REPUBLIQUE DU SENEGAL



Un Peuple - Un But - Une Foi

UNIVERSITE GASTON BERGER
L'excellence au service du développement

Université Gaston Berger de Saint-Louis

UFR DE SCIENCES APPLIQUEES ET DE TECHNOLOGIE

DEPARTEMENT D'INFORMATIQUE

MASTER SCIENCE ET TECHNOLOGIE

……MASTER 2……

Réseaux Sécurité et Systèmes Distribués

……R2SD……

Mise en Place d'une Détection Précoce avec des Algorithmes du Machine Learning dans un Réseau Informatique : Cas du DDoS

Présenté par :

MAMADOU BAILO BAH

Encadré par :

Pr Chérif DIALLO

Année académique 2022/2023

REMERCIEMENTS

Je tiens à exprimer mes plus sincères remerciements à **Ma Famille**,

- ✓ Véritable pilier de mon parcours académique. Leur soutien indéfectible, leurs encouragements constants et leur amour inconditionnel ont été des sources d'inspiration essentielles tout au long de la rédaction de ma mémoire de master.

Un grand merci à **Mon Encadrant, Pr Chérif Diallo**

- ✓ Dont les conseils éclairés, l'expertise et la disponibilité ont grandement contribué à la réussite de mon travail. Sa guidance précieuse a été un facteur déterminant dans l'aboutissement de ce projet.

À **Mes Amis**, **Daouda Basséne, Malal Dém, Et tout la famille Ndiaye**

- ✓ Je tiens à exprimer ma reconnaissance pour leur amitié, leur compréhension et leur soutien constant. Leurs discussions stimulantes ont enrichi mon expérience universitaire.

Je souhaite également adresser mes remerciements aux enseignants de l'UGB, qui m'ont transmis leurs connaissances et ont contribué à forger mon esprit critique.

Merci infiniment, **Ngoné Ndiaye**, pour ta générosité et ton impact positif, qui ont grandement enrichi cette réalisation.

Enfin, un merci tout particulier à **Mon Père**,

- ✓ Figure exemplaire de détermination et de sagesse. Son amour, ses conseils éclairés et son soutien indéfectible ont été des piliers majeurs de ma réussite académique.

Ces remerciements reflètent ma profonde gratitude envers ceux qui ont joué un rôle significatif dans la réalisation de ma mémoire de master. Merci du fond du cœur pour votre soutien et votre impact positif dans cette aventure académique.

DEDICACE

À mes chers parents,

À ma mère OUMOU LAMA BAH *et à* **mon père MAMADOU YAYA BAH**, *piliers de ma vie, dont l'amour et le dévouement ont été mes boussoles. Votre soutien indéfectible a illuminé chaque étape de ce parcours, et cette dédicace est le témoignage de ma gratitude éternelle.*

À mes frères et sœurs, *compagnons de vie et complices de toujours. Nos liens familiaux ont façonné ma personne, et je dédie ce travail à notre unité et à nos précieux moments partagés.*

À mes amis, *précieux joyaux qui ont enrichi mon existence de rires, de partages et d'encouragements. Cette dédicace symbolise la force de nos amitiés, ancrées dans le temps et l'échange.*

Ensemble, vous formez le socle sur lequel repose cette réussite. Merci d'avoir été mes piliers, mes confidents et mes inspirations. Cette réussite est la vôtre autant qu'elle est mienne.

Avec tout mon amour,

MAMADOU BAILO BAH

TABLE DES MATIERES

Université Gaston Berger de Saint-Louis I
REMERCIEMENTS I
DEDICACE II
TABLE DES MATIERES III
LISTE DES FIGURES VIII
LISTE DES TABLEAUX X
SIGLES ET ABREVIATIONS XI
INTRODUCTION GENERALE 1

CHAPITRE I : MENANCES de VULNERABILITES DANS LES RESEAUX INFORMATIQUES 2

INTRODUCTION 3
I.1 Définitions 3
I.2 Etudes des risques lié à la sécurité informatique 3
I.2.1 Types de risques informatiques 5
I.2.1.1 Risques Humains 5
I.2.1.2 Risques Matériel 6
I.2.2 Gestion des risques 6
I.2.2.1 Les risques potentiels 7
I.2.2.3 Formation et sensibilisation des utilisateurs 8
I.2.3 Etablissement et élément d'une politique de sécurité informatique 9
I.2.4 Les Défauts de sécurité informatiques 10
II Objectifs et différents types d'attaques informatiques 10
II.1. But des attaques informatique 10
II.2. Anatomie d'une attaque 12
II.3 Différents Classes des d'attaques informatiques 12
II.3.1 Classification selon l'effet de l'attaque 12
II.3.2 Classification selon la source de l'attaque 13
II.3.3 Classification selon la cible de l'attaque 13
II.3.4 Exemples d'attaques 13
II.3.4.1 Attaques par Dénis de Service (DoS) : 13
II.3.4.2 Attaques Distributed Denial of Service (DDoS 14
II.3.4.3 L'usurpation d'adresse IP (IP Spoofing) 14
II.3.4.4 Les analyseurs réseau (sniffer) 15

II.3.4.6 Les attaques de force brute 15
II.3.4.7 Les attaques de logiciel malveillant (Malware) 15
II.3.4.8 Les attaques par phishing 15
II.3.4.9 Les attaques de réseau sans fil 16
II.3.4.10 Les attaques par injection de code 16
II.3.4.11 Social engineering 16
II.4 Les failles exploitées 16
II.7 Mécanismes de défense contre les attaques 17
CHAPITRE II : SYSTEMES DE DETECTION D'ANOMALIES & DETECTION PRECOCE 19
INTRODUCTION 20
II.1 SYSTEMES DE DETECTION D'ANOMALIES 20
II.1.1 Introduction 20
II.1.2 Définition 20
II.1.3 Architecture et principe de fonctionnement d'un IDS 21
II.1.3.1 Architecture d'un IDS 21
II.1.3.2 Principe de fonctionnement d'un IDS 22
II.1.4 Classification des systèmes de détection d'intrusion 22
II.1.4.1 Méthode de détection 23
II.1.4.1.1 Approche Comportementale 23
II.1.4.1.2 Approche Scénarios 25
II.1.4.1.3 Autres Critères 27
II.1.4.2. Comportement après la détection ou types de réponses 28
II.1.4.2.1 IDS à réponse passive 28
II.1.4.2.2 IDS à réponse active 28
II.1.4.3 Emplacement de données 28
II.1.4.4 Fréquence d'utilisation 29
II.1.5 Emplacement des systèmes de détections d'Intrusions 29
II.1.5.1 Système de détection d'intrusion de type Hôte (HIDS) 29
II.1.5.2 Système de détection d'intrusion de type réseau (NIDS) 30
II.1.5.3 Détection d'intrusion hybrides 30
II.1.6 Avantages des systèmes de détections d'Intrusions 30
II.1.7 Limites des systèmes de détections d'Intrusions 31
II.2 Détection précoce dans un réseau informatiques 31
II.2.1 Définition 31

II.2.2 Architecture et principe de fonctionnement d'une détection précoce dans un Réseau 32
II.2.2.1 Architecture des alertes Précoces dans un réseau 32
II.2.2.2 Fonctionnement d'une détection précoce dans un réseau 33
II.2.3 Emplacement et avantage d'une détection précoce dans un réseau 34
II.2.3.1 Emplacement d'une alerte précoce dans un réseau 34
II.2.3.2 Avantages d'une détection précoce dans un réseau 35
II.3 Conclusion 35
CHAPITRE III : IMPORTANCE DU DEEP LEARNING DANS LES ATTAQUES RESEAUX . 37
INTRODUCTION 38
III.1 Notion de base sur l'apprentissage automatique (Machine Learning Basics) et sur l'apprentissage en Profondeur (Deep Learning) 38
III.1.1 Notion de base sur l'apprentissage automatique (Machine Learning Basics) 38
III.1.1.1 L'apprentissage automatique 38
III.1.1.2 Types d'apprentissage (Types of Learning) 39
III.1.1.3 ML vs. Deep Learning 39
III.1.2 Notion de base sur l'apprentissage en profondeur (Deep Learning) 40
III.1.2.1 Qu'est-ce que l'apprentissage en profondeur (DL) 40
III.1.2.2 Pourquoi DL est-il utile ? 41
III.2 Classification et Réseau de neurones 42
III.2.1 Classification 42
III.2.1.1 Concepts & Définition 42
III.2.1.3 Taxonomie de Classification 42
III.2.1.3.1 Classification Exclusive 43
III.2.1.3.1.1 Classification non supervisée 43
III.2.1.3.1.1.1 - Classification ascendante hiérarchique (CAH) : [31] 43
III.2.1.3.1.1.2 -Méthode des K-means (Centres mobiles) 43
III.2.1.3.1.2 Classification Supervisé 44
III.2.1.3.1.2.1 - K plus proches voisins 44
III.2.1.3.1.2.2 Arbres de décision 45
III.2.1.3.2 Classification Non Exclusive 45
III.2.2 Réseau de Neurones (Neural Network) 45
III.2.2.1 Concepts & Définitions 45
III.2.2.1.1 Neurone Biologique 46
III.2.2.1.2 Neurone Artificiel 46
III.2.2.1.3 Correspondance entre neurone biologique et neurone artificiel 47

III.2.2.2 Comportement de Neurone Artificiel 48
III.2.2.3 Les Types d'Apprentissage des Neurones 50
III.2.2.3.1 Apprentissage Non Supervisé 50
III.2.2.3.2 Apprentissage par renforcement 50
III.2.2.3.3 Apprentissage Supervisé 51
III.2.2.4 Règles d'apprentissage 51
III.2.2.5 Quelques Type de Modèle de réseaux de neurones 53
III.2.2.5.1 Perceptron Simple 53
III.2.2.5.2 Réseaux auto-organisateur (Réseau de Kohonen) 54
III.2.2.5.3 Perceptron Multicouche (MLP) 55
III.3 Quelques Algorithmes de Classifications 56
III.3.1 Naïve Bayes 56
III.3.2 Arbre de Décision 57
III.3.3 Machine à Vecteur de Support (SVM) 57
III.3.4 L'algorithme des K plus proche voisin (KNN) 58
III.3.4.1 Définition et Historique 58
III.3.4.2 Principe de l'algorithme 58
III.3.4.3 Calcul de Similarité dans l'algorithme de K-NN 59
III.3.4.3.1 Calcul de distance 59
III.3.4.3.2 Choix de K 60
III.4 Deep Learning de la sécurité des réseaux 60
IV.4.1 Identification de la circulation 61
IV.4.2 Facteurs de la sécurité réseau 61
IV.4.3 Technique utilisée pour détecter les anomalies dans la sécurité du réseau 62
III.5 CONCLUSION 63
CHAPITRE IV : MISE EN PLACE D'UNE DETECTION PRECOCE AVEC DES ALGORITHMES DU MACHINE LEARNING : CAS DU DDoS 64
IV.1 INTRODUCTION 65
IV.2 Mise en place du détection précoce avec Deep Learning 65
IV.2.1 Conception 65
IV.2.2 Outils de réalisation 66
IV.2.2.1 Choix du langage de programmation 66
IV.2.3 Méthodologies de recherche 67
IV.2.4 Présentation et exploration du dataset 67
IV.2.5 Importation des modules d'apprentissage 68
IV.2.6 Chargement des données du dataset 69

IV.2.7 Modèles D'apprentissage Et métriques 70
IV.2.7.1 LogisticRegression 70
IV.2.7.2 Machine à Vecteur de Support (SVM) 71
IV.2.7.3 K plus Proche Voisin 72
IV.2.7.4 Arbre de Décision 74
IV.2.7.5 RandomForest (Foret Aléatoire) 75
IV.2.7.6 Serveur gmail_smtp 76
IV.2.8 Fonction de détection précoce 76
IV.3 Analyse des données obtenus 78
IV.4 Les Modèles classiques d'apprentissage automatique : métriques & interprétation des résultats obtenus 87
I.V.4.1Analyse des performances du modèle de Régression Logistique : Précision, Choix du Solveur, et Temps d'Exécution 87
IV.4.2 Performance Comparative des Noyaux SVM dans la Classification (SVM) 88
IV.4.3 Optimisation des Hyperparamètres pour un Modèle d'Arbre de Décision 89
IV.4.4 Performance Exceptionnelle : Évaluation du Modèle Random Forest 92
IV.4.5 Performance Exceptionnelle : Évaluation du Modèle KNN 92
IV.5 Configuration du Paramètres SMTP Gmail pour l'alerte 95
IV.4 Conclusion 99
REFERENCE ET BIBLIOGRAPHIE 101

LISTE DES FIGURES

Figure I. 1 : Etat des Risques[2] 8
Figure I. 2 : Flux Normale de Transmission 10
Figure I. 3 : Cas D'interruption du Traffic 11
Figure I. 4 : Cas D'interception 11
Figure I. 5 : Cas de Modification 11
Figure I. 6 : Cas de Fabrication 12
Figure I. 7: Attaque DoS 14
Figure I. 8: Attaques DDoS[7] 14
Figure I. 9 : Attaques Man-in-the middle[2] 15

Figure II. 1 : Architecture d'un IDS [14] 21
Figure II. 2 : Fonctionnement d'un IDS [15] 22
Figure II. 3 : Classification des IDS [16] 23
Figure II. 4 : Méthode de détection [17] 23
Figure II. 5 : Approche Comportementale [18] 24
Figure II. 6 : Approche par Scénario ou Signature [18] 26
Figure II. 7 : Réponse Active [22] 28
Figure II. 8 : Détection avec HIDS [24] 30
Figure II. 9 : Schéma d'un NIDS [25] 30

Figure III. 1: Composant de IA [26] 38
Figure III. 2 : Méthode permettant d'apprendre et de prédire des données[22] 39
Figure III. 3 : Machine Learning en Pratique[28] 40
Figure III. 4 : Machine Learning VS Deep Learning[28] 40
Figure III. 5 : Apprentissage en profondeur dans la reconnaissance vocale[29] 41
Figure III. 6 : Google Trends l'Utilisation DL vers ML[29] 41
Figure III. 7 : Les méthodes de classification [30] 42
Figure III. 8: Exemple de dendrogramme [30] 43
Figure III. 9 : Exemple de Partition obtenue par les centres mobiles[31] 44
Figure III. 10 : Structures d'un Neurones Biologiques [35] 46
Figure III. 11 : Structures d'un Neurones Artificiel [35] 47
Figure III. 12 : Correspondance entre neurone artificiel et neurone biologique [35] 47
Figure III. 13 : Apprentissage d'un réseau de neurones 51
Figure III. 14 : Schéma général de perceptron Simple [38] 54
Figure III. 15 : Carte auto-adaptative de kohonen [38] 55
Figure III. 16 : Exemple de réseau de type perceptron Multicouche [38] 56
Figure III. 17 :Formule de naive bayes 56
Figure III. 18 : Schéma d'illustration 57
Figure III. 19 : Machine à Vecteurs de support (SVM) 58
Figure III. 20 : Calcul de la distance euclidienne 59
Figure III. 21: Distance de Manhattan 59
Figure III. 22: Algorithme KNN 60
Figure III. 23: Structure of Deep Learning [41] 61
Figure III. 24 : Analyse les données et détecte les activités suspectes à l'aide d'un apprentissage automatique non supervisé.[41] 62

Figure IV. 1: Solution 65
Figure IV. 2 : Phase de l'Algorithme de Classification 66
Figure IV. 3 : Ensemble de données du Dataset 68
Figure IV. 4 : Module utilisé 69
Figure IV. 5: Lecture et Affichage des données 70
Figure IV. 6: Définition de notre Modèle 70
Figure IV. 7: LogisticRégression 71
Figure IV. 8: Code Support Vecteur Machine 72
Figure IV. 9: Code Métrique Pour KNN 74
Figure IV. 10: Arbre de Décision 75
Figure IV. 11: Code de RandomForest 76
Figure IV. 12: Fonction d'envoi 76
Figure IV. 13: Envoi Avec SMTP_SSL 76
Figure IV. 14: Détection Précoce 78
Figure IV. 15: Présentation du Dataset 79
Figure IV. 16: Classification des requêtes bon et mauvais 80
Figure IV. 17: Valeurs Manquantes 81
Figure IV. 18: Analyse Exploratoire de la Base de Données Réseau 81
Figure IV. 19: Nombre Total de Requêtes 82
Figure IV. 20: Nombre de Requêtes d'attaques 82
Figure IV. 21: Nombre de requêtes 83
Figure IV. 22: Requêtes Des Différents Protocoles 84
Figure IV. 23: Durée 84
Figure IV. 24: Octets Transmis 85
Figure IV. 25: Octets envoyé par Seconde 85
Figure IV. 26: Répartitions des valeurs de SWITCH 86
Figure IV. 27: Caractéristiques de 'SWITCH' lors d'événements malveillants 87
Figure IV. 28: Évaluation approfondie de la Régression Logistique : Précision, Sélection du Solveur et Durée d'Exécution 88
Figure IV. 29: Évaluation et Sélection de Noyaux SVM pour la Prédiction 89
Figure IV. 30: Recherche des Meilleurs Paramètres : Modèle d'Arbre de Décision 89
Figure IV. 31: Évaluation Approfondie d'un Modèle avec un Arbre de Décision 91
Figure IV. 32: Optimisation des hyperparamètres 91
Figure IV. 33: Précision Maximale : Analyse des Résultats du Modèle Random Forest 92
Figure IV. 34: Analyse approfondie des Résultats du Modèle KNN 93
Figure IV. 35: Nombre de K 94
Figure IV. 36: Résultat du Détection précoce DDoS 95
Figure IV. 37: Configuration du Messagerie : Aller sur Sécurité 95
Figure IV. 38: Validation en deux étapes 95
Figure IV. 39: Saisir mot de passe 96
Figure IV. 40: Aller sur mot de passe des applications 97
Figure IV. 41: Saisi du nom de applications pour créer mot de passe 97
Figure IV. 42 : Mot de passe Générer 98
Figure IV. 43 : Le protocole SMTP_SSL et le mail destinataire 99
Figure IV. 44 : Message d'Alerte envoyé 99
Figure IV. 45 : Affichage du Mail reçu dans la boite l'ingénieur 99
Figure IV. 46 : Message lue par le destinataire 99

LISTE DES TABLEAUX

Tableau II. 1 : Comparaison des deux approches 27
Tableau III. 1 : Transition entre les neurones biologiques et les neurones formel 47
Tableau III. 2: Les fonctions de transfert les plus utilisées.....49

SIGLES ET ABREVIATIONS

TERMES	DEFINITION
R	Risque
M	Menace
V	Vulnérabilités
CM	Contre-Mesures
ISO	Organisation International de Normalisation
Dos	Déni de Services
DDoS	Déni de Service Distribué
Nmap	Network mapper
SNMP l	Simple Network Management Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
IDS	Intrusion Detection System
DHCP	Dynamic Host Configuration Protocol
RIP	Routing Information Protocol
ICMP	Internet Control Message Protocol
ARP	Address Resolution Protocol
DNS	Domain Name Service
TCP	Transmission Control Protocol
DNN	Deep Neural Network
IA	Intelligence Artificiel
CESIN	Club des Experts de Sécurité de l'Information et du Numérique
TPE	Très Petite Entreprise
PME	Petites et Moyennes Entreprises
ONU	Organisation des Nations Unies
VoIP	Voice over Internet Protocol
SMS	Short Message Service
MMS	Service de Messagerie Multimédia
PDG	Président Directeur General
HIDS	Host Intrusion Detection System

NIDS	Network Intrusion Detection System
NIDS	Système de Prévention des Intrusions
IDWG	Groupe de Travail sur l'Intégration des Données
IDMEF	Format d'Echange de Messages pour la Détection d'Intrusions
CPU	Unité Centrale de Traitement
OSSEC	Open-Source Security Information and Event Management
DNN	Deep Neural Network
ML	Machine Learning
MLP	Multi layer Perceptron
DL	Deep Learning
IA	Intelligence Artificiel
TIC	Technologies de l'Information et de la Communication
CAH	Classification Ascendante Hiérarchique
XOR	Ou Exclusive
SVM	Machine à Vecteurs de Support
K-NN	K-Nearest Neighbors
GAFAM	Google Apple Facebook Amazon Microsoft
GAR	Gaussian Belief Network
DBN	Deep Belief Network
RBM	Restricted Boltzmann Machine

INTRODUCTION GENERALE

Dans le paysage dynamique de la cybersécurité, la détection précoce des attaques est devenue impérative pour garantir l'intégrité et la disponibilité des réseaux informatiques. Parmi les menaces persistantes, les attaques par déni de service distribué (DDoS) demeurent un défi majeur, capables de paralyser des systèmes et des services critiques. Cette recherche se concentre sur l'élaboration d'une stratégie de détection précoce, en exploitant les avancées du Deep Learning, au sein d'un environnement réseau, avec un accent particulier sur le contexte spécifique des attaques DDoS.

L'évolution constante des techniques d'attaque nécessite des approches de sécurité innovantes et adaptatives. Ainsi, l'utilisation de méthodes basées sur le Deep Learning offre une opportunité prometteuse d'améliorer la capacité des systèmes à anticiper et à réagir rapidement aux attaques. Cette étude s'attache à explorer comment des algorithmes de classification avancés tels que KNN, SVM, RandomForest, et Arbre de Décision, renforcés par les principes du Deep Learning, peuvent être déployés pour une détection précoce efficace des attaques DDoS.

