République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 20 Août 1955- Skikda
Faculté des Sciences
Département d'Informatique

Mémoire

En vue de l'obtention du Diplôme de Master en Informatique

Spécialité : Réseaux et Systèmes Distribués (RSD)

Intitulé :

La détection automatique des intrusions dans les réseaux informatiques

Réalisé par :

- BEKKOUCHE Ilhem
- MAGHLAOUI Meriem

Devant le jury:

- Dr.BOULAICHE Mehdi	Président	Université de 20 Août 1955-Skikda
- Dr.NAFIR Abednacer	Examinateur	Université de 20 Août 1955-Skikda
- Dr.BOUGAMOUZA Fateh	Encadreur	Université de 20 Août 1955-Skikda

Année universitaire 2023/2024

DEDICACES

Je dédie ce modest travail à ...✍

A l'étre le plus cher de ma vie, ma mère.

Pour exprimer ma gratitude envers votre amour inconditionnel, votre soutien indéfectible, vos encouragements constants et vos innombrables sacrifices.

***À** mon chèr père*

Bien que tu ne sois plus physiquement présent, je sens ta force et ton soutien qui continuent de m'entourer. Tu resteras à jamais gravé dans mon cœur et dans mes souvenirs.

***À** mes chères sœurs*

Pour leurs encouragements , amour , conseils et leur soutien tout au long de mes études. Que Dieu les protège et leur accorde succès et bonheur.

***À** mon chère Neveu*

Firas

Tu es le petit personne qui la remplisse ma vie de joie et de bonheur.

A tous les amis et en particulier Meriem

Ilhem

DEDICACES

Je dédie ce modest travail à ...✍

À mes chers parents,

ce travail leur doit beaucoup,

qu'il soit pour eux le témoignage de mon infinie

reconnaissance pour leurs accompagnements

tout au long de mon parcours scolaire.

À ma sœur ,

À mes chères amis,

À mon binôme, ilhem Pour sa entant et sa sympathie

Enfin, je dédie ce travail à moi-même

Meriem

REMERCIEMENTS

Après cinq années d'efforts au cours desquelles nous avons acquis de nombreuses compétences, nous voilà arrivés à l'ultime étape avant l'obtention de nos diplômes.

Le plus grand merci est alhamdou lilleh. Puis merci énormément à monsieur Bougamouza qui m'a permis de bénéficier de son encadrement. Merci pour votre patience, conseils et merci pour l'honneur de travailler sous votre supervision.

Mes vifs remerciements aux membres du jury pour avoir accepté d'examiner ce modeste travail et d'enrichir, non seulement ce mémoire, mais aussi mes connaissances avec leur précieuses propositions. Et merci à tous mes enseignants durant toutes mes années d'études.

Je remercie profondément mes chers parents, à qui je dois ce que je suis et ce que je serais, qui n'ont jamais cessé de m'encourager et me conseiller, qui m'ont beaucoup aidé tout en long de mon parcours scolaire, grâce à leur amour, leur compréhension et leur patience sans jamais me quitter des yeux ni baisser les bras. C'est leur soutien inconditionnel qui m'a relevé quand je me suis lassé. Aucun remerciements ne peut exprimer ma reconnaissance que vous soyez mes parents.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Résumé

Au cours de la dernière décennie, le développement rapide des technologies de l'information et de la communication, en particulier Internet, a engendré une vague d'innovations sans précédent à travers le monde. Malheureusement, ces avancées ont été accompagnées par une augmentation des cyberattaques, rendant la protection des réseaux de communication un défi majeur pour les décennies à venir. En tant que première ligne de défense, les systèmes de détection d'intrusion (IDS) ont fait l'objet de nombreuses recherches et jouent un rôle crucial dans la sécurité des réseaux.

Ce travail présente l'état de l'art de la détection d'intrusion avec pour objectif de développer un modèle capable de détecter et de classifier un large éventail d'attaques. Les contributions de ce mémoire incluent l'exploration de diverses méthodes d'apprentissage automatique, l'étude approfondie d'une méthode d'apprentissage profond, l'autoencodeur, ainsi que l'analyse des résultats pour évaluer l'efficacité des différentes approches. Une comparaison des performances de l'apprentissage automatique et de l'apprentissage profond dans le contexte de la détection d'intrusions a été effectuée en utilisant la base de données publique CIC-IDS2017. Après un prétraitement et une normalisation des données, ces algorithmes ont été appliqués pour la classification et leurs performances ont été comparées à l'aide de différentes mesures d'évaluation.

En termes de performance, cette étude a révélé que les algorithmes d'apprentissage automatique surpassent l'auto-encodeur. Cette supériorité s'explique par la répartition déséquilibrée des échantillons de données entre les différentes classes d'attaques dans la base de données.

Mots clés : Système de détection d'intrusion (IDS), Intelligence artificielle, Apprentissage profond, Apprentissage automatique, CIC-IDS2017.

ملخص

خلال العقد الماضي، أدى التطور السريع لتقنيات المعلومات والاتصالات، وخاصة الإنترنت، إلى موجة من الابتكارات غير المسبوقة في جميع أنحاء العالم. ولسوء الحظ، رافقت هذه التطورات زيادة في الهجمات السيبرانية، مما جعل حماية شبكات الاتصال تحديًا كبيرًا للعقود القادمة. وكمستوى دفاع أول، كانت أنظمة كشف التسلل (IDS) موضوعًا للعديد من الأبحاث وتلعب دورًا حيويًا في أمن الشبكات.

يقدم هذا العمل أحدث ما توصلت إليه تقنية كشف التسلل بهدف تطوير نموذج قادر على اكتشاف وتصنيف مجموعة واسعة من الهجمات. تشمل مساهمات هذا البحث استكشاف طرق مختلفة لتعلم الآلة، ودراسة معمقة لطريقة التعلم العميق باستخدام المشفر التلقائي(auto-encoder) ، بالإضافة إلى تحليل النتائج لتقييم فعالية النهج المختلفة. كما تمت مقارنة أداء تعلم الآلة والتعلم العميق في سياق كشف التسلل باستخدام قاعدة البيانات العامة .CIC-IDS2017 بعد معالجة البيانات وتنظيمها، تم تطبيق هذه الخوارزميات للتصنيف ومقارنة أدائها باستخدام مقاييس تقييم مختلفة.

من حيث الأداء، أظهرت هذه الدراسة أن خوارزميات تعلم الآلة تتفوق على المشفر التلقائي. ويمكن تفسير هذا التفوق بعدم توازن توزيع عينات البيانات بين الفئات المختلفة للهجمات في قاعدة البيانات.

الكلمات المفتاحية : نظام كشف التسلل(IDS) ، الذكاء الاصطناعي، التعلم العميق، تعلم الآلة، CIC-IDS2017

ABSTRACT

Over the past decade, the rapid development of information and communication technologies, particularly the Internet, has led to an unprecedented wave of innovations worldwide. Unfortunately, these advancements have been accompanied by an increase in cyberattacks, making the protection of communication networks a major challenge for the coming decades. As the first line of defense, intrusion detection systems (IDS) have been the subject of extensive research and play a crucial role in network security.

This work presents the state of the art in intrusion detection, aiming to develop a model capable of detecting and classifying a wide range of attacks. The contributions of this thesis include the exploration of various machine learning methods, an in-depth study of a deep learning method (auto-encoder), and the analysis of results to evaluate the effectiveness of different approaches. A comparison of the performances of machine learning and deep learning in the context of intrusion detection was conducted using the public CIC-IDS2017 dataset. After preprocessing and normalizing the data, these algorithms were applied for classification, and their performances were compared using different evaluation metrics.

In terms of performance, this study revealed that machine learning algorithms outperform the auto-encoder. This superiority is explained by the imbalanced distribution of data samples among the different attack classes in the dataset.

Keywords : Intrusion Detection System (IDS), Artificial Intelligence, Deep Learning, Machine Learning, CIC-IDS2017.

Table de matière

Table de matière

Table de matière

Table de matière

Liste de figures

Chapitre 1: Généralités sur Intelligence Artificielle, Machine Learning et Deep Learning

Chapitre 2 : Les Systèmes de Détection d'Intrusions

Chapitre 3 : Conception, Experimentation et Resultats

Liste des tableaux

Chapitre 1: Généralités sur Intelligence Artificielle, Machine Learning et Deep Learning

Chapitre 2: Les Systèmes de Détection d'Intrusions

Chapitre 3: Conception, Experimentation et Resultats

Liste des abréviations

ABIDS Application-Based Intrusion Detection **S**ystem

AE Auto-Encodeur

AED L'auto-Encodeur de Débruitage

BNB Bernoulli Naive Bayes

CNN Réseaux des Neurones Convolutifs

DL Deep Learning

DR Dimensionality Reduction

DT Decision Tree

GNB Gaussian Naive Bayes

HIDS Host Based Intrusions Detection System

IA Intelligence Artificielle

IDS Intrusions Detection System

IDWG Intrusions Detection exchange formatWorking Group

KNN K Nearest Neighbors

LR Logistic Regression

ML Machine Learning

MNB Multinomial Naive Bayes

MNIST Base de Données de Chiffres Manuscrits

MSE Mode Sans Échec

NB Naive Bayes

NIDS Network Based Intrusions Detection System

NN Neural Network

PCA Principal Component Analysis

RF Random Forest

RL Reinforcement Learning

RNN Réseaux des Neurons Récurrents

SL **S**upervised Learning

SVM Support Vector Machine

TL Transfer Learning

UL Unsupervised Learning

Liste des abréviations

VAE L'auto-Encodeur Variationnel

Introduction Générale

1. Contexte

L'utilisation croissante d'Internet et des réseaux informatiques offre de nombreux avantages aux utilisateurs et à la société en général. Cependant, elle expose également à divers risques. Les systèmes de détection d'intrusion (IDS) jouent un rôle crucial dans la prévention des intrusions. Une intrusion se réfère à toute activité visant à compromettre la confidentialité, l'intégrité ou l'accessibilité des données, ou à contourner les mécanismes de sécurité d'un réseau informatique. La détection d'intrusion consiste à surveiller de manière proactive les incidents au sein d'un système ou d'un réseau, afin d'identifier et d'analyser les signaux indiquant une tentative d'intrusion. Les IDS peuvent être des systèmes logiciels, matériels, ou une combinaison des deux, spécifiquement conçus pour détecter ces intrusions.

2. Problématique

Les pirates informatiques exploitent activement les vulnérabilités des systèmes pour s'introduire et compromettre la sécurité, entraînant des conséquences dévastatrices telles que le vol de données sensibles et le sabotage des opérations. Ces attaques peuvent avoir un impact étendu, affectant non seulement les individus mais aussi les entreprises et les infrastructures critiques à travers le globe. Renforcer les systèmes de protection devient donc impératif pour prévenir et contrer efficacement ces menaces croissantes à la sécurité informatique.

3. Les objectifs

Dans cette étude, nous explorons plusieurs approches d'apprentissage automatique et d'apprentissage profond qui ont prouvé leur fiabilité dans le domaine de l'intelligence artificielle. Les principaux objectifs visés sont les suivants :

- Exploration de diverses méthodes d'apprentissage automatique.
- Exploration d'une méthode d'apprentissage profond : l'autoencodeur.
- Analyse des résultats obtenus pour évaluer l'efficacité des différentes approches
- Comparaison des performances de l'apprentissage automatique et de l'apprentissage profond dans le contexte de la détection d'intrusions.

4. Plan de mémoire

Outre cette introduction et la conclusion générale, ce mémoire est divisé en trois chapitres principaux :

Chapitre 01 : Ce chapitre est consacré à présenter les concepts généraux relatifs l'intelligence artificielle (IA), machine learning (ML) et deep learning (DL).

Chapitre 02 : Ce chapitre est consacré à présenter une architecture globale d'un systeme de detection d'instrusion (IDS), la définition et le mode de fonctionnement de ce dernier. Ainsi la classification et les types des IDSs et la méthode de détection d'une intrusion. Enfin l'état de l'art sur les IDSs.

Chapitre 03 : Il regroupe la description de notre système, ainsi que les différentes expérimentations réalisées et l'analyse des résultats obtenus.

L'annexe contient un complément : Outils et environnement de réalisation et Bibliothèques essentielles en Python

Chapitre 1:

Généralités sur Intelligence Artificielle , Machine Learning et Deep Learning

1. Introduction

Le mode de vie de la société moderne a considérablement changé ces dernières années avec l'expansion rapide des technologies d'intelligence artificielle, d'apprentissage automatique et d'apprentissage profond qui impactent presque tous les aspects technologiques de la société. Le réseau neuronal est une technique d'apprentissage automatique inspirée par la fonctionnalité sophistiquée du cerveau humain et qui ressemble au système nerveux humain et à la structure du cerveau. Il peut être utilisé dans une variété de problèmes.

Ce chapitre présente un groupe des techniques d'apprentissage automatique et des techniques d'apprentissage profond basées sur la technologie du réseau neuronal.

2. Intelligence artificielle

L'intelligence artificielle (en anglais Artificial Intelligence, IA) englobe un large éventail des techniques visant à conférer aux machines des capacités cognitives. Dans le domaine de l'IA, L'apprentissage automatique (en anglais Machine Learning, ML) se concentre sur le développement d'algorithmes permettant aux systèmes d'apprendre à partir des données sans une programmation explicite. À son tour, l'apprentissage profond (en anglais Deep Learning, DL) une sous-discipline du ML, exploite des réseaux de neurones artificiels profonds pour apprendre des représentations hiérarchiques des données. Cela facilite la compréhension et l'analyse de structures complexes dans des domaines tels que la vision par ordinateur et le traitement du langage naturel [1]. (Voir figure 1.1)

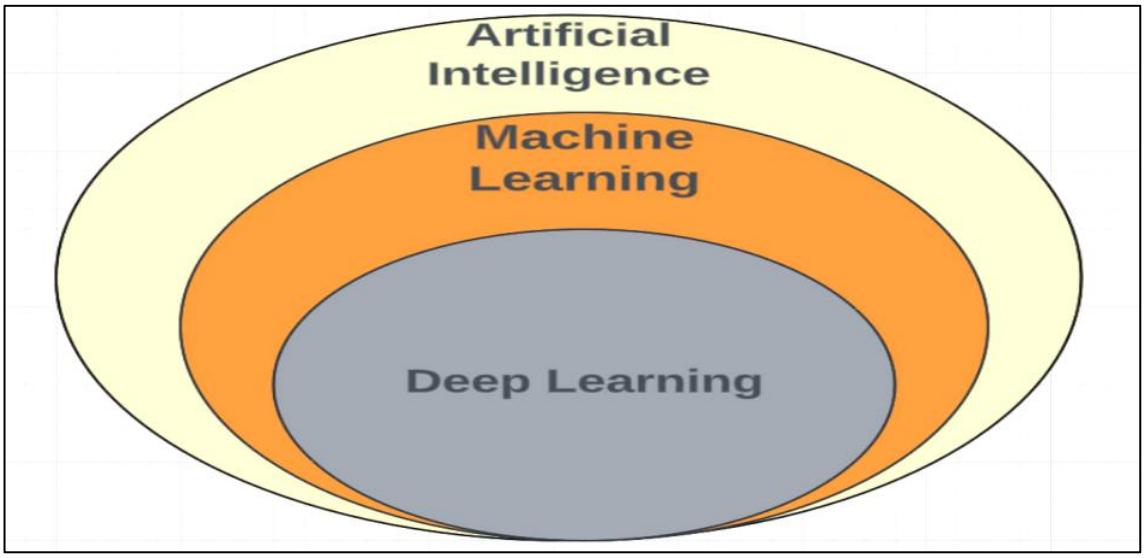

Figure 1.1 : Intelligence artificielle

3. Apprentissage automatique

L'ML est une discipline scientifique axée sur le développement d'algorithmes capables d'apprendre à résoudre des tâches spécifiques sans une programmation explicite [1]. En d'autres termes, le ML est une branche de l'intelligence artificielle qui permet à un logiciel d'apprendre et de reconnaître des modèles complexes de données, de manière similaire à un être humain. La capacité d'apprentissage d'une machine s'accroît avec la quantité de données collectées. Les méthodes d'ML peuvent être classées en plusieurs catégories, notamment l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé, l'apprentissage par renforcement et le transfert d'apprentissage [2]. (Voir figure 1.2)

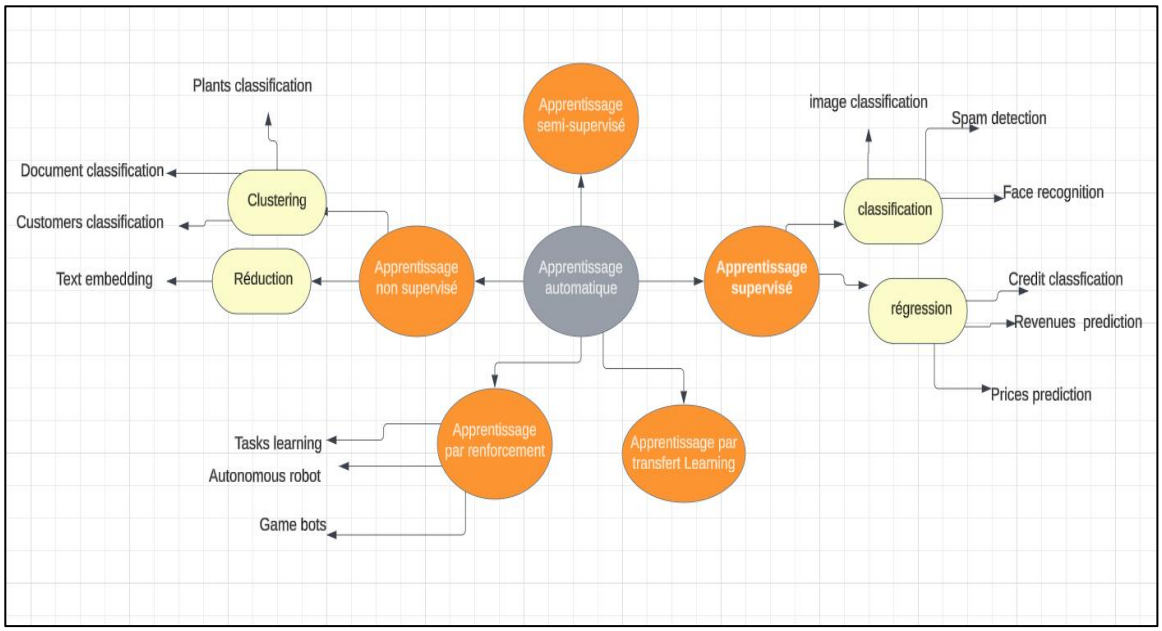

Figure 1.2 : Les méthodes d'apprentissage automatique

3.1. Types d'apprentissage automatique

3.1.1. Apprentissage supervisé

L'apprentissage supervisé (en anglais supervised learning, SL) constitue une sous-catégorie fondamentale et bien connue du ML. Cette méthode repose sur un ensemble d'exemples préalablement classifiés, où la catégorie de chaque entrée utilisée comme exemple est connue à l'avance. L'objectif de l'SL est de former un système artificiel capable d'apprendre la relation entre les entrées et les sorties, afin de prédire la sortie du système en fonction de nouvelles entrées. Typiquement, l'SL s'applique dans les contextes de classification et de régression [3].

3.1.2. Apprentissage non supervisé

L'apprentissage non supervisé (en anglais unsupervised learning, UL) est une méthode bien établie et simple du ML. Elle repose sur un ensemble d'exemples préalablement étiquetés, où chaque entrée est associée à une catégorie connue à l'avance. Son objectif est de former un système capable d'apprendre à prédire des sorties en fonction des entrées [4], L'UL comprend deux catégories d'algorithmes : Algorithmes de Regroupement (Clustering en anglais) et les algorithmes de Réduction de la dimentialité [5].

3.1.3. Apprentissage semi-supervisé

L'apprentissage semi-supervisé est une approche qui combine à la fois des éléments supervisés et non supervisés, où seule une partie des données est étiquetée. Cette partie étiquetée est ensuite utilisée pour déduire des informations sur la partie non étiquetée, comme c'est le cas dans les systèmes de récupération de texte ou d'image.

3.1.4. Apprentissage par renforcement

L'apprentissage par renforcement (en anglais reinforcement learning, RL) se distingue de SL et UL en enseignant à une machine à exécuter une séquence d'actions. Les chercheurs programment un algorithme pour accomplir une tâche dans le RL, fournissant des rétroactions positives ou négatives pendant qu'il travaille sur la façon d'exécuter la tâche. Le programmeur définit les règles pour les récompenses, mais laisse à l'algorithme la liberté de déterminer les étapes à suivre pour maximiser la récompense et accomplir la tâche [6],[7].

3.1.5. Apprentissage par transfert Learning

L'apprentissage par transfert (en anglais transfer learning, TL) nous commençons par former un réseau de base sur un ensemble de données et une tâche de base, puis nous réutilisons ou transférons les fonctionnalités apprises vers un second réseau cible, qui sera entraîné sur un ensemble de données et une tâche cibles distincts. Ce processus réussit généralement lorsque les fonctionnalités sont générales, adaptées à la fois aux tâches de base et cibles, plutôt que spécifiques à la tâche de base [w1].

3.2. Techniques d'apprentissage automatique

Parmi les nombreuses techniques d'apprentissage automatique, certaines se distinguent par leur efficacité et leur utilisation répandue :

3.2.1. K-Nearest Neighbors

Les K plus proches voisins (en anglais K-Nearest Neighbors, KNN) une méthode de Machine Learning supervisée, se révèle être une approche simple mais efficace pour la

classification et la régression. Cette méthode repose sur la similarité entre les données, mesurée à l'aide de diverses techniques telles que la distance euclidienne, de Chebyshev ou encore cosinus. Ce qui distingue KNN, c'est sa rapidité, car il ne nécessite pas de phase de formation préalable pour générer des prédictions. Cependant, sa performance diminue lorsque la taille des données augmente, ce qui constitue son principal inconvénient. De plus, les données de grande dimension peuvent compliquer les calculs [8].

3.2.2. Arbre de décision

L'arbre de décision (en anglais Decision Tree, DT) une méthode de Machine Learning supervisée, est utilisé pour résoudre des problèmes de classification et de régression.

Cet algorithme crée un modèle à partir des données d'entraînement, utilisant des règles de décision simples pour prédire la classe ou la valeur d'une variable cible. La prédiction commence par le nœud racine de l'arbre et progresse jusqu'au nœud terminal. Contrairement à d'autres méthodes, l'arbre de décision nécessite moins de préparation des données lors du prétraitement, et la normalisation des données n'est pas requise. Cependant, de petites variations dans les données peuvent entraîner des changements significatifs dans la structure de l'arbre, ce qui peut rendre le modèle instable. De plus, l'entraînement d'un arbre de décision peut être chronophage et parfois le calcul devient très complexe [9].

