, et elle possède un certain nombre de fonctionnalités qui la rendent bien adaptée à une variété de tâches d'apprentissage automatique. Voici comment cela fonctionne:

- XGBoost commence par ajuster un modèle faible, tel qu'un arbre de décision, aux données d'apprentissage.
- Le modèle faible est utilisé pour faire des prédictions sur les données d'apprentissage.
- Les valeurs prédites sont ensuite comparées aux valeurs réelles et l'erreur est calculée.
- Le modèle faible est ensuite mis à jour en fonction de l'erreur.
- Les étapes 2 à 4 sont répétées jusqu'à ce qu'un nombre spécifié de modèles faibles aient été ajustés, ou jusqu'à ce que les performances du modèle atteignent un certain seuil.
- Le modèle final est un ensemble de tous les modèles faibles, et il est utilisé pour faire des prédictions sur de nouvelles données.

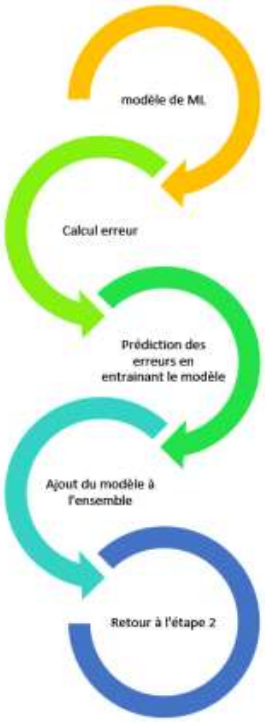


Figure 11:Le processus de l'algorithme XGBoost .

1.7. SVM

SVM (Support Vector Machine ou Machine à vecteurs de support) : Les SVMs sont une famille d'algorithmes d'apprentissage automatique qui permettent de résoudre des problèmes tant de classification que de régression ou de détection d'anomalie. Ils sont connus pour leurs solides garanties théoriques, leur grande flexibilité ainsi que leur simplicité d'utilisation même sans grande connaissance de data mining.

SVM a pour but de trouver un hyperplan dans un espace à N dimensions (N est le nombre de caractéristiques) qui sépare au maximum les différentes classes, figure 12.

Les SVM sont particulièrement bien adaptés à la classification d'ensembles de données complexes mais de petite ou moyenne taille. Ils sont également efficaces dans les espaces de grande dimension et dans les cas où le nombre de dimensions est supérieur au nombre d'échantillons.

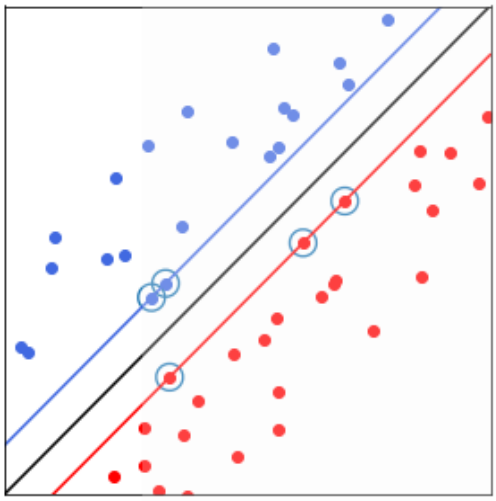


Figure 12:La séparation des données par un hyperplan.

2. Dataset

2.1. L'origine du Dataset

Pour comprendre le dataset utilisé dans ce projet, il est nécessaire de connaître l'origine de ce dernier. Tariq MOUATASSIM étudiant en INPT, le réalisateur de ce projet afin d'obtenir le Master en Cyber sécurité a utilisé le jeu de données 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. L'ensemble de données comprend à la fois le trafic normal et d'attaque, et il est couramment utilisé pour évaluer les performances des algorithmes IDS.

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.

NF-UQNIDS-v2 est largement utilisé dans la recherche sur les systèmes de détection d'intrusion, et il a fait l'objet de nombreux articles et études. Il est souvent utilisé pour évaluer les performances de différents algorithmes IDS, et c'est une ressource précieuse pour les chercheurs et les praticiens travaillant dans ce domaine.

Le jeu de données utilisé dans ce projet a déjà été nettoyé et préparé, on s'intéresse plus d'améliorer les résultats obtenus dans le projet de fin d'études de Tariq MOUATASSIM.

2.2. Aperçu du Dataset

Notre Dataset contient 81951 observations et 44 caractéristiques, figure 13.

```
df
```

	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

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

Figure 13:Dataset.