Par ailleurs, l'intégration de données issues de l'environnement SDN vient enrichir la pertinence de la recherche en permettant une adaptation contextualisée des modèles de détection. Cette démarche vise à élever le niveau de sécurité des réseaux informatiques, tout en offrant une contribution significative à la compréhension et à la résilience face aux attaques DDoS.

CHAPITRE I : MENANCES de VULNERABILITES dans les RESEAUX INFORMATIQUES

INTRODUCTION

Le progrès technologique, le développement des moyens de communication, l'ouverture du monde aux nouvelles technologies et la transmission de divers types de données à travers les réseaux, ainsi que d'autres facteurs, ont introduit un risque d'accès et de manipulation non autorisés des données par des individus non autorisés ou des concurrents. Par conséquent, la sécurité de l'information est devenue une nécessité cruciale pour la construction de systèmes sécurisés, visant à éliminer ces vulnérabilités. Dans ce premier chapitre, nous exposons les concepts fondamentaux de la sécurité informatique, en commençant par des définitions des différentes notions liées à la sécurité informatique, puis en abordant les attaques informatiques et leur classification, illustrées par des exemples concrets.

I.1 Définitions

La sécurité informatique, c'est comme un ensemble de protections pour empêcher que des problèmes, qu'ils soient accidentels ou causés par des personnes malveillantes, n'endommagent un système. L'objectif de la sécurité informatique, c'est de s'assurer que tout le matériel et les programmes d'un ordinateur ne sont utilisés que de la manière prévue et par les bonnes personnes. On doit définir clairement ce que les utilisateurs attendent en matière de sécurité informatique :

- **La confidentialité** : Seules les personnes concernées sont autorisées à voir les données. Personne d'autre ne doit pouvoir les regarder. Pour éviter cela, les données sont rendues illisibles sauf pour les personnes qui font partie de la transaction, car elles ont une sorte de clé spéciale pour les comprendre.
- **L'intégrité** : Il est important de s'assurer en tout temps que les données que l'on envoie sont les bonnes et qu'elles n'ont pas été modifiées d'une manière incorrecte pendant leur transfert. Pour ce faire, nous devons vérifier que les données sont complètes, exactes, vraies et valides.
- **La disponibilité** : Il est essentiel que le système fonctionne correctement et que l'on puisse utiliser un service ou des choses quand on en a besoin. On peut mesurer si quelque chose est disponible en regardant combien de temps cela fonctionne par rapport au temps où ça aurait dû fonctionner.
- **La non-répudiation** : Personne ne peut dire qu'ils n'ont pas fait une transaction, car il y a une preuve que les données ont été reçues. Cette preuve est obtenue à l'aide de certificats numériques et d'une clé spéciale que seul le propriétaire connaît.
- **L'authentification :** La sécurité informatique permet seulement aux bonnes personnes d'entrer. Avant de partager des informations, il faut être sûr de qui est la personne qui les demande. En résumé, la sécurité d'un système est aussi solide que la partie la plus fragile. Donc, même si le système est très bien protégé par la technologie, si les gens qui l'utilisent ne font pas attention, cela peut rendre tout le système vulnérable.

I.2 Etudes des Risques liée à la sécurité informatique

Les problèmes informatiques peuvent coûter cher, tout comme les mesures de sécurité pour les prévenir. Il est important de faire une étude pour identifier les problèmes qui pourraient survenir et combien cela coûterait pour les résoudre. Ensuite, on doit organiser ces solutions de

manière à former une politique de sécurité logique, en fonction de combien de risques on est prêt à prendre. Cela nous donne une liste de ce que nous devons protéger. Il faut cependant prendre conscience que les principaux risques restent : *câble arraché, coupure secteur, crash disque, mauvais profil utilisateur ...* **[1]**

Voici quelques éléments pouvant servir de base à une étude de risque :

- Quelle est la valeur des équipements, des logiciels et surtout des informations ?
- Quel est le coût et le délai de remplacement ?
- Faire une analyse de vulnérabilité des informations contenues sur les ordinateurs en réseau (programmes d'analyse des paquets, logs…).
- Quel serait l'impact sur la clientèle d'une information publique concernant des intrusions sur les ordinateurs de la société ?

Avec l'essor d'Internet, de nombreuses entreprises permettent à leurs partenaires ou fournisseurs d'accéder à leur système informatique, ce qui devient plus complexe avec une structure en trois étapes ou davantage. Il est donc essentiel de savoir quelles parties de l'entreprise doivent être protégées et comment contrôler qui peut accéder au système et ce qu'ils sont autorisés à faire. Cependant, la sécurité informatique doit trouver un équilibre entre les coûts, les risques et les contraintes. On comprendra mieux le poids d'un risque en se fiant à la formule suivante :[1]

$$\boldsymbol{Risque} = \frac{\textbf{Menace x Vulnerabilite}}{\textbf{Contre mesure}}$$

Risque : C'est la probabilité qu'une menace exploite une vulnérabilité. Autrement dit, c'est une possibilité qu'un fait dommageable se produise.

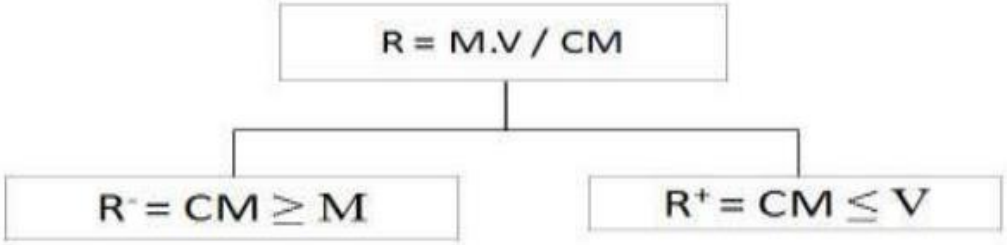
Vulnérabilité : C'est une faiblesse inhérente à un système (software ou hardware). Appelée parfois faille ou brèche, elle représente le niveau d'exposition face à la menace dans un contexte particulier.

Menace : c'est le danger (interne ou externe) tel qu'un hacker, un virus, etc.

Contre-mesure : c'est un moyen permettant de réduire le risque dans une organisation.

Conséquences de la formule :

- Le risque est d'autant plus réduit que les contre-mesures sont nombreuses ;
- Le risque est plus important si les vulnérabilités sont nombreuses.



L'utilisation de l'outil informatique est susceptible de nous exposer à plusieurs types des risques. Il importe donc de pouvoir mesurer ces risques en fonction de la probabilité ou de la fréquence de leurs survenances et aussi en mesurant leurs effets possibles. Ces effets peuvent avoir des conséquences négligeables ou catastrophiques

- Le traitement informatique en cours échoue : il suffit de le relancer, éventuellement par une autre méthode si on craint que la cause ne réapparaisse ;
- L'incident est bloquant et on doit procéder à une réparation ou une correction avant de poursuivre le travail entrepris. Il est cependant à noter que ces mêmes incidents peuvent avoir des conséquences beaucoup plus fâcheuses :
- Données irrémédiablement perdues ou altérées, ce qui les rend inexploitables par la suite;
- Données ou traitements durablement indisponibles, pouvant entrainer l'arrêt d'une production ou d'un service ;
- Divulgation d'informations confidentielles ou erronées pouvant profiter à des sociétés concurrentes ou nuire à l'image de marque de l'entreprise ;
- Déclenchement d'actions pouvant provoquer des accidents physiques ou induire des humains.

I.2.1 Types de Risques Informatiques

Les Risques peuvent être : ***Des Risques Humain, Risques Matériels*** **[1]**

I.2.1.1 Risques Humains

Ce sont les plus importants, même s'ils sont le plus souvent ignorés ou minimisés. Ils concernent les utilisateurs mais également les informaticiens eux-mêmes.

On peut citer :

- **La maladresse :** commettre des erreurs ou exécuter de traitement non souhaité, ou effacer involontairement des données ou des programmes ; etc.
- **L'inconscience et l'ignorance :** Certains utilisateurs peuvent involontairement installer des programmes nuisibles sans le savoir, par exemple, en ouvrant un email. Beaucoup de gens qui utilisent des ordinateurs ne sont pas conscients ou ne connaissent pas les dangers qu'ils peuvent créer pour les systèmes qu'ils utilisent. Ils peuvent également effectuer des actions imprudentes, que ce soit avec des logiciels ou du matériel ;
- **La malveillance** : Au cours des dernières années, il est devenu difficile de ne pas être au courant des problèmes liés aux virus et aux vers informatiques. Il arrive que certaines personnes sabordent délibérément les systèmes informatiques en y insérant sciemment des virus ou en ajoutant intentionnellement de fausses informations dans une base de données. Cette forme de comportement est parfois qualifiée de "cybercriminalité" ;
- **L'ingénierie sociale :** une méthode pour obtenir d'une personne des informations confidentielles, que l'on n'est pas normalement autorisé à obtenir, en vue de les exploiter à d'autres fins. Elle consiste à :
  - ✓ Se faire passer pour quelqu'un que l'on n'est pas (en général un administrateur réseau) ;

- ✓ Demander des informations personnelles (nom de connexion, mot de passe, données confidentielles, etc.) en intervenant un quelconque prétexte (problème dans le réseau, modification de celui-ci, etc.) ; Elle peut se faire soit au moyen d'une simple communication téléphonique ; soit par mail, soit en se déplaçant directement sur place.

- **L'espionnage** : En particulier dans le domaine industriel, les entreprises utilisent les mêmes méthodes, parmi d'autres, pour recueillir des informations sur ce que font leurs concurrents, comment ils fabriquent leurs produits, quels projets ils ont en cours, quels nouveaux produits ils prévoient de lancer, comment ils définissent leurs prix, et quelles sont leurs relations avec leurs clients et leurs prospects, etc.

I.2.1.2 Risques Matériel

Ces problèmes sont dus aux inévitables erreurs et dysfonctionnements que connaissent tous les équipements informatiques, qu'ils soient matériels ou logiciels. La fréquence de ces incidents varie en fonction de la qualité de la fabrication et de la rigueur des tests effectués avant la mise en service des ordinateurs et des logiciels. Certaines de ces pannes ont des causes indirectes, voire très difficiles à anticiper. On peut citer :

- **Les incidents liés au matériel :** La plupart des pièces électroniques modernes fabriquées en grande quantité peuvent avoir des problèmes à cause de défauts de fabrication. À un moment donné, elles peuvent cesser de fonctionner. Certaines de ces pannes sont compliquées à détecter parce qu'elles se produisent de temps en temps ou sont rares. Parfois, ces problèmes sont dus à des erreurs dans la façon dont les pièces ont été conçues.
- **Les incidents liés au logiciel :** Ces problèmes sont les plus courants. Les systèmes d'exploitation et les logiciels deviennent de plus en plus compliqués car ils font de plus en plus de tâches différentes. Ils sont créés par de nombreux développeurs, parfois des dizaines, des centaines, voire des milliers. Ces développeurs peuvent faire des erreurs, que ce soit individuellement ou ensemble, et même les meilleures méthodes de travail et les meilleurs outils de vérification ou de test ne peuvent pas éliminer complètement ces erreurs.
- **Les incidents liés à l'environnement :** Les ordinateurs et les réseaux de communication peuvent être endommagés par des changements de température, d'humidité ou par des champs électromagnétiques. Ainsi, il est possible qu'un ordinateur cesse de fonctionner complètement ou de manière intermittente en raison de conditions météorologiques inhabituelles ou de l'impact de systèmes électriques, notamment dans les environnements industriels.

I.2.2Gestion des risques

La gestion des risques informatiques est un ensemble d'opérations de gérer et de diriger les différentes incidences liées à la manipulation de l'outil informatique. La gestion des risques consiste en trois actions majeures :

- Etudier les risques potentiels (identifier/mettre au jour ces risques) ;
- Imposer des règles de sécurité adéquates pour réduire ces risques ;

- Formation des utilisateurs.

I.2.2.1 Les Risques Potentiels

Cette phase consiste à faire un examen intégral de la méthodologie de l'étude des risques informatique en vigueur. Cela se matérialise aux moyens :

- **Définition de l'environnement :** Définition des acteurs et leurs intérêts ; Importance de la sécurité dans la stratégie de l'entreprise ; Type de données impliquées ; Visibilité extérieure de la sécurité (importance pour la clientèle, le public).
- **Etude des menaces :** Identifier la nature de la menace : accidentelles (désastre, bugs…) ou intentionnelles (attaques, vols…) ; S'enquérir des sources de la menace : personnel non autorisé, intrus, logiciel ; Localiser la menace : procédures manuelles, informatique (software, réseau, stockage, hardware), infrastructure (concrète et abstraite).
- **Etude des vulnérabilités :** Etudes des faiblesses engendrées par l'exécution d'une menace.
- **Etude des risques :** Probabilité d'occurrence de ces menaces conduisant à une vulnérabilité.
- **Estimation du risque et du plan stratégique** : Le risque signifie combien d'argent nous pourrions perdre à court, moyen et long terme à cause d'un problème, comparé à combien ça coûterait pour éviter ce problème. Le plan stratégique est un calendrier qui dit quand nous allons mettre en place des mesures de sécurité et comment nous allons les suivre, en prenant en compte nos besoins futurs en matière de sécurité.**[1]**
- **Mise en place du plan de sécurité :** Les règles de sécurité peuvent rendre les choses difficiles pour les utilisateurs, et plus le réseau devient grand, plus ces règles peuvent devenir compliquées. Par conséquent, il est important de concevoir la sécurité informatique de manière à ce qu'elle n'empêche pas les utilisateurs de faire ce dont ils ont besoin et qu'ils puissent utiliser le système en toute confiance. Raison pour laquelle il est nécessaire de définir dans un premier temps une politique de sécurité dont la mise en œuvre s'effectue en quatre phases :
  - ✓ Identifier les besoins en termes de sécurité, les risques informatiques pesant sur l'entreprise et leurs éventuelles conséquences ;
  - ✓ Elaborer des règles et des procédures à mettre en œuvre dans les différents services de l'organisation pour les risques identifiés ;
  - ✓ Surveiller et détecter les vulnérabilités du système d'information et se tenir informé des failles sur les applications et matériels utilisés ;
  - ✓ Définir les actions à entreprendre et les personnes à contacter en cas de détection d'une menace.**[1]**
- **Audit de sécurité** : L'audit de sécurité signifie faire appel à une personne ou à une entreprise de confiance, de préférence spécialisée en sécurité informatique, pour vérifier que les mesures de protection mises en place sont en accord avec la politique de sécurité. En d'autres termes, l'audit vise à s'assurer que toutes les règles de sécurité sont correctement suivies et que toutes les mesures prises s'assemblent de manière logique et cohérente.

NB : **Vulnérabilité + Menace = Risque**

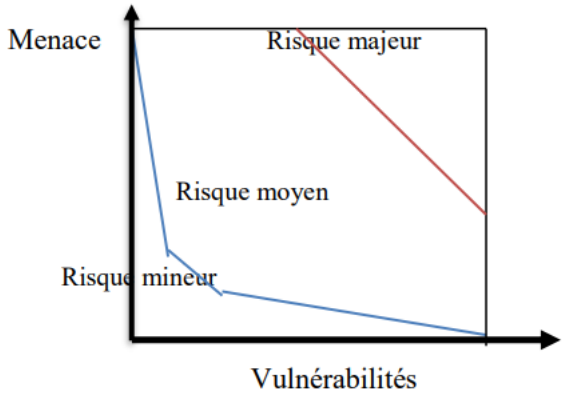


Figure I. 1 : Etat des Risques[2]

I.2.2.2 Règles de Sécurité à Imposer

Ceci consiste en la définition de procédures internes à l'entreprise basées sur ;

- Des règles administratives : Suivre des standards de sécurité (normes ISO) ; Suivre les lois.
- Des règles physiques : Gardes, caméras, alarmes, verrous et Accès aux locaux sécurisés par biométrie.
- Des règles techniques : Déterminer des niveaux de classification des données ; Définir des niveaux d'accès à ces données ; Utiliser la cryptographie pour le traitement et le stockage de l'information ; Mettre en place un firewall matériel et/ou logiciel, ...

I.2.2.3 Formation et sensibilisation des Utilisateurs

Il est de plus en plus admis que la sécurité est essentielle. Les coûts engendrés par les pertes de données dues aux attaques réseaux et autres malwares diminuent sensiblement d'années en années. Il est beaucoup plus simple de corrompre l'utilisateur et ce qui l'entoure que l'algorithme de chiffrement utilisé comme par exemple :

- L'utilisateur ne connait pas les risques engendrés par la conservation de la liste des mots de passe utilisés à côté de l'ordinateur ;
- Il est souvent plus simple de s'introduire dans l'ordinateur de l'utilisateur afin de retrouver le texte en clair (hacking, vol, . . .) ;
- Il est possible de l'espionner, le pousser à la délation, pratiquer le shoulder-surfing ou tout autre technique dite de "social engineering", ... **[2]**

Nous n'allons pas enseigner aux employés comment les algorithmes fonctionnent en détail, mais nous allons leur expliquer comment et quand ils doivent les utiliser en établissant des règles qu'ils ne doivent pas enfreindre. Il y a également plusieurs manières de réagir à un risque, des plus « sûres » aux plus inconscientes :

- Transférer les risques à une compagnie d'assurances ;

- Réduire les risques en implémentant des contre-mesures qui peuvent être :

- Dissuasives : empêcher une attaque ;
- Préventives : faire échouer une attaque ;
- Correctrices : réduire les dommages causés par une attaque ;
- Ignorer/Négliger les risques ;
- Accepter les risques si les contre-mesures sont trop onéreuses

Certes, il y a toujours un risque, aussi infime soit-il. Il faudra donc peser le pour et le contre lors de la mise en place éventuelle d'une contre-mesure. Toutefois, en 2007, on remarque une remontée de la somme totale des pertes, due à la fraude financière**.[1]**

I.2.3 Etablissement et Elément d'une Politique de sécurité Informatique

L'élément de politique de sécurité est l'ensemble des orientations suivies par une organisation en termes de sécurité. Elle est élaborée au niveau de système de pilotage (Direction), car elle concerne tous les utilisateurs du système. La sécurité informatique de l'entreprise repose sur une bonne connaissance des règles par les employés, grâce à des actions de formation et de sensibilisation auprès des utilisateurs, mais elle doit aussi aller au-delà de cela tout en couvrant les champs ci-après :

- Mise en place des correctifs ;
- Définition de la police de sécurité ;
- Objectifs, Portée, Responsables ;
- Une stratégie de sauvegarde correctement planifiée ;
- Description de la sécurité (de l'infrastructure physique, des données informatiques, des applications, du réseau) ;
- Plan en cas de sinistre (Un plan de reprise après incident) ;
- Sensibilisation du personnel aux nouvelles procédures
- Sanctions en cas de manquements.

Suite à l'étude des risques et avant de mettre en place des mécanismes de protection, il faut préparer une politique à l'égard de la sécurité. C'est elle qui fixe les principaux paramètres, notamment les niveaux de tolérance et les coûts acceptables. Voici quelques éléments pouvant aider à définir une politique :

- Quels furent les coûts des incidents informatiques passés ?
- Quel degré de confiance pouvez-vous avoir envers vos utilisateurs internes ?
- Qu'est-ce que les clients et les utilisateurs espèrent de la sécurité ?
- Quel sera l'impact sur la clientèle si la sécurité est insuffisante, ou tellement forte qu'elle devient contraignante ?
- Y a-t-il des informations importantes sur des ordinateurs en réseaux ? Sont-ils accessibles de l'externe ?
- Quelle est la configuration du réseau et y a-t-il des services accessibles de l'extérieur ?
- Quelles sont les règles juridiques applicables à votre entreprise concernant la sécurité et la confidentialité des informations (exemple : loi « informatique et liberté », archives

comptables…) ? Il ne faut pas également perdre de vue que la sécurité est comme une chaîne, guère plus solide que son maillon le plus faible. En plus de la formation et de la sensibilisation permanente des utilisateurs, la politique de sécurité peut être découpée en plusieurs parties : **[2]**

Défaillance matérielle, Défaillance logiciel, Accident (pannes, incendies, inondations…), Erreur Humaine, Vol via des dispositifs physique (disques et bandes), Virus provenant de disquettes, Piratage et virus réseau **[1]**

I.2.4 Les Défauts de Sécurité Informatiques

Les défauts de sécurité peuvent être considérés comme des modifications accidentelles ou inconscientes du fonctionnement normal des équipements informatiques. Les défauts de sécurité d'un système d'information les plus souvent constatés sont :

- Installation des logiciels et matériels par défaut ;
- Mises à jour non effectuées ;
- Mots de passe inexistants ou par défaut ;
- Services inutiles conservés (Netbios…) ;
- Traces inexploitées ;
- Pas de séparation des flux opérationnels des flux d'administration des systèmes ;
- Procédures de sécurité obsolètes ;
- Eléments et outils de test laissés en place dans les configurations en production ;
- Authentification faible ;
- Télémaintenance sans contrôle fort.

II Objectifs et Différents Types d'Attaques Informatiques

II.1. But des Attaques informatique

Il existe plusieurs objectifs pour les attaques :

- Ici on est en présence d'un **Traffic normal**



Figure I. 2 : Flux Normale de Transmission

- **Interruption :** vise la **disponibilité** des informations (DoS, . . .)