3.2.3. Forêt aléatoire

La forêt aléatoire (en anglais Random Forest, RF) est une méthode flexible et largement utilisée en apprentissage supervisé, adaptée à la fois à la classification et à la régression [10]. Pour assurer des estimations stables et précises, RF construit plusieurs arbres de décision à l'aide du bagging, puis les combine. Ses paramètres par défaut sont souvent efficaces pour produire des prédictions précises tout en restant facilement interprétables. Cependant, RF peut être lent et inefficace dans des scénarios nécessitant une classification en temps réel avec un grand nombre d'arbres.

3.2.4. Naive Bayes

L'algorithme Naïve Bayes (NB), basé sur le théorème de Bayes, est prisé pour sa simplicité de construction et son efficacité, surtout avec de vastes ensembles de données [11]. Contrairement aux variables numériques, il s'adapte bien aux variables catégorielles. Il offre diverses variantes comme le Naïve Bayes gaussien (GNB), multinomial (MNB), de Bernoulli (BNB), entre autres. Un défi majeur réside dans sa tendance à attribuer une probabilité nulle aux variables catégorielles absentes dans l'ensemble d'entraînement, rendant la prédiction impossible pour ces variables dans l'ensemble de test. Toutefois, des

techniques de lissage peuvent être employées pour pallier ce problème de fréquence zéro. Malgré ses avantages, il est important de souligner que le Naïve Bayes est généralement considéré comme un estimateur peu fiable, constituant ainsi une limite [12].

3.2.5. Xgboost

Xgboost (eXtreme Gradient Boosting) est une technique de Machine Learning très populaire chez les Data Scientists. Il s'agit d'un modèle amélioré de l'algorithme d'amplification de gradient (Gradient Boost). Cette technique d'apprentissage machine est également utilisé pour résoudre les problématiques courantes d'entreprises tout en se basant sur une quantité minimale de ressources.

En Machine Learning, le boosting de gradient extrême est une méthode qui est utilisée pour réduire le nombre d'erreurs dans l'analyse prédictive des données. XGBoost est un assemblage d'arbres décisionnels (weak learners) qui prédisent les résidus et corrige les erreurs des arbres décisionnels précédents. La particularité de cet algorithme réside dans l'arbre décisionnel utilisé [w2].

4. L'apprentissage profond

L'apprentissage en profondeur (en anglais deep learning, DL) représente un ensemble d'algorithmes d'apprentissage automatique qui s'efforcent d'apprendre à plusieurs niveaux, correspondant à divers niveaux d'abstraction [13]. Cette approche est capable d'extraire des caractéristiques à partir des données brutes en utilisant des couches de traitement multiples, composées de transformations linéaires et non linéaires. Ce processus permet à l'algorithme d'apprendre progressivement sur ces caractéristiques à travers chaque couche, avec une intervention humaine minimale.

4.1. La différence entre machine learning et deep learning

L'apprentissage automatique et l'apprentissage profond simulent tous deux le processus d'apprentissage du cerveau humain. Leur principale distinction réside dans les types d'algorithmes employés, bien que l'apprentissage profond soit plus étroitement lié à l'apprentissage humain en raison de son utilisation de neurones. En général, l'apprentissage automatique utilise des techniques telles que les arbres de décision et les réseaux de neurones peu profonds, tandis que l'apprentissage profond fait appel à des réseaux de neurones profonds, plus complexes. De plus, les deux approches peuvent être mises en œuvre de manière supervisée ou non supervisée [w3].(Voir tableau 1.1)

	Machine Learning	**Deep Learning**
Ressource de calcul et de stockage	Moins	Plus
Base de donnees	Contrôlable	>1 million de donnees
Champ d'application	Actions simples de routine	Tache complexes
Extraction des caractéristiques	Manuelle	Automatique
Complexité	Moins	Plus complexe

Tableau 1.1: Comparaison entre ML et DP

4.2. Réseaux neuronaux artificiels

Réseaux neuronaux artificiels (en anglais Artificial Neural Network, ANN) sont des structures de graphes orientés étiquetés, où chaque nœud effectue un calcul élémentaire. Conformément à la théorie des graphes, un graphe orienté est constitué d'un ensemble de nœuds (ou sommets) et d'un ensemble de connexions (ou arêtes) reliant ces nœuds entre eux. Chaque nœud dans un tel graphe exécute un calcul simple, et chaque connexion transporte le résultat de ce calcul d'un nœud à un autre, portant un poids qui indique l'amplification ou la réduction du signal. Certains poids de connexion sont significativement positifs, soulignant l'importance du signal dans la classification, tandis que d'autres sont négatifs, réduisant l'impact du signal sur la classification finale. Un système est considéré comme un réseau neuronal artificiel lorsqu'il se présente sous la forme d'une structure de graphe avec des poids de connexion modifiables, ajustés par un algorithme d'apprentissage [14].

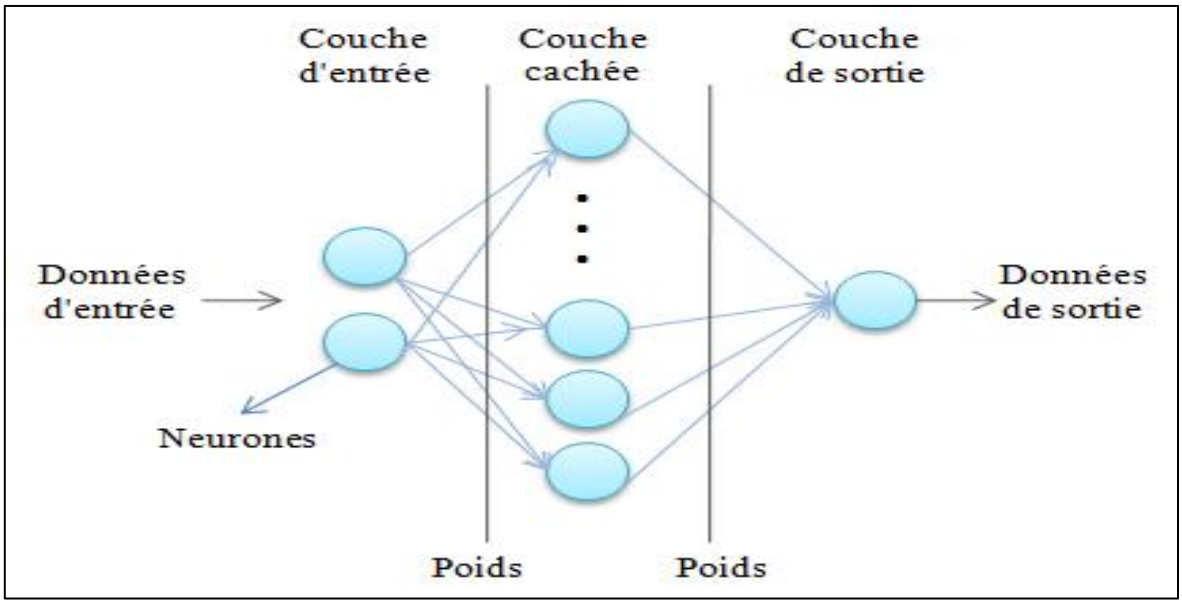

Figure 1.3 : Architecture de réseau neuronal simple [w4]

4.2.1. Réseaux des neurones convolutifs

Les Réseaux des neurones convolutifs (en anglais Convolutional Neural Networks, CNN) sont un type de réseau de neurones spécialisés dans le traitement de données ayant une topologie semblable à une grille. qui s'est sont avérés très efficaces dans des domaines tells que la reconnaissance et la classification d'images et vidéos. CNN a réussi à identifier les visages, les objets, panneaux de circulation et auto-conduite des voitures. Récemment, les CNN ont été efficaces dans plusieurs tâches de traitement du langage naturel (telles que la classification des phrases). Dans le ML, un réseau convolutif est un type de réseau de neurones feed-forward, il a été inspiré par des processus biologiques [15].

4.2.2. Réseaux des neurons récurrents

Les réseaux de neurones récurrents (en anglais Recurrent Neural Networks, RNN) sont conçus pour utiliser des informations séquentielles, contrairement aux réseaux neuronaux traditionnels qui supposent que toutes les entrées (et les sorties) sont indépendantes les unes des autres. Cette indépendance supposée n'est pas adaptée à de nombreuses tâches, notamment lorsqu'il s'agit de prédire le prochain mot dans une phrase. Dans ce cas, il est essentiel de connaître les mots qui précèdent pour faire une prédiction précise.

Les RNN sont appelés récurrents parce qu'ils exécutent la même tâche pour chaque élément d'une séquence, où chaque sortie dépend des calculs précédents. Une autre façon de concevoir les RNN est de les imaginer avec une "mémoire" qui capture l'information sur ce qui a été calculé jusqu'ici. En théorie, les RNN peuvent utiliser des informations dans

des séquences arbitrairement longues. Cependant, en pratique, ils sont souvent limités à prendre en compte seulement quelques étapes en arrière en raison de contraintes computationnelles et de problèmes tels que la dissipation ou l'explosion des gradients [16].

5. Réseaux Auto-encodeurs

Les auto-encodeurs (en anglais Auto-Encoder, AE) sont des réseaux de neurones particuliers ayant le même nombre de neurones dans leur couche d'entrée et leur couche de sortie. Leur objectif est de produire une sortie aussi proche que possible de l'entrée. L'apprentissage est auto-supervisé, car la perte à minimiser est le coût de reconstruction entre la sortie et l'entrée.

Les données n'ont pas besoin d'être labellisées, car elles servent de leurs propres labels, ce qui fait de ce modèle un modèle non supervisé.

5.1. Architecture d'un auto-encodeur

L'auto-encodeur se compose de deux parties principales : un réseau d'encodeurs et un réseau de décodeurs. Pendant la formation, les encodeurs et les décodeurs sont utilisés ensemble. Comme toute architecture de DL, l'AE fonctionne par couches de neurones et est entraîné par rétro-propagation.

Les couches sont divisées en couches d'encodeur et de décodeur distinctes. L'entrée est connectée à la première couche de l'encodeur. Dans chaque couche successive de l'encodeur, le nombre de neurones est réduit jusqu'à atteindre la couche codée finale, qui a le moins de neurones et représente les caractéristiques du goulot d'étranglement. Autrement dit, les entrées sont transformées couche par couche jusqu'à ce que cette couche codée représente une version dimensionnellement réduite des données, capable de capturer les caractéristiques les plus importantes. Après cette couche codée, les couches du décodeur sont définies, où le nombre de neurones augmente dans chaque couche jusqu'à ce que la couche de sortie ait le même nombre de neurones que la couche d'entrée [17] .

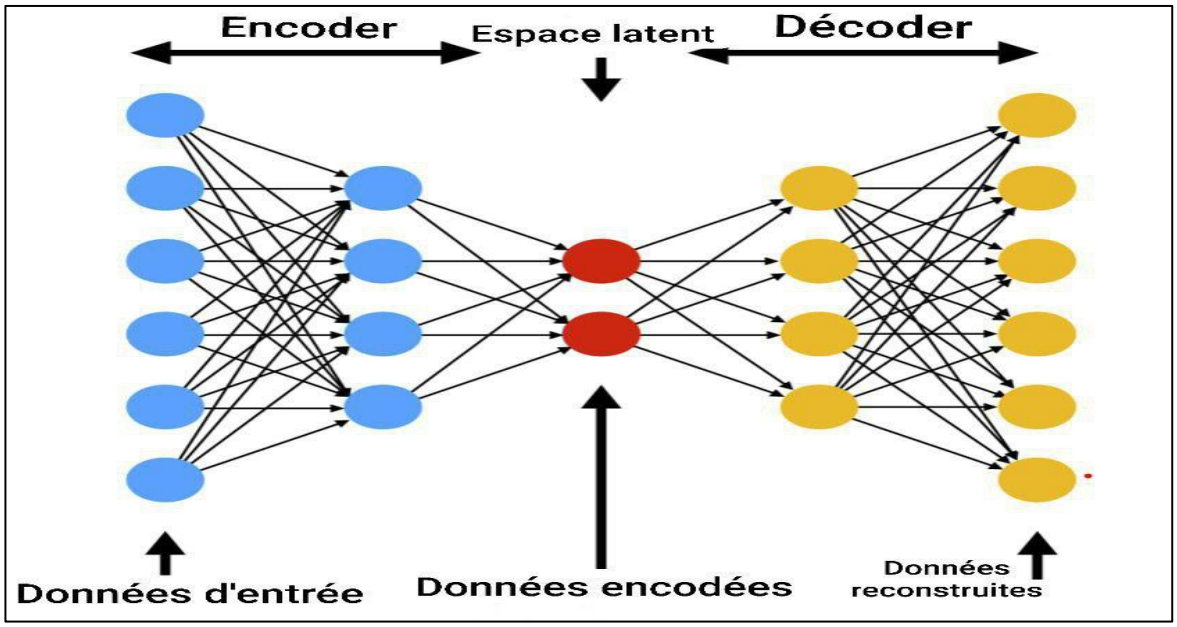

Figure 1.4 : Architecture d'un auto-encodeur [17]

5.2. Apprentissage des auto-encodeurs

Nous entraînons un réseau de neurones monocouche et la valeur cachée h est prédite par la sortie z de l'entrée x. Il s'agit d'une tâche appelée reconstruction (voir figure 1.5).

Par conséquent, le critère est de minimiser l'erreur de reconstruction L(x;z).

Ainsi, la couche cachée doit contenir des informations relatives à la reconstruction. Par exemple : si la couche cachée contient moins d'unités que l'entrée, pour reconstruire correctement, elle doit apprendre à résumer l'entrée, car toutes les informations nécessaires sont contenues dans la couche cachée. En conséquence, des caractéristiques pertinentes pour la, reconstruction doivent être extraites.

Quatre hyper-paramètres doivent être définis avant l'apprentissage d'un auto-encodeur : [w5]

5.2.1. Taille du code

La taille du code ou la taille du goulot d'étranglement est le hyper-paramètre le plus important utilisé pour ajuster l'auto-encodeur. La taille du goulot d'étranglement décide de la quantité de compression des données. Cela peut également agir comme un terme de régularisation.

5.2.2. Nombre de couches

Comme pour tous les réseaux neuronaux, un hyper-paramètre important pour ajuster les auto-encodeurs est la profondeur de l'encodeur et du décodeur. Alors qu'une profondeur plus élevée augmente la complexité du modèle, une profondeur plus faible permet un

traitement plus rapide.

5.2.3. Nombre de nœuds par couche

Le nombre de nœuds par couche définit les poids que nous utilisons par couche. En général, le nombre de nœuds diminue avec chaque couche successive dans l'auto-encodeur, car l'entrée de chaque couche devient plus petite à travers les couches.

5.2.4. Perte de reconstruction

La fonction de perte que nous utilisons pour entraîner l'auto-encodeur dépend du type d'entrée et de sortie que nous voulons que l'auto-encodeur adapte. Les fonctions de perte les plus populaires pour la reconstruction de données d'image sont la perte quadratique moyenne (MSE) et la perte L1. Si les entrées et les sorties sont dans la plage [0,1], comme dans MNIST (base de données de chiffres manuscrits), l'entropie croisée binaire peut également être utilisée comme perte de reconstruction.

5.3. Utilisation des auto-encodeurs

Les auto-encodeurs sont principalement utilisés pour exploiter l'encodeur et l'espace latent dimensionnel pour diverses applications, telles que l'annotation d'images et la détection d'anomalies. Le décodeur sert généralement à visualiser la qualité des sorties reconstruites. Les auto-encodeurs peuvent être vus comme une version plus avancée des techniques de réduction de la dimensionnalité, telles que l'analyse en composantes principales (PCA).

5.4. Types d'auto-encodeurs

Les auto-encodeurs codent les valeurs d'entrée x à l'aide d'une fonction f. Décode ensuite les valeurs codées f(x) à l'aide d'une fonction «g» pour créer des valeurs de sortie identiques à l'entrée valeurs.Il existe de nombreux types d'encodeurs automatiques, allant des auto-encodeurs simples aux auto-encodeurs à débruitage.

5.4.1. Auto-encodeurs sous-complet

Les auto-encodeurs sous-complets (en anglais Undercomplete Autoencoders) figurent parmi les types les plus rudimentaires d'auto-encodeurs. Ils se caractérisent par une dimension du code inférieure à celle de l'entrée. En apprenant une représentation sous-complète, l'auto-encodeur est contraint de saisir les aspects les plus significatifs des données d'entraînement. Le principe de fonctionnement est assez simple :

- l'AE sous-complet prend en entrée une image et vise à prédire la même image en sortie, reconstruisant ainsi l'image à partir de la zone compressée du code, également appelée goulot d'étranglement [18].

L'objectif est de minimiser la fonction de perte en pénalisant le g(f(x)) parce qu'il est

différent à partir de l'entrée x.

$$L = |x - x^{\wedge}| \text{.....(1)}$$

$$L = |x - g(f(x))| \text{.....(2)}$$

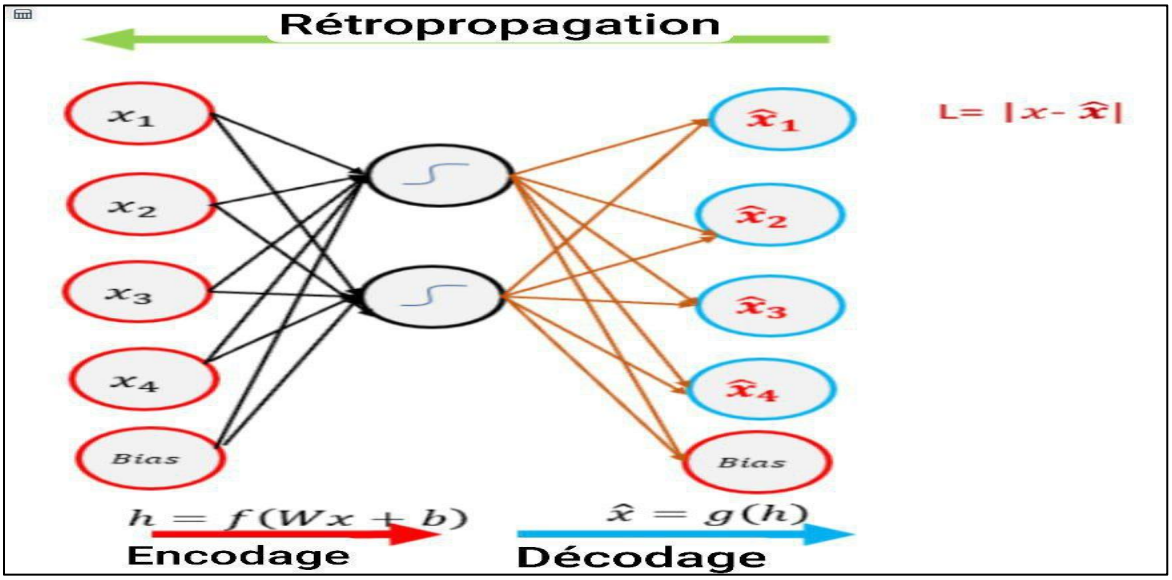

Figure 1.5 : auto-encodeur sous-complet [18]

5.4.2. Auto-encodeurs parcimonieux

Les auto-encodeurs parcimonieux (en anglais Sparse Autoencoders) sont similaires aux auto-encodeurs sous-complets en ce qu'ils utilisent la même image en entrée et en vérité terrain. Cependant, les moyens par lesquels le codage de l'information est régulé diffèrent considérablement.L'auto-encodeur parcimonieux est régulé en modifiant le nombre de nœuds à chaque couche cachée [18].

Un AE parcimonieux est caractérisé par un nombre de couches cachées supérieur à celui de la couche d'entrée dans une approche de généralisation visant à réduire le surajustement. Cela limite la fonction de perte et empêche l'auto-encodeur de surutiliser tous ses nœuds [19].

Les AEs parcimonieux sont soumis à une pénalité de parcimonie (h), une valeur proche de zéro mais non nulle. Cette pénalité de parcimonie est appliquée sur la couche cachée en plus de l'erreur de reconstruction, ce qui prévient le surajustement .

$$L = |x - g(f(x))| + \Omega(h) \text{.....(3)}$$

Les AEs parcimonieux sélectionnent les valeurs d'activation les plus élevées dans la couche cachée et mettent à zéro le reste des nœuds cachés. Cela empêche les auto-encodeurs d'utiliser tous les nœuds cachés à la fois et force uniquement un nombre réduit de nœuds cachés à être utilisés. Nous activons et désactivons les nœuds cachés pour chaque ligne

dans l'ensemble de données. Chaque nœud caché extrait une caractéristique des données [20].

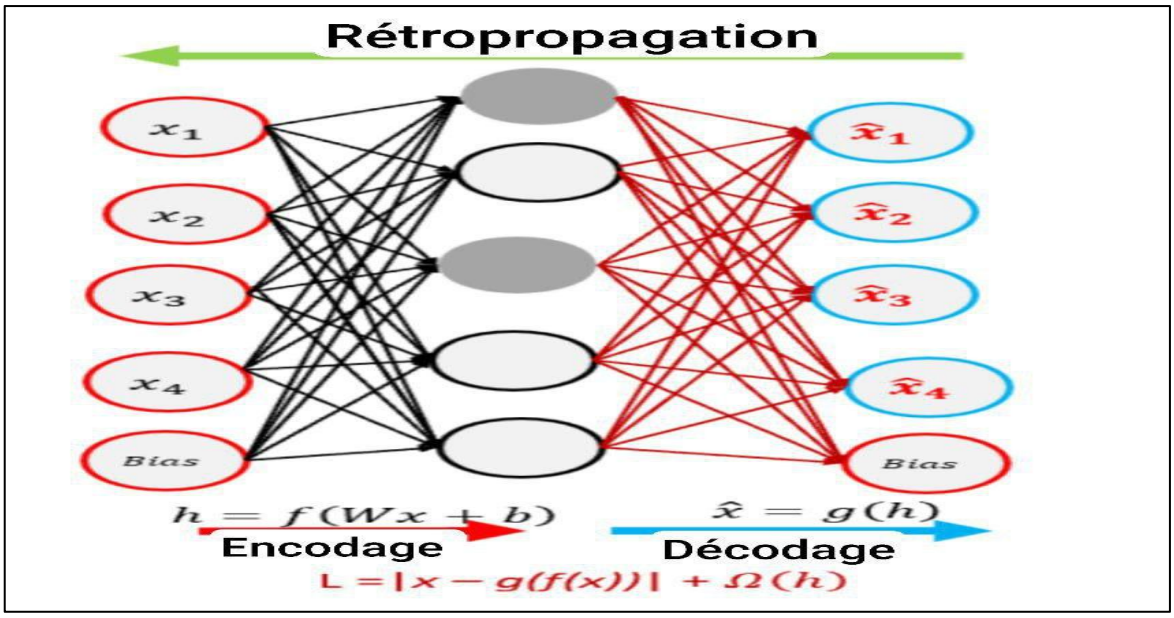

Figure 1.6 : Les auto-encodeurs parcimonieux [20]

5.4.3. Auto-encodeurs contractifs

Les AEs contractifs (en anglais Contractive Autoencoders) effectuent la tâche d'apprentissage d'une représentation de l'image tout en la faisant passer par un goulot d'étranglement et en la reconstruisant dans le décodeur.