Le Dataset contient plusieurs étiquettes, pour notre problème de classification binaire on s'intéresse plus sur l'étiquette « label », qui montre est ce que une observation est déterminé comme une intrusion ou bien non.

La méthode isnull( ) a été appliquée sur le Dataset en vue de rechercher les champs vides. Aucun champ vide n'est présent dans notre Dataset, figure 2.10.

```
# verifier s'il y a des champs vides
df.isnull().any().any()
```

```
False
```

Figure 2.10 : Absence de champs vides dans le Dataset

Le Dataset contient des classes équilibres, figure 14. (Classe 0: Pas une intrusion, Classe 1: Une intrusion)

```
: df.Label.value_counts()

: 0    41002
  1    40949
  Name: Label, dtype: int64
```

Figure 14: L'équilibre entres les classes.

3. Les Approches Proposées

Afin d'avoir des résultats améliorés en comparaison avec les unes des premiers projets, on a mis à la connaissance des changements au niveau de l'application des modèles de classification, parmi les approches utilisées pour atteindre le but est les suivants :

3.1. 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. Les hyperparamètres sont des paramètres qui sont définis avant de commencer l'apprentissage d'un modèle et qui influencent son comportement et ses performances. Ils sont généralement choisis en fonction de la nature des données et du problème à résoudre.

Il existe plusieurs techniques pour estimer les hyperparamètres d'un modèle, notamment :

a. La validation croisée : cette technique consiste à diviser le jeu de données en plusieurs sous-ensembles et à utiliser l'un de ces sous-ensembles comme ensemble de validation pour évaluer les performances du modèle avec différentes valeurs d'hyperparamètres.
b. L'optimisation de grille : cette technique consiste à essayer toutes les combinaisons possibles d'hyperparamètres et à choisir celles qui donnent les meilleures performances.
c. L'optimisation aléatoire : cette technique consiste à générer aléatoirement des valeurs d'hyperparamètres et à choisir celles qui donnent les meilleures performances. celle-ci qui nous intéresse, pour réaliser cette

tâche on a créé une boucle for, dont chaque iteration le modèle va entrainer sur l'ensemble d'entrainement avec une valeur d'hyperparamètre aléatoire, et à chaque fois on calcule le score sur l'ensemble d'entrainement et de test, pour avoir la valeur d'hyperparamètre qui donne le meilleur score

Chaque modèle de machine Learning a des hyperparamètres qu'on peut ajuster leurs valeurs dans le but d'augmenter les performances de modèle, parmi ces hyperparamètres, on mentionne :

a. Pour Random Forest

   n_estimators : est le nombre d'arbres à utiliser dans la forêt.

   max_depth : la profondeur maximale de l'arbre, pour contrôler le sur-apprentissage.

b. Pour AdaBoost

   n_estimators : Le nombre maximal d'estimateurs auquel l'amplification est terminée

   learning_rate : Pondération appliquée à chaque classifieur à chaque itération de boosting. Un learning_rate plus élevé augmente la contribution de chaque classifieur.

c. Pour Gradient Boosting

   n_estimators : Le nombre d'étapes de boosting à effectuer

   Critère : La fonction pour mesurer la qualité d'une division. Les critères pris en charge sont 'friedman_mse' pour l'erreur quadratique moyenne avec le score d'amélioration de Friedman, 'squared_error' pour l'erreur quadratique moyenne.

3.2. Training Loop

Le training loop (boucle d'apprentissage) 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 entrainé sur l'ensemble d'entrainement plusieurs fois afin de bien prédit sur l'ensemble de test.

Pour réaliser cette tache, Nous ferons connaissance des deux termes importants dans le training loop

a.Epoch : (ou époque en français) est une itération complète sur l'ensemble des données d'entraînement dans un modèle de machine learning. Par exemple, si vous avez un jeu de données d'entraînement composé de 1000 exemples, une époque consiste à utiliser tous ces exemples pour mettre à jour les poids du modèle.

b. Batch : (ou lot en français) est un sous-ensemble des données d'entraînement utilisé pour mettre à jour les poids du modèle. Par exemple, si vous définissez le lot à 100, cela signifie que vous utilisez 100 exemples à chaque itération pour mettre à jour les poids du modèle.

Conclusion

Dans ce chapitre nous avons énuméré et détaillé les différents modèles de classification qui seront utilisés dans notre projet, à savoir : Décision Tree, Random Forest, AdaBoost, Gradient Boosting, Logistic Regression, XGBoost et SVM. Ensuite, un aperçu du Dataset utilisé a été présenté, accompagné d'une brève représentation de son origine. Enfin, le chapitre s’est achevé par une présentation des différentes approches proposées pour augmenter la performance et l'évaluation des modèles.