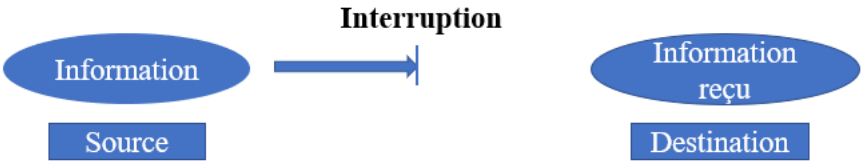


Figure I. 3 : Cas D'interruption du Traffic

- **Interception :** vise la **confidentialité** des informations (capture de contenu, analyse de trafic, . . .).

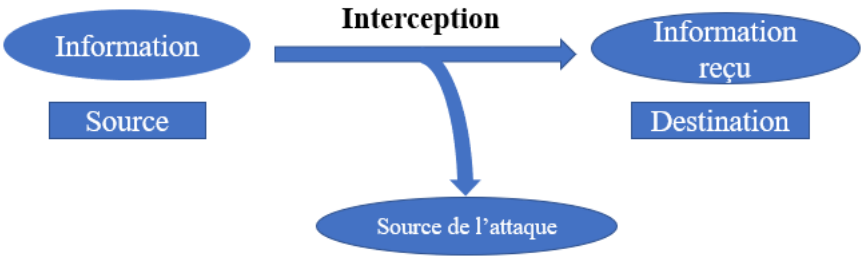


Figure I. 4 : Cas D'interception

- **Modification :** vise **l'intégrité** des informations (modification, rejet, . . .).

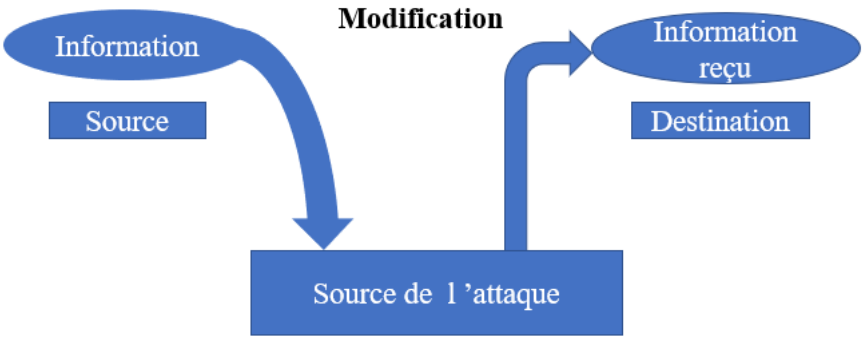


Figure I. 5 : Cas de Modification

- **Fabrication :** vise **l'authenticité** des Informations (Masquerade)

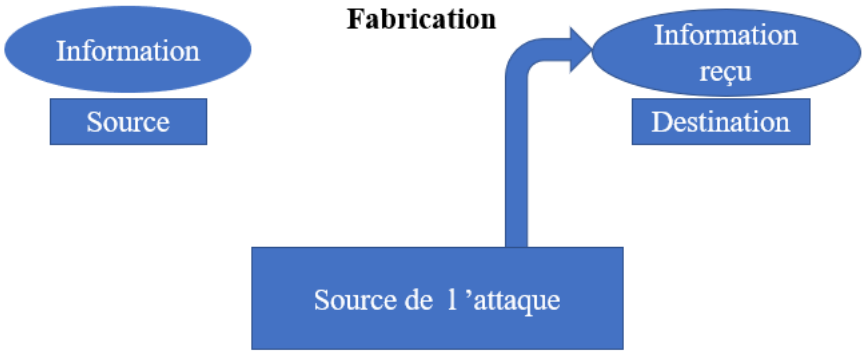


Figure I. 6 : Cas de Fabrication

II.2. Anatomie d'une attaque

Fréquemment appelés « les 5 P », ces cinq verbes anglophones constituent le squelette de toute attaque informatique : **Probe, Penetrate, Persist, Propagate, Paralyze**.

Observons le détail de chacune de ces étapes :

Probe : consiste en la collecte d'informations par le biais d'outils comme whois, Arin, DNS lookup. La collecte d'informations sur le système cible peut s'effectuer de plusieurs manières, comme par exemple un scan de ports grâce au programme Nmap pour déterminer la version des logiciels utilisés, ou encore un scan de vulnérabilités à l'aide du programme Nessus. Pour les serveurs web, il existe des outils comme Nikto qui permet de rechercher les failles connues ou les problèmes de sécurité. Des outils comme hping ou SNMP Walk permettent quant à eux de découvrir la nature d'un réseau. **[3]**

Penetrate : utilisation des informations collectées pour pénétrer un réseau. Des techniques comme le brute force ou les attaques par dictionnaires peuvent être utilisées pour outrepasser les protections par mot de passe. Une autre alternative pour s'infiltrer dans un système est d'utiliser vulnérabilités du système.

Persist : création d'un compte avec des droits de super utilisateur pour pouvoir se réinfiltrer ultérieurement. Une autre technique consiste à installer une application de contrôle à distance capable de résister à un reboot (ex : un cheval de Troie).

Propagate : cette étape consiste à observer ce qui est accessible et disponible sur le réseau local.

Paralyze : cette étape peut consister en plusieurs actions. Le pirate peut utiliser le serveur pour mener une attaque sur une autre machine, détruire des données ou encore endommager le système d'exploitation dans le but de planter le serveur. Après ces cinq étapes, le pirate peut éventuellement tenter d'effacer ses traces, bien que cela ne soit rarement utile**.[5]**

II.3 Différents Classes des d'attaques Informatiques

II.3.1 Classification selon l'effet de l'attaque

Selon les effets résultant de l'attaque on peut classifier les attaques en deux groupes principaux : les attaques passives et les attaques actives.

- **Les attaques passives :** consistent à accéder, utiliser ou à observer le système cible sans modifier les données ou dysfonctionné les ressources de ce dernier, elles sont généralement indétectables (ex. : capture de contenu, analyse de trafic).
- **Les attaques actives :** consistent à effectuer des changements non autorisés sur les données des systèmes, à s'introduire dans des équipements réseau ou à perturber leurs fonctionnements, les attaques de ce type sont bien évidemment plus dangereuses. (Ex. : mascarade et déni de service).

II.3.2 Classification selon la source de l'attaque

En termes de relation intrusion-victime, les attaques sont classées comme suit :

- **Les attaques internes :** provenant des employés de leur entreprise ou de leurs partenaires commerciaux ou clients,
- **Les attaques externes :** venant de l'extérieur, fréquemment via Internet.

II.3.3 Classification selon la cible de l'attaque

- **Les attaques réseaux :** Les attaques réseaux s'appuient sur des vulnérabilités liées directement aux protocoles ou à leur implémentation. Il existe un grand nombre. Néanmoins, la plupart d'entre elles ne sont que des variantes des cinq attaques réseaux les plus connues aujourd'hui.
- **Les attaques applicatives :** Les attaques applicatives s'appuient principalement sur des vulnérabilités spécifiques aux applications utilisées.

II.3.4 Exemples d'attaques

Il existe un nombre énorme d'attaques qui menacent les systèmes et les réseaux informatiques, néanmoins, la plupart d'entre elles ne sont que des variantes des autres. Voici des exemples d'attaques les plus connues aujourd'hui ciblant les réseaux informatiques.

II.3.4.1 Attaques de Dénis de Service (DoS) :

Est une attaque informatique ayant pour but de rendre indisponible un service, d'empêcher les utilisateurs légitimes d'un service de l'utiliser. Il peut s'agir de :

- L'inondation d'un réseau afin d'empêcher son fonctionnement ;
- La perturbation des connexions entre deux machines, empêchant l'accès à un service particulier ;
- L'obstruction d'accès à un service à une personne en particulier ;
- Également le fait d'envoyer des milliards d'octets à un box internet. L'attaque par déni de service peut ainsi bloquer un serveur de fichiers, rendre impossible l'accès à un serveur web ou empêcher la distribution de courriel dans une entreprise, les principales attaques qu'on peut trouver sont Apache2, Back, Land, Mail bomb, SYN Flood, Ping of death, Process table, Smurf, Syslogd, Teardrop, Udp storm.

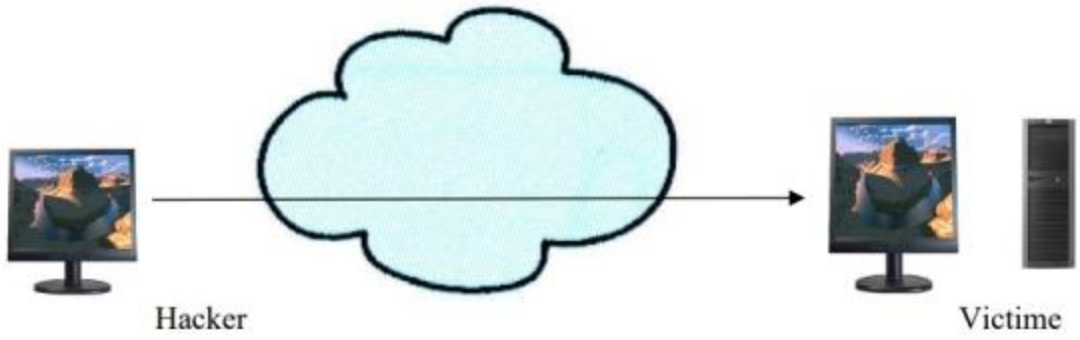


Figure I. 7: Attaque DoS

II.3.4.2 Attaques Distributed Denial of Service (DDoS)

Les attaques de déni de service distribué (DDoS) sont des attaques où plusieurs ordinateurs sont utilisés pour attaquer un seul système, souvent en utilisant des botnets.

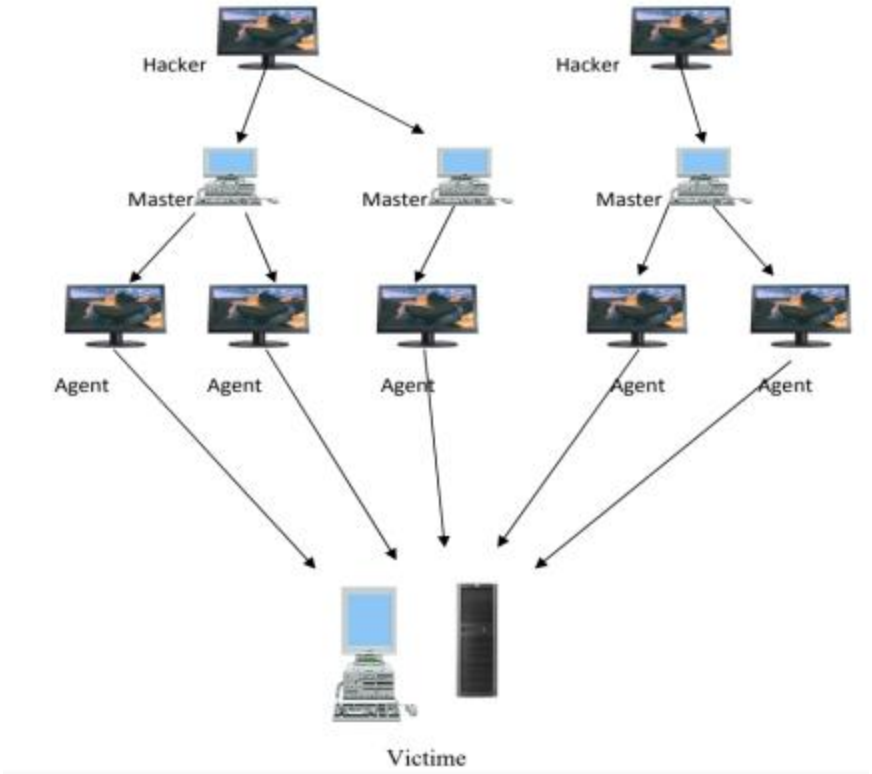


Figure I. 8: Attaques DDoS[7]

II.3.4.3 L'usurpation d'adresse IP (IP Spoofing)

Le principe de fonctionnement de cette attaque est d'envoyer des paquets IP en utilisant une adresse IP source qui n'a pas été allouée à l'ordinateur qui émet ces paquets pour le but de masquer l'identité de l'attaquant lors d'une attaque d'un serveur ou n'importe quelle cible dans

le réseau, ou d'usurper l'identité d'un autre équipement du réseau pour bénéficier des services auxquels il a accès.**[2]**

II.3.4.4 Les analyseurs réseau (sniffer)

Est un dispositif permettant d'écouter » le trafic d'un réseau, c'est-à-dire de capturer les informations qui y circulent, vu que les données dans un réseau non commuté sont envoyées à toutes les machines du réseau et dans une utilisation normale les machines ignorent les paquets qui ne leur sont pas destinés. Le sniffer peut également servir cette propriété à une personne malveillante ayant un accès physique au réseau pour collecter des informations (ex : les mots de passes), sniffer peut aussi être utilisé comme un outil positif pour le but d'étudier et de capturer le trafic d'un réseau par les administrateurs réseaux et les détecteurs d'intrusion (IDS).

II.3.4.5 L'homme du milieu (Man in the Middle)

Les attaques de l'homme du milieu consistent à intercepter les communications entre deux systèmes pour voler des informations ou modifier les données. Les points sensibles permettant cette technique sont les plus souvent les protocoles : **DHCP, RIP, ICMP, ARP, DNS, Proxy http, Virus**.



Figure I. 9 : Attaques Man-in-the middle[2]

II.3.4.6 Les attaques de force brute

Les attaques de force brute consistent à essayer de deviner un mot de passe en essayant toutes les combinaisons possibles jusqu'à ce que le bon mot de passe soit trouvé.

II.3.4.7 Les attaques de logiciel malveillant (Malware)

Les logiciels malveillants, tels que les virus, les chevaux de Troie et les ransomwares, sont utilisés pour infecter un système et voler des données ou bloquer l'accès à des fichiers ou des systèmes. **Exemple : Ransomware**

II.3.4.8 Les attaques par Phishing

Les attaques de phishing visent à tromper les utilisateurs en leur faisant croire qu'ils sont sur un site légitime alors qu'ils sont en réalité sur un site malveillant, qui vise à voler des

II.3.4.9 Les attaques de réseau sans fil

Les attaques de réseau sans fil sont des attaques sur des réseaux Wi-Fi ou Bluetooth, dans le but de pirater des informations ou d'injecter du code malveillant dans le système.

II.3.4.10 Les attaques par injection de code

Les attaques par injection de code sont des attaques qui visent à injecter du code malveillant dans un système en exploitant des failles de sécurité dans les applications web ou les serveurs.

Il est important de se protéger contre ces attaques en utilisant des logiciels de sécurité, en gardant à jour les mises à jour de sécurité, et en utilisant des mots de passe forts et différents pour chaque compte.

II.3.4.11 Social engineering

L'ingénierie sociale est une attaque utilisée pour collecter des informations sur des cibles potentielles ou les pousser à faire des transactions/paiements. L'ingénierie sociale se base sur une faiblesse commune de tous les systèmes d'informations existants : le facteur humain. L'ingénierie sociale est donc le principe d'exploiter des faiblesses psychologiques sociales pour obtenir des informations ou de l'argent d'une personne, le contact est fait via différents moyens : téléphone, réseaux sociaux, messagerie, rencontre personnelle, etc… C'est une attaque qui demande peu de connaissances techniques et est souvent utilisée pour la récolte d'informations ou sur des cibles psychologiquement faibles (personnes âgées/handicapées).

II.4 Les failles exploitées

Il existe une diversité d'attaques possibles pour les assaillants. Pour réussir, ces attaques exploitent des vulnérabilités de sécurité qui sont inhérentes au domaine de l'informatique. En effet, pour sécuriser efficacement ces vulnérabilités, il est d'abord nécessaire de les repérer. Cela signifie qu'il y aura toujours des failles de sécurité, et par conséquent, des individus malveillants cherchant à les exploiter. De plus, les outils et techniques de piratage évoluent rapidement, ce qui donne aux pirates un avantage en termes de découverte et d'exploitation des vulnérabilités de sécurité. Les failles exploitées par les attaquants peuvent prendre différentes formes, que l'on peut classer en deux catégories : les failles techniques et les failles humaines.

Cependant, toutes les failles de sécurité ne sont pas d'origine technique ; certaines sont liées à des erreurs humaines, telles que des manipulations incorrectes, des erreurs d'inattention ou de la négligence. Ces failles humaines sont plus prévisibles. Dans ces cas, les attaquants cherchent à tromper leur cible en utilisant des tactiques visant à convaincre la victime de la légitimité de l'attaque, dans le but de la faire baisser sa garde. Cette approche est appelée "ingénierie sociale", et elle intervient lorsque les pirates exploitent nos vulnérabilités psychologiques pour obtenir frauduleusement des informations, des biens, des services, des transferts bancaires, un accès physique ou informatique, ou la divulgation d'informations confidentielles, entre autres. Les fraudeurs peuvent utiliser diverses techniques psychologiques, ainsi que leurs compétences en informatique et leur charisme pour abuser de la confiance, de l'ignorance ou de la crédule de leur

cible, en vue d'obtenir ce qu'ils souhaitent. Ces techniques sont bien connues, ce qui facilite leur prévention. Néanmoins, la prévention de ces failles est parfois négligée dans certaines entreprises, même si elle est généralement moins coûteuse que celle des failles techniques.**[5][6]**

II.7 Mécanismes de défense contre les attaques

C'est l'ensemble de procédures ou dispositifs qui sont conçu pour détecter, prévenir ou récupérer les attaques qui menacent la sécurité informatique, il existe plusieurs outils de prévention contre-attaques informatiques, Nous avons cité ci-dessous quelques mécanismes :

- **Chiffrement :** Algorithme généralement basé sur des clefs et transformant les données. Sa sécurité est dépendante du niveau de sécurité des clefs.
- **Signature numérique :** Données ajoutées pour vérifier l'intégrité ou l'origine des données.
- **Bourrage de trafic :** Données ajoutées pour assurer la confidentialité, notamment au niveau du volume du trafic.
- **Notarisation :** Utilisation d'un tiers de confiance pour assurer certains services de sécurité.
- **Contrôle d'accès :** Vérifie les droits d'accès d'un acteur aux données. N'empêche pas l'exploitation d'une vulnérabilité.
- **Antivirus :** Logiciel censé protéger l'ordinateur contre les logiciels (ou fichiers potentiellement exécutables) néfastes. Ne protège pas contre un intrus qui emploie un logiciel légitime, ou contre un utilisateur légitime qui accède à une ressource alors qu'il n'est pas autorisé à le faire.
- **Le pare-feu :** Un élément (logiciel ou matériel) du réseau informatique contrôlant les communications qui le travers. Il a pour fonction de faire respecter la politique de sécurité du réseau, celle-ci définissant quelles sont les communications autorisées ou interdites. N'empêche pas un attaquant d'utiliser une connexion autorisée pour attaquer le système. Ne protège pas contre une attaque venant du réseau intérieur (qui ne le traverse pas).
- **Détection d'intrusion :** Repère les activités anormales ou suspectes sur le réseau surveillé. Ne détecte pas les accès incorrects mais autorisés par un utilisateur légitime. Mauvaise détection : taux de faux positifs, faux négatifs.
- **Journalisation ("logs") :** Enregistrement des activités de chaque acteur. Permet de constater que des attaques ont eu lieu, de les analyser et potentiellement de faire en sorte qu'elles ne se reproduisent pas.
- **Analyse des vulnérabilités ("Security audit") :** Identification des points de vulnérabilité du système. Ne détecte pas les attaques ayant déjà eu lieu, ou lorsqu'elles auront lieu. Mais Aucun des mécanismes de sécurité ne suffit par lui-même, et pour cela dans la plupart du temps en vue d'atteindre un niveau acceptable de sécurité informatique plusieurs mécanismes sont utilisés en même temps.

II.CONCLUSION

Après cette exploration approfondie de la sécurité informatique sous différents angles, nous pouvons conclure que la sécurité totale à 100% est une illusion et qu'atteindre un risque zéro est tout simplement impossible. En réalité, ce que nous devons faire, c'est protéger de manière judicieuse les ressources informatiques dont nous disposons, car l'information est le moteur principal d'une entreprise et sa sécurité est inestimable. Ainsi, pour assurer cette sécurité, il est essentiel de considérer deux aspects : ce qu'il faut protéger et comment le protéger.

CHAPITRE II : SYSTEMES DE DETECTION D'ANOMALIES & DETECTION PRECOCE

INTRODUCTION

L'internet a facilité le flux d'informations, personnels, financières et autres. En même temps, il a promu autant de dangers. Les utilisateurs malveillants recherchent des proies vulnérables comme les systèmes sans correctifs, les systèmes affectés par des chevaux de Troie et les réseaux exécutant des services peu surs. Des alertent sont nécessaires pour prévenir les administrateurs et les membres de l'équipes de sécurité qu'une effraction s'est produite afin qu'ils puissent anticiper cela ou répondre en temps réel au danger. Les systèmes de détection d'intrusion ont été conçus pour jouer le rôle d'un tel système d'alerte de même que les détections précoces.