L'AE contractif possède également un terme de régularisation pour empêcher le réseau d'apprendre la fonction identité et de mapper l'entrée sur la sortie.

Les AEs contractifs fonctionnent sur la base que des entrées similaires devraient avoir un encodage similaire et une représentation spatiale latente similaire. Cela signifie que l'espace latent ne doit pas varier considérablement pour de légères variations de l'entrée [21].

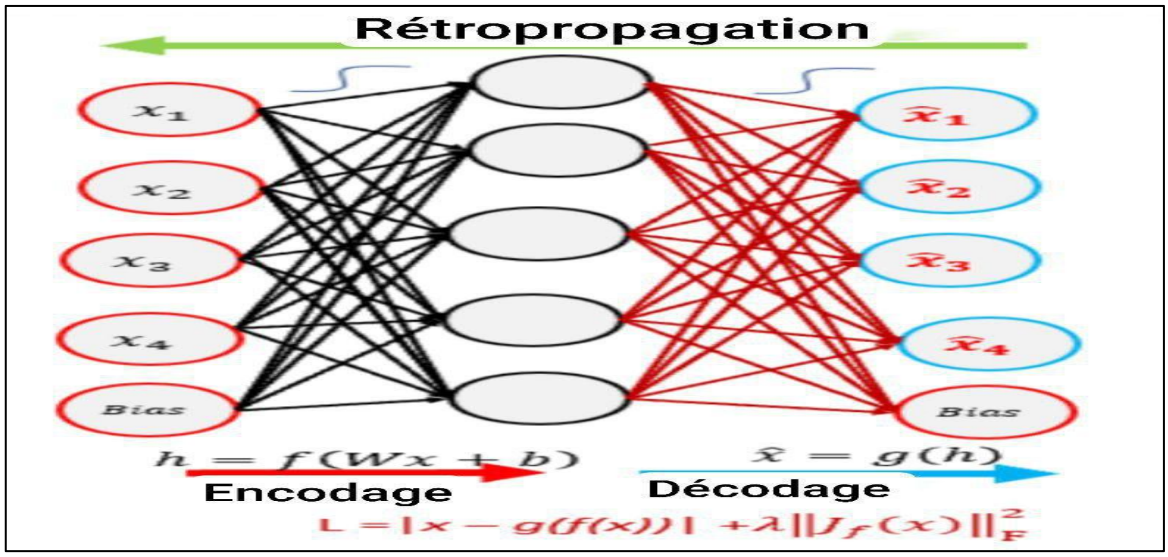

Figure 1.7 : Auto-encodeurs contractifs [21]

5.4.4. Auto-encodeurs débruitages

L'AE débruitage (en anglais Denoising Autoencoders) est un auto-encodeur qui reçoit un point de données corrompu en entrée et est entraîné à prédire le point de données original et non corrompu en sortie. Les auto-encodeurs qui éliminent le bruit d'une image.

Dans les auto-encodeurs de débruitage, ils alimentent une version bruitée de l'image, où du bruit a été ajouté via des altérations numériques. L'image bruitée est envoyée à l'architecture encodeur-décodeur, et la sortie est comparée à l'image de référence [20].

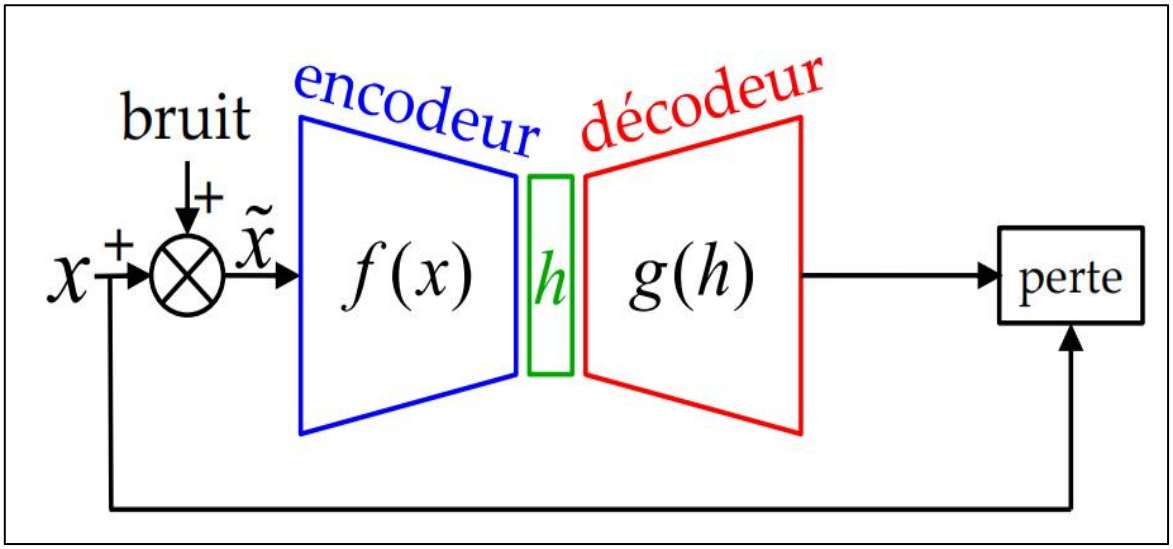

Figure 1.8 : Auto-encodeurs de débruitage [20]

5.4.5. Auto-encodeurs variationnels

L'AE variationnel (en anglais Variational Autoencoder, VAE) peut être défini comme un auto-encodeur dont l'entraînement est régularisé pour éviter le surajustement et garantir

que l'espace latent possède de bonnes propriétés permettant un processus génératif. Le VAE peut être vu comme deux modèles couplés, mais paramétrés indépendamment : le modèle d'encodeur ou de reconnaissance, et le modèle de décodeur ou génératif. Ces deux modèles se soutiennent mutuellement. Le modèle de reconnaissance fournit au modèle génératif une approximation de sa distribution a posteriori sur les variables aléatoires latentes, dont il a besoin pour mettre à jour ses paramètres à l'intérieur d'une itération d'apprentissage par "espérance-maximisation". Inversement, le modèle génératif est une sorte d'échafaudage pour que le modèle de reconnaissance apprenne des représentations significatives des données, y compris éventuellement des étiquettes de classe. Le modèle de reconnaissance est l'inverse approximatif du modèle génératif selon la règle de Bayes [22].

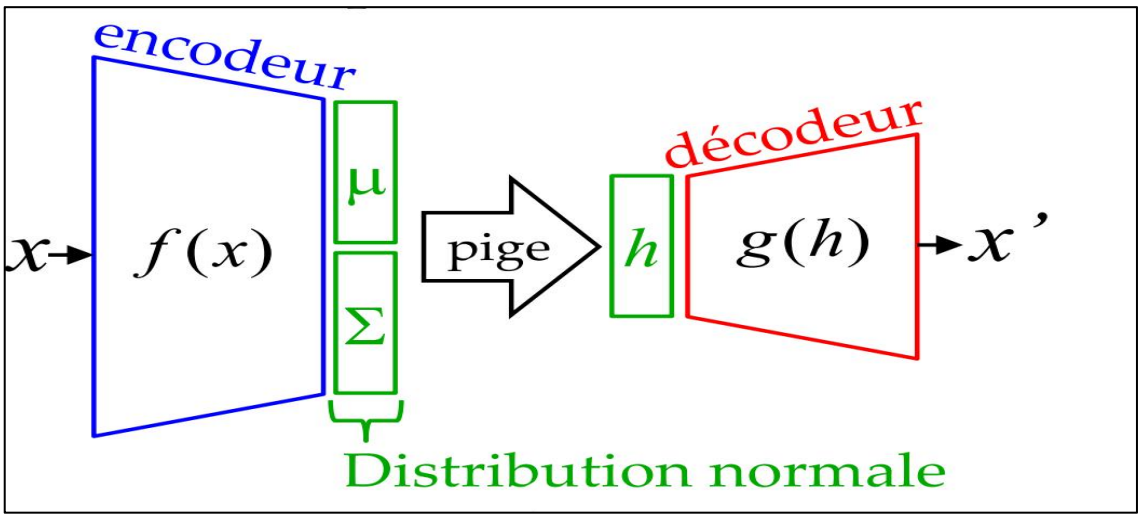

Figure 1.9 : Architecture d'encodeur automatique variationnel [22]

5.4.6. Auto-encodeurs convolutionnels profonds

L'AE convolutionnel (en anglais Deep Convolutional Autoencoders) est une variante des réseaux de neurones convolutionnels utilisés comme outils pour l'apprentissage non supervisé des filtres de convolution. Ils sont généralement appliqués à la tâche de reconstruction d'images afin de minimiser les erreurs de reconstruction en apprenant les filtres optimaux. Une fois qu'ils sont entraînés dans cette tâche, ils peuvent être appliqués à n'importe quelle entrée pour extraire des caractéristiques.

Les AEs convolutionnels sont des extracteurs de caractéristiques polyvalents, contrairement aux AE généraux qui ignorent complètement la structure d'image 2D. Dans les auto-encodeurs, l'image doit être déroulée en un seul vecteur et le réseau doit être construit en suivant la contrainte sur le nombre d'entrées [w6].

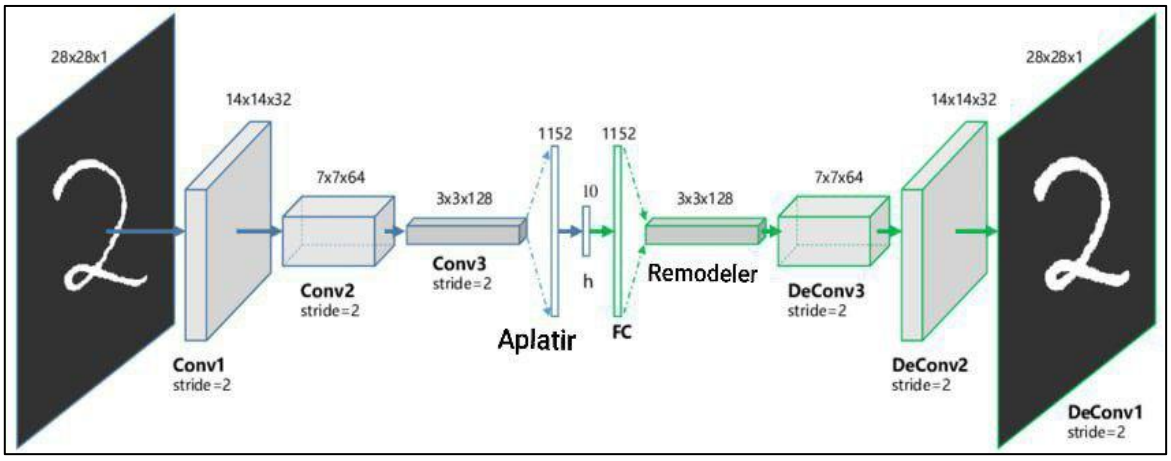

Figure 1.10 : La structure de Convolutional AutoEncoder [w6]

5.5. Fonctionnement d'un auto-encodeur

Le fonctionnement d'un AE peut se décomposer en plusieurs étapes :

- Préparation des données : Les données d'entrée sont préparées sous une forme appropriée pour être utilisées dans le réseau de neurones. Cela peut inclure la normalisation, la mise en forme et la division en jeux de données d'entraînement et de test.
- Définition de l'architecture de l'auto-encodeur : L'architecture de l'auto-encodeur est définie en spécifiant le nombre de couches et le nombre de neurones dans chaque couche. Les couches d'entrée et de sortie ont le même nombre de neurones, tandis que les couches cachées ont généralement un nombre de neurones inférieur, ce qui permet de comprimer les données dans un espace de dimensionnalité latent.
- Formation de l'auto-encodeur : L'AE est formé en entraînant le réseau de neurones sur les données d'entraînement. L'objectif de la formation est de minimiser la perte de reconstruction entre les données d'entrée et les données reconstruites à partir de l'espace de dimensionnalité latent.
- Encodage des données : Une fois l'AE formé, il peut être utilisé pour encoder les données dans un espace de dimensionnalité latent. Cela est fait en passant les données d'entrée à travers l'encodeur du réseau de neurones.
- Décodage des données : Les données encodées peuvent être décodées à partir de l'espace de dimensionnalité latent en passant les données encodées à travers le décodeur du réseau de neurones.
- Utilisation de l'auto-encodeur : L'AE peut être utilisé pour une variété de tâches, telles que la réduction de dimensionnalité, la détection d'anomalies, la génération de données et l'apprentissage de représentations de caractéristiques.

6. Conclusion

Nous avons consacré ce chapitre à la présentation des notions de base telles que l'IA, ainsi que les concepts fondamentaux de l'apprentissage automatique. Nous avons d'abord défini l'ML et présenté ses différents types afin de fournir une vision globale de ce domaine. Ensuite, nous avons expliqué les algorithmes en détail, en mettant l'accent sur certains algorithmes clés.

L'DL fait référence à une classe de techniques d'ML qui utilisent des algorithmes pour imiter le fonctionnement du cerveau humain. Nous avons également présenté la théorie nécessaire pour comprendre les réseaux de neurones, en particulier les auto-encodeurs.

Dans le prochain chapitre, nous présenterons le système de détection d'intrusion.

Chapitre 2 :
Les Systèmes de Détection d'Intrusions

1. Introduction

Les Systèmes de Détection d'Intrusions (en anglais Intrusion Detection System, IDS) sont des systèmes logiciels ou matériels qui automatisent le processus de surveillance des événements se produisant dans un système informatique ou un réseau en effectuant une certaine analyse pour détecter les signes de problèmes de sécurité. Comme les attaques de réseau ont augmenté en nombre et en gravité au cours des dernières années, les IDSs sont devenus nécessaires pour assurer la sécurité de la plupart des organisations. En effet, les IDSs sont considérés comme une des lignes de défense efficaces contre les attaques réseau dirigées contre les systèmes informatiques.

Dans ce chapitre nous présentons tout d'abord la notion d'IDS ainsi que son architecture. Nous présentons également la classification de ce dernier, dans ce cadre plusieurs critères sont pris en compte Nous commençons par la classification selon la méthode d'analyse qui découpe les IDS en deux approches (comportementale et par signatures), enfin nous allons mettre le point sur la détection d'intrusions réseau.

2. Définition

La détection des intrusions est le processus de surveillance des événements qui se trouvent dans un système des ordinateurs ou du réseau et les analysant pour détecter les signes des intrusions, défini comme des tentatives pour compromettre la confidentialité, l'intégrité, la disponibilité ou éviter des mécanismes de sécurité de l'ordinateur ou du réseau. L'intrusion est causée par les attaques accédant au système via Internet, autorisée l'utilisateur du système qui essaye de gagner les privilèges supplémentaires pour lesquels ils n'ont pas été autorisés, et autorisé les utilisateurs qui abusent les privilèges donnés. Le système de détection des intrusions est un logiciel ou un matériel qui automatise des surveillances et les processus analysés [23].

3. Présentation d'un Systèmes de Détection d'Intrusions

Les IDS protègent un système contre les attaques, les mauvaises utilisations et les compromis. Ils peuvent également surveiller l'activité du réseau, analyser les configurations du système et du réseau contre toute vulnérabilité, analyser l'intégrité de données et bien plus.

Un IDS a quatre fonctions principales : l'analyse, la journalisation, la gestion et l'action.

- Analyse : Analyse des journaux du système pour identifier des intentions dans la masse de données recueillie par l'IDS.
- Journalisation : Enregistrement des évènements dans un fichier de log.

- Gestion : Les IDS doivent être administrés de manière permanente. On peut assimiler un IDS à une caméra de sécurité.

- Action : Alerter l'administrateur quand une attaque dangereuse est détectée.

3.1. Architecture d'un Systèmes de Détection d'Intrusions [24]

Plusieurs schémas ont été proposés pour décrire les composants d'un d'IDS. Parmi eux, nous avons retenu celui issu des travaux d'Intrusions detection exchange formatWorking Group (IDWG) de l'Internet Engineering Task Force (IETF) comme base de départ, car il résulte d'un large consensus parmi les intervenants du domaine.

L'objectif des travaux du groupe IDWG est la définition d'un standard de communication entre certains composants d'un système de détection d'intrusions. La figure 2.1 illustre ce modèle et permet d'introduire un certain nombre de concepts :

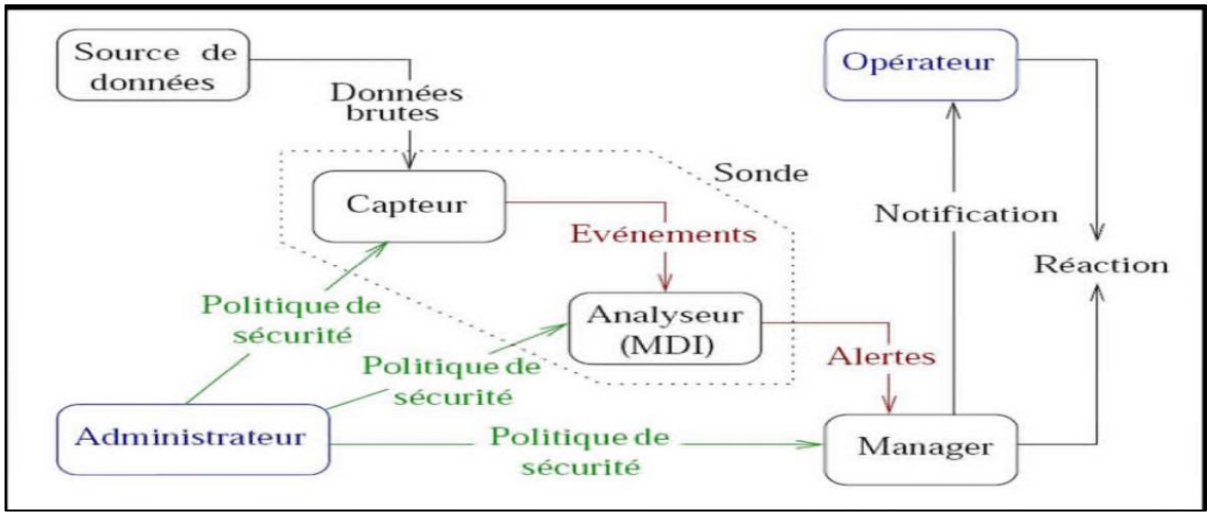

Figure 2.1 : Modèle générique de la détection d'intrusions proposé par l'IDWG

3.1.1. Les différents éléments de cette architecture

- Administrateur : Personne chargée de mettre en place la politique de sécurité et par conséquent, de déployer et configurer les IDS.

- Alerte : Message formaté émis par un analyseur s'il trouve des activités intrusives dans une source de données.

- Analyseur : C'est un outil logiciel qui met en œuvre l'approche choisie pour la détection (comportementale ou par scénarios), il génère des alertes lorsqu'il détecte une intrusion.

- Capteur : Logiciel générant des événements en filtrant et formatant les données brutes provenant d'une source de données.

- Événement : Message formaté et renvoyé par un capteur. C'est l'unité élémentaire utilisée pour représenter une étape d'un scénario d'attaques connu.

- Manager : Composant d'un IDS permettant à l'opérateur de configurer les différents

éléments d'une sonde et de gérer les alertes reçues et éventuellement la réaction.

- Notification : La méthode par laquelle le manager d'IDS met au courant l'opérateur de l'occurrence d'alerte.
- Opérateur : Personne chargée de l'utilisation du manager associé à l'IDS. Elle propose ou décide de la réaction à apporter en cas d'alerte. C'est, parfois, la même personne que l'administrateur.
- Réaction : Mesures passive ou actives prises en réponse à la détection d'une attaque, pour la stopper ou pour corriger ses effets.
- Sonde : Un ou des capteurs couplés avec un analyseur.
- Source de données : Dispositif générant de l'information sur les activités des entités du système d'information [25].

3.2. Vocabulaire de la détection d'intrusions

La détection d'intrusions utilise une vocabulaire bien définis qui ne se trouve pas dans le modèle récédent et qui est comme suit:

- Attaque ou intrusion : Action qui permet de violer la politique de sécurité.
- Audit de sécurité : C'est l'ensemble des mécanismes permettant la collecte d'informations sur les actions faites sur un système d'information.
- Détection d'intrusions : recherche de traces laissées par une intrusion dans les données produites par une source.
- Faux positif : Alerte en l'absence d'attaque.
- Faux négatif : Absence d'alerte en présence d'attaque.
- Vulnérabilité : Faille de conception, d'implémentation ou de configuration d'un système logiciel ou matériel.
- Log (trace d'audit) : C'est un fichier système à analyser.
- Exploit : Terme utilisé pour désigner un programme d'attaque.
- Scénario : Suite constituée des étapes élémentaires d'une attaque.
- Signature : Suites des étapes observables d'une attaque, utilisée par certains analyseurs pour rechercher dans les activités des entités, des traces de scénarios d'attaques connus.
- Système de détection d'intrusions: Ensemble constitué d'un ou plusieurs capteurs, un ou plusieurs analyseurs et un ou plusieurs managers.
- Corrélation : C'est l'interprétation conceptuelle de plusieurs événements visant à leur assigner une meilleure sémantique et à réduire la quantité globale d'événements [26].

3.3. Caractéristiques d'un système de détection d'intrusions

Les caractéristiques suivantes sont souhaitables dans un IDS :

- Fonctionnement en permanence avec une supervision manuelle minimale.
- Être tolérant aux pannes dans le sens où il doit récupérer après une défaillance ou une réinitialisation de la machine.
- Résister aux tentatives de corruption, c'est-à-dire, il doit pouvoir détecter s'il a subit lui-même une modification indésirable.
- Utiliser un minimum de ressources pour implémenter une politique de sécurité spécifique d'un réseau.
- Être facilement configurable pour implémenter une politique de sécurité spécifique d'un réseau.
- S'adapter au cours du temps aux changements du système surveillé et du comportement des utilisateurs.
- Être difficile à tromper.

Comme la taille des réseaux a tendance à croître, on peut ajouter les caractéristiques suivantes :

- Être scalable.
- Être robuste, c'est-à-dire l'arrêt d'un composant ne doit pas entraîner une défaillance totale.

3.4. Emplacement d'un système de détection d'intrusions [27]

Il existe plusieurs endroits stratégiques où il convient de placer un IDS.