Chapitre3 : Résultat et discussion

Introduction :

Dans le dernier chapitre on va citer les résultats de notre modèles aussi l'évaluation et choix des meilleurs entre eux dans la partie de discussion.

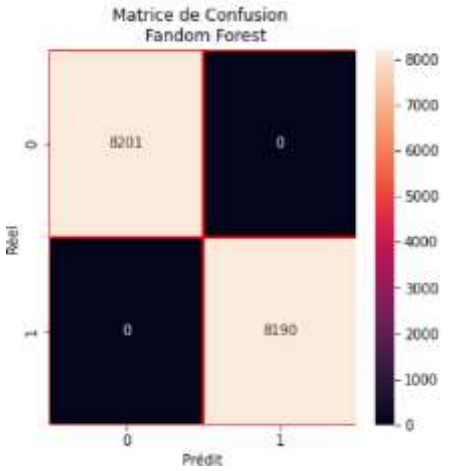
1.Résultat

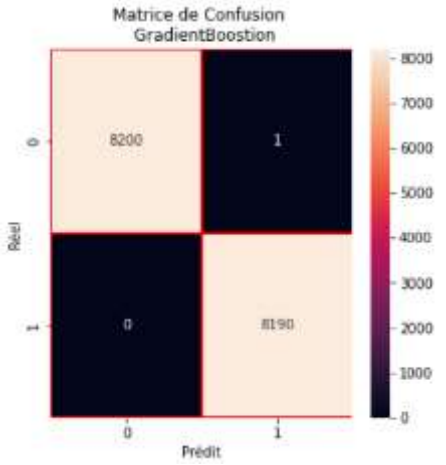
Après appliquées les modèles sur notre problématique d'implémenter un modèle parfait pour détecter les attaques dans un réseau. Ainsi l'évaluation de ces algorithmes portera sur le critère F1 score avec l'exactitude, tout en prenant en considération le temps d'entraînement et de prédiction. On a trouvé les résultats suivants

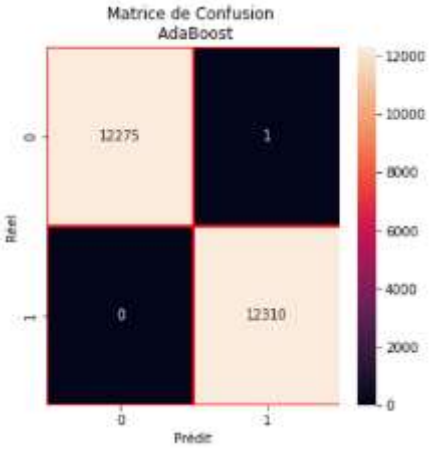
	Random Forest	Gradient Boosting	Logistique regression	Xgboost	Decision Tree	Adaboost	SVM
Epoch	15	10	1	1	20	20	10
Hyperparametre	N_estimators = 11	N_estimators = 100	solver='saga' max_iter =1000	n_estimators = 240		n_estimators = 100	C = 52
Temp entrainem	12.539312601089478	296.1378493309021	4.167687177658081	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	6.100909035444690 4e-05	0.14148008053199923	6.100909035444690 4e-05	6.100909035444690 4e-05	4.067355405512618e-05	0.0014642181685070588
F1 - SCORE	1.0	0.9999389909119165	0.858514510003111	0.9999389909119165	0.9999389900884211	0.9999593264435898	0.9985357819949923
Recall	1.0	0.9999389909096456	0.8585199194680008	0.9999389909096456	0.9999389909096456	0.9999593264459449	0.9985357818314929

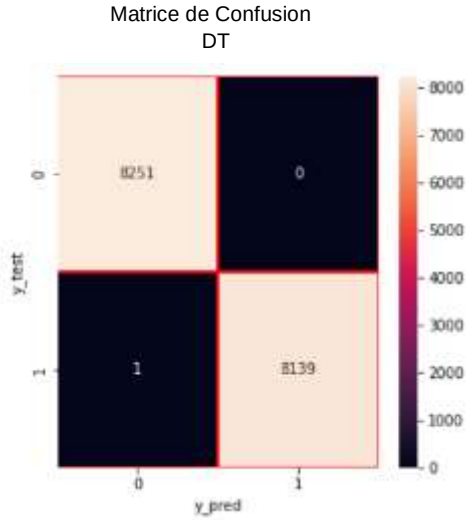
:

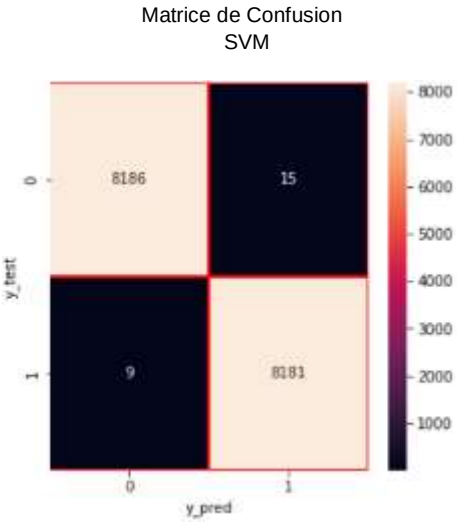
Figure 15:Performance des méthodes machine learning sur la classification binaire.

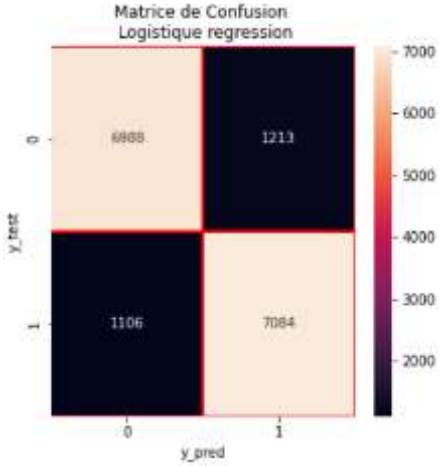


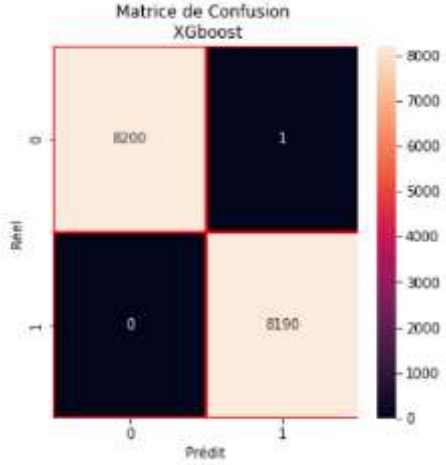












2.Discussion

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 prendraient plusieurs temps d'entrainement car ils ont entrainé pour 15 à 20 epochs. Mais plus vite dans la prédiction.

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

Conclusion :

Ce chapitre nous a permis de découvrir et comparer les différentes techniques de machine Learning pour détection d'intrusion et leurs fonctionnements.

Conclusion et perspective

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 à la base de données du NF-UQ-NIDS-v2.

L'adoption du Dataset récent NF-UQ-NIDS-v2, eu égard à ses caractéristiques excellentes : variété et pertinence des attaques, variété des protocoles et présence des attaques complexes.

Les résultats obtenus sont prometteurs en comparaison avec les articles scientifiques les plus récents traitant de la même problématique. En effet, la classification binaire a enregistré une exactitude et un F1 score de 100% en utilisant Random Forest, et un temps de prédiction qui est inférieur à 1 ms pour Decision Tree et 3 ms pour Random forest tout en préservant une précision et une exactitude d'entre 99.99% et 100%. Sauf régression logistique qui a enregistré des performances en dessous de 86%.

Par ailleurs, il serait judicieux de penser à améliorer la solution en pensant à d'autres méthodes d'optimisation, en explorant les algorithmes du Deep Learning et en cherchant à minimiser les attributs, par le 'Feature Selection', pour améliorer davantage le temps de prédiction et celui d'entraînement et aussi de classifier le dataset par des classes multiples c'est- à-dire le type d'attaque.

Annexes et Références

- K. A. Scarfone et P. M. Mell, « Guide to Intrusion Detection and Prevention Systems (IDPS) », National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800- 94, 2007. doi: 10.6028/NIST.SP.800-94.

- Y. Maleh, M. Shojafar, M. Alazab, et Y. Baddi, Éd., Machine Intelligence and Big Data Analytics for Cybersecurity Applications, vol. 919. Cham : Springer International Publishing, 2021. doi: 10.1007/978-3-030-57024-8.

- Tariq MOUATASSIM, Détection des intrusions dans le domaine de la Cybersécurité Approche avec Machine Learning, Rapport de Stage .

- Emad E. Abdallah*, Wafa' Eleisah, Ahmed Fawzi Otoom, Intrusion Detection Systems using Supervised Machine Learning Techniques: A survey