Dans ce chapitre nous présenterons tout d'abord la notion de système de détection d'intrusions. Nous présenterons également les détections précoces en deuxième lieux

II.1 SYSTEMES DE DETECTION D'ANOMALIES

II.1.1 Introduction

La détection d'intrusion englobe toutes les méthodes et mécanismes utilisés pour repérer les erreurs susceptibles de causer des vulnérabilités en matière de sécurité, ainsi que pour identifier les tentatives d'attaques. Un IDS constitue la mise en pratique concrète de ces méthodes et mécanismes de détection d'intrusion. En conséquence, les IDS ont pour mission de détecter et/ou de prévenir (dans ce cas, on parle d'un système de prévention des intrusions, ou Intrusion Prevention System) les attaques qui se manifestent au sein d'un système ou d'un réseau. Ils peuvent être déployés soit sur l'ordinateur surveillé, auquel cas il s'agit d'un système de détection d'intrusion basé sur l'hôte (HIDS, pour Host Based Intrusion Detection System), soit sur le réseau surveillé, ce qui correspond à un système de détection d'intrusion basé sur le réseau (NIDS, pour Network Based Intrusion Detection System).

II.1.2 Définition

Un système de détection d'intrusion (IDS), abréviation d'Intrusion Detection System en anglais, se présente sous la forme d'un dispositif ou d'un processus actif qui scrute en permanence l'activité d'un système informatique et d'un réseau. Son objectif est d'identifier toute tentative d'accès non autorisé ou toute activité présentant des intentions malveillantes. Les méthodes employées par un IDS pour repérer des anomalies peuvent varier considérablement. Toutefois, l'objectif fondamental de tout IDS demeure d'intercepter les auteurs avant qu'ils ne puissent causer des dommages significatifs à vos ressources.

Les IDS jouent un rôle crucial dans la protection des systèmes en contrant les attaques, les abus et les vulnérabilités potentielles. En outre, ils assurent une surveillance constante de l'activité du réseau, passent au crible les paramètres du système et de la configuration réseau à la recherche de failles, vérifient l'intégrité des données, et bien d'autres fonctions encore.

Les IDS s'appuient généralement sur deux critères traditionnels :

- **Fiabilité :** Tout incident d'intrusion doit impérativement générer une alerte. L'absence de signalement en cas d'intrusion constitue un échec de détection, connu sous le nom de "faux négatif".
- **Pertinence des alertes :** Chaque alerte émise doit correspondre à une intrusion réelle. Les fausses alertes, aussi appelées "faux positifs", réduisent l'efficacité de l'IDS. Un IDS

est considéré comme parfaitement fiable en l'absence de faux négatifs et pertinent en l'absence de faux positifs.

II.1.3 Architecture et Principe de fonctionnement d'un IDS

L'architecture d'un IDS repose généralement sur une combinaison de capteurs et d'analyseurs. Les capteurs surveillent l'activité du système ou du réseau, collectant des données telles que les journaux d'événements. Les analyseurs examinent ensuite ces données à la recherche de schémas suspects ou d'anomalies, déclenchant des alertes en cas de détection d'une intrusion présumée. Les IDS peuvent être basés sur l'hôte (HIDS), examinant le trafic d'une machine spécifique, ou basés sur le réseau (NIDS), surveillant le trafic sur l'ensemble du réseau.

II.1.3.1 Architecture d'un IDS

Plusieurs architectures ont été proposées pour décrire les différents éléments intervenants dans un système de détection d'intrusion. L'architecture la plus simple est composée de trois modules : la source de données, l'analyseur des données et le module des réponses. L'architecture générale d'un IDS proposée par IDWG (Intrusion Détection Working Group) est montrée dans la (**Figure II.1).** Dans l'architecture proposée par le groupe IDWG de l'IETF2 on trouve les trois modules cités précédemment couplés avec d'autres composants, Dans cette architecture, l'objectif été la définition d'un standard de communication entre les composants du système de détection d'intrusion. Cette architecture définit un format d'échange de message pour les IDS : Intrusion Détection Message Exchange Format (IDMEF), qui contient implicitement un modèle de données. Proposée par IDWG.

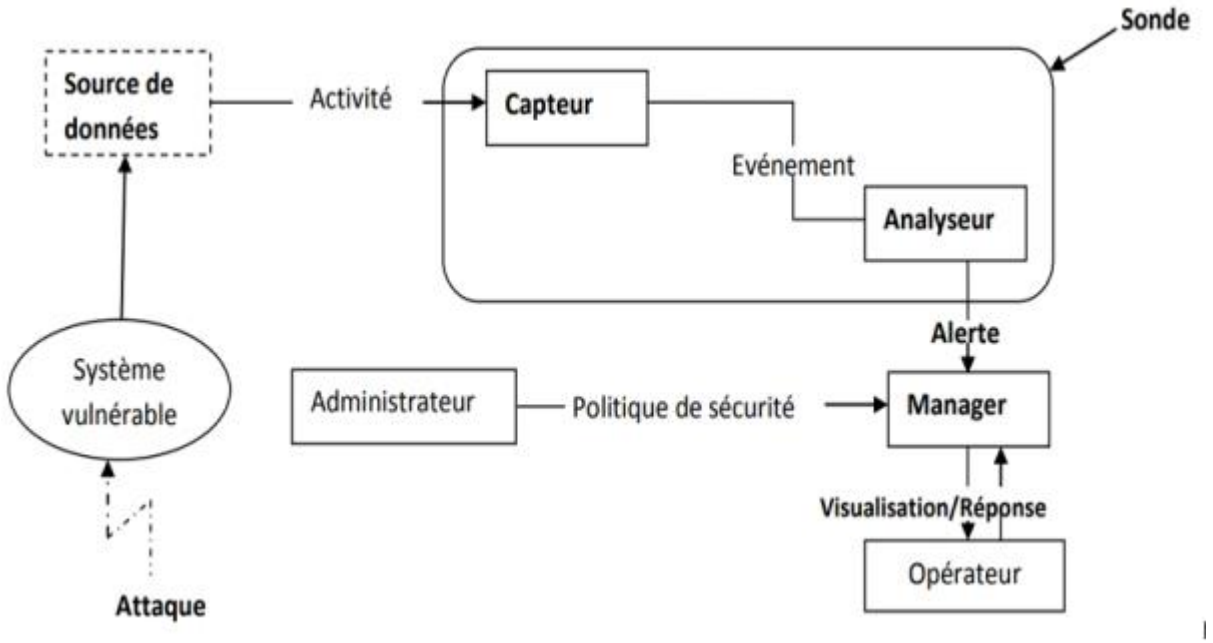


Figure II. 1 : Architecture d'un IDS [14]

Cette architecture est composée des modules suivants :

- **Source de données :** C'est l'interface entre le système surveillé et l'IDS, elle fait la collecte d'informations sur les activités du système.

- **Capteur :** Chargé de filtrer et formater les informations brutes envoyées par la source de données. Le résultat de ce traitement sera un message formaté, appelé aussi événement, il représente l'unité de base dans un scenario d'attaque.
- **Analyseur :** Permet d'analyser les événements générés par le capteur. S'il détecte une activité intrusive il émet une alerte, qui est un message sous un format standard. Dans cette architecture, le capteur et l'analyseur forment ensemble une sonde.
- **Alertes :** Lorsqu'un IDS détecte une intrusion, il doit la signaler à l'administrateur à travers les alertes, Ces dernières générées par les IDS sont généralement stockées dans les journaux du système ou utilisés pour prendre des actions contre les attaques (cela dépend du type d'IDS : à réaction active ou passive). Cependant il existe une norme qui permet d'en formaliser le contenu, afin de permettre à différents éléments de sécurité d'inter-opérer. Ce format s'appelle IDMEF 3 (Intrusion Détection Message Exchange Format), où il est possible de les visualiser ultérieurement par un expert de sécurité.
- **Manager :** En plus de la notification des alertes, il offre à l'administrateur la possibilité de configurer une sonde et de gérer les alertes envoyées par l'analyseur.

II.1.3.2 Principe de Fonctionnement d'un IDS

Pour bien gérer un système de détection d'intrusions, il est important de savoir comment il repère les intrusions. En d'autres termes, comment fait-il quand il y a une attaque parmi les données normales ? Pour répondre à ces questions, nous avons étudié comment fonctionne un système de détection d'intrusions. En résultat, nous avons identifié deux approches utilisées pour détecter les attaques : l'observation du comportement et la mise en place de scénarios. Il existe aussi d'autres critères pour classer ces systèmes, comme la fréquence d'utilisation, les sources de données analysées, et comment le système réagit après avoir détecté une intrusion.

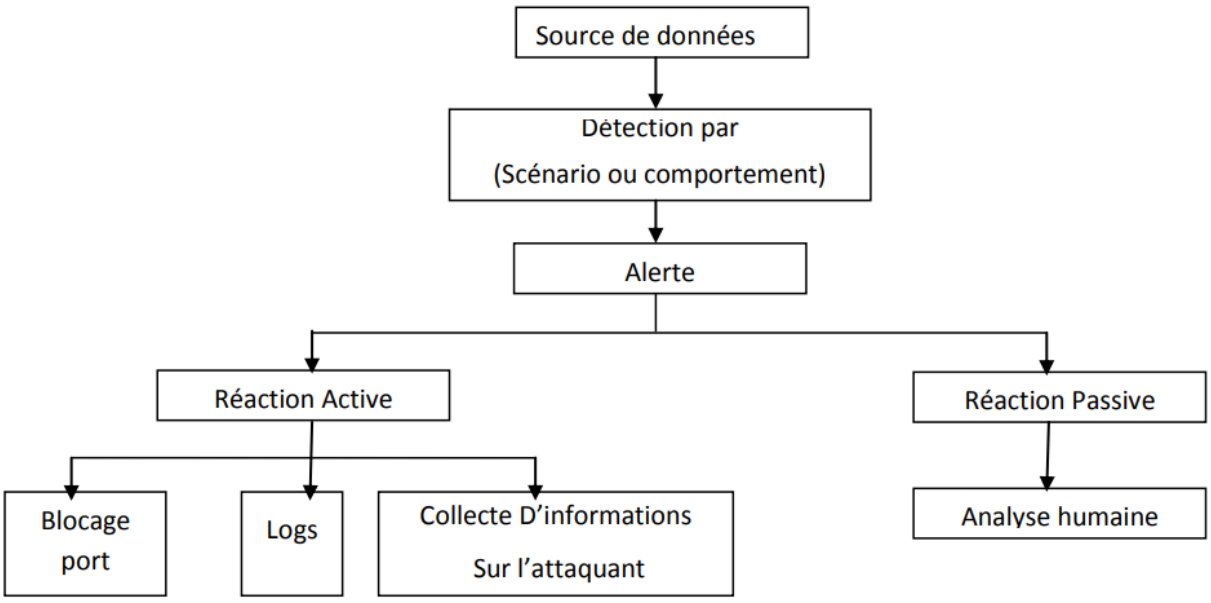


Figure II. 2 : Fonctionnement d'un IDS [15]

II.1.4 Classification des Systèmes de Détection d'intrusion

Il existe plusieurs classifications des systèmes de détection des intrusions, nous avons choisi le modèle apparu dans la **Figure II.3** :

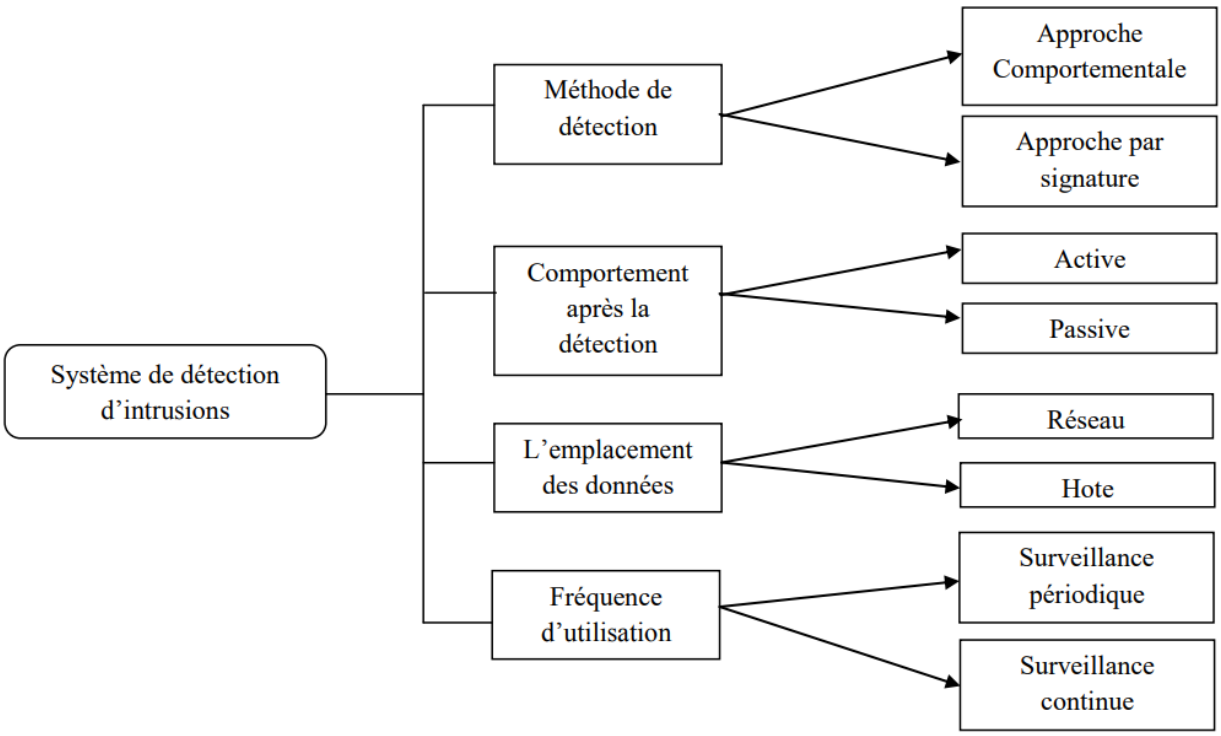


Figure II. 3 : Classification des IDS [16]

II.1.4.1 Méthode de détection

Comme illustré dans la **Figure II.4**, il y a deux principales méthodes de détection, à savoir la détection par signatures (appelée également détection d'intrusions par signatures ou détection par correspondance de modèles) et la détection par comportements (appelée aussi détection d'anomalies). La détection par signatures implique la recherche d'attaques en vérifiant si les observations correspondent à des modèles d'attaques déjà connus. En revanche, la détection par comportements (ou détection d'anomalies) consiste à repérer une attaque en vérifiant si les observations ne correspondent pas aux comportements légitimes établis en référence. Certains IDS combinent ces deux approches pour améliorer leur efficacité.

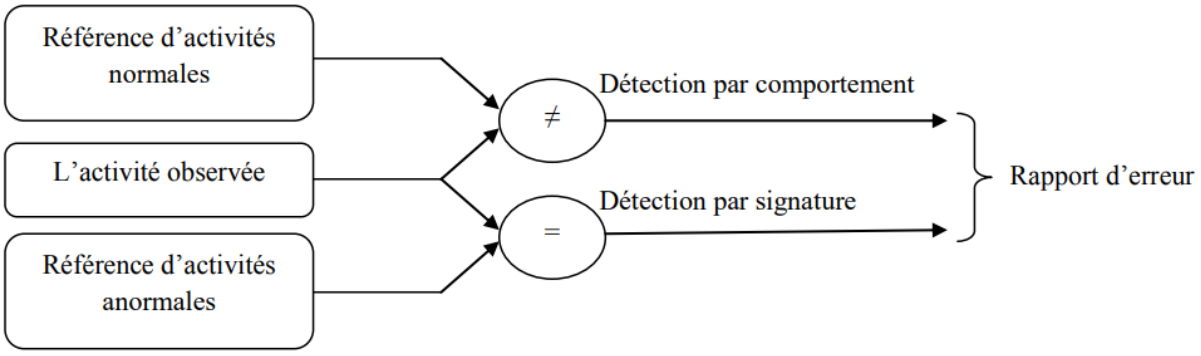


Figure II. 4 : Méthode de détection [17]

II.1.4.1.1 Approche Comportementale

Cette approche a été proposée par J. Anderson dans [18] en 1980, puis révisée et étendue dans [19] par D. Denning en 1987. Elle consiste à détecter si un utilisateur a fait un comportement

anormal par rapport à ses habitudes. Elle utilise pour cela un modèle statistique développé par Denning dans **[19]** et elle se base sur un profil du comportement normal de l'utilisateur, au vu de plusieurs variables aléatoires. Lors de l'analyse, on calcule un taux de déviation entre le comportement courant et le comportement passé.

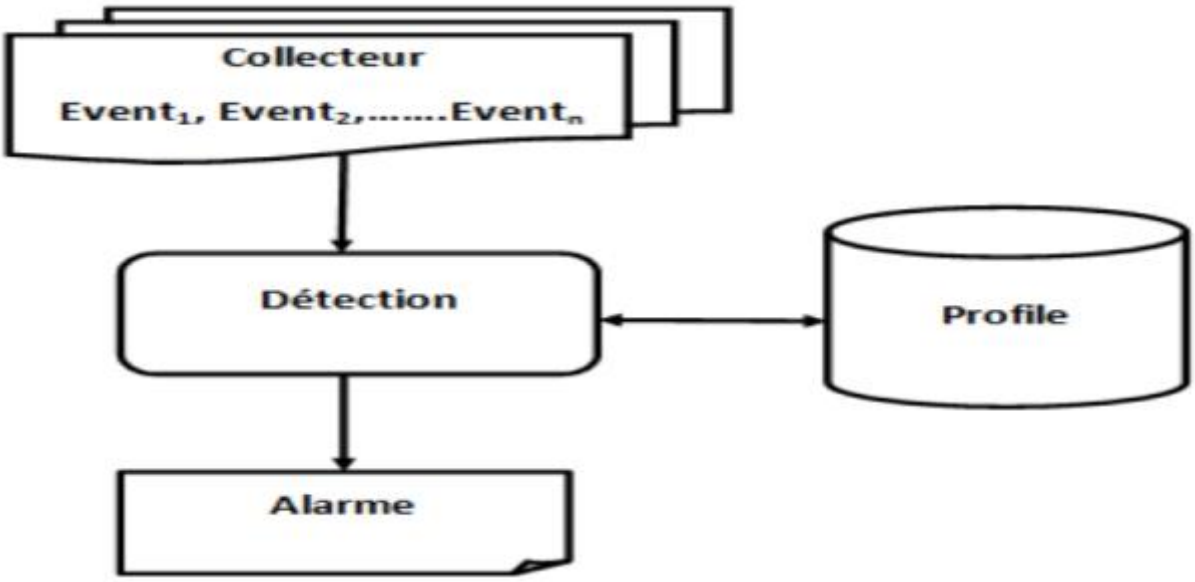


Figure II. 5 : Approche Comportementale [18]

Lorsque ce taux dépasse un certain seuil, le système considère qu'il y a une attaque en cours. Par exemple, si un employé se connecte à des heures inhabituelles, en plus de sa connexion habituelle en journée, l'IDS pourrait signaler ce comportement comme étant anormal. Les IDS comportementaux ont un avantage majeur : ils peuvent détecter de nouveaux types d'attaques. Contrairement aux IDS signature qui recherchent des attaques spécifiques, les IDS comportementaux sont conçus pour repérer toute activité qui sort de l'ordinaire. La Figure II.5 présente un exemple de détection basée sur le comportement, utilisant une méthode de profil. Cette approche compare les événements actuels à un profil construit à partir d'événements passés **[21]**. Cependant, cette méthode peut générer de nombreuses fausses alertes et parfois ne pas détecter certaines attaques. L'approche comportementale est basée sur plusieurs techniques, on citera dans ce qui suit quelques-unes :

- ✓ **L'observation de seuils** : implique l'utilisation de la méthode de classification de BAYE. Cette méthode consiste à définir ce qui est considéré comme un comportement normal pour un utilisateur en établissant des limites pour certaines mesures (par exemple, le nombre maximum de mots de passe incorrects autorisés). Si le comportement d'un utilisateur dépasse ces limites, le système de détection déclenche une alarme
- ✓ **Le profilage des utilisateurs [20]** : consiste à créer des profils spécifiques pour chaque utilisateur du système. Ces profils évoluent à mesure que l'utilisateur modifie ses activités. Cependant, il peut être compliqué de définir un profil pour un utilisateur qui a des habitudes irrégulières ou très changeantes

- ✓ **Le profilage des groupes** : consiste à regrouper les utilisateurs en catégories pour réduire le nombre de profils à gérer. Ensuite, un profil global est créé pour chaque groupe en se basant sur les activités passées des membres. On s'assure que les membres d'un groupe travaillent de manière cohérente et ne s'écartent pas du profil établi pour ce groupe. Cependant, il peut être difficile de déterminer quel groupe convient le mieux à une personne. Parfois, il est même nécessaire de créer un groupe pour une seule personne.
- ✓ **Le profilage de l'utilisation des ressources** : implique de surveiller comment certaines ressources, telles que les unités centrales de traitement (CPU), les ports de communication, les comptes d'utilisateurs, les applications et la mémoire, sont utilisées sur de longues périodes. Ensuite, on compare ces activités actuelles à ce qui a été observé dans le passé. On peut également examiner les changements dans l'utilisation des protocoles réseau et détecter les ports où le trafic augmente de manière anormale. Cependant, il est généralement difficile d'interpréter les différences par rapport au comportement normal.
- ✓ **Le profilage des programmes exécutables** : implique d'observer comment certains programmes de confiance utilisent les ressources du système, comme les fichiers ou les imprimantes, pour détecter toute activité inhabituelle. En d'autres termes, on surveille comment ces programmes se comportent normalement et on les compare ensuite à des comportements différents. Par exemple, on peut détecter qu'un serveur commence à écouter sur des ports qu'il n'utilise généralement pas.
- ✓ **Le profilage statistique** : tel que défini par Denning dans [**19],** consiste en une approche basée sur des statistiques pour créer des modèles du comportement des utilisateurs dans un système. Ce modèle statistique permet de décider si une nouvelle observation (par exemple, la valeur xn+1) est acceptable ou non en se basant sur les n observations précédentes (r1, ..., rn) effectuées sur une variable x. En d'autres termes, un profil est constitué d'un ensemble de mesures qui représentent différentes statistiques liées à des événements, comme le nombre de fois qu'un utilisateur exécute une commande système spécifique ou la quantité de temps CPU utilisée par un programme, sur une certaine période de temps.
- ✓ **L'approche immunologique :** vise à imiter le fonctionnement des systèmes immunitaires réels pour distinguer ce qui est normal de ce qui ne l'est pas. Contrairement à l'approche comportementale qui se base sur la connaissance de ce qui est correct et surveille constamment l'activité du système, l'approche immunologique cherche à identifier ce qui est incorrect en comprenant d'abord ce qui est correct. En d'autres termes, elle consiste à créer un modèle du comportement normal des services en utilisant de courtes séquences d'appels système. Pendant la phase d'apprentissage, on établit une base de données de séquences d'appels système normaux en observant le comportement d'un service sur une période donnée. Toute séquence qui ne correspond pas à cet ensemble est considérée comme une possible intrusion.