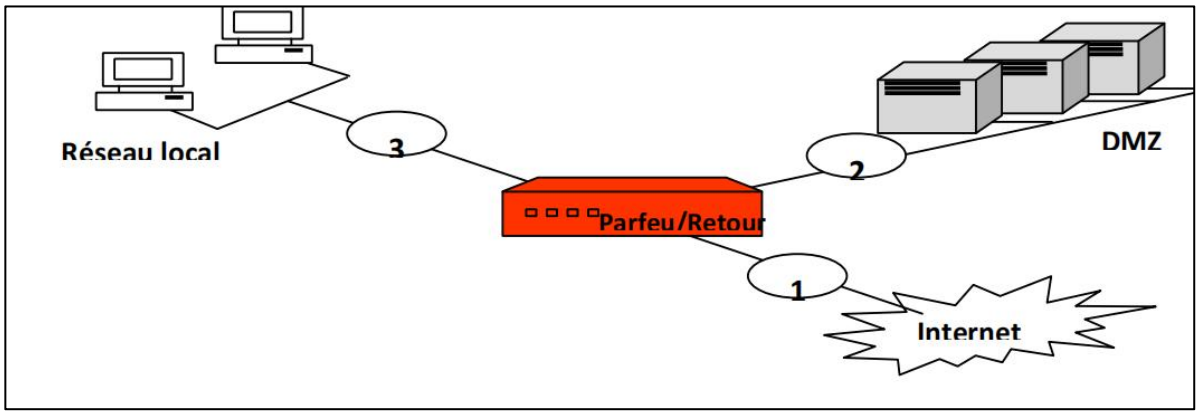

Figure 2.2 : Endroits typiques pour un système de détection d'intrusions

Position (1) : Sur cette position, l'IDS va pouvoir détecter l'ensemble des attaques frontales, provenant de l'extérieur, en amont du firewall. Ainsi, beaucoup (trop?) d'alertes seront remontées ce qui rendra les logs difficilement consultables.

Position (2) : Si l'IDS est placé sur la DMZ, il détectera les attaques qui n'ont pas été

filtrées par le firewall et qui relèvent d'un certain niveau de compétence. Les logs seront ici plus clairs à consulter puisque les attaques bénins ne seront pas recensées.

Position (3) : L'IDS peut ici rendre compte des attaques internes, provenant du réseau local de l'entreprise. Il peut être judicieux d'en placer un à cet endroit étant donné le fait que 80% des attaques proviennent de l'intérieur. De plus, si des trojans ont contaminé le parc informatique (navigation peu méfiante sur internet)

3.5. Classification des systèmes de détection d'intrusions [25]

Nous pouvons classifier les systèmes de détection d'intrusions selon cinq critères :

- La source des données à analyser.
- Le lieu de l'analyse des données.
- La fréquence de l'analyse.
- Le comportement en cas d'attaque détectée.
- La méthode de détection utilisée.

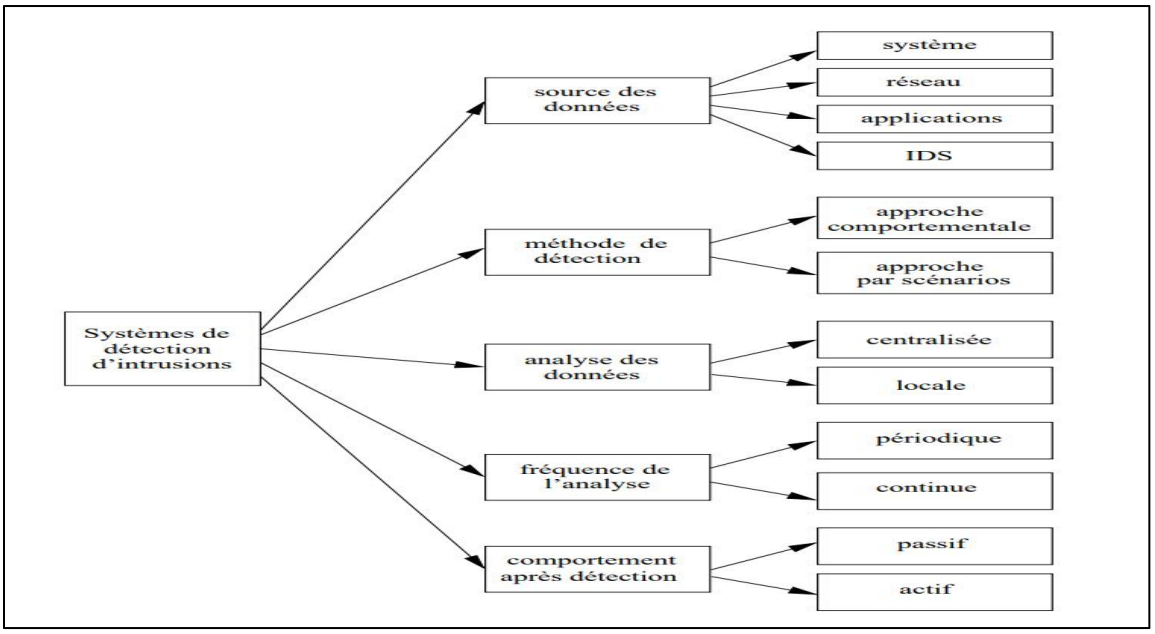

Figure 2.3 : Classification des systèmes de détection d'intrusions

3.5.1. Source des données à analyser

Les sources possibles de données à analyser sont une caractéristique essentielle des systèmes de détection d'intrusions puisque ces données constituent la matière première du processus de détection. Les données proviennent soit de logs générés par le système d'exploitation, soit de logs applicatifs, soit d'informations provenant du réseau, soit encore d'alertes générées par d'autres IDS.

A. Source d'information système

Un système d'exploitation fournit généralement plusieurs sources d'information :

- Commandes systèmes : Presque tous les systèmes d'exploitation fournissent des commandes pour avoir un «instantané» de ce qui se passe.

- Accounting : L'accounting fournit de l'information sur l'usage des ressources partagées par les utilisateurs .

-Audit de sécurité : Tous les systèmes d'exploitation modernes proposent ce service pour fournir des événements système, les associer à des utilisateurs et assurer leur collecte dans un fichier d'audit.

- Les utilisateurs : Accès en lecture à un fichier, exécution d'une application, etc.

B. Source d'information réseau

Des dispositifs matériels ou logiciels (snifer) permettent de capturer le trafic réseau. Cette source d'information est particulièrement adaptée lorsqu'il s'agit de rechercher les attaques en déni de service qui se passent au niveau réseau ou les tentatives de pénétration à distance. Le processus d'interception des paquets peut être rendu quasiment invisible pour l'attaquant car on peut utiliser une machine dédiée juste reliée à un brin du réseau, configurée pour ne répondre à aucune sollicitation extérieure et dont personne ne soupçonnera l'existence. Néanmoins, il est difficile de garantir l'origine réelle de l'attaque que l'on a détectée car il est facile de masquer son identité en modifiant les paquets réseau.

C.Source d'information applicative

Les applications peuvent également constituer une source d'information pour les IDS.

Les capteurs applicatifs sont de deux natures :

- Capteur interne : Le filtrage sur les activités de l'application est alors exécuté par le code de l'application.

- Capteur externe : Le filtrage se fait à l'extérieur de l'application.

Plusieurs méthodes sont utilisées : un processus externe peut filtrer les logs produits par l'application ou bien l'exécution de l'application peut être interceptée (au niveau de ses appels de librairies ou d'un proxy applicatif) Prendre ses informations directement au niveau de l'application présente plusieurs avantages. Premièrement, les données interceptées ont réellement été reçues par l'application.

Il est donc difficile d'introduire une désynchronisation entre ce que voit passer le capteur applicatif et ce que reçoit l'application contrairement à ce qu'il peut se passer avec les capteurs réseau. Ensuite, cette source d'information est généralement de plus haut niveau que les sources système et réseau. Cela permet donc de filtrer des événements qui ont une

sémantique plus riche. Finalement, si l'on prend l'exemple d'une connexion web chiffrée par SSL, un capteur réseau ne verra passer que des données pseudo-aléatoires tandis qu'un capteur associé au serveur web pourra analyser le texte en clair de la requête.

D. Source d'information basée IDS

Une autre source d'information, souvent de plus haut niveau que les précédentes, peut être exploitée. Il s'agit des alertes remontées par des analyseurs provenant d'un IDS. Chaque alerte synthétise déjà un ou plusieurs événements intéressants du point de vue de la sécurité. Elles peuvent être utilisées par un IDS pour déclencher une analyse plus fine à la suite d'une indication d'attaque potentielle. De surcroît, en corrélant plusieurs alertes, on peut parfois détecter une intrusion complexe de plus haut niveau. Il y aura alors génération d'une nouvelle alerte plus synthétique que l'on qualifie de méta-alerte.

3.5.2. Localisation de l'analyse des données

On peut également faire une distinction entre les IDS en se basant sur la localisation réelle de l'analyse des données :

- Analyse centralisée : Certains IDS ont une architecture multi-capteurs (ou multisondes). Ils centralisent les événements (ou alertes) pour analyse au sein d'une seule machine. L'intérêt principal de cette architecture est de faciliter la corrélation entre événements puisqu'on dispose alors d'une vision globale. Par contre, la charge des calculs (effectués sur le système central) ainsi que la charge réseau (due à la collecte des événements ou des alertes) peuvent être lourdes et risquent de constituer un goulet d'étranglement.

- Analyse locale : Si l'analyse du flot d'événements est effectuée au plus près de la source de données (généralement en local sur chaque machine disposant d'un capteur), on minimise le trafic réseau et chaque analyseur séparé dispose de la même puissance de calcul. En contrepartie, il est impossible de croiser des événements qui sont traités séparément et l'on risque de passer à côté de certaines attaques distribuées.

3.5.3. Fréquence de l'analyse

Une autre caractéristique des systèmes de détection d'intrusions est leur fréquence d'utilisation :

- Périodique : Certains systèmes de détection d'intrusions analysent périodiquement les fichiers d'audit à la recherche d'une éventuelle intrusion ou anomalie passée. Cela peut être suffisant dans des contextes peu sensibles (on fera alors une analyse journalière, par exemple).

- Continue : La plupart des systèmes de détection d'intrusions récents effectue leur analyse des fichiers d'audit ou des paquets réseau de manière continue afin de proposer une

détection en quasi temps-réel. Cela est nécessaire dans des contextes sensibles (confidentialité) et/ou commerciaux (confidentialité, disponibilité). C'est toutefois un processus coûteux en temps de calcul car il faut analyser à la volée tout ce qui se passe sur le système.

3.5.4. Comportement après détection

Une autre façon de classer les systèmes de détection d'intrusions consiste à les classer par type de réaction lorsqu'une attaque est détectée :

- Passive : La plupart des systèmes de détection d'intrusions n'apportent qu'une réponse passive à l'intrusion. Lorsqu'une attaque est détectés, ils génèrent une alarme et notifient l'administrateur système par e-mail, message dans une console, voire même par beeper. C'est alors lui qui devra prendre les mesures qui s'imposent

- Active : D'autres systèmes de détection d'intrusions peuvent, en plus de la notification à l'opérateur, prendre automatiquement des mesures pour stopper l'attaque en cours. Par exemple, ils peuvent couper les connexions suspectes ou même, pour une attaque externe, reconfigurer le pare-feu pour qu'il refuse tout ce qui vient du site incriminé. Des outils tels que RealSecure ou NetProwler proposent ce type de réaction. Toutefois, il apparait que ce type de fonctionnalité automatique est potentiellement dangereux car il peut mener à des dénis de service provoqués par l'IDS. Un attaquant déterminé peut, par exemple, tromper l'IDS en usurpant des adresses du réseau local qui seront alors considérées comme la source de l'attaque par l'IDS. Il est préférable de proposer une réaction facultative à un opérateur humain (qui prend la décision finale).

3.5.5. Méthode de détection

Deux approches de détection ont été proposées :

- L'approche comportementale : Cette approche se base sur l'hypothèse selon laquelle nous pouvons définir un comportement normal de l'utilisateur et que toute déviation par rapport à celui-ci est potentiellement suspect.

- L'approche par signature : Elle s'appuie sur un modèle constitué des sections interdites dans le système d'informatique, ce modèle s'appuie sur la connaissance des techniques employées par les attaquants : On tire des scénarios d'attaque et on recherche dans les traces d'audit leur éventuelle survenue.

4. Les types des Systèmes de détection d'intrusion [27]

La classification des IDSs se fait à travers un critère important qui est le champ de la surveillance. Il existe une surveillance sur une petite zone qui prend le nom de détection basée sur l'hôte et sur une zone vaste qui appelle détection d'intrusion basée sur le réseau.

4.1. Systèmes de détection d'intrusion en réseau

Systèmes de détection d'intrusion en réseau (en anglais Networked Intrusion Detection Systems, NIDS) est responsable de la surveillance du réseau par la détection des données non autorisées et anormales qui circulent dans le réseau surveillé par le NIDS, en conséquence la détection des intrusions dans les trafics réseaux.

Il existe une différence entre le pare-feu et le NIDS, le pare-feu permet d'autoriser ou refuser l'accès à un réseau à base d'un ensemble des règles. Mais le NIDS capture tous les paquets arrivant au réseau et puis cherche dans la liste des signatures, le résultat de la recherche peut être une intrusion ou non.

Le NIDS s'exécute soit en tant qu'indépendante machine autonome qui surveille tout le trafic ou en tant qu'une machine cible qui fait la surveillance du son propre trafic.

Les avantages des NIDS sont les suivants : Les capteurs peuvent être bien sécurisés puisqu'ils se contentent d'observer le trafic et permettent donc une surveillance discrète du réseau, les attaques de type scans sont facilement détectées, et il est possible de filtrer le trafic.

Les NIDS sont très utilisés et remplissent un rôle indispensable, mais ils présentent néanmoins de nombreuses faiblesses. En effet, la probabilité de faux négatifs (attaques non détectées comme telles) est élevée et il est difficile de contrôler le réseau entier. De plus, ils doivent principalement fonctionner de manière cryptée d'où une complication de l'analyse des paquets. Pour finir, à l'opposé des IDS basés sur l'hôte, ils ne voient pas les impacts d'une attaque.

Voici quelques exemples de NIDS : NetRanger, Dragon, NFR, Snort, DTK et ISS RealSecure

4.2. Systèmes de détection d'intrusion basée sur l'hôte

Systèmes de détection d'intrusion basée sur l'hôte (en anglais Host-Based Intrusion Detection Systems, HIDS) est la solution de problème des intrus dans les organisations, car il dépend sur un système de vérification de réseau de l'organisation.

Le HIDS est un système qui fonctionne sur un seul machine. C'est une technique de détections des activités malveillante qui l'implémente sur un seul hôte.

Il travaille avec un logiciel de surveillance le système d'exploitation avec des journaux spécifiques tel que les journaux systèmes, etc .

On peut installer le HIDS dans une machine à distance ou sur un ensemble des machines sur le réseau.

Ce type d'IDS possède un certain nombre d'avantages : Il est possible de constater immédiatement l'impact d'une attaque et donc de mieux réagir. Grâce à la quantité des informations étudiées, il est possible d'observer les activités se déroulant sur l'hôte avec précision et d'optimiser le système en fonction des activités observées.

De plus, les HIDS sont extrêmement complémentaires des NIDS. En effet, ils permettent de détecter plus facilement les attaques de type «Cheval de Troie», alors que ce type d'attaque est difficilement détectable par un NIDS. Les HIDS permettent également de détecter des attaques impossibles à détecter avec un NIDS, car elles font partie de trafic crypté.

Néanmoins, ce type d'IDS possède également ses faiblesses, qui proviennent de ses qualités : Du fait de la grande quantité de données générées, ce type d'IDS est très sensible aux attaques de type DoS, qui peuvent faire exploser la taille des fichiers de logs.

Un autre inconvénient tient justement à la taille des fichiers de rapport d'alertes à examiner, qui est très contraignante pour le responsable sécurité. La taille des fichiers peut en effet atteindre plusieurs Mégaoctets. Du fait de cette quantité de données à traiter, ils sont assez gourmands en CPU et peuvent parfois altérer les performances de la machine hôte.

Enfin, ils ont moins de facilité à détecter les attaques de type hôte que les IDS réseaux.

Les HIDS sont en général placés sur des machines sensibles, susceptibles de subir des attaques et possédantes des données sensibles pour l'entreprise. Les serveurs, web et applicatifs, peuvent notamment être protégés par un HIDS.

Pour finir, voici quelques HIDS connus : Tripwire, SWATCH, DragonSquire, Tiger et Security Manager.

4.3. Le système hybride de détection d'intrusion

Le système hybride est la combinaison entre les NIDS et le HIDS. Le système hybride prend les avantages de chaque système avec le complètement des faiblesses de chaque système pour l'augmentation de la sécurité du réseau. Le système hybride offre une grande flexibilité dans les options de déploiement.

Les avantages des IDS hybrides sont multiples :

- Moins de faux positifs .

- Meilleure corrélation (la corrélation permet de générer de nouvelles alertes à partir de celle existantes).
- Possibilité de réaction sur les analyseurs.

4.4. System de détection d'Intrusions basée sur une application

Les IDS basés sur les applications (en anglais Application-Based Intrusion Detection ABIDS) sont un sous-groupe des HIDS. Ils contrôlent l'interaction entre un utilisateur et un programme en ajoutant des fichiers de log afin de fournir de plus amples informations sur les activités d'une application particulière. Puisque vous opérez entre un utilisateur et un programme, il est facile de filtrer tout comportement notable. Un ABIDS se situe au niveau de la communication entre un utilisateur et l'application surveillée.

L'avantage de cet IDS est qu'il lui est possible de détecter et d'empêcher des commandes particulières dont l'utilisateur pourrait se servir avec le programme et de surveiller chaque transaction entre l'utilisateur et l'application. De plus, les données sont décodées dans un contexte connu, leur analyse est donc plus fine et précise.

Par contre, du fait que cet IDS n'agit pas au niveau du noyau, la sécurité assurée est plus faible, notamment en ce qui concerne les attaques de type «Cheval de Troie». De plus, les fichiers de log générés par ce type d'IDS sont des cibles faciles pour les attaquants et ne sont pas aussi sûrs, par exemple, que les traces d'audit du système.

Ce type d'IDS est utile pour surveiller l'activité d'une application très sensible, mais son utilisation s'effectue en général en association avec un HIDS. Il faudra dans ce cas contrôler le taux d'utilisation CPU des IDS afin de ne pas compromettre les performances de la machine.

5. Les mécanismes des Systèmes de détection d'intrusion [28]

IDS est un système qui permet de connaître et découvrir les accès non autorisés dans le système de l'entreprise et les réseaux informatiques. La détection d'intrusion n'est pas une nouvelle technique, utilisée dans le passé pour les guerres. Avec l'évolution de la technologie, la détection d'intrusion devient un système pour protéger les systèmes des entreprises et les réseaux informatiques.

Il existe deux modèles de mécanismes de détection d'intrusion :

5.1. détection basée sur les anomalies

Le système basé sur les anomalies est un système qui travaille avec le comportement du système, ont créé des profils sur l'état du système normal.

Ces profils sont les comportements normaux du système. Dans le cas de l'intrusion on fait

une comparaison entre le comportement anormal et le comportement normal du système et en faire une mise à jour sur les comportements normaux.

5.2. détection basée sur les signatures

Ce type de détection porte le nom de la détection basée sur la mauvaise utilisation, il fonctionne avec une base de signatures des activités normales. Ce modèle doit créer une liste sur toutes les activités anormales ou les actions non autorisées, à cause de nombre limité des éléments de la liste. Pour la facilité de la gestion de la liste, on a trois classes d'activité :

- l'accès non autorisé.
- la modification non autorisée.
- dénis de service.

5.3. Comparaison entre les deux approches

Le tableau suivant établi une comparaison entre les caractéristiques des deux précédentes approches (voir tableau 2.1)

Scénarion	Comportementale
Pas de faux positifs	Prise en compte de nouvelles
Pas de détection d'attaques non connues	Faux positifs nombreux attaques
Mise à jour rapide	Mise à jour délicate

Tableau 2.1 : Comparaison entre l'approche par scénario et l'approche comportementale [29]

6. Les différents types d'attaques dans les IDSs

plusieurs critères sont essentiels pour évaluer et garantir la robustesse des mesures de sécurité mises en place. Ces critères fournissent un cadre indispensable pour comprendre et mesurer l'efficacité des stratégies de protection des données et des systèmes contre les menaces potentielles. Parmi les critères de sécurité les plus fondamentaux, on retrouve la confidentialité, l'intégrité, la disponibilité et le contrôle. Chacun de ces aspects joue un rôle crucial dans la création d'un environnement informatique sécurisé et fiable.

- Confidentialité : Les conséquences de l'attaque sont que la confiance mutuelle envers un utilisateur a changé (en général en sa faveur), c'est-à-dire qu'il n'a plus à s'authentifier pour un programme donné. De telles circonstances sont encore très répandues de nos jours : si par exemple, une confiance mutuelle est instaurée entre deux hôtes, l'utilisateur d'un hôte peut se loger dans un autre hôte sans mot de passe. Si un attaquant parvient à un équivalent de cette situation par son attaque, il sera souvent très difficile de découvrir cet "abus de confiance".

- Intégrité : Si l'utilisateur modifie d'importants fichiers système ou de configuration, remplace des binaires par les siens... Souvent, des binaires (comme par exemple /bin/login) sont remplacés par des versions "personnelles". L'utilisateur doit alors fournir un mot de passe défini (dans le code source) et obtient (principalement) des privilèges de root. De telles attaques servent à augmenter les privilèges d'un utilisateur.

De plus, des erreurs dans la configuration et l'utilisation sont souvent dévastatrices et font de Tripwire un risque supplémentaire dans ces cas-là.

- Disponibilité : L'accessibilité du système est affectée. Ceci peut empêcher certains utilisateurs de se connecter de manière définitive ou occasionnelle. Le but est de travailler sans être dérangé et de ne pas être découvert.

- Contrôle : Par exemple, si l'attaque est destinée à prendre le contrôle du système, c'est-à-dire les fichiers, les programmes... Si cela se produit, vous pouvez considérer que l'attaquant bénéficie également des trois autres possibilités. S'il prend le contrôle total sur le système, il peut modifier, restreindre tout ce qu'il veut [30].

Il existe un grand nombre de type d'attaques, que nous pouvons classifier en différentes catégories selon plusieurs critères [31].

6.1. Les attaques d'accès

C'est des attaques qui visent à voler les données sensibles de l'utilisateur (login, mot de passe) ou bien à usurper son identité.

6.1.1. Le Sniffing

Cette attaque est utilisée par des personnes malveillantes pour obtenir les mots de passe de différents comptes utilisateurs. L'idée est de configurer l'interface réseau de la machine en mode écoute (monitor mode), afin de visualiser et de « dévier » le chemin de toutes les trames qui circulent sur le réseau vers la machine du pirate, sans en être forcément le destinataire. Elle permet de récupérer les login/mot de passe des comptes utilisateurs utilisant FTP et TELNET [32].