II.1.4.1.2 Approche Scénarios

Cette approche consiste à chercher des indices ou des signatures d'attaques bien connues dans les activités d'un système surveillé. Chaque signature décrit une attaque spécifique, et une

attaque peut être détectée en observant un événement ou une séquence d'événements collectés à partir de diverses sources, comme des fichiers de journalisation ou des informations réseau. La **Figure II.6** montre un schéma de base d'un système de détection d'intrusions adapté à cette méthode.

Cependant, cette approche présente des similitudes avec les antivirus et partage leurs limitations. Elle ne peut détecter que les attaques pour lesquelles elle possède des signatures préalables. De plus, elle nécessite des mises à jour régulières de sa liste de signatures, et son efficacité dépend de la qualité de ces signatures. Si les signatures sont incorrectes ou manquantes, le système devient inefficace. En revanche, il est relativement simple à mettre en place et à améliorer.

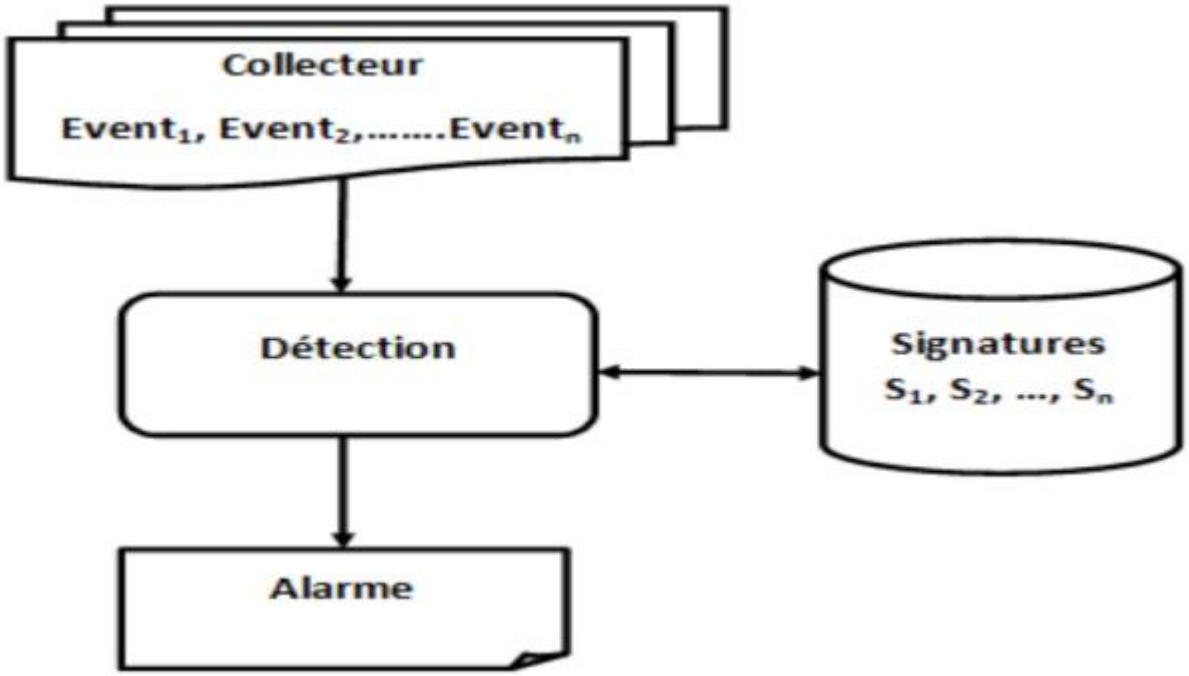


Figure II. 6 : Approche par Scénario ou Signature [18]

Plusieurs mécanismes ont été proposés afin de localiser les signatures d'attaques dans les traces d'audit. Parmi ces mécanismes, on peut citer :

- ✓ **L'analyse des transitions d'états** : consiste à considérer les signatures d'attaques comme des séquences d'événements marqués. En commençant par un état initial et en examinant les actions successives effectuées sur le système, nous pouvons repérer des états non souhaités qui signalent des tentatives d'intrusions.
- ✓ **Les réseaux de neurones :** sont souvent utilisés pour classifier un groupe d'actions en différentes catégories. Dans le cas de la détection d'intrusions, ce groupe d'actions correspond à toutes les actions effectuées sur un système, et l'objectif est de les classer en au moins deux catégories : les actions malveillantes ou suspectes et les
- ✓ **Actions non malveillantes** : Les réseaux de neurones sont choisis en raison de leur souplesse et de leur rapidité, ce qui permet d'analyser efficacement le flux d'audit en temps réel. Cependant, il peut être difficile de comprendre pourquoi une action est classée dans une catégorie plutôt que dans une autre dans certains cas.

- ✓ **La reconnaissance de forme**, ou le "Pattern Matching", consiste à représenter les modèles de comportement des attaques sous forme de séquences abstraites d'événements, parfois en incluant des éléments variables. Ensuite, on compare ces modèles avec les séquences d'actions concrètes du trafic pour voir s'il est possible de trouver des correspondances.

Approche	**Avantages**	**Inconvénients**
Comportementale	Détection d'intrusion inconnue possible	• Choix délicat des mesures à retenir pour un système cible donner. • Pour un utilisateur au comportement erratique, toute activité est « normale ». • En cas de profonde modification de l'environnement du système cible, déclenchement d'un flot ininterrompu d'alarmes (faux positifs). • L'utilisateur pouvant changer lentement de comportement dans le but d'habituer le système a un comportement intrusif (faux négatifs)
Par signature ou scenarios	Prise en compte des comportements exacts des attaques potentiels.	• Base de règles délicates à construire. • Seules les attaques contenues dans la base sont détectées.

Tableau II. 1 : Comparaison des deux approches

II.1.4.1.3 Autres Critères

Parmi les autres critères de classification existants, nous pouvons citer entre autres :

- Les informations que nous pouvons examiner sont une partie fondamentale des systèmes de détection d'intrusions. Ces informations peuvent être obtenues en écoutant le trafic sur le réseau, en collectant des fichiers générés par le système d'exploitation, ou encore en analysant des fichiers produits par des applications.
- Un autre moyen de catégoriser les systèmes de détection d'intrusions consiste à examiner comment ils réagissent lorsqu'ils repèrent une attaque. Certains systèmes se contentent de déclencher une alarme en réponse, ce qui est une réaction passive.
- Un autre aspect important des systèmes de détection d'intrusions concerne la fréquence à laquelle ils sont utilisés, soit de manière périodique, soit de manière continue. Certains systèmes vérifient de temps en temps les journaux d'audit à la recherche de signes d'intrusion passée, ce qui peut suffire dans des environnements moins critiques. En revanche, la plupart des systèmes de détection d'intrusions récents effectuent une analyse continue des journaux d'audit ou du trafic réseau pour détecter les problèmes presque en

temps réel, ce qui est essentiel dans des environnements sensibles, bien que cela nécessite une puissance de calcul élevée car il faut examiner en permanence tout ce qui se passe sur le système.

II.1.4.2. Comportement après la Détection ou Types de réponses

Ils existent deux types de réaction des IDS qui sont actifs et passifs

II.1.4.2.1 IDS à réponse passive

Une réponse passive d'un IDS signifie qu'il enregistre les intrusions qu'il détecte dans un fichier de journal, que le responsable de la sécurité peut examiner ultérieurement. Il peut également générer des alertes, envoyer des e-mails à un ou plusieurs utilisateurs, etc. Cette approche permet de corriger les vulnérabilités de sécurité pour éviter que des attaques similaires ne se reproduisent, mais elle n'empêche pas directement une attaque de se produire.

II.1.4.2.2 IDS à réponse active

La réponse active, à la différence de la réponse passive, vise à arrêter une attaque dès qu'elle est détectée, sans attendre une intervention humaine. Il existe deux techniques pour cela : la reconfiguration du pare-feu et l'interruption d'une connexion TCP.

La reconfiguration du pare-feu consiste à bloquer le trafic malveillant au niveau du pare-feu en fermant le port utilisé ou en interdisant l'adresse de l'attaquant. Cependant, cette fonctionnalité dépend du type de pare-feu utilisé, car tous les modèles ne permettent pas à un IDS de reconfigurer le pare-feu. De plus, la reconfiguration est limitée par les capacités du pare-feu. L'IDS peut également interrompre une session établie entre un attaquant et sa cible, ce qui empêche le transfert de données ou la modification du système attaqué. **[22]**

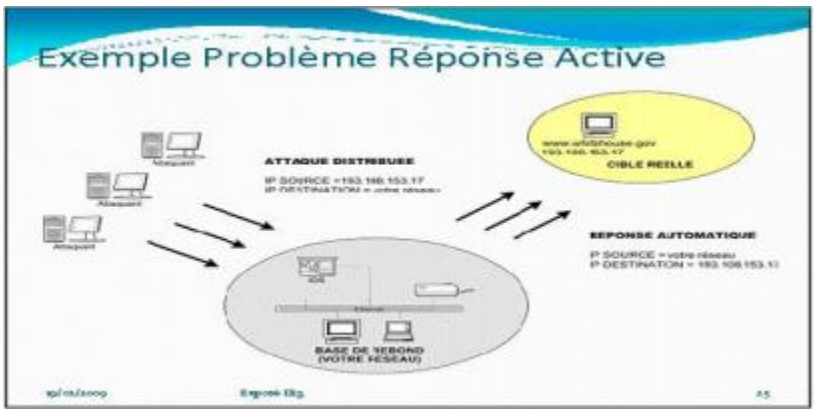


Figure II. 7 : Réponse Active [22]

II.1.4.3 Emplacement de Données

Il existe des IDS qui veillent à la sécurité du réseau en surveillant et en analysant les paquets de données qui circulent à travers le réseau (appelés NIDS pour "Network Intrusion Detection System"). D'autres se concentrent sur la sécurité des ordinateurs individuels en examinant les informations générées par le système d'exploitation ou par les applications installées sur les machines locales (appelés HIDS pour "Host Intrusion Detection System"). Certains IDS hybrides combinent à la fois NIDS et HIDS pour obtenir des alertes de sécurité plus précises. **[23].**

II.1.4.4 Fréquence d'utilisation

Une autre caractéristique des systèmes de détection d'intrusions concerne la fréquence à laquelle ils effectuent leur surveillance, soit de manière régulière à intervalles, soit en permanence.

Surveillance périodique : Dans ce scénario, les systèmes de détection d'intrusions examinent de manière récurrente les sources de données pour détecter d'éventuelles intrusions ou anomalies passées. Cette approche peut être adéquate dans des contextes où la sensibilité à la sécurité est relativement faible, par exemple en effectuant une analyse une fois par jour.

Surveillance continue : La plupart des systèmes de détection d'intrusions modernes effectuent une analyse continue des données sur l'ordinateur local ou du trafic réseau pour offrir une détection presque en temps réel. Cette approche est nécessaire dans des environnements sensibles, notamment en ce qui concerne la confidentialité. Cependant, il est important de noter que cela nécessite une puissance de calcul considérable, car il faut examiner en temps réel tout ce qui se déroule sur le système.

II.1.5 Emplacement des Systèmes de Détections d'Intrusions

II.1.5.1 Système de détection d'intrusion de type Hôte (HIDS)

Les HIDS, abréviation de "Host-Based Intrusion Detection System", sont configurés sur un ordinateur particulier et surveillent toutes les activités qui s'y déroulent, sans analyser le réseau. Ils peuvent surveiller ce que fait l'utilisateur (quels programmes et commandes sont utilisés, quand et combien de temps ils sont utilisés, etc.), ce que fait la machine (combien de ressources elle utilise, combien de processus sont en cours d'exécution, etc.), et détecter toute activité suspecte (comme des virus ou des chevaux de Troie). Leur principal avantage est qu'ils génèrent peu de fausses alertes, mais l'inconvénient est qu'il faut configurer un HIDS pour chaque machine, en fonction des différents systèmes et sous-configurations utilisés.

Un exemple de HIDS est OSSEC, un logiciel open-source qui a été développé initialement pour analyser les journaux de différents serveurs, mais qui est devenu un outil de détection d'intrusions complet capable d'analyser divers formats de journaux, y compris ceux d'Apache, Syslog et Snort, et qui intègre également un analyseur d'intégrité système. **[24]**

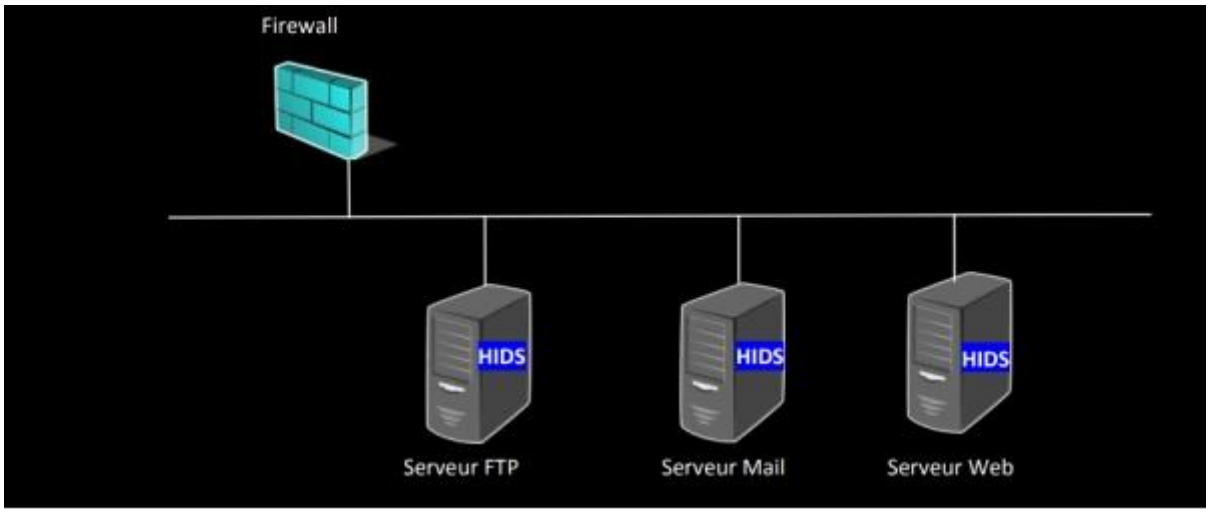


Figure II. 8 : Détection avec HIDS [24]

II.1.5.2 Système de détection d'intrusion de type réseau (NIDS)

Les NIDS, abréviation de "Network Intrusion Detection System", analysent de manière passive les données en transit sur un réseau et identifient les intrusions en temps réel. En d'autres termes, ils écoutent le trafic réseau, l'examinent et génèrent des alertes si des paquets semblent suspects. Les NIDS sont particulièrement importants et utiles en raison de la présence généralisée des réseaux dans notre vie quotidienne. Lorsqu'ils sont déployés à différents endroits, ils permettent une surveillance efficace de l'ensemble d'un réseau. Un exemple de NIDS est SNORT, un outil capable d'analyser le trafic en temps réel et de consigner les paquets sur un réseau IP. Il peut effectuer diverses analyses, rechercher des correspondances de contenu et détecter de nombreuses attaques et tentatives d'intrusion, telles que les débordements de tampons, les scans de vulnérabilités, les attaques contre les systèmes de détection, et bien d'autres. Dans la suite de notre travail, nous nous concentrerons sur les NIDS. **[25]**

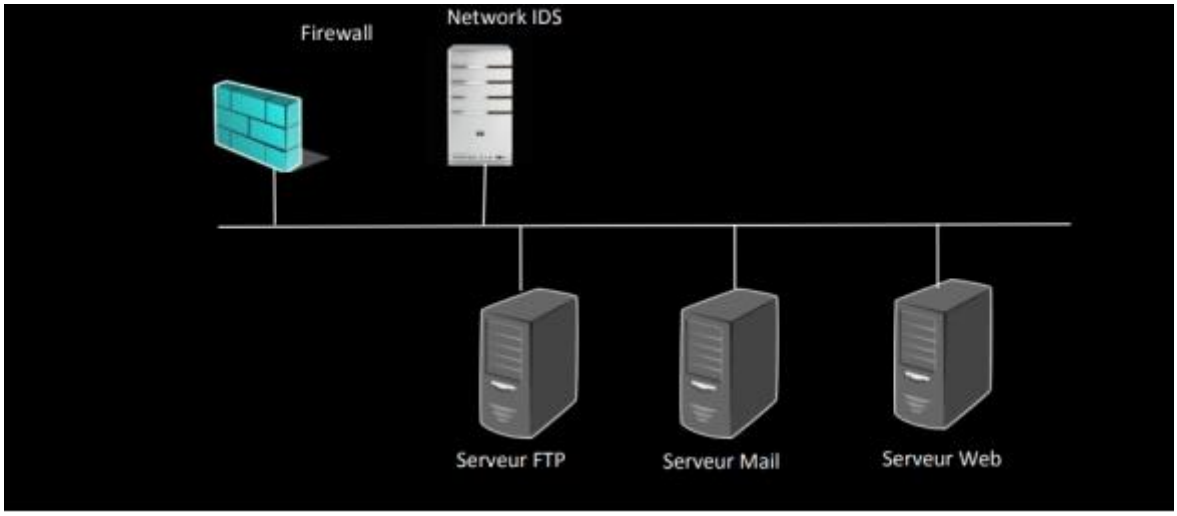


Figure II. 9 : Schéma d'un NIDS [25]

II.1.5.3 Détection d'intrusion hybrides

Ces systèmes, généralement utilisés dans des environnements décentralisés, ont la capacité de collecter des informations provenant de diverses sondes réparties sur le réseau. Ils sont appelés "hybrides" car ils peuvent rassembler des données à la fois d'un HIDS et d'un NIDS. L'exemple le plus célèbre dans le monde de l'Open Source est Prélude. Ce Framework permet de stocker des alertes issues de différents systèmes, qu'il s'agisse de Snort utilisé comme NIDS, ou d'autres logiciels comme Samhain en tant que HIDS. En combinant ces puissants outils, Prélude permet d'avoir une vue d'ensemble centralisée des attaques. **[24]**

II.1.6 Avantages des Systèmes de Détections d'Intrusions

- **Déjouer les attaques attendues sur le réseau** : Les IDS (Systèmes de Détection d'Intrusion) protègent les systèmes contre divers types d'attaques réseau, tels que la détection des accès non autorisés, la détection des usurpations d'adresse IP, la prévention des attaques par déni de service (DoS), la détection des vers, chevaux de Troie, virus, Botnet, rootkits, logiciels espions et d'autres menaces potentielles qui pourraient causer des dommages au réseau. Les IDS actifs prennent automatiquement des mesures pour contrer ces menaces et les risques de sécurité.

- Avertis Administrateur réseau d'alerte pour les événements de sécurité potentiels : L'objectif principal des systèmes de détection d'intrusions est de signaler la présence de menaces, qu'elles proviennent de l'extérieur ou de l'intérieur du réseau, ou qu'elles représentent des violations des règles de sécurité du réseau. Ils fournissent également des informations détaillées à l'administrateur sur le flux des données à l'intérieur du réseau.

- Gagnez du temps : L'utilisation des IDS permet de surveiller le réseau en continu, ce qui permet d'économiser beaucoup de temps et d'efforts pour comprendre ce qui se passe. De plus, ils peuvent fonctionner en permanence sans nécessiter une supervision humaine constante.

- Contrôle des Programmes utilisés par les employés pour surveiller l'Internet : Les IDS peuvent contribuer à identifier les logiciels qui interagissent avec Internet, ce qui facilite une meilleure gestion et protection du réseau.

- Avoir la confiance des clients : Les IDS aident les organisations à prévenir le vol de données et les violations de la sécurité, ce qui renforce la confiance de leurs clients et partenaires et maintient une réputation positive pour l'entreprise.