6.1.2. L'ingénierie sociale

L'ingénierie sociale (social engineering) n'est pas vraiment une attaque informatique, c'est plutôt une méthode qui consiste à jouer le rôle de quelqu'un d'autre (un administrateur de serveur par exemple), afin d'avoir des mots de passe, des informations et l'accès en inventant une cause par exemple un blocage de réseau [31].

«Lorsque quelqu'un désire pénétrer dans un système informatique, sa première arme est le "Baratin". Il n'y a généralement pas d'attaques réussies sans relations humaines. On

appel ceci l'ingénierie sociale (social engineering)» [32].

6.2. Les attaques de modification

Une attaque de type modification a pour but la modification des informations, c'est-à-dire qu'elle affecte l'intégrité de l'information.

6.2.1. Virus

Un virus informatique est un logiciel de petite taille, transmis d'ordinateur à ordinateur via les pièces jointes d'un message électronique, ou bien des messages instantanés ou via les différents disques amovibles. Il perturbe le fonctionnement d'une machine, endommage ou supprime des données [Microsoft, 2012].

Parmi les «symptômes» que provoque l'infection par un virus, nous citons :

- L'ordinateur fonctionne plus lentement que d'habitude .
- Les disques ou les lecteurs de disques deviennent inaccessibles.
- Des messages d'erreur inhabituels s'affichent.
- Les programmes disparaissent involontairement.

6.2.2. Les vers

Un ver est un programme parasite. Il n'est pas forcément auto-propageable. Son but est de grignoter des ressources système : CPU, mémoire, espace disque, etc [SecuriteInfo, 2012].

6.2.3. Les chevaux de Troie

Ce sont des programmes informatiques cachés dans d'autres programmes. Ils peuvent voler les mots de passe, copier les données, exécuter des actions nuisibles. Ils peuvent aussi créer des failles dans le système [Microsoft, 2012].

6.3. Les attaques par saturation

Une attaque par déni de service (en anglais Denial Of Services, DOS) est un type d'attaque visant à rendre indisponible, pendant un temps indéterminé, les services ou ressources d'une organisation.

Son principe est de saturer les serveurs et de les paralyser en leur envoyant des milliers de requêtes depuis des stations différentes, qui ne sont pas forcément situées dans le même emplacement.

Il existe différentes attaques par saturation :

6.3.1. TCP-SYN flooding

Il s'appuie sur une faille dans le protocole TCP. Il exploite la connexion en 3 phases de TCP (3-way handshake).

Le but du SYN flooding est d'ouvrir «à moitié» un nombre de connexions dépassent la

taille du backlog (file permettant de stocker les connections en attentes). Cela a pour conséquence de bloquer totalement les nouvelles demandes de connexion vers la station visée [31].

6.3.2. Smurf

Le pirate fait des requêtes ICMP ECHO à des adresses de broadcaste en indiquant l'adresse de la machine cible. Cette machine cible va recevoir un nombre énorme de réponses, car toutes les machines vont lui répondre, et ainsi utiliser toute sa bande passante [33].

Le nombre de paquets ICMP envoyé à la machine évolue de manière exponentielle ce qui provoque son blocage ou sa saturation.

6.4. Les attaques de répudiation

Appelée aussi attaque contre la responsabilité. Elle consiste à envoyer de fausses informations ou à effacer la trace d'une quelconque transaction déjà faite [31].

L'attaque la plus connue de ce type est IP spoofing.

- IP spoofing :

La mystification d'adresse IP (spoofing) est une attaque qui consiste à remplacer l'adresse IP de l'expéditeur par l'adresse IP d'une autre machine [32]. Ce n'est pas réellement le changement de l'adresse IP, mais plutôt le déguisement de l'adresse IP au niveau des paquets envoyés.

7. Les mesures d'évaluation de l'IDS [24]

Les mesures qui nous permettent d'évaluer l'efficacité globale des systèmes de détection d'intrusion selon sont :

- La precision : Le système IDS est précis lorsqu'il détecte les attaques sans faire des fausses alarmes. La non-précision survient lorsqu'il déclare comme anormale ou instructive une action légitime dans l'environnement.
- La performance de traitement : Est mesurée par la vitesse dans laquelle les événements sont traités. Lorsque le système IDS est plus performant alors la détection en temps réel sera possible.
- La complétude : C'est la capacité d'un IDS de détecter toutes les attaques.
- La tolérance aux pannes: la plupart des systèmes de détection d'intrusion s'exécutent dans des systèmes d'exploitation ou des matériels qui sont connus pour être vulnérables aux attaques. Donc un IDS devrait être résistant à ces attaques en particulier les attaques de déni de service.
- La rapidité : L'IDS doit être plus rapide dans l'analyse et l'exécution pour minimiser le

temps de réagir, et aussi pour empêcher l'attaquant d'altérer la source de vérification ou interrompre le fonctionnement du système.

8. Les limites d'un IDS [34], [35]

La majorité des IDS actuels soufrent d'au moins deux des problèmes suivants :

8.1. Problème d'intégrité des données analysées

Il se peut que les informations en route vers l'IDS pour être analysées, soient compromises par l'intrus qui tenterait de dissimuler ses traces. C'est dans ce cas que se pose le problème d'intégrité des informations analysées par l'IDS par rapport aux informations provenant de la source.

8.2. Problème de fiabilité

L'IDS est lui-même un système qui peut subir des actions malveillantes de sabotage ou de reconfiguration ou même de désactivation de ses diffèrent modules. Il n'est pas à exclure aussi qu'un IDS, comme tous les systèmes d'ailleurs, puisse contenir des défauts de conception, d'implémentation ou de configuration.

8.3. Problème d'utilisation de ressources

Ce problème se pose principalement pour les NIDS qui sont fonctionnel en temps réel et qui consomment par conséquent beaucoup de ressources système, surtout avec des débits réseaux de plus en plus élevés et des bases de signatures dont la taille augmente exponentiellement.

8.4. Perte de la capacité de détection

Ce problème est dû au volume important de données à analyser. C'est-à-dire que lorsque la dimensionnalité des données est très grande, le système a du mal à garder un bon taux de détection.

9. Les Avantages d'utilisation des IDS [36]

Le déploiement d'un IDS sur votre réseau présente notamment les avantages suivants :

9.1. Déjouer les attaques attendues sur le réseau

Les IDS protègent les systèmes contre les attaques réseaux par : Détection de porte dérobée, détection d'usurpation d'adresse IP, Dos, les vers, les chevaux de Troie, virus, Botnet, rootkit, Spyware, et autres menaces qui pourraient nuire au réseau, Les IDS actifs prennent des mesures automatiques contre les menaces de sécurité et les risques auxquels font face.

9.2. Avertis Administrateur réseau d'alerte pour les événements de sécurité potentiels

La fonction de base des systèmes de détection d'intrusions est de générer des avertissements là où existent des menaces externes, internes ou de violations de la politique de sécurité réseau, et aussi de fournir à l'administrateur des informations détaillées sur le mouvement des données au sein du réseau.

9.3. Gagnez du temps

- L'utilisation des IDS fournit beaucoup de temps et d'effort pour connaitre de ce qui se passe dans le réseau, peut aussi tourner en permanence sans superviseur humain.
- Contrôle des Programmes utilisés par les employés pour surveiller l'Internet
- IDS peut aider à découvrir les programmes qui traitent de l'internet, cela permet de mieux contrôler et de protéger le réseau.

9.4. Avoir la confiance des clients

Les IDS aident les organisations de protéger les données de ses clients contre le vol et la violation de la sécurité, Cela permet d'avoir la confiance des clients et partenaires et garder une bonne réputation sur l'organisation.

9.5. Économisez de l'argent

Grace aux IDS les organisations peuvent déterminer les mouvements suspects dans le réseau et signaler les responsables pour prendre des mesures proactives en protéger le réseau et gagner l'argent qui sera dépensé si la violation de la sécurité est arrivée dans le réseau ou si le vol de renseignements personnels a eu lieu.

10. État de l'art

L'application de l'apprentissage profond a pris une importance croissante pour aborder certaines détections d'intrusion émergentes. Grâce à leur capacité à traiter directement les données brutes, les méthodes d'apprentissage profond permettent d'apprendre des caractéristiques et de réaliser des classifications simultanément. Cela les rend l'approche de référence dans les études sur les systèmes de détection d'intrusion (voir Tableau). Parallèlement, les hackers publient leurs performances et les nouvelles méthodes qu'ils ont créées, tandis que les systèmes de détection d'intrusion sont continuellement développés pour être de plus en plus efficaces contre ces malwares.

Etudes	Année	Méthode	Jeux de données utilisés	Mesures de performance	Valeurs
Salama et al. [37]	2011	DBN-SVM	NSL-KDD	Accuracy	90%
Ferrag et al.[38]	2020	Deep disciriminative models. (DNN, RNN, CNN)Unsupervised models (RBM, DBN, DBM, DA)	CSE-CIC IDS2018	Accuracy	97.37%
Patil et al. [39]	2019	Random Forest.	CSE-CIC-IDS2017 UNSW-NB15	Accuracy	99.96% 96.94%
Pujol-Perich et al.[40]	2020	GNN (graph neural network)	CSE-CIC IDS2018	Accuracy	99%
Kanimozhi et al. [41]	2019	ANN, RF, k-NN, SVM, Adaboost, NB	CSE-CIC IDS2018	Accuracy Precision Recall Score-F1	99.91% 99.95% 99.98% 99.93%
Rezvy et al. [42]	2018	Deep auto-encoder	NSL-KDD	Accuracy	99.3%
Alom et al. [43]	2015	DBN	NSL-KDD	Accuracy	97.5%
Wu et al. [44]	2018	CNN	NSL-KDD	Accuracy	97.88%

Tableau 2.2 : Les travaux connexes pour les systèmes de détection d'intrusion basés sur le deep learning

11. Conclusion

Dans ce chapitre, nous avons présenté le system détection d'intrusions. nous avons d'abord Donné une définition à cette notion, ensuite nous avons présenté une classification des systèmes de détection d'intrusions (IDS) en se basant sur la source de données de ces derniers ainsi que sur leurs emplacements dans le système informatique. Cette classification comporte les IDS basés hôte (HIDS, les IDS basés sur l'application (AB-IDS) les IDS basés réseau (N-IDS). Nous avons abordé aussi les deux grandes approches utilisées par les Modes de détection IDS à savoir : La détection d'anomalies et La reconnaissance de signature Ensuite nous avons présenté les deux réponses apportées par les systèmes après détection d'une intrusion, que ce soit la réponse active qui englobe plusieurs techniques comme la reconfiguration du firewall et l'interruption d'une connexion TCP, ou bien la réponse passive dont le principe repose sur l'émission des alertes, la chose qui est assurée par la plupart des IDS actuels.

Dans le chapitre suivant, nous présenterons la structure générale de notre système de détection d'intrusions , ainsi que le protocole expérimental suivi et les résultats obtenus.

Chapitre 3 :
Conception,
Experimentation et Résultats

1. Introduction

Ces dernières années, avec la généralisation d'Internet, le nombre croissant d'ordinateurs interconnectés dans notre quotidien a été remarquable. Cependant, cette expansion a également exposé les vulnérabilités des serveurs, facilitant l'accès des pirates informatiques à ces systèmes par le biais non seulement de méthodes d'attaque bien connues, mais aussi de nouvelles approches émergentes.

Dans ce chapitre, nous nous concentrerons sur la conception et la mise en œuvre de systèmes de détection d'intrusion (IDS) exploitant les techniques de ML et de DL. Nous détaillerons les étapes clés de notre recherche, en commençant par la présentation de la base de données utilisée et les prétraitements nécessaires pour l'adapter à notre étude. Ensuite, nous aborderons le choix des algorithmes sélectionnés, puis nous passerons à l'apprentissage des modèles où nous diviserons les données en ensembles d'apprentissage et de test. Enfin, nous procéderons à une analyse approfondie des différentes mesures de performance qui seront utilisées pour comparer ces algorithmes.

2. Architecture de notre système

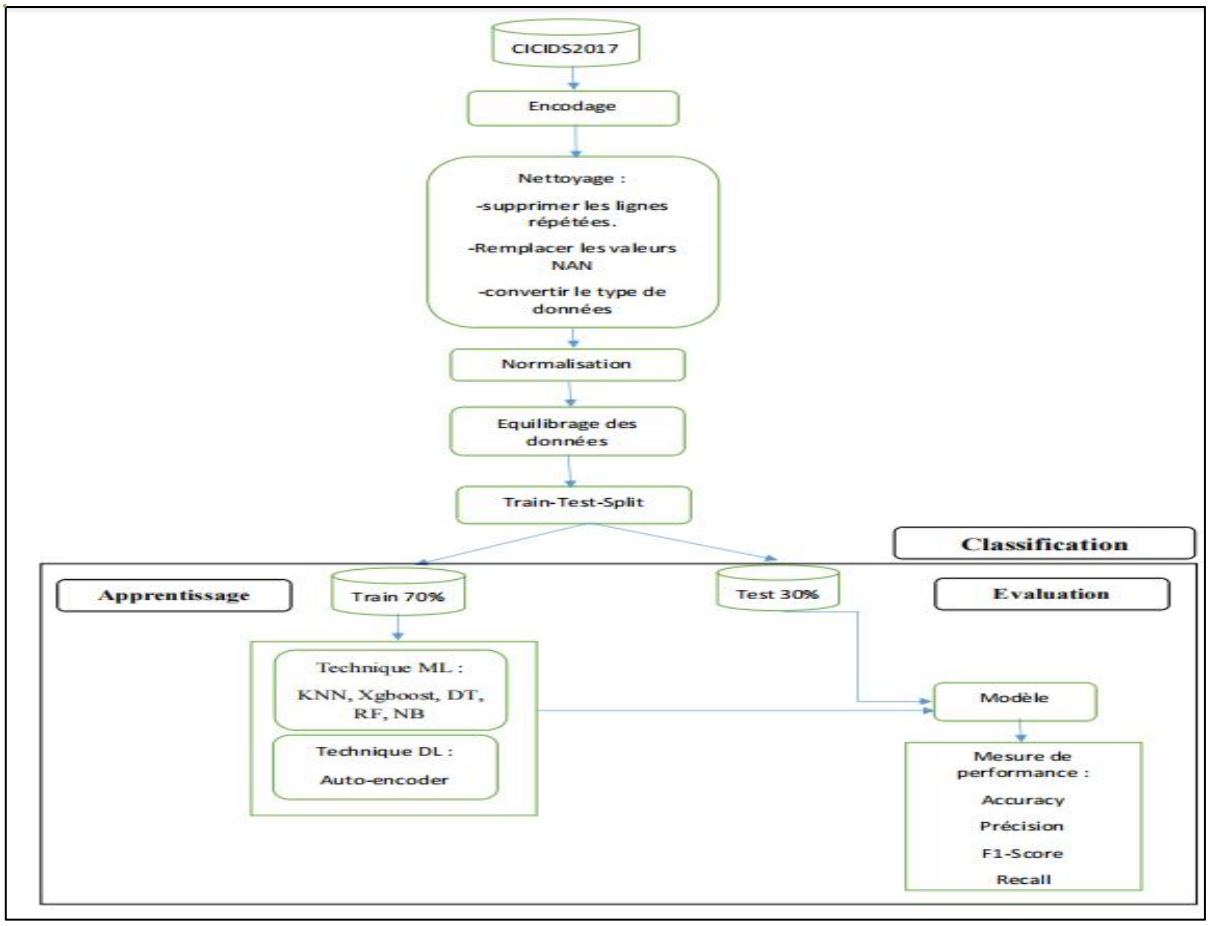

Figure 3.1 : Architecture globale de notre système

3. Présentation détaillée du système

3.1. Choix du Dataset

Depuis sa création, l'ensemble de données CIC-IDS2017 (Canadian Institute for Cybersecurity Intrusion Detection Dataset 2017) a suscité l'intérêt des chercheurs pour l'analyse et le développement de nouveaux modèles et algorithmes. Ce dataset se compose de 8 fichiers distincts contenant des données de trafic normal et d'attaques sur une période de 5 jours [w7].

CIC-IDS2017 inclut à la fois des attaques simulées et des exemples d'attaques courantes, reproduisant ainsi des scénarios similaires aux données réelles du monde. L'ensemble de données intègre également les résultats de l'analyse du trafic réseau à l'aide de CICFlowMeter. Les flux sont étiquetés en fonction de l'horodatage, des adresses IP source et destination, des ports source et destination, des protocoles, ainsi que des types d'attaques. CICFlowMeter génère des flux bidirectionnels (Biflow), où chaque flux est caractérisé par plus de 80 fonctionnalités, incluant des statistiques telles que la durée, le nombre de paquets, le nombre d'octets, la longueur des paquets, etc. Ces caractéristiques sont calculées séparément pour les directions avant (source vers destination) et arrière (destination vers source).

Les données sont exportées au format CSV avec 6 colonnes principales : FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, et Protocol.

En résumé, le tableau 3.2 présente une vue détaillée de chaque fonctionnalité de trafic réseau avec sa description respective, offrant ainsi une base solide pour l'analyse et le développement de techniques avancées de détection d'intrusion [45].

Nom des fichiers	Activité de Jour	Attaques trouvées
MondayWorkingHours.pcap_ISCX.csv	Monday	Benign (normal activities)
TuesdayWorkingHours.pcap_ISCX.csv	Tuesday	Benign, FTP-Patator, SSH-Patator
WednesdayworkingHours.pcap_ISCX.csv	Wednesday	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed

ThursdayWorkingHoursMorningWebAttacks.pcap_ ISCX.csv	Thursday	Benign, Web Attack – BruteForce, WebAttak–Sql Injection, Web Attack XSS
Thursday-WorkingHours-AfternoonInfilteration.pcap_ ISCX.csv	Thursday	Benign, Infiltration
Friday-WorkingHoursMorning.pcap_ISCX.csv	Friday	Benign, Bot
Friday-WorkingHourAfternoonPortScan.pcap_ISCX.csv	Friday	Benign, PortScan
Friday-WorkingHours-AfternoonDDos.pcap_ISCX.csv	Friday	Benign, DDoS

Tableau 3.1 : Description des fichiers contenant l'ensemble de données CIC-IDS2017

Le tableau 3.2 montre que l'ensemble de données contient des informations sur les attaques sous la forme de données de trafic sur 5 jours. Les données du jeudi après-midi et du vendredi conviennent bien à la classification binaire. De même, les données du mardi, du mercredi et du jeudi matin conviennent mieux à la conception d'un modèle de détection multiclasse. Il convient toutefois de noter qu'un modèle de détection optimal doit être capable de détecter les attaques de tout type. Par conséquent, pour concevoir un tel IDS typique, les données de trafic de tous les jours doivent être fusionnées pour former un seul ensemble de données à utiliser par l'IDS. C'est exactement ce que nous avons fait pour fusionner ces fichiers. En fusionnant les fichiers présentés dans le tableau 3.2, nous avons trouvé la forme complète d'un ensemble de données qui contient 3119345 instances et 78 caractéristiques contenant 15 étiquettes de classe (1 normale + 14 étiquettes d'attaque). En outre, l'examen des instances des fichiers combinés a révélé que l'ensemble de données contient 288602 instances dont l'étiquette de classe est manquante et 203 instances dont l'information est manquante. En supprimant ces instances manquantes, nous avons obtenu un ensemble de données combiné de CIC-IDS2017 contenant 2830540 instances. À ce stade, nous avons recherché d'éventuelles instances redondantes. De manière surprenante, aucune instance redondante n'a été trouvée.Les caractéristiques de l'ensemble de données combiné et l'occurrence détaillée par classe sont présentées dans le tableau 3.3 et le tableau 3.4 [w8].

Remarque :

Nom de la fonction : les noms des fonctionnalités (caractéristiquess) présentes dans le jeu de données CIC-IDS2017.

Numéro des classes : le nombre de classes uniques présentes dans chaque colonne

Nom de la fonction	Description	Numéro des classes
Flow ID	un identifiant unique attribué aux flux de réseau.	1085071
Source IP	L'adresse IP de source de la connexion.	17005
Source Port	Port de source de la connexion.	64640
Destination IP	L'adresse IP de destination de la connexion.	19112
Destination Port	Port de destination de la connexion.	53805
Protocol	Protocole utilisé lors de la connexion	3
Timestamp	Temps à laquelle la connexion a eu lieu	27965
Flow Duration	Durée du flux en microsecondes	1050899
Total Fwd Packets	Total des paquets dans le sens direct	1432
Total Backward Packets	Total des paquets dans le sens inverse	1747
Total Length of Fwd Packets	Taille totale des paquets dans le sens direct	17928
Total Length of Bwd Packets	Taille totale du paquet dans le sens inverse	64698
Fwd Packet Length Max	Taille maximale du paquet dans la direction avant	5279
Fwd Packet Length Min	Taille minimale du paquet dans la direction avant	384
Fwd Packet Length Mean	Taille moyenne du paquet dans la direction avant	109091
Fwd Packet Length Std	taille du paquet dans la direction avant	254384
Bwd Packet Length Max	Taille maximale du paquet en sens inverse	4838
Bwd Packet Length Min	Taille minimale du paquet dans le	583

	sens inverse	
Bwd Packet Length Mean	Taille moyenne du paquet en sens inverse	154284
Bwd Packet Length Std	taille du paquet dans le sens inverse	249206
Flow Bytes/s	Nombre d'octets de flux par seconde	1595244
Flow Packets/s	Nombre de paquets de flux par seconde	1242273
Flow IAT Mean	Temps moyen entre deux paquets envoyés dans le flux	1170377
Flow IAT Std	Écart-type du temps entre deux paquets envoyés dans le flux	1057046
Flow IAT Max	Temps maximum entre deux paquets envoyés dans le flux	580289
Flow IAT Min	Temps minimum entre deux paquets envoyés dans le flux	136316
Fwd IAT Total	Temps total entre deux paquets envoyés dans le sens direct	493098
Fwd IAT Mean	Temps moyen entre deux paquets envoyés dans le sens direct	738963
Fwd IAT Std	Écart-type entre deux paquets envoyés dans le sens direct	700372
Fwd IAT Max	Temps maximum entre deux paquets envoyés dans le sens direct	437316
Fwd IAT Min	Temps minimum entre deux paquets envoyés dans le sens direct	110631
Bwd IAT Total	Temps total entre deux paquets envoyés en sens Inverse	414928
Bwd IAT Mean	Temps moyen entre deux paquets envoyés en sens inverse	671985
Bwd IAT Std	Écart-type du temps entre deux paquets envoyés dans le sens inverse	709055
Bwd IAT Max	Temps maximum entre deux	368285

	paquets envoyés dans le sens inverse	
Bwd IAT Min	Temps minimum entre deux paquets envoyés dans le sens inverse	66074
Fwd PSH Flags	Nombre de fois où l'indicateur PSH a été activé dans les paquets circulant dans le sens direct (0 pour UDP	2
Bwd PSH Flags	Nombre de fois où l'indicateur PSH a été activé dans les paquets voyageant dans le sens inverse (0 pour UDP)	1
Fwd URG Flags	Nombre de fois où l'indicateur URG a été activé dans des paquets circulant dans le sens direct (0 pour UDP)	2
Bwd URG Flags	Nombre de fois où l'indicateur URG a été activé dans les paquets voyageant dans le sens inverse (0 pour UDP)	1
Fwd Header Length	Total des octets utilisés pour les en-têtes dans le sens direct	3771
Bwd Header Length	Nombre total d'octets utilisés pour les en-têtes dans le sens inverse	3945
Fwd Packets/s	Nombre de paquets aller par seconde	1222680
Bwd Packets/s	Nombre de paquets en sens inverse par seconde	1109941
Min Packet Length	Longueur minimale d'un paquet	215
Max Packet Length	Longueur maximale d'un paquet	5708
Packet Length Mean	Longueur moyenne d'un paquet	226511
Packet Length Std	Longueur de l'écart-type d'un paquet	413401
Packet Length Variance	Variance de la longueur d'un paquet	408238
FIN Flag Count	Nombre de paquets avec FIN	2
SYN Flag Count	Nombre de paquets avec SYN	2

RST Flag Count	Nombre de paquets avec RST	2
PSH Flag Count	Nombre de paquets avec PUSH	2
ACK Flag Count	Nombre de paquets avec ACK	2
URG Flag Count	Nombre de paquets avec URG	2
CWE Flag Count	Nombre de paquets avec CWR	2
ECE Flag Count	Nombre de paquets avec ECE	2
Down/ Patio	Taux de téléchargement et de téléversement	31
Average Packet Size	Taille moyenne des paquets	212207
Avg Fwd Segment Size	Taille moyenne observée dans le sens direct	99719
Avg Bwd Segment Size	Taille moyenne observée dans le sens inverse	147611
Fwd Header Length.1	Nombre total d'octets utilisés pour les en-têtes dans le sens direct	3771
Fwd Avg Bytes /Bulk	Nombre moyen d'octets en vrac dans le sens direct	1
Fwd Avg Packets /Bulk	Nombre moyen de paquets en vrac dans le sens direct	1
Fwd Avg Bulk Rate	Nombre moyen de paquets en vrac dans le sens direct	1
Bwd Avg Bytes /Bulk	Nombre moyen d'octets en vrac dans le sens inverse	1
Bwd Avg Packets /Bulk	Nombre moyen de paquets en vrac dans le sens inverse	1
Bwd Avg Bulk Rate	Nombre moyen d'octets en masse dans le sens inverse	1
Subflow Fwd Packets	Nombre moyen de paquets dans un sous-flux dans le sens direct	1432
Subflow Fwd Bytes	Nombre moyen d'octets dans un	17928

	sous-flux dans le sens direct	
Subflow Bwd Packets	Nombre moyen de paquets dans un sous-flux dans le sens inverse	1747
Subflow Bwd Bytes	Le nombre moyen d'octets dans un sous-flux dans le sens inverse.	64738
Init_Win_bytes_forward	Nombre total d'octets envoyés dans la fenêtre initiale dans le sens direct	12151
Init_Win_bytes_backward	Nombre total d'octets envoyés dans la fenêtre initiale dans le sens inverse.	13112
act_data_pkt_fwd	Nombre de paquets contenant au moins 1 octet de données TCP dans le sens direct	1093
min_seg_size_forward	Taille minimale d'un segment observée dans le sens direct	28
Active Mean	Durée moyenne d'activité d'un flux avant qu'il ne devienne inactif	326583
Active Std	Écart-type du temps d'activité d'un flux avant l'inactivité	202829
Active Max	Durée maximale d'activité d'un flux avant qu'il ne devienne inactif	299565
Active Min	Durée minimale pendant laquelle un flux a été actif avant de devenir inactif	175670
Idle Mean	Durée moyenne d'inactivité d'un flux avant qu'il ne devienne actif	222137
Idle Std	Écart-type de la durée d'inactivité d'un flux avant qu'il ne devienne actif	197617
Idle Max	Durée maximale d'inactivité d'un flux avant qu'il ne devienne actif	149737
Idle Min	Durée minimale d'inactivité d'un flux avant qu'il ne devienne actif	223888

Label	représente la classification du trafic réseau en tant que normal ou lié à une intrusion ou à une attaque particulière	15

Tableau 3.2 : Fonctionnalités de trafic réseau avec la description

Encodage	Normal/ Attaque Labels	Numéro d'instances
1	BENIGN	2273097
2	Bot	1966
3	DDoS	128027
4	DoS GoldenEye	10293
5	DoS Hulk	231073
6	DoS Slowhttptest	5499
7	DoS slowloris	5796
8	FTP-Patator	7938
9	Heartbleed	11
10	Infiltration	36
11	Port Scan	158930
12	SSH-Patator	5897
13	Web Attack – Brute Force	1507
14	Web Attack – Sql Injection	21
15	Web Attack – XSS	652

Tableau 3.3 : Occurrence des instances par classe dans l'ensemble de données CIC-IDS2017

3.2. Prétraitement

Le prétraitement des données est une technique d'exploration des données qui transforme les données brutes en formats compréhensibles, en s'attaquant à des problèmes tels que l'incomplétude, l'incohérence et les erreurs dans les grands ensembles de données [46].

3.2. Labellisation (étiquetage)

consiste à attribuer des étiquettes ou des catégories à des données pour les classifier. Cela permet aux modèles d'apprentissage automatique d'associer les caractéristiques des données à leurs étiquettes correspondantes, facilitant ainsi la prédiction de nouvelles données.

Label textuelle	Label numérique
BENIGN	0
Bot	1
DDoS	2
DoS GoldenEye	3
DoS Hulk	4
DoS Slowhttptest	5
DoS slowloris	6
FTP-Patator	7
Heartbleed	8
Infiltration	9
Port Scan	10
SSH-Patator	11
Web Attack – Brute Force	12

Web Attack – Sql Injection	13
Web Attack – XSS	14

Tableau 3.4: Numérisation des attaques

3.2.2. Nettoyage des données

Assurer un nettoyage efficace des données est une étape cruciale avant l'analyse ou la modélisation des données, même si cette tâche peut s'avérer laborieuse. Nous avons commencé par supprimer les lignes en double en tant qu'étape initiale du processus de nettoyage des données. En outre, nous avons supprimé les espaces et remplacé les valeurs manquantes (NaN) par les moyennes des colonnes et convertir les colonnes "Object" en valeurs numériques. [w9]

3.2.3. Normalisation des données

La plupart du temps, en machine Learning, les Data Set proviennent avec des ordres de grandeurs différents. Cette différence d'échelle peut conduire à des performances moindres. Pour palier à cela, des traitements préparatoires sur les données existent. Notamment le caractéristique Scaling qui comprend la Standardisation et la Normalisation. Min-Max Scaling peut être appliqué quand les données varient dans des échelles différentes. A l'issue de cette transformation, les caractéristiques seront comprises dans un intervalle fixe [0,1]. Le but d'avoir un tel intervalle restreint est de réduire l'espace de variation des valeurs d'une caractéristique et par conséquent réduire l'effet de valeurs aberrantes. La normalisation peut être effectuée par la technique du Min-Max Scaling. La transformation se fait grâce à la formule suivante :

$$Xnormalise = \frac{\mathrm{X} - \mathrm{Xmin}}{\mathrm{Xmax} - \mathrm{Xmin}}$$

Avec :

- X min : la plus petite valeur observée pour la caractéristique X
- X max : la plus grande valeur observée pour la caractéristique X
- X : La valeur de la caractéristique qu'on cherche à normaliser [w10]

3.2.4. Ré-équilibrage

En complément du choix d'un critère pertinent, il peut être intéressant de tenter de ré-

équilibrer l'échantillon pour aider les algorithmes à mieux détecter les individus de la classe minoritaire. Les méthodes classiques consistent à créer de nouvelles observations de la classe minoritaire (oversampling) et/ou supprimer des individus de la classe minoritaire (undersampling). et SMOTE pour Synthetic Minority OverSampling Technique consiste à sur-échantillonner en se basant sur les proches voisins de la classe minoritaire. [47]

3.2.5. Étape de division

Dans cette étape, nous divisons l'ensemble de données en 70 % de données d'entraînement et 30 % de données de Test.

3.3. Classification

La sélection de l'algorithme d'apprentissage est une étape critique dans notre étude, nécessitant une évaluation minutieuse des performances sur notre jeu de données. Nous avons exploré plusieurs approches pour identifier celle qui répond le mieux à nos objectifs de classification.

3.3.1. Les algorithmes de l'apprentissage automatique

Nous avons évalué les performances des principaux algorithmes d'apprentissage machine suivants :

- K-Nearest Neighbors
- Arbre de décision
- Forêt aléatoire
- Gaussian Naive Bayes
- Xgboost

Chacun de ces algorithmes a été testé et comparé pour déterminer celui qui convient le mieux à notre cas d'étude.

3.3.2. Les algorithmes d'apprentissage profond

Nous avons également exploré l'utilisation d'algorithmes d'apprentissage profond, notamment :

- Auto-encodeur

L'auto-encodeur a été étudié en profondeur pour ses capacités à extraire des caractéristiques pertinentes et à améliorer la précision de la classification dans notre contexte spécifique.

3.4. Les mesures d'évaluation des modèles

On a Les mesures suivants :

- Précision (Pr) : le pourcentage des attaques identifiées comme des attaques TP parmi tous les exemples prédit comme attaque, cette métrique, également relative à chaque catégorie, renseigne sur la probabilité qu'une prédiction d'une catégorie donnée soit correcte. Il est donné par :

$$Pr = TP/(TP+FP)$$

- Accuracy (le taux de réussite) : Indique la façon dont la technique de détection est correcte. C'est une métrique qui traduit également le rapport entre les détections correctes et les détections totales obtenues.

$$Accuracy = (TP+TN)/(TP+TN+FP+FN)$$

- Recall (Rc) Le taux de détection (Rappel) ou bien TPR (True Positive Rate) : le pourcentage des attaques identifiées comme des attaques TP parmi tous les attaques dans l'ensemble de données, C'est le rapport entre le nombre d'intrusions correctement détectées et le nombre total d'intrusions. Et décrit par la formule :

$$TPR \text{ ou bien } Rc = TP/(TP+FN)$$

- F1-score (F1) : La moyenne harmonique F combine le rappel et la précision en un nombre compris entre 0 et 1, il est donné par :

$$F1 = 2*(Pr*Rc)/(Pr+Rc)$$

4. Résultat et discussion

Nous utilisant une technique de caractéristiques sélection qui s'appelle «SelectKBest» pour obtenu les caractéristiques nécessaire, après nous donnes notre résultat à la technique «Train_Test_Split()» pour faire division de dataset, après nous appliquons l'algorithmes «K-Nearest Neighbors», «Arbre de décision», «Forêt aléatoire», «Gaussian Naive Bayes», «Xgboost», et «Les réseau de neurones artificiels (Auto-encodeur)»

Dans cette section, nous allons montrer et interpréter les résultats que nous avons obtenus à travers nos tests. Nous commençons d'abord par les résultats des techniques de ML ensuite le technique de DL (AE).

Partie 1

4.1. Précision détaillée par classe

Nous montrons respectivement les résultats des mesures "precision", "recall" et "F1-score" respectivement dans les tableaux. la moyenne seulement est montrée sous forme de pourcentage avec arrondi au plus proche en deux chiffres après la virgule.

Expérience 1 : Après avoir configuré l'algorithme KNN avec 3 voisins, puis avec 5 voisins, nous avons observé que les résultats étaient plus fiables avec 3 voisins.

Voici le résultat de la technique avec K=3 :

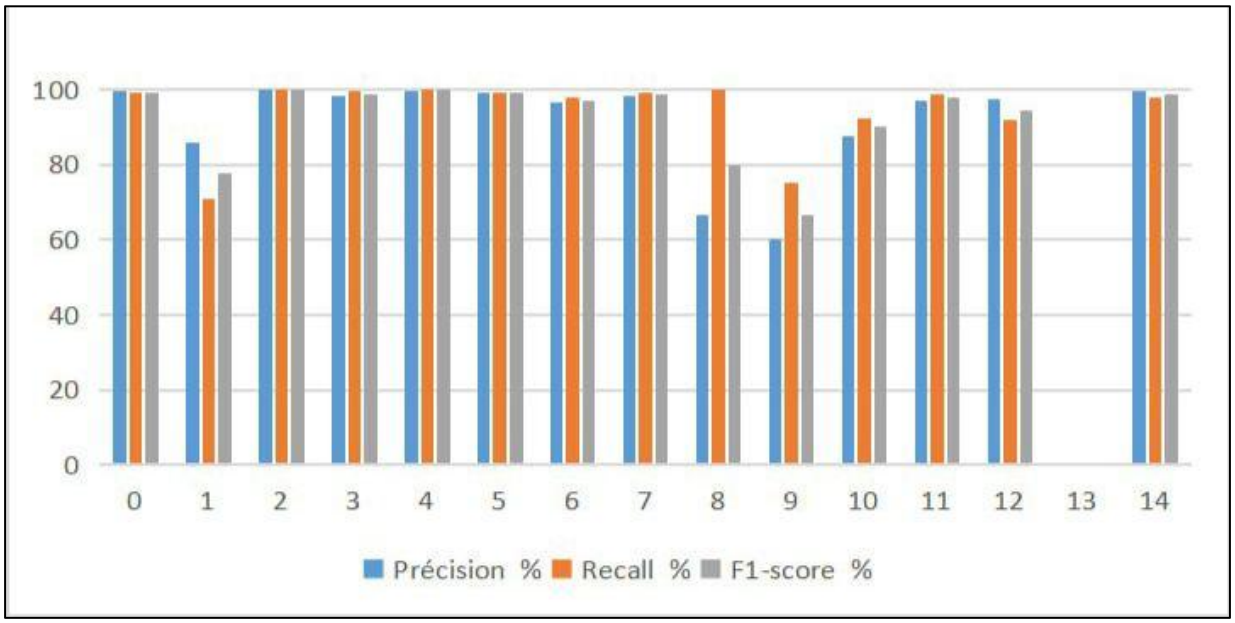

Figure 3.2 : Le résultat de la technique K-Nearest Neighbors avec K=3

Voici le résultat de la technique avec K=5 :

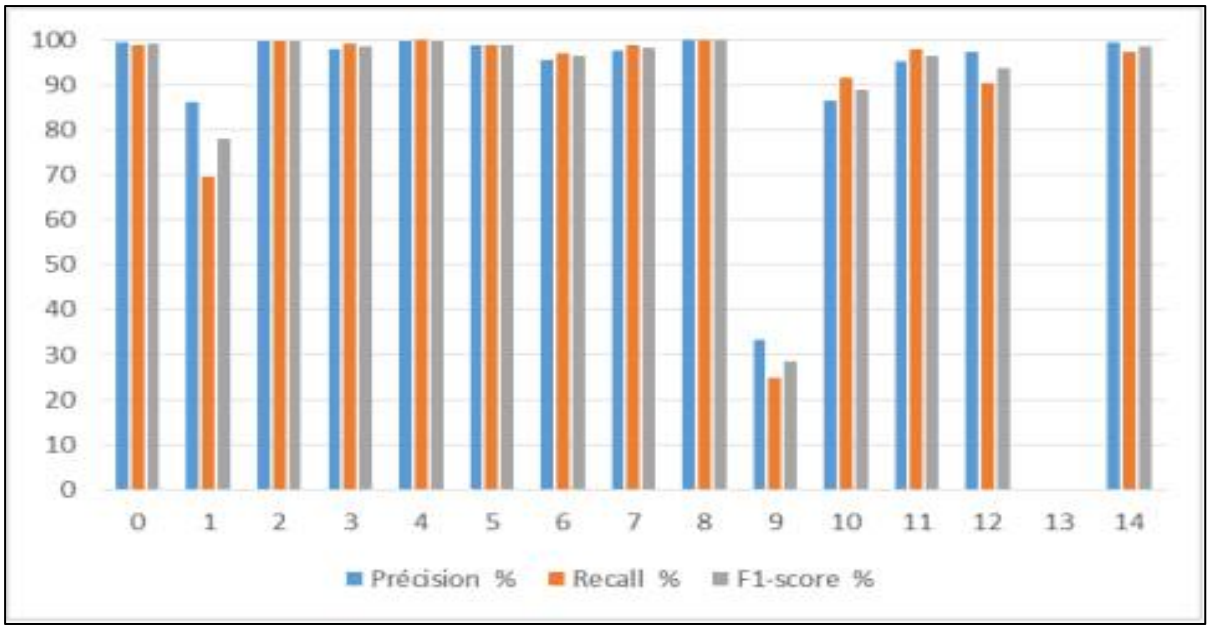

Figure 3.3 : Le résultat de la technique K-Nearest Neighbors avec K=5

Expérience 2 : Après avoir configuré l'algorithme d'arbre de décision avec l'indice 'gini' et un autre avec 'entropy', nous avons constaté que les résultats étaient fiables avec les deux méthodes.

Voici le résultat de la technique avec l'indice 'Entropy' :

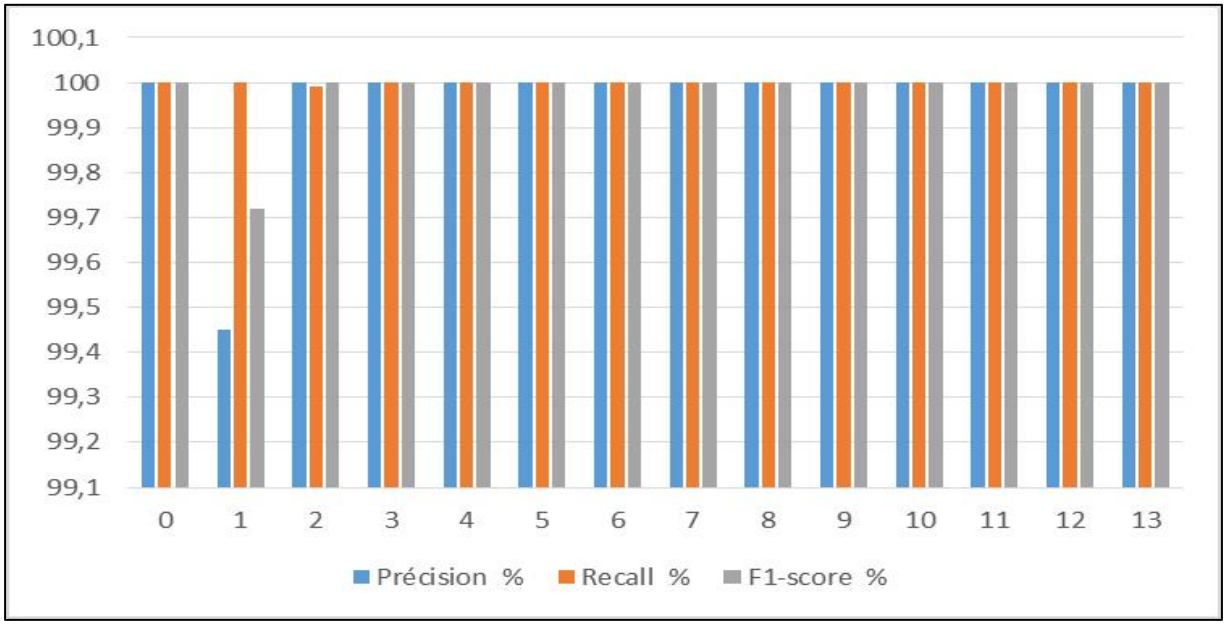

Figure 3.4 : Le résultat de la technique Arbre de décision, le critère de diversité est "Entropy"

Voici le résultat de la technique avec l'indice 'Gini' :

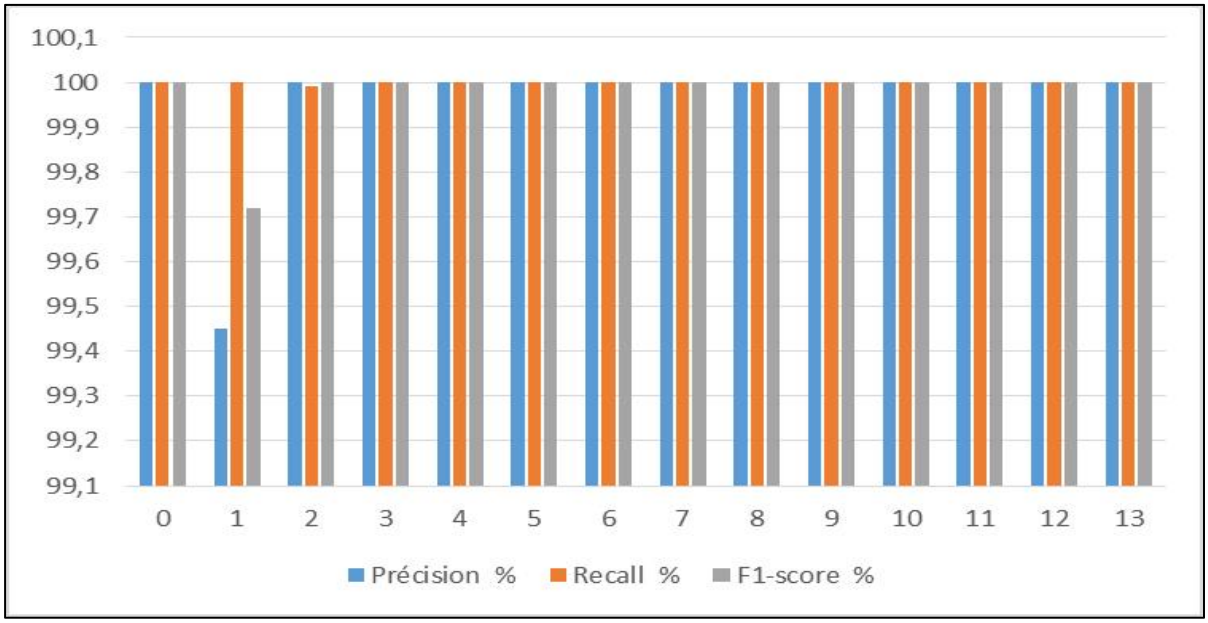

Figure 3.5 : Le résultat de la technique Arbre de décision, le critère de diversité est "Gini"

Expérience 3 : Après avoir configuré l'algorithme Forêt aléatoire avec l'indice 'Gini' et un autre avec 'Entropy', nous avons constaté que les résultats étaient fiables avec les deux méthodes, particulièrement avec l'indice 'Gini'.