- Économisez de l'argent : Grâce aux IDS, les organisations peuvent détecter des activités suspectes dans leur réseau et identifier les responsables, ce qui leur permet de prendre des mesures préventives pour protéger leur réseau. Cela évite également les coûts potentiels liés à une violation de la sécurité ou au vol d'informations personnelles

II.1.7 Limites des Systèmes de Détections d'Intrusions

- Les systèmes de détection d'intrusions actuels peuvent avoir des lacunes, soit parce qu'ils ne parviennent pas à détecter certains types d'attaques, soit parce qu'ils peuvent être vulnérables à des attaques elles-mêmes.
- Les systèmes de détection d'intrusions actuels sont souvent trop rigides, ce qui limite leur capacité à être comparés en termes de performances, à coopérer entre eux, et à établir des normes d'interopérabilité entre différents outils pour résoudre ce problème.
- Il y a un décalage entre les preuves requises par les tribunaux et celles fournies par les outils de détection d'intrusions. Cela signifie qu'il faut effectuer des travaux supplémentaires pour extraire les preuves nécessaires, comme les fichiers journaux du système, une partie du trafic réseau capturé lors de l'attaque, les adresses IP incriminées et divers autres fichiers, lorsque des poursuites en justice sont envisagées. Le problème réside dans la difficulté à prouver que ces éléments n'ont pas été altérés.
- Les systèmes de détection d'intrusions ont tendance à générer un grand nombre de fausses alertes (faux positifs).

II.2 Détection Précoce Dans un Réseau informatiques

II.2.1 Définition

La "Détection Précoce dans un Réseau Informatique" (ou Early Detection in Computer Networks en anglais) fait référence à la capacité d'un système de surveillance ou d'un ensemble d'outils informatiques à identifier et à signaler rapidement les activités suspectes, les intrusions ou les menaces potentielles dans un réseau informatique. L'objectif principal de la détection

précoce est de repérer les comportements anormaux ou les indicateurs d'attaques dès qu'ils se produisent, afin de prendre des mesures immédiates pour contrer ces menaces avant qu'elles ne causent des dommages importants au réseau ou aux systèmes qui y sont connectés. Cette approche vise à minimiser les perturbations et à renforcer la sécurité globale du réseau en réagissant rapidement aux incidents de sécurité. **[27]**

II.2.2 Architecture et Principe de fonctionnement d'une détection précoce dans un Réseau

II.2.2.1 Architecture des Alerte Précoce dans un réseau

Une architecture d'alerte précoce dans un réseau informatique est conçue pour détecter rapidement les menaces potentielles ou les activités suspectes et pour alerter les administrateurs ou les équipes de sécurité afin qu'ils puissent prendre des mesures immédiates. Voici les principaux composants et concepts d'une telle architecture :

Capteurs : Les capteurs sont des dispositifs ou des logiciels spécialement conçus pour surveiller le réseau. Ils collectent des données sur le trafic réseau, les journaux d'événements, les activités des utilisateurs et d'autres informations pertinentes.

Collecte de données : Les données collectées par les capteurs sont agrégées et centralisées dans un emplacement central, souvent appelé "concentrateur de données" ou "collecteur de données". Cela permet une analyse plus approfondie et une gestion centralisée des informations.

Analyse en temps réel : Les données collectées sont analysées en temps réel à la recherche de comportements anormaux, de modèles de trafic suspects ou de signaux d'intrusion potentiels. Cette analyse peut être basée sur des règles prédéfinies, des algorithmes d'apprentissage automatique ou d'autres méthodes avancées de détection des menaces.

Détection d'incidents : Lorsqu'une activité suspecte est détectée, un incident est généré. Cet incident comprend des informations sur la nature de la menace, la source de l'activité suspecte, l'heure à laquelle elle a été détectée, etc.

Alertes : Les alertes sont ensuite générées à partir des incidents détectés. Ces alertes sont transmises aux administrateurs ou aux équipes de sécurité via différents canaux, tels que des notifications par e-mail, des messages texte, des tableaux de bord de surveillance, ou d'autres moyens de communication.

Corrélation d'alertes : Dans certains cas, les systèmes d'alerte précoce peuvent corréler plusieurs alertes pour identifier des attaques complexes ou des menaces coordonnées qui pourraient passer inaperçues lors de l'analyse individuelle des alertes.

Prise de décision automatisée : En fonction de la gravité de l'alerte et de la politique de sécurité définie, des mesures automatisées peuvent être déclenchées pour contrer la menace. Cela peut inclure la mise en quarantaine d'un périphérique, le blocage d'une adresse IP, la réinitialisation d'un mot de passe, etc.

Rapports et journaux : Toutes les données relatives aux alertes, aux incidents et aux mesures prises sont enregistrées dans des rapports et des journaux. Ces informations sont utiles pour les audits de sécurité, l'analyse post-incident et l'amélioration continue de la sécurité du réseau.

Réponse humaine : En fin de compte, les équipes de sécurité prennent des décisions et prennent des mesures en fonction des alertes et des incidents signalés. Cela peut inclure une enquête plus approfondie, une réponse immédiate à l'incident ou une révision de la politique de sécurité.

L'architecture d'alerte précoce vise à réduire au minimum les délais de réaction aux menaces et à renforcer la sécurité globale d'un réseau informatique en combinant une surveillance continue, une analyse en temps réel et des actions proactives en cas de détection d'incidents.

II.2.2.2 Fonctionnement d'une détection précoce dans un réseau

Le fonctionnement d'une alerte précoce dans un réseau informatique repose sur un processus de surveillance, de détection, de notification et de réaction rapide aux activités suspectes ou aux menaces potentielles. Voici les étapes clés du fonctionnement d'une telle alerte précoce :

La détection précoce dans un réseau informatique est un processus essentiel pour identifier rapidement les anomalies, les menaces potentielles et les problèmes de performance avant qu'ils ne deviennent des incidents graves. Voici comment fonctionne généralement la détection précoce dans un réseau informatique :

Collecte de données :

- Les données sont collectées à partir de diverses sources au sein du réseau. Cela peut inclure des journaux système, des paquets de données réseau, des informations de configuration, des mesures de performance, etc.
- Des capteurs, des agents logiciels et des dispositifs de surveillance peuvent être déployés pour collecter des données en temps réel à partir de divers endroits du réseau.

Traitement des données :

- Les données collectées sont agrégées et normalisées pour s'assurer qu'elles sont cohérentes et exploitables.
- Les données sont souvent stockées dans une base de données centralisée ou un entrepôt de données, où elles peuvent être analysées efficacement.

Analyse des données :

- L'analyse des données peut se faire en temps réel ou à intervalles réguliers, en fonction des besoins.
- Des algorithmes d'analyse sont utilisés pour détecter les anomalies, les modèles de comportement inhabituels et les seuils dépassés.
- Les méthodes d'analyse peuvent varier, allant de simples règles préconfigurées à des techniques d'apprentissage automatique et d'intelligence artificielle plus avancées pour détecter des menaces plus subtiles.

Détection d'anomalies :

- La détection d'anomalies repose sur la comparaison entre les données collectées et les critères de normalité établis.
- Les alertes sont générées lorsque des écarts significatifs par rapport aux modèles attendus sont détectés. Ces alertes indiquent la présence d'anomalies.

Notification des alertes :

- Les alertes sont notifiées aux membres du personnel appropriés, tels que les équipes de sécurité informatique ou d'exploitation, via des mécanismes de notification, tels que des e-mails, des messages texte ou des systèmes de gestion des alertes.
- Les alertes sont généralement classées en fonction de leur gravité pour permettre une réponse appropriée.

Réaction et résolution :

- Lorsqu'une alerte est déclenchée, les équipes responsables enquêtent sur l'incident pour en déterminer la cause sous-jacente.
- Des mesures correctives sont prises pour résoudre le problème, que ce soit en isolant une menace de sécurité, en optimisant les performances du réseau ou en corrigeant une configuration erronée.

Rétroaction et amélioration :

- Les données sur les incidents sont enregistrées et analysées pour identifier les tendances et les domaines d'amélioration potentiels.
- Sur la base de ces informations, des ajustements sont apportés à l'architecture de détection précoce, aux politiques de sécurité ou aux configurations réseau pour renforcer la résilience du réseau.

La détection précoce dans un réseau informatique est essentielle pour garantir la sécurité, la disponibilité et les performances du réseau. Elle permet de réagir rapidement aux menaces et aux problèmes, minimisant ainsi les interruptions de service et les impacts négatifs sur l'entreprise.

II.2.3 Emplacement et Avantage d'une détection précoce dans un réseau

L'emplacement et les avantages d'une alerte précoce dans un réseau sont essentiels pour garantir une sécurité efficace et une réponse rapide aux menaces potentielles. Voici des informations sur ces aspects :

II.2.3.1 Emplacement d'une Alerte Précoce dans un réseau

Périphérie du Réseau : Les alertes précoces peuvent être positionnées à la périphérie du réseau, près des points d'entrée et de sortie. Cela permet de détecter rapidement les activités suspectes provenant de l'extérieur du réseau, telles que les tentatives d'intrusion, les attaques DDoS (Distributed Denial of Service), les scans de ports, etc.

Au Niveau des Serveurs et des Postes de Travail : Des alertes précoces peuvent également être déployées sur les serveurs et les postes de travail à l'intérieur du réseau. Elles surveillent les activités internes, telles que les comportements d'utilisateurs, les tentatives d'accès non autorisées aux systèmes, et la détection de logiciels malveillants.

Sur les Systèmes de Passerelles et de Pare-feu : Les dispositifs de sécurité réseau tels que les pares-feux et les systèmes de passerelles peuvent intégrer des alertes précoces pour surveiller le trafic entrant et sortant. Cela permet de bloquer rapidement les activités malveillantes avant qu'elles n'atteignent les systèmes internes.

II.2.3.2 Avantages d'une Détection Précoce dans un Réseau :

Détection Rapide des Menaces : L'un des principaux avantages est la capacité à détecter rapidement les menaces et les comportements suspects dès leur apparition. Cela permet une réponse proactive avant que la menace ne puisse causer des dommages importants.

Réduction des Délais de Réaction : Une détection précoce permet de réduire les délais de réaction aux incidents de sécurité. Cela signifie que les équipes de sécurité peuvent intervenir plus rapidement pour atténuer les menaces et minimiser les perturbations.

Protection des Données Sensibles : Les alertes précoces aident à protéger les données sensibles en identifiant les tentatives d'accès non autorisé ou de fuite de données dès le départ. Cela contribue à maintenir la confidentialité des informations.

Minimisation des Pertes Financières : En détectant rapidement les menaces, une alerte précoce peut contribuer à minimiser les pertes financières résultant d'attaques ou d'incidents de sécurité. Cela peut inclure la prévention du vol de données, des temps d'arrêt coûteux, etc.

Réputation et Confiance : Une réponse proactive à la sécurité renforce la réputation de l'organisation en montrant son engagement envers la sécurité des données et la confidentialité des clients. Cela renforce également la confiance des clients et des partenaires commerciaux.

Meilleure Visibilité : Les alertes précoces offrent une meilleure visibilité sur les activités du réseau, ce qui permet aux équipes de sécurité de mieux comprendre les tendances et les schémas d'attaque.

En résumé, l'emplacement et les avantages d'une alerte précoce dans un réseau dépendent des besoins spécifiques de sécurité de l'organisation. Ils contribuent à renforcer la sécurité, à réduire les risques et à minimiser les conséquences des incidents de sécurité potentiels.

II.3 Conclusion

Dans ce chapitre Nous avons présenté aussi le principe de fonctionnement des IDS ainsi leurs classifications selon différents critères avec leurs avantages et inconvénients, parmi les critères de classification abordés la méthode de détection qui divise les IDS en deux types, les IDS comportementaux et à base de signatures, le principe de ce dernier consiste à rechercher dans l'activité de l'élément surveillé les empreintes (ou signatures) d'attaques connues d'une façon similaire à celle des antivirus, ce type d'IDS est simple d'implémenter mais il présente des difficultés comme la nécessité de mettre à jour d'une façon quotidienne la base de signature, en

plus seuls les attaques contenues dans la base sont détectables, cela peut baissé l'efficacité des systèmes de détection des intrusions à base des signature surtout avec l'énorme développement des attaques réseaux d'aujourd'hui, et pour que la détection d'intrusions soit fiable. Les détections précoces sont des compléments des IDS, en résolvant certaines failles des IDS comme nous l'avons bien expliqué dans les derniers paragraphes de ce chapitre. Dans le chapitre suivant nous allons expliquer le Deep Learning et leur utilité ainsi que l'algorithme utilisé pour notre projet.

CHAPITRE III : IMPORTANCE DU DEEP LEARNING DANS LES ATTAQUES RESEAUX

INTRODUCTION

Dans ce chapitre, nous allons expliquer ce qu'est l'apprentissage en profondeur (DL) et les réseaux de neurones profonds (DNN). Ces derniers sont en fait des réseaux de neurones qui ont plusieurs couches cachées. Vous pourriez vous demander pourquoi nous utilisons plusieurs couches cachées, surtout si vous avez entendu parler du théorème d'approximation universelle. C'est une question compréhensible, car pendant longtemps, les réseaux de neurones étaient utilisés de manière plus simple.

Pour simplifier, une des raisons pour utiliser plusieurs couches cachées est que pour approximer de manière précise une fonction complexe, il faudrait un très grand nombre de neurones dans une seule couche cachée, ce qui rendrait l'utilisation peu pratique. Les réseaux de neurones profonds sont capables d'apprendre des abstractions à partir des exemples d'entrée, de comprendre les caractéristiques fondamentales de ces exemples, et ensuite de faire des prédictions en se basant sur ces caractéristiques. Cela représente un niveau d'abstraction plus élevé que ce que peuvent faire d'autres méthodes d'apprentissage automatique de base et les réseaux neuronaux peu profonds.

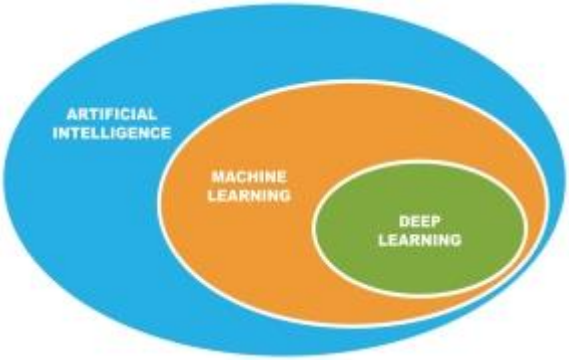


Figure III. 1: Composant de IA [26]

III.1 Notion de base sur l'apprentissage automatique (Machine Learning Basics) et sur l'apprentissage en Profondeur (Deep Learning)

III.1.1 Notion de base sur l'apprentissage automatique (Machine Learning Basics)

III.1.1.1 L'apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle (IA) qui permet aux ordinateurs d'apprendre sans nécessiter une programmation explicite. Les chercheurs de ce domaine s'efforcent de reproduire au mieux le fonctionnement du cerveau humain, ainsi que les modes de traitement de l'information et de communication que l'on observe dans le système nerveux biologique. L'objectif est de donner aux machines la capacité d'apprendre à partir de données, de les interpréter, et de prendre des décisions éclairées.

Les méthodes d'apprentissage automatique ont connu un succès notable dans divers produits et applications des technologies de l'information et de la communication (TIC) au cours des dernières années, tels que la reconnaissance d'image, la traduction automatique, le diagnostic médical, et bien d'autres secteurs technologiques tels que les véhicules autonomes et les robots

intelligents.

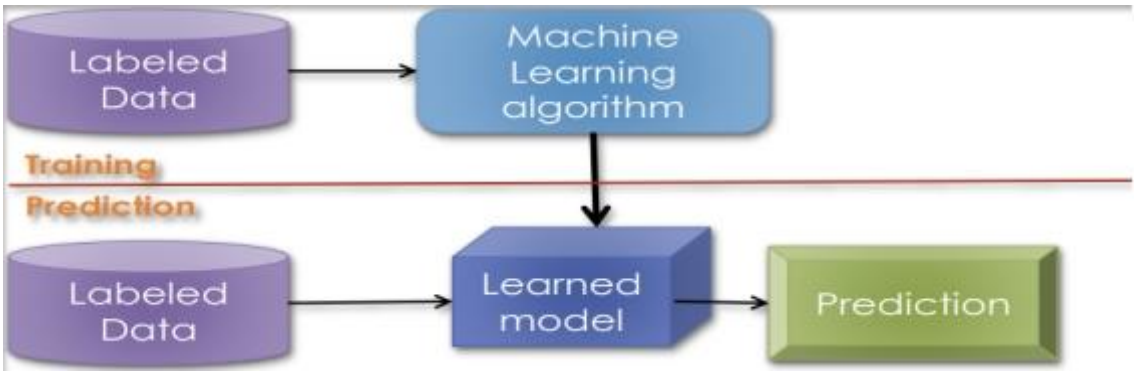


Figure III. 2 : Méthode permettant d'apprendre et de prédire des données[22]

III.1.1.2 Types d'apprentissage (Types of Learning)

Apprentissage supervisé implique l'utilisation d'un ensemble de formation comprenant des données étiquetées, comme la classification d'e-mails en utilisant des e-mails déjà catégorisés.

En revanche, l'apprentissage non supervisé vise à découvrir des modèles au sein de données non étiquetées, par exemple, en regroupant des documents similaires en fonction de leur contenu textuel.

Quant à l'apprentissage par renforcement, il s'agit d'apprendre à prendre des actions en se basant sur des retours ou récompenses, comme apprendre à jouer au jeu de Go, où la récompense est de gagner ou de perdre **[27].**

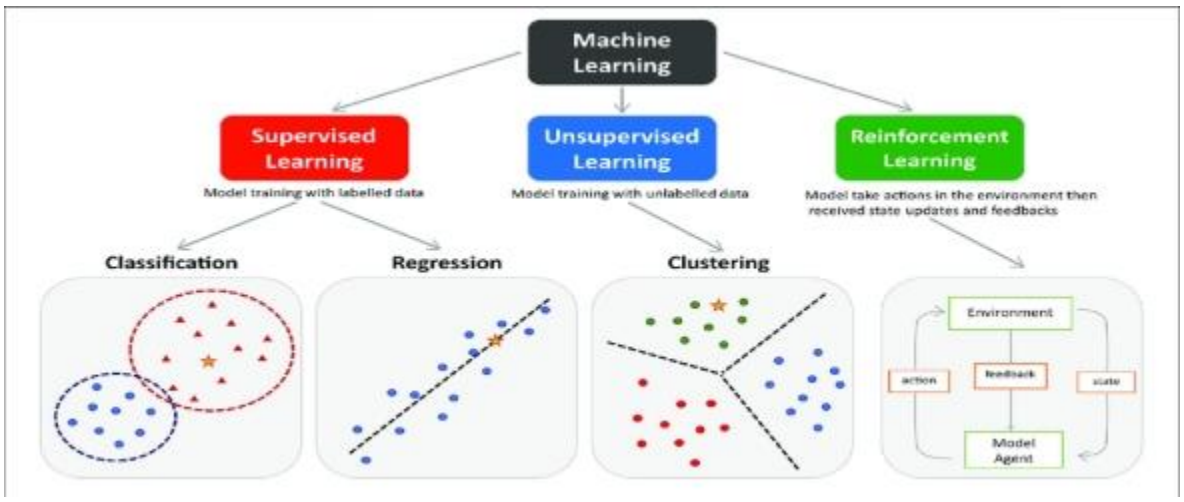


Figure III.3 *:* Exemple de Classification, Régression et Clustering [27]

III.1.1.3 ML vs. Deep Learning

La plupart des techniques d'apprentissage automatique (Machine Learning) obtiennent de bons résultats en raison des représentations et des fonctions d'entrée préalablement conçues par des humains. Dans ce contexte, l'apprentissage automatique consiste principalement à ajuster les poids de manière à améliorer la précision des prédictions finales.

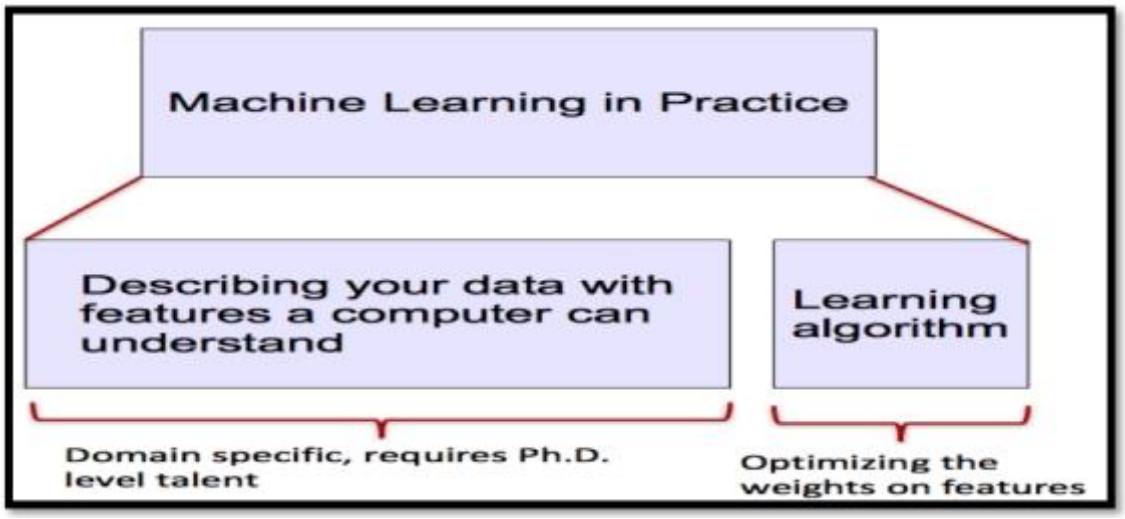


Figure III. 3 : Machine Learning en Pratique[28]

III.1.2 Notion de base sur l'apprentissage en profondeur (Deep Learning)

III.1.2.1 Qu'est-ce que l'apprentissage en profondeur (DL)

Une sous-discipline de l'apprentissage automatique se concentre sur la création de représentations de données pour l'apprentissage. C'est particulièrement efficace pour la création de modèles d'apprentissage. Les algorithmes d'apprentissage profond cherchent à apprendre des représentations à plusieurs niveaux en utilisant une hiérarchie de couches multiples. Lorsque vous alimentez le système avec une grande quantité d'informations, il commence à comprendre ces données et à réagir de manière utile.