Voici le résultat de la technique avec l'indice 'Entropy' :

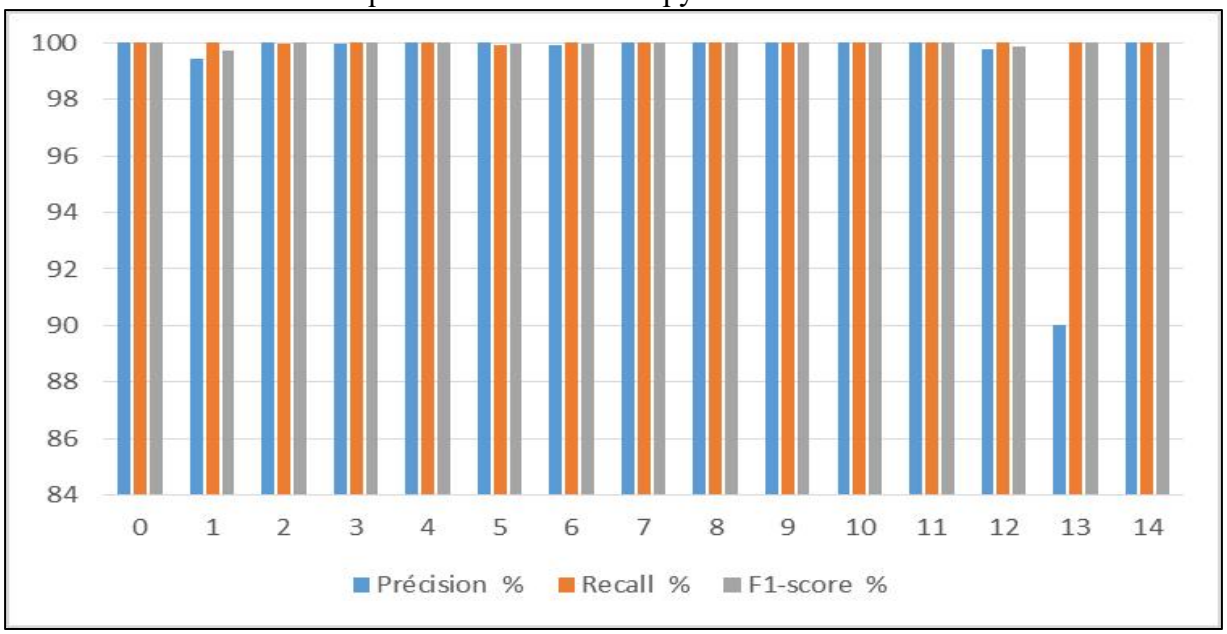

Figure 3.6 : Le résultat de la technique Forêt aléatoire, le critère de diversité est "Entropy"

Voici le résultat de la technique avec l'indice 'Gini' :

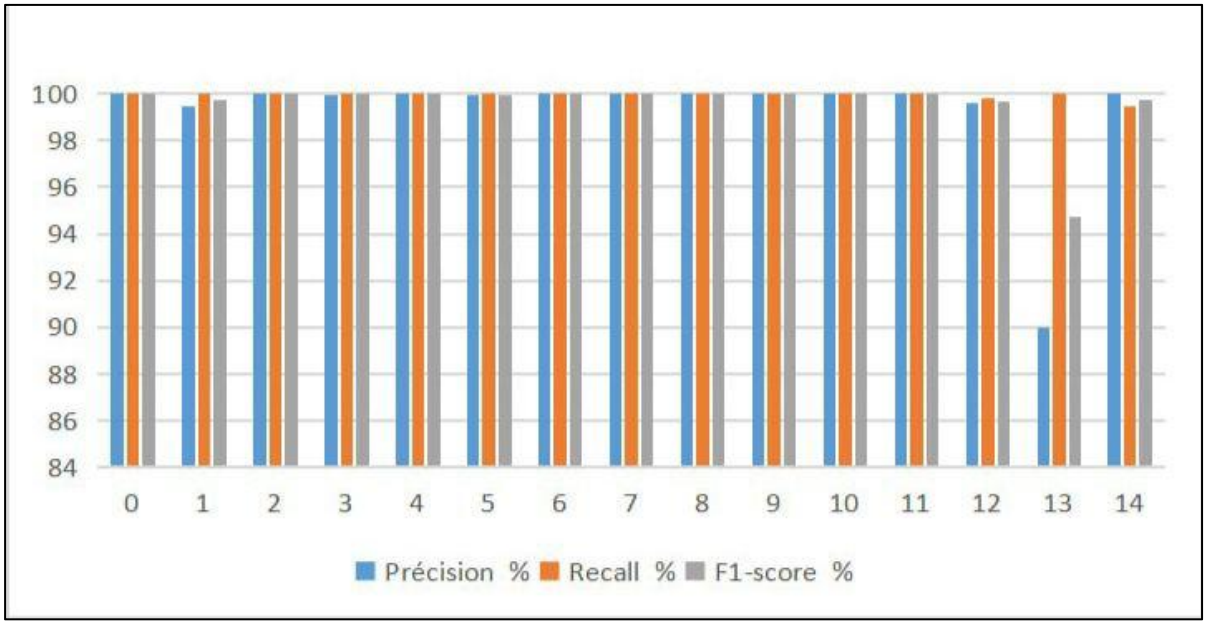

Figure 3.7 : Le résultat de la technique Forêt aléatoire, le critère de diversité est "Gini"

Expérience 4 : Après avoir configuré l'algorithme de Gaussian Naive Bayes

Voici le résultat de la technique :

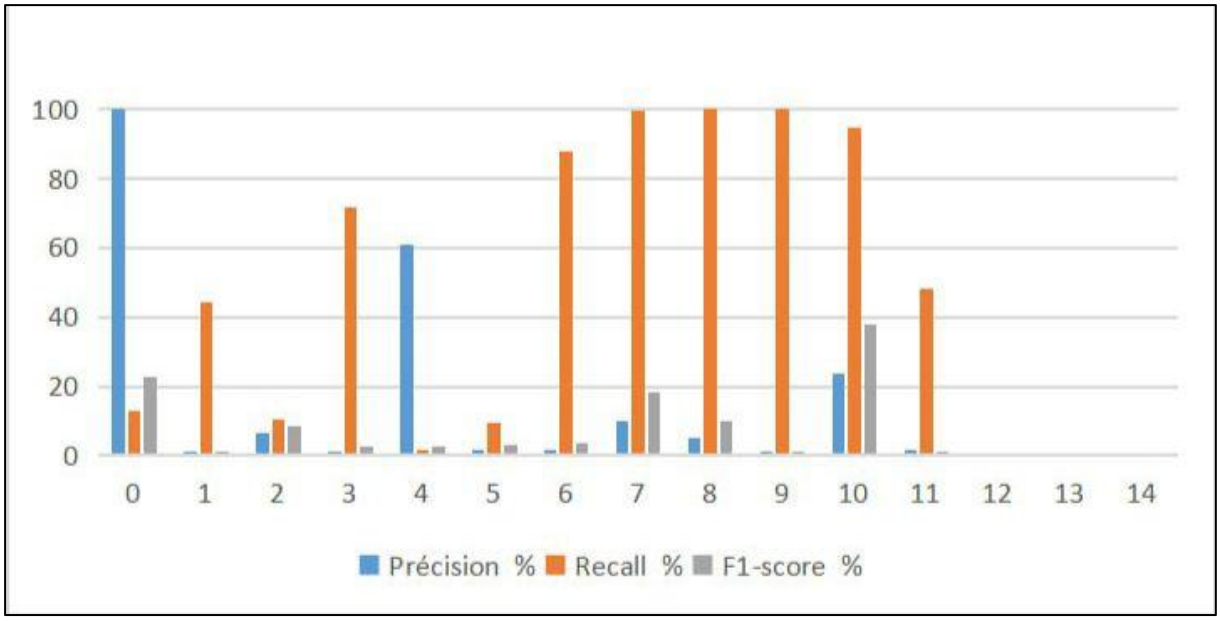

Figure 3.8 : Le résultat de la technique Gaussian Naive Bayes

Expérience 5 : Après avoir configuré l'algorithme de Xgboost

Voici le résultat de la technique :

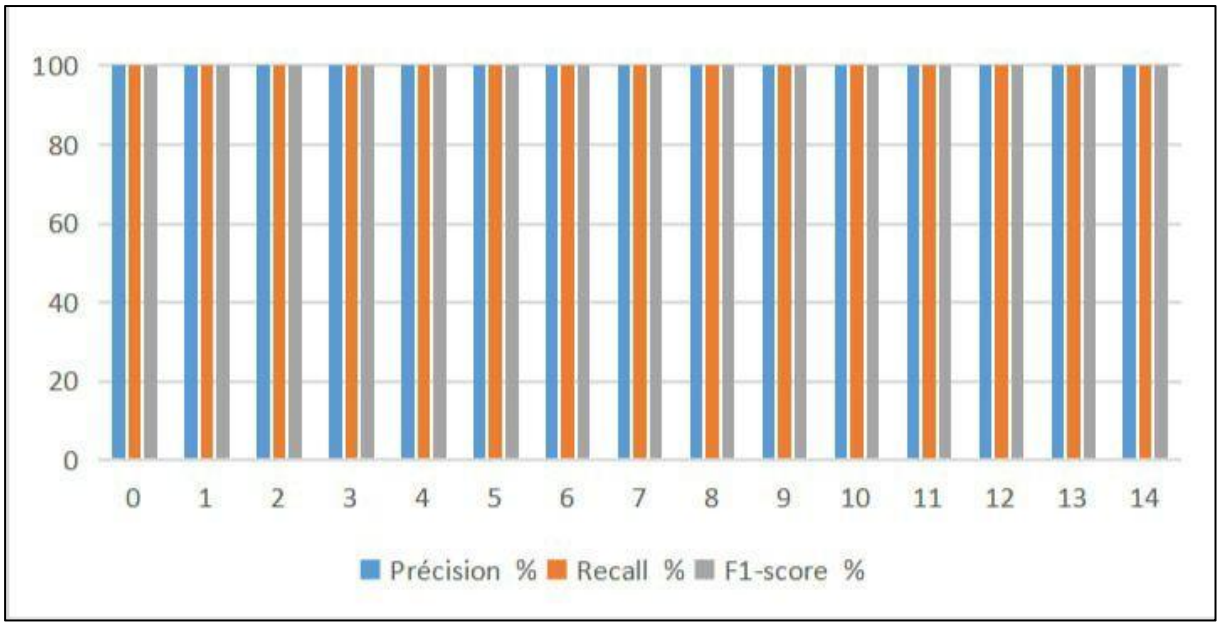

Figure 3.9 : Le résultat de la technique Xgboost

Nous présentons les résultats de l' accuracy dans le tableau ci-dessous. La moyenne est affichée en pourcentage arrondi au plus proche de deux chiffres après la virgule

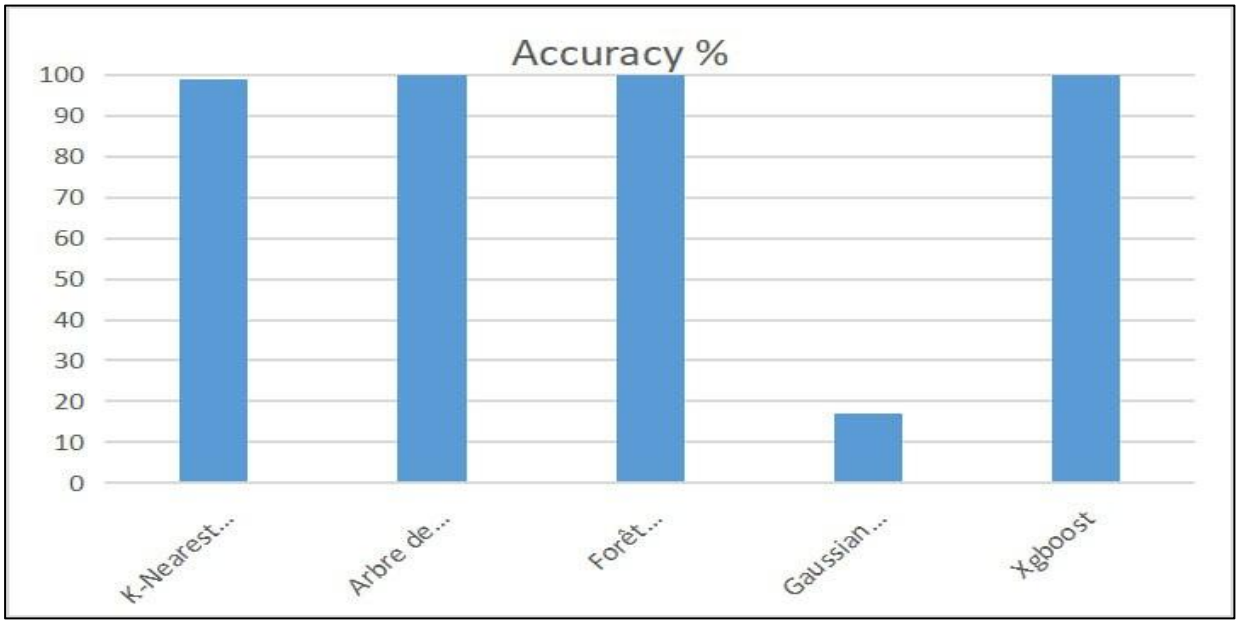

Figure 3.10 : Les résultats des mesures accuracy

Les trois algorithmes avec accuracy la plus élevée (100%) sont l'Arbre de Décision, la Forêt Aléatoire et XGBoost. Ensuite, l'algorithme K-Nearest Neighbors suit de près avec accuracy de 98%. En revanche, l'algorithme le moins performant parmi ceux testés est la classification Gaussian naïve bayésienne , avec une exactitude ne dépassant pas les 20%.

Partie 2 :

Nous avons développé et entraîné un modèle d'AE utilisant des sous-ensembles de données du jeu de données CICIDS-2017. Voici l'architecture utilisée pour ce modèle.

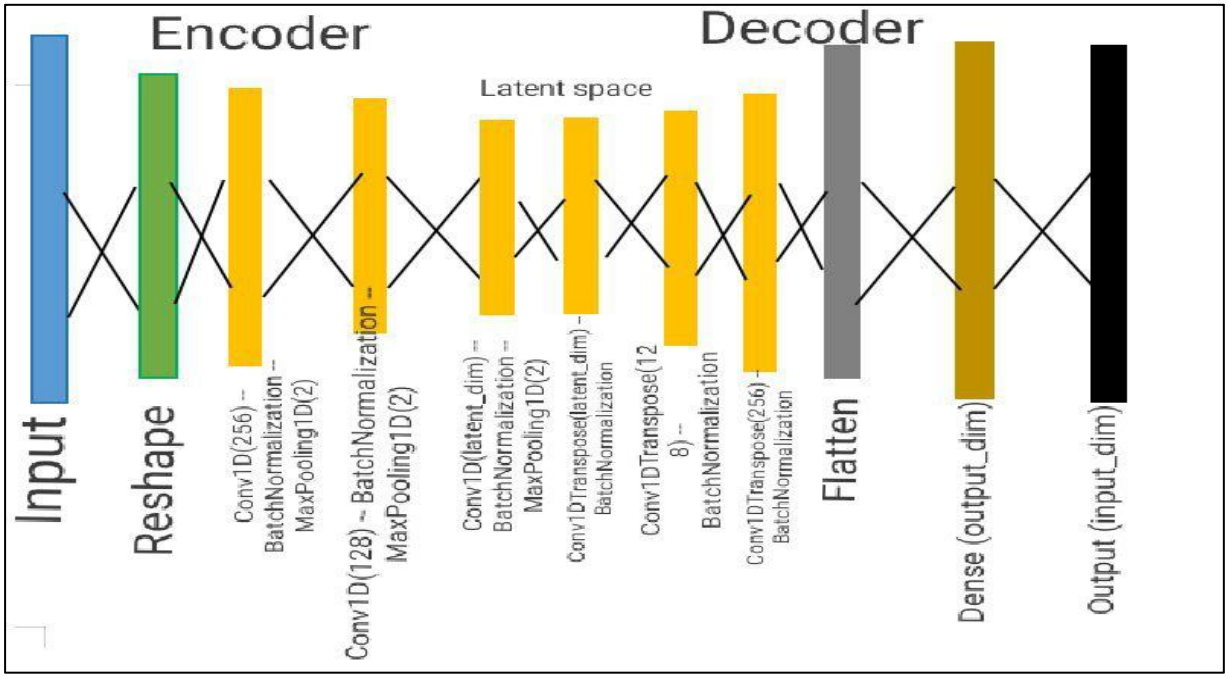

Figure 3.11 : l'architecture utilisée pour le modèle l'AE

Voici le résultat d'AE :

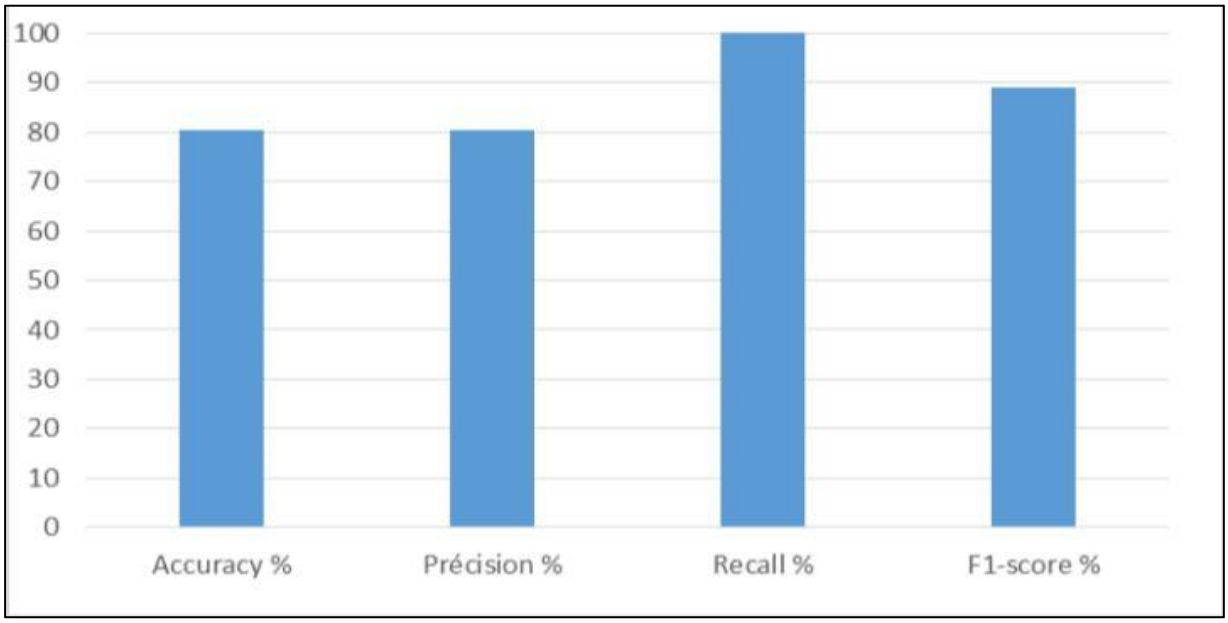

Figure 3.12: Résultats d'AE

L'auto-encodeur présente une précision et une exactitude de 80.30%, avec un score F1 proche de 90% et un rappel (recall) très élevé de 100%. Nous pensons que l'exactitude de ce modèle

pourrait encore être améliorée en utilisant un ensemble de données plus vaste.

Enfin, nous avons évalué les performances de différents modèles d'ML ainsi que d'un modèle DL, spécifiquement un auto-encodeur, pour la détection d'intrusions sur la base de données CICIDS2017. Nos résultats indiquent que parmi les modèles testés :

Les techniques d'ML pour la détection d'intrusions présentent des variations significatives en termes de performance et d'applicabilité. Les méthodes telles que les forêts aléatoires ,les arbres de décision et les algorithmes de boosting comme XGBoost sont souvent préférées en raison de leur capacité à gérer des données complexes tout en évitant le surapprentissage. En comparaison, les KNN sont simples mais peuvent devenir inefficaces sur de grands ensembles de données et Le résultat obtenu avec la méthode Naive Bayes est généralement inférieur.

En revanche, les AEs, sont capables de modéliser des relations complexes, mais exigent des ressources de calcul substantielles et de vastes ensembles de données pour une performance optimale. Cependant, nous croyons que l'utilisation d'un grand ensemble de données pourrait encore améliorer l'efficacité des AEs.

En conclusion dans la littérature, le DL est souvent vanté pour sa capacité à gérer efficacement des données complexes et non structurées à grande échelle. Cependant, pour notre étude telles que la détection d'intrusions avec un sous-ensemble de données limité et spécifique de CIC-IDS2017, les techniques de ML traditionnelles peuvent également être tout à fait appropriées et efficaces.

Le choix de la technique dépend donc des caractéristiques spécifiques des données, des ressources disponibles et des objectifs particuliers en matière de détection d'intrusions.

5. Conclusion

Diverses approches et architectures d'ML et d'DL ont été appliquées pour détection d'intrusion. Ces algorithmes proposés ont des performances varient en fonction des jeux de données sélectionnés et des caractéristiques entrées. Cependant, en utilisant les mêmes méthodes et techniques d'apprentissage ne garantit pas toujours le même résultat, pour différentes classes les attaques peuvent varier

Conclusion générale

La recherche sur les techniques d'apprentissage automatique pour l'analyse des comportements réseau et la détection des menaces a connu des progrès significatifs ces dernières années. Avec l'augmentation rapide du volume de données disponibles, la fréquence des attaques de sécurité a également augmenté de manière préoccupante. Les systèmes de détection d'intrusion sont devenus des outils essentiels pour sécuriser les réseaux. Ces systèmes scrutent le trafic réseau à la recherche d'activités anormales et déclenchent des alertes lorsque des menaces potentielles sont identifiées.

Notre étude a utilisé la base de données CIC-IDS2017 pour évaluer l'efficacité de plusieurs algorithmes d'apprentissage automatique et d'apprentissage profond dans la détection des intrusions. Nous avons testé et comparé des algorithmes tels que K-Nearest Neighbors, l'arbre de décision, la forêt aléatoire, Gaussian Naive Bayes, Xgboost, et l'auto-encodeur.

Les résultats de notre analyse indiquent que les algorithmes d'arbre de décision, de forêt aléatoire et de Xgboost offrent les meilleures performances en termes de précision ("accuracy").

Nous pensons que la performance de l'auto-encodeur pourrait être améliorée avec des jeux de données plus volumineux et une meilleure configuration des paramètres. Pour des travaux futurs, une exploration plus approfondie des paramètres et des configurations de chaque algorithme pourrait optimiser davantage leurs performances. Cette optimisation renforcerait la robustesse des systèmes de détection d'intrusion, permettant de mieux protéger les réseaux contre des menaces de plus en plus sophistiquées.