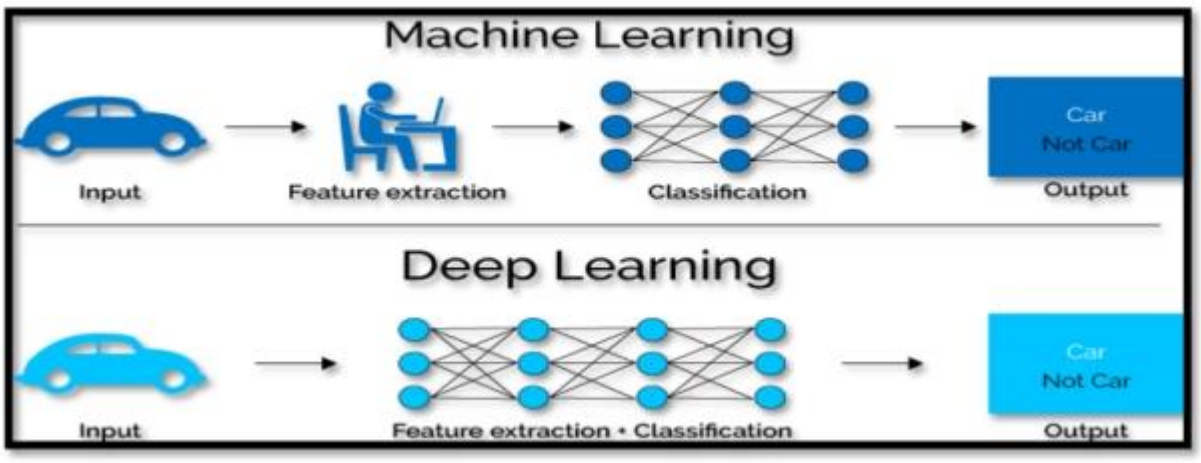


Figure III. 4 : Machine Learning VS Deep Learning[28]

III.1.2.2 Pourquoi DL est-il utile ?

1. Les fonctionnalités conçues manuellement sont souvent surspécifiées, incomplètes et prennent beaucoup de temps à concevoir et à valider.

2. Les fonctionnalités apprises sont faciles à adapter, rapides à apprendre.

3. L'apprentissage en profondeur fournit un cadre très souple, universel (presque ?) Et pouvant être appris pour représenter des informations mondiales, visuelles et linguistiques.

4. Peut apprendre à la fois sans surveillance et sous surveillance.

5. Apprentissage efficace du système conjoint de bout en bout.

6. Utiliser de grandes quantités de données d'entraînement **[29]**.

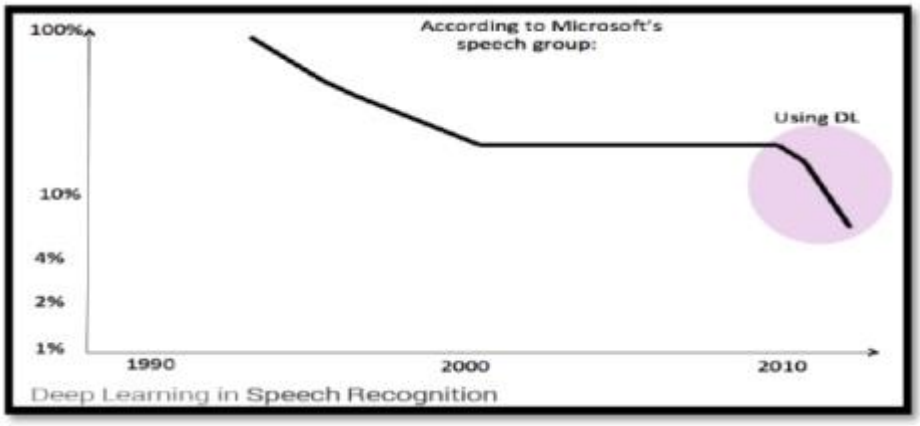


Figure III. 5 : Apprentissage en profondeur dans la reconnaissance vocale[29]

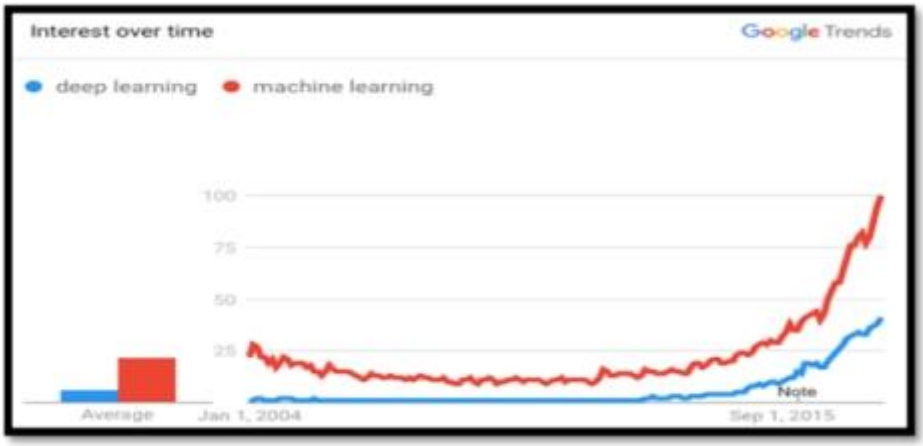


Figure III. 6 : Google Trends l'Utilisation DL vers ML[29]

- Plusieurs améliorations majeures de la PNL au cours des dernières années
  - ✓ Traduction automatique
  - ✓ Analyse des sentiments
  - ✓ Agents de dialogue
  - ✓ Question répondant

- ✓ Classification du texte…

- Tirer parti des différents niveaux de représentation
  - ✓ Mots et caractères.
  - ✓ Syntaxe et sémantique.

III.2 Classification et Réseau de neurones

III.2.1 Classification

III.2.1.1 Concepts & Définition

La classification est le processus, qui permet de grouper des objets (observations ou individus) dans des classes (clusters) de manière à ce que les objets appartenant à la même classe soient plus similaires entre eux qu'aux objets appartenant aux autres classes. Le calcul de la proximité entre objets se fait sur une série de variables mesurées sur tous les objets. **[30]** La classification connait une large utilisation dans plusieurs domaines notamment de l'intelligence artificielle comme l'analyse financière (prévision d'évolution de marchés), Marketing (établir un profil client, mailing), Banque (attribution de prêts), Médecine (aide au diagnostic), Télécom (détection de fraudes). Biométrie, Robotique, Reconnaissance de forme (OCR, Transcription de la parole, Compréhension/Dialogue), Recherche d'information (moteur internet, moteur multimédia) ou enfin la détection Précoce des attaques dans des réseaux informatiques qui est le sujet de notre projet de fin d'études.

III.2.1.3 Taxonomie de Classification

Un grand nombre de méthodes est établi pour résoudre à peu près tous les problèmes de classification, cependant du fait que certaines de ces approches partagent des caractéristiques communes, soit dans la façon d'appréhender le problème (apprentissage supervisé ou non), soit dans la nature de la sortie réalisée (groupes disjoints ou classification flou), la Figure III.9 montre le regroupement de ces méthodes sous la forme d'une hiérarchie (taxonomie). **[30]**

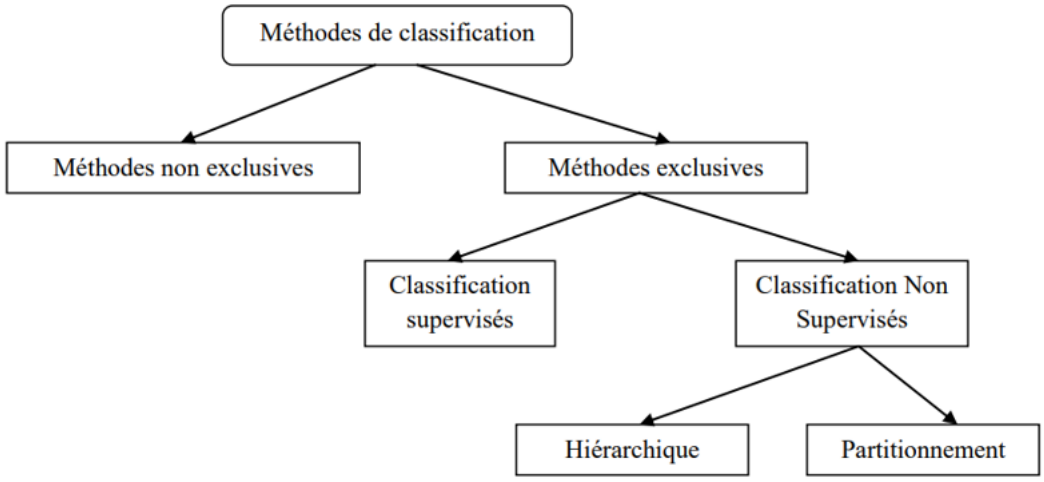


Figure III. 7 : Les méthodes de classification [30]

Nous donnons dans ce qui suit une rapide description des méthodes décrites dans la figure ci-dessus.

III.2.1.3.1 Classification Exclusive

Dans la classification exclusive un objet ne peut être qu'à une seule classe dans la partition finale.

III.2.1.3.1.1 Classification non supervisée

La classification non supervisée, également appelée clustering, implique le regroupement d'individus similaires dans le même groupe, sans avoir préalablement connaissance de la structure des classes. Les classes ne sont pas prédéfinies, et les exemples à disposition ne sont pas étiquetés. L'objectif principal est de former des groupes. Parmi les méthodes couramment utilisées en classification non supervisée, on trouve le **K-Means** (centres mobiles) et la méthode hiérarchique. **[32]**

III.2.1.3.1.1.1 - Classification ascendante hiérarchique (CAH) : [31]

La classification ascendante hiérarchique vise à créer une série de partitions imbriquées des données en un certain nombre de classes, allant de **"n"** classes à **"n-1"** classes, puis à **"n-2"** classes, et ainsi de suite jusqu'à une seule classe. Cela peut être représenté visuellement sous la forme d'un dendrogramme, comme illustré dans la Figure III.10.

Le principe de cette méthode est le suivant :

- À l'étape initiale, chaque l'un des **"n"** individus est considéré comme une classe à part entière.
- Ensuite, on calcule les distances entre tous les individus deux par deux, et on regroupe les deux individus les plus proches dans une seule classe.
- On répète ce processus en calculant à chaque fois la distance entre la nouvelle classe ainsi formée et les **"n-2"** individus restants, puis en fusionnant à nouveau les deux éléments (classes ou individus) les plus proches.

Ainsi, ce processus itératif permet de construire une hiérarchie de regroupements des données en classes de différentes tailles, jusqu'à obtenir une seule classe globale.

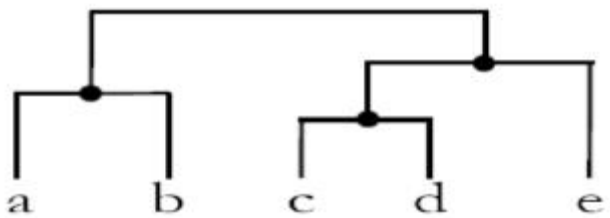


Figure III. 8: Exemple de dendrogramme [30]

III.2.1.3.1.1.2 -Méthode des K-means (Centres mobiles)

Cet algorithme fut longtemps utilisé sur les grands jeux de données en raison de sa rapidité. On s'intéresse tout d'abord à l'algorithme lui-même, puis à ses propriétés.

Principe : On suppose qu'il existe K classes distinctes. On commence par designer K centres de classes $\mu 1, ..., \mu K$ parmi les individus. Ces centres peuvent être soit choisis par l'utilisateur pour leur "représentativité", soit désignés aléatoirement. On réalise ensuite itérativement les deux étapes suivantes :

- Pour chaque individu qui n'est pas un centre de classe, on regarde quel est le centre de classe le plus proche. On définit ainsi K classes *C1, ..., CK, où Ci = {ensemble des points les plus proches du centre μi},*

- Dans chaque nouvelle classe Ci, On définit le nouveau centre de classe μi comme étant le barycentre des points de Ci. **[31]**

L'algorithme s'arrête suivant un critère d'arrêt fixé par l'utilisateur qui peut être choisi parmi les suivants :

- Soit le nombre limite d'itérations est atteint, soit l'algorithme converge, c'est-à dire qu'entre deux itérations les classes formées restent les mêmes,

- Soit l'algorithme a "presque" converge, c'est-à-dire que l'inertie intra-classe ne s'améliore quasiment plus entre deux itérations.

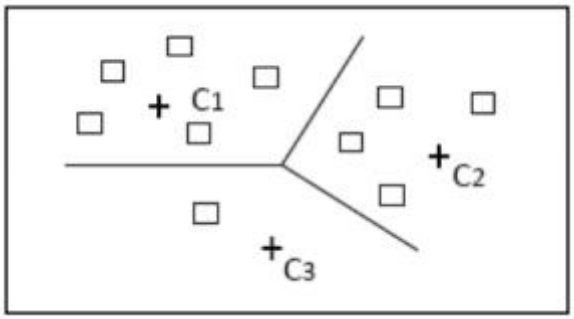


Figure III. 9 : Exemple de Partition obtenue par les centres mobiles[31]

III.2.1.3.1.2 Classification Supervisé

Dans la classification supervisée, les classes sont préalablement définies, et un ensemble d'objets déjà étiquetés est utilisé comme ensemble d'apprentissage. L'objectif est de créer un modèle capable d'attribuer une classe appropriée à un nouvel objet en se basant sur les exemples déjà étiquetés, qui servent de données d'apprentissage. Parmi les méthodes couramment utilisées en classification supervisée, on trouve la méthode des "k plus proches voisins" et la méthode de l'arbre de décision.**[32]**

III.2.1.3.1.2.1 - K plus proches voisins

La méthode des "k plus proches voisins" est l'une des méthodes de classification supervisée les plus élémentaires. Son principe se résume de la manière suivante : pour classer un objet x (c'est-à-dire prédire à quelle classe appartient un nouveau cas), on calcule la distance entre x et tous les échantillons de référence (qui représentent les différentes classes). Ensuite, on choisit les K échantillons les plus proches de x et on attribue à x la classe qui est la plus fréquemment représentée parmi ces K échantillons.

III.2.1.3.1.2.2 Arbres de décision

Les arbres de décision sont les méthodes les plus connues en classification. Le principe de cette méthode est de réaliser la classification d'un exemple par une suite de tests sur les attributs qui le décrivent. Concrètement, dans la représentation graphique d'un arbre où :

- Un nœud interne correspond à un test sur la valeur d'un attribut.
- Une branche part d'un nœud et correspond à une ou plusieurs valeurs de ce test.
- Une feuille est un nœud d'où ne part aucune branche et correspond à une classe.

Une règle de décision (de la forme si . . . alors . . .) est créée pour chaque chemin partant de la racine de l'arbre et parcourant les tests (en faisant des conjonctions) jusqu'à la feuille qui est l'étiquette de la classe Il y a d'autre méthodes de la classification supervisée, tel que la méthode de réseaux de neurones et vu que nous nous basons sur ces réseaux pour réaliser notre modèle de détection précoce, ces derniers vont être étudiés en détails dans les sections suivantes de ce chapitre. **[33]**

III.2.1.3.2 Classification Non Exclusive

Chaque objet est associé à une de densité de probabilité qui indique pour chacune des classes la probabilité que l'objet considéré y appartienne (la classification floue).

III.2.2 Réseau de Neurones (Neural Network)

Les réseaux de neurones artificiels ont permis de formaliser de manière significative le fonctionnement du cerveau humain. Les chercheurs ont progressivement compris comment le cerveau humain fonctionne en termes de capacité d'adaptation, de transférabilité de l'apprentissage, de reconnaissance d'objets et de prise de décision. Le cerveau, qui est composé de milliards de neurones interconnectés de manière extrêmement complexe, forme un vaste réseau de neurones interagissant les uns avec les autres. Cette interconnexion entre les cellules nerveuses confère au cerveau la capacité de stocker, d'accéder et de traiter des informations, des images, des sons et des séquences de signaux reçus à travers les multiples neurones. De plus, les réseaux de neurones permettent également l'apprentissage par la répétition et la correction des erreurs.

III.2.2.1 Concepts & Définitions

L'histoire des réseaux de neurones artificiels remonte à 1943, lorsque Mac Culloch et Pitt ont introduit des modèles de neurones formels qui imitaient le fonctionnement des neurones biologiques et qui pouvaient mémoriser des fonctions booléennes simples. Ces premiers réseaux de neurones artificiels, basés sur ces neurones formels, ont été inspirés par le système nerveux biologique. Ils ont été conçus pour reproduire certaines caractéristiques des mémoires biologiques, notamment leur capacité à fonctionner de manière massivement parallèle, à apprendre et à stocker des informations dans les connexions entre les neurones, ainsi qu'à traiter des données incomplètes.

En 1949, Hebb a mis en évidence l'importance de la connectivité synaptique dans le processus d'apprentissage, soulignant comment les liaisons entre les neurones se renforcent ou se dégradent en fonction de l'interaction du cerveau avec son environnement. Le premier modèle opérationnel, appelé "perceptron simple" et capable d'apprentissage, a été présenté en 1958 par Rosenblatt.

Toutefois, les limitations du perceptron monocouche en termes de performances ont été démontrées en 1969 par les mathématiciens Minsky et Papert.

Les travaux de Hopfield en 1982 ont démontré que les réseaux de neurones artificiels pouvaient résoudre des problèmes d'optimisation, tandis que ceux de Kohonen, également en 1982, ont montré leur capacité à accomplir des tâches de classification et de reconnaissance.**[34]**

III.2.2.1.1 Neurone Biologique

Le neurone est une cellule qui comprend un corps cellulaire et un noyau. Le corps cellulaire se divise en plusieurs branches, appelées dendrites, qui peuvent parfois être très nombreuses, créant ainsi une structure dendritique complexe. Les dendrites sont responsables de l'acheminement de l'information depuis l'extérieur du neurone jusqu'au soma, le corps principal du neurone. Une fois traitée par le neurone, l'information est ensuite transmise le long de l'axone, qui est unique, pour être relayée vers d'autres neurones. Il est important de noter que la transmission de l'information entre deux neurones n'est pas directe. En réalité, il existe un minuscule espace intercellulaire, de l'ordre de quelques dizaines d'angströms (10-9 mètres), entre l'axone du neurone qui envoie l'information (appelé neurone afférent) et les dendrites (on parle d'une dendrite) du neurone récepteur. Cette zone de contact entre deux neurones est appelée la synapse. (**Figure III.12**) **[35]**

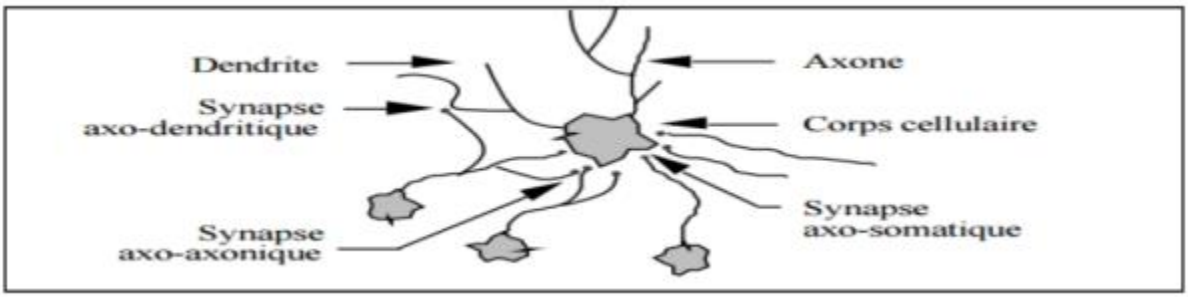


Figure III. 10 : Structures d'un Neurones Biologiques [35]

III.2.2.1.2 Neurone Artificiel

Un neurone artificiel (formel) est un processeur élémentaire stimulé par des neurones qui le précèdent, qu'on appellera ses entrées, à chacune de ces entrées est associé un poids représentatif de la force de la connexion w et, en fonction de cette stimulation, on lui attribue une valeur qu'on appellera sa sortie. Qui se ramifie ensuite pour alimenter un nombre variable de neurones avals.