[1] ROY, Jean-Francis. *Apprentissage automatique avec garanties de généralisation à l'aide de méthodes d'ensemble maximisant le désaccord*. 2018. Thèse de doctorat. Université Laval.

[2] SAMBA, Alassane. *Science des données au service des réseaux d'opérateur: proposition de cas d'utilisation, d'outils et de moyens de déploiement*. 2018. Thèse de doctorat. Ecole nationale supérieure Mines-Télécom Atlantique.

[3] LIU, Bing et LIU, Bing. Supervised learning. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, 2011, p. 63-132.

[4] ALLOGHANI, Mohamed, AL-JUMEILY, Dhiya, MUSTAFINA, Jamila, *et al.* Une revue systématique sur les algorithmes d'apprentissage automatique supervisés et non supervisés pour la science des données. *Apprentissage supervisé et non supervisé pour la science des données*, 2020, p. 3-21.

[5] BONACCORSO, G. Machine learning algorithms Packt Publishing Ltd. 2017.

[6] ANDREW, Alex M. REINFORCEMENT LEARNING : AN INTRODUCTION par Richard S. Sutton et Andrew G. Barto, série Adaptive Computation and Machine Learning, MIT Press (Bradford Book), Cambridge, Mass., 1998, xviii+ 322 pp, ISBN 0-262-19398-1, (relié, 31,95 £). *Robotica*, 1999, vol. 17, no 2, p. 229-235.

[7] KAELBLING, Leslie Pack, LITTMAN, Michael L., et MOORE, Andrew W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 1996, vol. 4, p. 237-285.

[8] ZHANG, Zhongheng. Introduction à l'apprentissage automatique : k plus proches voisins. *Annales de médecine translationnelle*, 2016, vol. 4, no 11.

[9] SAFAVIAN, S. Rasoul et LANDGREBE, David. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 1991, vol. 21, no 3, p. 660-674.

[10] BREIMAN, L. Random forests Mach Learn 45 (1): 5–32. 2001.

[11] BERRAR, Daniel. Bayes' Theorem and Naive Bayes Classifier. 2019.

[12] XU, Shuo. Bayesian Naïve Bayes classifiers to text classification. *Journal of Information Science*, 2018, vol. 44, no 1, p. 48-59.

[13] DENG, Li, YU, Dong, *et al.* Deep learning: methods and applications. *Foundations and trends® in signal processing*, 2014, vol. 7, no 3–4, p. 197-387.

[14] ROSEBROCK, Adrian. *Deep learning for computer vision with python: Starter bundle*. PyImageSearch, 2017.

[15] HUBARA, Itay, COURBARIAUX, Matthieu, SOUDRY, Daniel, *et al.* Réseaux neuronaux quantifiés : Entraînement de réseaux neuronaux avec des poids et des activations de faible précision. *Journal de la recherche sur l'apprentissage automatique*, 2018, vol. 18, n° 187, p. 1-30.

[16] MIKOLOV, Tomas, KARAFIÁT, Martin, BURGET, Lukas, *et al.* Recurrent neural network based language model. In : *Interspeech*. 2010. p. 1045-1048.

[17] SEWAK, Mohit, SAHAY, Sanjay K., et RATHORE, Hemant. An overview of deep learning architecture of deep neural networks and autoencoders. *Journal of Computational and Theoretical Nanoscience*, 2020, vol. 17, no 1, p. 182-188.

[18] BANK, Dor, KOENIGSTEIN, Noam, et GIRYES, Raja. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, 2023, p. 353-374.

[19] IVAN, Vasilev. Python deep learning: exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow/Ivan Vasilev [and four others]. 2019.

[20] GOODFELLOW, Ian, BENGIO, Yoshua, et COURVILLE, Aaron. *Deep learning*. MIT press, 2016.

[21] RIFAI, Salah, VINCENT, Pascal, MULLER, Xavier, *et al.* Auto-encodeurs contractifs : Invariance explicite lors de l'extraction des caractéristiques. Dans : *Actes de la 28e conférence internationale sur la conférence internationale sur l'apprentissage automatique*. 2011. p. 833 à 840.

[22] KINGMA, Diederik P., WELLING, Max, *et al.* An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 2019, vol. 12, no 4, p. 307-392.

[23] FERHAT, IKRAM. ALGORITHME NSGA-III pour la sélection Des fonctionnalités utilisée dans un problème de classification déséquilibrée

[24] DEBAR, Hervé, MORIN, Benjamin, CUPPENS, Frédéric, *et al.* Détection d'intrusions: corrélation d'alertes. *Revue des Sciences et Technologies de l'Information-Série TSI: Technique et Science Informatiques*, 2004, vol. 23, p. 32 pages.

[25] MICHEL, Cédric. *Langage de description d'attaques pour la détection d'intrusions par corrélation d'événements ou d'alertes en environnement réseau hétérogène*. 2003. Thèse de doctorat. Universite Rennes 1.

[26] ABBES, Tarek. *Classification du trafic et optimisation des règles de filtrage pour la détection d'intrusions*. 2004. Thèse de doctorat. Henri Poincaré University, Nancy, France.

[27] SELMANI, Elgharbi. *Mise en place d'un IDS pour sécuriser un réseau en utilisant Snort*. 2020. Thèse de doctorat. Université Mouloud Mammeri

[28] KIZZA, Joseph Migga, KIZZA, Wheeler, et WHEELER. *Guide to computer network security*. Berlin : Springer, 2013.

[29] ZIMMERMANN, Jacob et MÉ, Ludovic. Les systèmes de détection d'intrusions: principes algorithmiques. *Supélec, équipe SSIR*, 2002.

[30] MÜLLER, Klaus. IDS− Système de Détection d'Intrusion, Partie II. 2003.

[31] GILL, Stéphane. Type d'attaques. *Document soumis à la licence GNU FDL. http://sgill. ep. profweb. qc. ca/spip/IMG/pdf/02_ TypeAttaque. pdf*, 2003, p.

[32] Gerphagnon Jean-Olivier et al, Attaques Informatique, Centro Brasileiro de Pesquisas Fisicas – CBPF/CNPq Coordena c ao de Atividade Técnicas – CAT

[33] BURGERMEISTER, David et KRIER, Jonathan. Les systèmes de détection d'intrusions. *Article disponible sur http://dbprog. developpez. com*, 2006.

[34] AXELSSON, Stefan. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 2000, vol. 3, no 3, p. 186-205.

[35] TABIA, Karim. *Modèles graphiques et approches comportementales pour la détection d'intrusions*. 2008. Thèse de doctorat. Artois.

[36] ABID, Malika et BENEDDINE, Mustapha Mohamed Amine. *Conception d'un IDS basé sur le Deep Learning et RBN*. 2021. Thèse de doctorat. Université Ibn Khaldoun-Tiaret-.

[37] SALAMA, Mostafa A., EID, Heba F., RAMADAN, Rabie A., *et al.* Hybrid intelligent intrusion detection scheme. In : *Soft computing in industrial applications*. Springer Berlin Heidelberg, 2011. p. 293-303.

[38] FERRAG, Mohamed Amine, MAGLARAS, Leandros, MOSCHOYIANNIS, Sotiris, *et al.* Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 2020, vol. 50, p. 102419.

[39] PATIL, Rajendra, DUDEJA, Harsha, et MODI, Chirag. Designing an efficient security framework for detecting intrusions in virtual network of cloud computing. *Computers & Security*, 2019, vol. 85, p. 402-422.

[40] PUJOL-PERICH, David, SUÁREZ-VARELA, José, CABELLOS-APARICIO, Albert, *et al.* Unveiling the potential of graph neural networks for robust intrusion detection. *ACM SIGMETRICS Performance Evaluation Review*, 2022, vol. 49, no 4, p. 111-117.

[41] KANIMOZHI, V. et JACOB, T. Prem. Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing. In : *2019 international conference on communication and signal processing (ICCSP)*. IEEE, 2019. p. 0033-0036.

[42] REZVY, Shahadate, PETRIDIS, Miltos, LASEBAE, Aboubaker, *et al.* Intrusion detection and classification with autoencoded deep neural network. In : *International Conference on Security for Information Technology and Communications*. Cham : Springer International Publishing, 2018. p. 142-156.

[43] ALOM, Md Zahangir, BONTUPALLI, VenkataRamesh, et TAHA, Tarek M. Intrusion detection using deep belief networks. In : *2015 National Aerospace and Electronics Conference (NAECON)*. IEEE, 2015. p. 339-344.

[44] WU, Kehe, CHEN, Zuge, et LI, Wei. A novel intrusion detection model for a massive network using convolutional neural networks. *Ieee Access*, 2018, vol. 6, p. 50850-50859.

[45] BOUROUBA, Hadjer et CHAOUCHE, Ouidad. *ptimisation des IDS du Cloud Computing par les techniques de machines Learning*. 2020. Thèse de doctorat. Université Ibn Khaldoun-Tiaret-.

[46] TRIBHUWAN, Meghana et GODSE, Snehal. Évaluation des prix des logements à l'aide

de techniques d'apprentissage automatique prédictif. Dans : *Actes de la Conférence internationale sur l'informatique et la communication innovantes (ICICC).* 2022.

[47] AJAKAN, Nora. Ingénierie de la représentation des variables pour la classification binaire à partir des données déséquilibrées. 2022.

[w1] https://machinelearningmastery.com/transfer-learning-for-deep-learning/, “A Gentle Introduction to Transfer Learning for Deep Learning”, Jason Brownlee , 16 Septembre 2019, extrait le 07 Juin 2024.

[w2] https://www.jedha.co/formation-ia/, “algorithme-xgboost ”, Myriam Emilion , 03 mai 2024, extrait le 8 juin 2024.

[w3] https://blog.bismart.com/en/difference-between-machine-learning-deep-learning/, Quelle est la différence entre le machine learning et le deep learning ?“,Maria Gorini, extrait le 07 Juin 2024

[w4] http://onlineresize.club/2021-club.html

[w5] https://towardsdatascience.com/applied/, **“**deep-learning-part-3-autoencoders”, Arden Dertat, 3 octobre 2017, extrait le 07 Juin 2024

[w6]https://analyticsindiamag.com/how-to-implement-convolutional-autoencoder-in-pytorchwith-cuda, “How to Implement Convolutional Autoencoder in PyTorch with CUDA ”

[w7]https://www.scribd.com/document/403095164/IJET-22797/, “A detailed analysis of CIC-IDS2017 dataset fordesigning Intrusion Detection Systems”, IJET-22797, 6 juin 2023, , extrait le le 8 juin 2024.

[w8]https://blog.arcoptimizer.com/pretraitement-des-donnees-dans-lapprentissage automatique/, “Prétraitement des données dans l’apprentissage automatique”, Johnny, 7 octobre 2022, le 8 juin 2024.

[w9] https://lrouviere.github.io/TUTO_ML/dondes.html/, **“**Chapitre7Données déséquilibrées | Machine learning ”, Rouvière, Laurent, extrait le 8 juin 2024

Annexe :
Outils Implémentation et Application

1. Introduction

Dans cette annexe, nous commençons par introduire les outils et langages que nous avons utilisés pour mettre en œuvre notre modèle. Ensuite, nous détaillons les différentes étapes du travail sur la base de données, notamment le nettoyage et la normalisation des données. Enfin, nous avons présenté les expériences menées et fourni des captures d'écran illustrant l'exécution de notre application.

2. Matériel et logiciels utilisés

2.1. Python

Définition du langage Python en informatique Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels. En effet, parmi ses qualités, Python permet notamment aux développeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font. Il a libéré les développeurs des contraintes de formes qui occupaient leur temps avec les langages plus anciens. Ainsi, développer du code avec Python est plus rapide qu'avec d'autres langages.

2.2. Kagele

Kaggle est une plateforme populaire de concours de science des données. Elle met en relation des data scientists avec des organisations qui ont besoin d'aide pour résoudre des problèmes de données.

Les organisations publient des défis sur Kaggle, et les scientifiques des données s'affrontent pour les résoudre.

3. Bibliothèques Supplémentaires

Une bibliothèque est un ensemble de fonctions prédéfinies. Celles-ci sont regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire. Python est un langage de programmation très riche avec ses bibliothèques. On a utilisé plusieurs paquets (bibliothèques) dans ce travail qui sont :

3.1. NumPy

est une bibliothèque Python fondamentale pour le calcul scientifique, spécialisée dans la manipulation des tableaux (array), essentiellement les vecteurs et les matrices.

3.2. Pandas

La librairie Pandas est une librairie Python qui a pour objectif de vous faciliter la vie en matière de manipulation de données. C'est donc un élément indispensable qui faut maîtriser en tant que data scientiste. Les structures de données gérées par Pandas peuvent contenir tout type d'éléments à savoir (dans le jargon Pandas) des Séries et DataFrame et des Panel. Dans le cadre de nos expérimentations on utilisera plutôt les Dataframe car ils offres une vue bidimentionnelle des données (comme un tableau excel), et c'est exactement ce que l'on va chercher à utiliser pour nos modèles.

3.3. Scikit-learn

une bibliothèque Python, fournit une large gamme d'algorithmes d'apprentissage supervisé et non supervisé. Elle s'appuie sur des technologies familières telles que NumPy, pandas et Matplotlib.

3.4. TensorFlow

est une bibliothèque de deep Learning open source développée par Google utilisée pour effectuer des opérations numériques complexes et plusieurs autres tâches pour modéliser les architectures de Deep Learning. Il peut déployer facilement des calculs sur plusieurs plates-formes comme les CPU, les GP

3.5. Keras

est une API de haut niveau qui vise à créer et à entraîner des modèles de Deep Learning basé sur python. Elle a été développée dans le but de permettre des expérimentations rapides.

3.6. Matplotlib

est une bibliothèque complète permettant de créer des visualisations statiques, animées et interactives en Python. Matplotlib rend les choses faciles et les choses difficiles possibles.

3.7. Os

Le module OS en python fournit des fonctions d'interaction avec le système d'exploitation. 'OS' fait partie des modules utilitaires standard de Python. Ce module fournit un moyen portable d'utiliser les fonctionnalités dépendant du système d'exploitation. Les modules 'os' et 'os.path' incluent de nombreuses fonctions pour interagir avec le système de fichiers.

3.8. PyTorch

C'est une bibliothèque open-source d'apprentissage automatique développée par Facebook, principalement utilisée pour le développement et la mise en œuvre de réseaux de neurones profonds

4. Étapes de classification

Voici les étapes pour mettre en œuvre le système:

4.1. Importation des bibliothèques

Nous avons utilisé les libraires suivantes :

```
#Importer les bibliothèques néccessaires.
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import LabelEncoder
import numpy as np
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import SMOTES
```

Figure 1 : Importer les bibliothèques néccessaires

4.2. Concaténation des tableaux

Nous avons utilisé la fonction pd.read_csv () de la bibliothèque pandas pour charger un fichier CSV dans un DataFrame. Dans notre cas, le fichier CSV que nous avons chargé Il est localisé dans le chemin spécifié.

```
#Concaténation des tableaux
d1 = pd.read_csv('/kaggle/input/network-intrusion-dataset/Friday-WorkingHours-Morning.pcap_ISCX.csv')
d2 = pd.read_csv('/kaggle/input/network-intrusion-dataset/Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv')
d3 = pd.read_csv('/kaggle/input/network-intrusion-dataset/Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv')
d4 = pd.read_csv('/kaggle/input/network-intrusion-dataset/Monday-WorkingHours.pcap_ISCX.csv')
d5 = pd.read_csv('/kaggle/input/network-intrusion-dataset/Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv')
d6 = pd.read_csv('/kaggle/input/network-intrusion-dataset/Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv')
d7 = pd.read_csv('/kaggle/input/network-intrusion-dataset/Tuesday-WorkingHours.pcap_ISCX.csv')
d8 = pd.read_csv('/kaggle/input/network-intrusion-dataset/Wednesday-workingHours.pcap_ISCX.csv')
df = pd.concat([d1,d2,d3,d4,d5,d6,d7,d8], axis=0)
```

Figure 2 : Chargement de données

4.3. Extraction caractéristiques et Labels

La dernière colonne du DataFrame df contient les étiquettes de données.

```
df[' Label'].value_counts().sum
```

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of  Label
BENIGN                        2273097
DoS Hulk                       231073
PortScan                       158930
DDoS                           128027
DoS GoldenEye                   10293
FTP-Patator                      7938
SSH-Patator                      5897
DoS slowloris                    5796
DoS Slowhttptest                 5499
Bot                              1966
Web Attack ? Brute Force         1507
Web Attack ? XSS                  652
Infiltration                       36
Web Attack ? Sql Injection         21
Heartbleed                         11
Name: count, dtype: int64>
```

Figure 3 : Extraction de la colonne de données cible du DataFrame

4.4. convertir le type de donnes

Notre type de dataset définie Object nous utilisant le command ci-dessus qui faire la conversion Object ver numérique.

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df[' Label']= encoder.fit_transform(df[' Label'])
```

Figure 4 : convertir le type de donnes

4.5. Nettoyage des données

Nettoyage de données est l'étape la plus importante avant d'analyser ou modéliser des données.

```
#Nettoyage des données..
df = df.fillna(0)  # Replace NaN with 0
df = df.replace([np.inf, -np.inf], 0)
```

```
df.isnull().sum()
```

```
 Destination Port                0
 Flow Duration                   0
 Total Fwd Packets               0
 Total Backward Packets          0
Total Length of Fwd Packets      0
                                ..
Idle Mean                        0
 Idle Std                        0
 Idle Max                        0
 Idle Min                        0
 Label                           0
Length: 79, dtype: int64
```

Figure 5: nettoayer les donnes

4.6. Normalisation des données

Cette étape est couramment utilisée dans le prétraitement des données pour améliorer les performances des modèles d'apprentissage automatique, en particulier pour les algorithmes sensibles à l'échelle des caractéristiques.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Figure 6: normaliser des donnes

4.7. Étape de division

la bibliothèque scikit-learn (sklearn) en Python, qui est largement utilisée pour diviser un ensemble de données en ensembles d'entraînement et de test.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3, random_state=42)
```

Figure 7: Séparation des données

4.8. Création du modèle autoencoder

```
class AutoEncoder(Model):
    def __init__(self, input_dim, latent_dim):
        super(AutoEncoder, self).__init__()
        self.input_dim = input_dim
        self.latent_dim = latent_dim

        self.encoder = tf.keras.Sequential([
            layers.Input(shape=(input_dim,)),
            layers.Reshape((input_dim, 1)),  # Reshape to 3D for Conv1D
            layers.Conv1D(128, 3, strides=1, activation='relu', padding="same"),
            layers.BatchNormalization(),
            layers.MaxPooling1D(2, padding="same"),
            layers.Conv1D(128, 3, strides=1, activation='relu', padding="same"),
            layers.BatchNormalization(),
            layers.MaxPooling1D(2, padding="same"),
            layers.Conv1D(latent_dim, 3, strides=1, activation='relu', padding="same"),
            layers.BatchNormalization(),
            layers.MaxPooling1D(2, padding="same"),
        ])
        # Previously, I was using UpSampling. I am trying Transposed Convolution this time around.
        self.decoder = tf.keras.Sequential([
            layers.Conv1DTranspose(latent_dim, 3, strides=1, activation='relu', padding="same"),
#             layers.UpSampling1D(2),
            layers.BatchNormalization(),
            layers.Conv1DTranspose(128, 3, strides=1, activation='relu', padding="same"),
#             layers.UpSampling1D(2),
            layers.BatchNormalization(),
            layers.Conv1DTranspose(128, 3, strides=1, activation='relu', padding="same"),
#             layers.UpSampling1D(2),
            layers.BatchNormalization(),
            layers.Flatten(),
            layers.Dense(input_dim)
        ])

    def call(self, X):
        encoded = self.encoder(X)
        decoded = self.decoder(encoded)
        return decoded


input_dim = X_train.shape[-1]
latent_dim = 32

model = AutoEncoder(input_dim, latent_dim)
model.build((None, input_dim))
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.01), loss="mae")
model.summary()
```

Figure 8 : Création du modèle

4.9. Les couches de auto-encodeur

```
Model: "auto_encoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 10, 32)            63264

 sequential_1 (Sequential)   (None, 79)                167151

=================================================================
Total params: 230415 (900.06 KB)
Trainable params: 229263 (895.56 KB)
Non-trainable params: 1152 (4.50 KB)
_________________________________________________________________
```

Figure 9 : Résumé du modèle construit

4.10. Entraînement du modèle

```
Epoch 1/50
11188/11188 [==============================] - 649s 58ms/step - loss: 650400.0625 - val_loss: 506165.
9062
Epoch 2/50
11188/11188 [==============================] - 643s 58ms/step - loss: 335628.8125 - val_loss: 271590.
4062
Epoch 3/50
11188/11188 [==============================] - 646s 58ms/step - loss: 146122.2969 - val_loss: 183092.
0625
Epoch 4/50
11188/11188 [==============================] - 645s 58ms/step - loss: 98399.8984 - val_loss: 196015.6
094
Epoch 5/50
11188/11188 [==============================] - 636s 57ms/step - loss: 78212.5156 - val_loss: 382857.4
062
Epoch 6/50
11188/11188 [==============================] - 637s 57ms/step - loss: 65739.3047 - val_loss: 187292.4
531
Epoch 7/50
11188/11188 [==============================] - 633s 57ms/step - loss: 58160.9805 - val_loss: 132306.7
656
Epoch 8/50
11188/11188 [==============================] - 633s 57ms/step - loss: 51849.0703 - val_loss: 183636.1
250
```

```
Epoch 9/50
11188/11188 [==============================] - 642s 57ms/step - loss: 47754.7148 - val_loss: 328592.4
375
Epoch 10/50
11188/11188 [==============================] - 641s 57ms/step - loss: 43978.0000 - val_loss: 282408.2
188
Epoch 11/50
11188/11188 [==============================] - 641s 57ms/step - loss: 40954.3945 - val_loss: 156910.3
594
Epoch 12/50
11188/11188 [==============================] - 641s 57ms/step - loss: 39494.0938 - val_loss: 136729.3
906
Epoch 13/50
11188/11188 [==============================] - 640s 57ms/step - loss: 36886.3945 - val_loss: 456797.7
500
Epoch 14/50
11188/11188 [==============================] - 640s 57ms/step - loss: 35581.3828 - val_loss: 152773.2
812
Epoch 15/50
11188/11188 [==============================] - 643s 57ms/step - loss: 34820.9180 - val_loss: 138943.8
438
Epoch 16/50
11188/11188 [==============================] - 644s 58ms/step - loss: 32925.9453 - val_loss: 154338.7
188
```

Figure 10 : Entraînement du modèle

5. Conclusion

Dans cette annexe, nous avons exploré en détail le développement de notre système, en fournissant une description complète de sa mise en œuvre. Nous avons examiné les différentes composantes et étapes clés qui ont contribué à la réalisation du système.