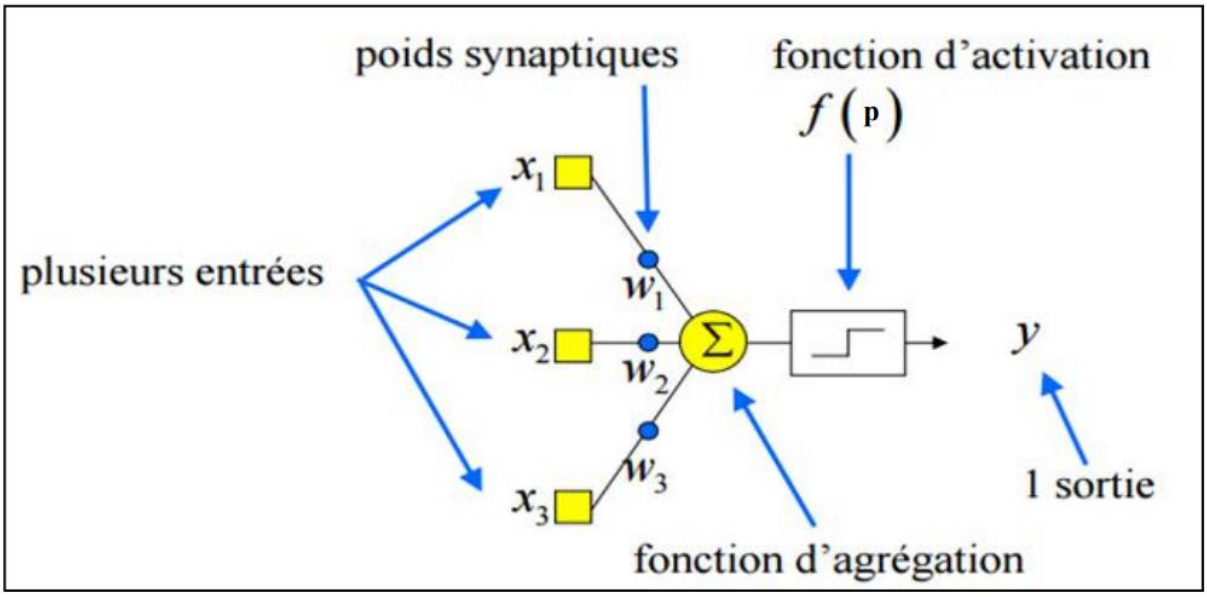


Figure III. 11 : Structures d'un Neurones Artificiel [35]

III.2.2.1.3 Correspondance entre neurone biologique et neurone artificiel

La structure d'un neurone artificiel est en fait inspirée de la structure des neurones biologiques. Les principales structures biologiques des neurones ont toutes leurs équivalents artificiels, ceci ayant pour but de reproduire leur fonctionnement de la meilleure façon possible (d'une manière logique, simple et facilement représentable sur informatique).

Neurone biologique	Neurone artificiel
Synapses	Poids de connexions
Axones	Signal de sortie
Dentride	Signal d'entrée
Somma	Fonction d'activation

Tableau III. 1 : Transition entre les neurones biologiques et les neurones formel

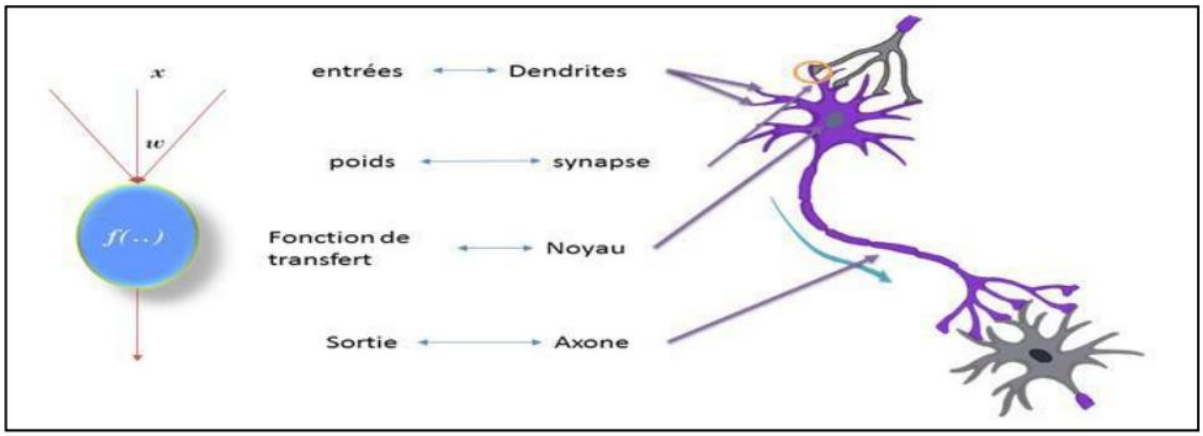


Figure III. 12 : Correspondance entre neurone artificiel et neurone biologique [35]

III.2.2.2 Comportement de Neurone Artificiel

- **Entrées du neurone « X » :** Elles proviennent soit d'autres éléments 'processeurs' (neurones), soit de l'environnement.
- **Le poids « W » (coefficient synaptique) :** Est une valeur numérique associée à une connexion entre deux unités (neurones) qui reflète la force de relation (connexion) entre ces deux unités i et j, et il est noté par Wi.
- **La fonction d'agrégation (combinaison) « P » :** Elle combine les entrées et les poids en calculant l'influence de chaque entrée en tenant en compte de son poids. Cette influence est calculée via la formule suivante : $P = \sum Wi\mathrm{X}i$, Où Wi est le poids de la connexion à l'entrée i, Xi : est le signal de l'entrée i.
- **La fonction de transfert (d'activation) :** La fonction d'activation (la fonction de transfert) joue un rôle très important dans le comportement du neurone. Elle retourne une valeur représentative de l'activation du neurone, cette fonction a comme paramètre la somme pondérée des entrées ainsi que le seuil d'activation. Elle calcule la valeur de sortie à partir du résultat de la fonction de combinaison : S = F(P).

Où : **S** : est la valeur de sortie, **F** : est la fonction de transfert.

La nature de cette fonction diffère selon le réseau, les plus courantes sont présentées sur le Tableau III.2, avec leurs équations mathématiques. Sachant que la différence avec les neurones biologiques est que l'état de ces derniers est binaire, par contre la plupart des fonctions de transfert sont continuées et offrant une infinité de valeurs comprises dans l'intervalle **[0, +1] ou [-1, +1]. [33]**

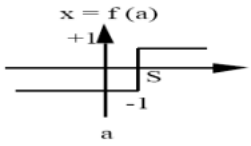
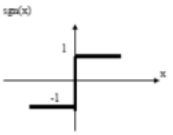
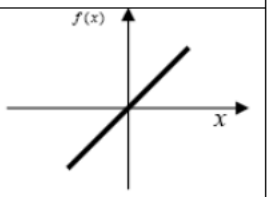
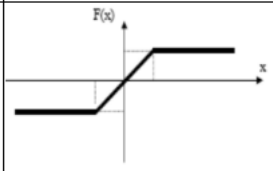
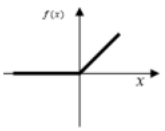
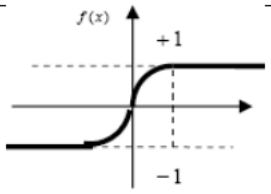
Catégorie	La fonction	La Formule mathématique	Le graphe
seuil	Fonction de Heaviside	$X=\begin{cases}1 \text{ Si } a\geq 0\\ 0 \text{ Si } a<0\end{cases}$	
	Fonction de signe	$S=\begin{cases}1 \ si \ \ a\geq 0\\ -1 \ si \ \ a<0\end{cases}$ Tel que : S = seuil	
Linéaire	Identité	f(x) = x	
	Fonction linéaire à seuil ou multi-seuil	$F(x)=\begin{cases}x & x\in[u,v]\\ v & si\ x\geq v\\ u & si\ x\leq u\end{cases}$	
	Fonction Linéaire positif	$f(x)\begin{cases}0 \text{ si } x<0\\ x \text{ si } x\geq 0\end{cases}$	
Non Linéaire	Logistique (sigmoïde)	$f(x)=\frac{1}{1+e^{-x}}$	

Tableau III. 2: Les fonctions de transfert les plus utilisées

La fonction sigmoïde(dérivable) :

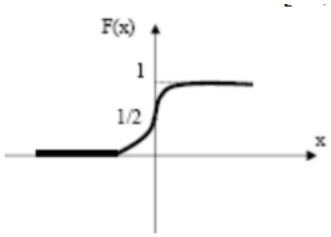
Est la plus utilisée car elle introduit de la non-linéarité, mais c'est aussi une fonction continue, différentiable, C'est la fonction que nous avons utilisée pour mettre en œuvre notre modèle, Une fonction sigmoïde est définie par :

$$f_{sig}(x)=\frac{1}{1+e^{-x}}$$

Possède les propriétés importantes évoquées précédemment (elle n'est pas polynomiale et est indéfiniment continument dérivable). En outre, une propriété simple permet d'accélérer le calcul de sa dérivée, ce qui réduit le temps de calcul nécessaire à l'apprentissage d'un réseau de neurones. On a en effet :

$$\frac{d}{dx} f_{sig}(x) = f_{sig}(x)\,(1 - f_{sig}(x))$$

On peut donc calculer la dérivée de cette fonction en un point de façon très efficace à partir de sa valeur en ce point. De plus, la fonction sigmoïde est à valeurs dans l'intervalle [0 ; 1], ce qui permet d'interpréter la sortie du neurone comme une probabilité.



III.2.2.3 Les Types d'Apprentissage des Neurones

L'apprentissage est sans doute la propriété la plus intéressante des réseaux neuronaux, elle ne concerne cependant pas tous les modèles, mais les plus utilisés. L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré, « C'est-à-dire un changement dans la valeur des poids qui relient les neurones d'une couche à l'autre » **[14].** Soit le poids wij.

Reliant le neurone i à son entrée j. Au temps τ, un changement △wij(τ) de poids peut s'exprimer simplement de la façon suivante :

$$\triangle wij(\tau) = wij(\tau+1) - wij(\tau)$$

Par conséquent, **wij(τ+1) = wij(τ)+ △wij(τ),** avec **wij(τ+1)** et **wij(τ)** représentent respectivement les nouvelles valeurs et les anciennes aussi du poids **wij.** On distingue plusieurs types d'apprentissages, parmi lesquels on cite les trois principaux : non supervisé, par renforcement et supervisé.**[33] [36]**

III.2.2.3.1 Apprentissage Non Supervisé

Le réseau de neurones doit repérer des similitudes entre les exemples en ajustant les pondérations, de manière à produire des sorties similaires pour des entrées ayant des caractéristiques similaires. L'apprentissage non supervisé est particulièrement approprié pour la représentation de données complexes telles que des images ou des sons, qui sont généralement symboliques et pour lesquelles les règles déterminant le comportement du système modélisé par les réseaux de neurones sont moins précises.

III.2.2.3.2 Apprentissage par renforcement

Dans cette situation, même si les sorties souhaitées ne sont pas explicitement connues, il existe un moyen de déterminer si les sorties du réseau de neurones artificiels se rapprochent ou

s'éloignent de l'objectif visé. Par conséquent, les poids sont ajustés de manière plus ou moins aléatoire, et ces ajustements sont conservés s'ils ont un effet positif, sinon ils sont rejetés.

III.2.2.3.3 Apprentissage Supervisé

Comme nous l'avons vu précédemment, un réseau de neurones non bouclé réalise une fonction algébrique entre ses entrées et ses sorties. Donc on peut effectuer à un tel réseau la tâche qui consiste à réaliser une fonction algébrique non linéaire, on fournit à ce réseau un couple (entrée, sortie) et on modifie les poids en fonction de l'erreur entre la sortie désirée et la sortie obtenue, **(Figure III.18).**

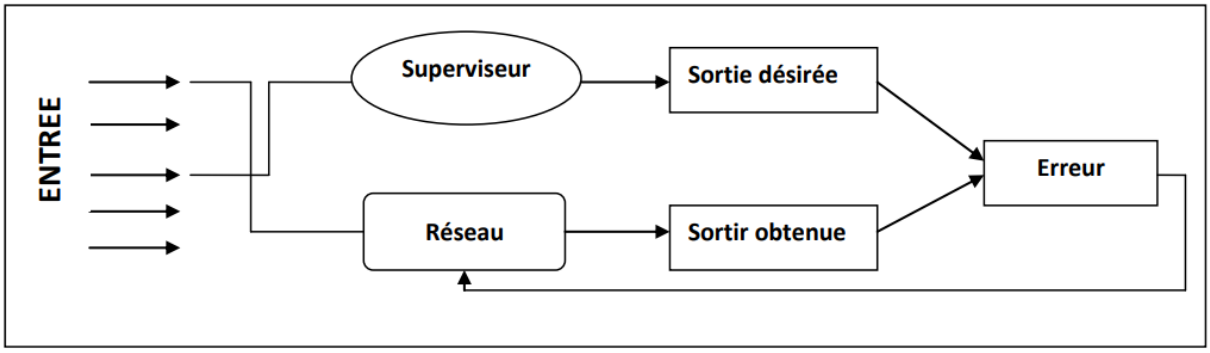


Figure III. 13 : Apprentissage d'un réseau de neurones

La fonction accomplie par ce réseau peut être divisée en deux composantes : une fonction connue analytiquement, où le réseau effectue une approximation, et une fonction dont nous ne connaissons pas l'expression analytique, mais pour laquelle nous disposons d'un ensemble fini de données, qui peuvent provenir de mesures réalisées sur un processus physique, chimique, etc. Dans ce contexte, le réseau effectue une adaptation statique ou une régression. Dans ce chapitre, nous nous concentrons sur l'apprentissage supervisé, en particulier sur la modélisation statique. L'algorithme d'apprentissage que nous utilisons dans notre étude est la rétropropagation des erreurs, car il est particulièrement adapté à la modélisation statique avec des perceptrons multicouches (MLP).

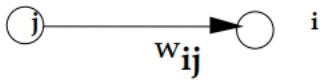
III.2.2.4 Règles d'apprentissage

Il existe plusieurs types de règles d'apprentissage (modification de poids), Parmi lesquelles on trouve :

- **Correction d'erreur Loi de Widrow-Hoff (Règle de delta)** : **ai** activation produite par le réseau, ***di*** réponse désirée par l'expert humain par exemple si la sortie est inférieure à la réponse désirée, il va falloir augmenter le poids de la connexion à condition bien sûr que l'unité j soit excitatrice (égale à 1). On est dans l'hypothèse d'unités booléennes {0,1}.

	ai = 0	ai = 1
di = 0	$\Delta W_{ij} = 0$	$\Delta W_{ij} = -R$
di = 1	$\Delta W_{ij} = R$	$\Delta W_{ij} = 0$

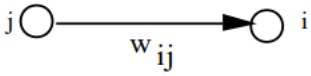
$\Delta W_{ij}=R(d_i_a_i)a_j$



- **Loi de Grossberg :** On augmente les poids qui entrent sur l'unité gagnante ai s'ils sont trop faibles, pour les rapprocher du vecteur d'entrée **aj**.

C'est la règle d'apprentissage utilisée dans les cartes auto-organisatrices de Kohonen

$$\Delta W_{ij} = Ra_i\left(a_j - W_{ij}\right)$$



- **Apprentissage de Boltzmann :**

Les réseaux de Boltzmann sont des réseaux symétriques récurrents. Ils possèdent deux sous-groupes de cellules, le premier étant relié à l'environnement (cellules dites visibles) et le second ne l'étant pas (cellules dites cachées), les machines de Boltzmann opèrent en deux modes distincts:

Le mode figé (clamped), dans ce cas les cellules visibles sont affectées par une valeur déterminée par l'environnement ;

Le mode libre évolution (free-running) dans lequel l'ensemble des cellules, qu'elles soient visibles ou cachées, peuvent changer d'état librement. La règle d'apprentissage est de type stochastique, elle est dérivée de la théorie de l'information et des principes de la thermodynamique. L'objectif de cet apprentissage est d'ajuster les poids des connexions, de sorte que l'état des cellules visibles satisfasse une distribution probabiliste souhaitée. En accord avec la règle d'apprentissage de Boltzmann, les modifications se font par :

Δwi j=h(*r* i j -ri j) où h est le taux d'apprentissage et(r i j), (resp. ri j) sont les corrélations entre les états i et j lorsque le système est en mode figé (resp. Libre-évolution), Les valeurs ***r* i j et ri j** sont classiquement estimées à l'aide de la méthode de Monte-Carlo5 qui est extrêmement lente. **[32]**

- **Règle de Hebb :** ***« Si deux unités connectées sont simultanément actives, Alors le poids de la connexion qui les relie doit être renforcé »,*** **[37],** Selon la formule ci-dessous, R est une constante positive qui représente la force d'apprentissage (Learning-rate).

	ai = -1	ai = 1
aj = -1	$\Delta Wji = R$	$\Delta Wji = -R$
aj = 1	$\Delta Wji = -R$	$\Delta Wji = R$

$\Delta Wji = Ra_ia_j$

- **Apprentissage par compétition :**

  A la différence de la règle de Hebb (dans laquelle plusieurs neurones peuvent être activés en sortie), cet apprentissage n'active qu'un seul neurone. Le principe de cet apprentissage est de regrouper les données en catégories. Les patrons similaires vont donc être rangés dans une même classe, en se basant sur les corrélations des données, et seront représentés par un seul neurone, on parle de « winner-take-all ». Dans un réseau à compétition simple, chaque neurone de sortie est connecté aux neurones de la couche d'entrée, aux autres cellules de la couche de sortie (connexions inhibitrices) et à elle-même (connexion excitatrice).

La sortie va donc dépendre de la compétition entre les connexions inhibitrices et excitatrices, Le résultat de la compétition est de choisir la cellule ayant la plus grande (ou la plus petite) entrée. A noter que seules les connexions du vainqueur sont mises à jour.

III.2.2.5 Quelques Type de Modèle de réseaux de neurones

III.2.2.5.1 Perceptron Simple

Le perceptron est le premier modèle de réseau de neurones inventé en 1957 par Frank Rosenblatt. Le but du perceptron est d'associer des formes en entrée à des réponses. Le perceptron se compose de deux couches : la couche d'entrée et la couche de sortie qui donne la réponse correspondant à la stimulation présente en entrée. Les cellules de la première couche répondent en oui/non. La réponse « oui » correspond à une valeur **« 1 »** et la réponse « non » correspond à une valeur **« 0 »** à la sortie du neurone. Les cellules d'entrée sont reliées aux cellules de sortie grâce à des synapses d'intensité variable. L'apprentissage du perceptron s'effectue en modifiant l'intensité de ces synapses.

Les cellules de sortie évaluent l'intensité de la stimulation en provenance des cellules de la couche d'entrée en effectuant la somme des intensités des cellules actives. Le perceptron doit trouver l'ensemble des valeurs à donner aux synapses pour que les configurations d'entrée se traduisent par des réponses voulues.

Pour cela, on utilise la règle d'apprentissage de Windrow-Hoff (Correction d'erreur). Pour apprendre, le perceptron simple (**Figure III.19**) doit savoir qu'il a commis une erreur, et doit

connaitre la réponse qu'il aurait dû donner. De ce fait, on parle d'apprentissage supervise, La règle d'apprentissage est locale dans ce sens que chaque cellule de sortie apprend sans avoir besoin de connaitre la réponse des autres cellules. La cellule ne modifie l'intensité de ses synapses (apprend) que lorsqu'elle se trompe. Marvin Lee Minsky a montré qu'une forme toute simple **(le XOR)** ne peut être apprise par un neurone de type perceptron.

Un neurone ne peut séparer que deux régions séparables par un hyper plan, avec plusieurs neurones, ça va déjà mieux mais il est vite clair qu'une seule couche de perceptron ne peut pas apprendre des figures complexes. **[38]**

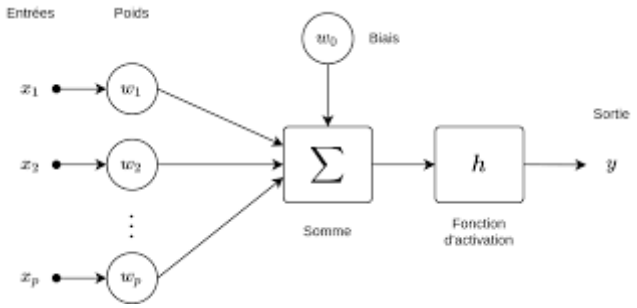


Figure III. 14 : Schéma général de perceptron Simple [38]

III.2.2.5.2 Réseaux auto-organisateur (Réseau de Kohonen)

Les cartes topologiques ou cartes auto organisatrices ont été introduites pour la première fois par T. Kohonen en 1981. Les premiers modèles cherchaient tout particulièrement à représenter des données multidimensionnelles. La particularité la plus importante des cartes auto-organisatrices est qu'elles rendent possible la comparaison des groupements qui ont réalisé directement à partir des données. Une observation est affectée à un groupe qui est projeté en un nœud de la carte. La comparaison des projections liées à deux observations distinctes permet d'apprécier la proximité des groupes dont elles sont issues. **[39]**

Dans la plupart des applications, les neurones d'une carte de Kohonen sont disposés sur une grille 2D (**Figure III.20**). Chaque neurone **i** de la carte effectue un calcul de la distance euclidienne entre le vecteur d'entrée et le vecteur poids **Wi**, Dans les réseaux de Kohonen, la mise à jour des paramètres des neurones s'effectue sur tout un voisinage d'un neurone **i,** Un rayon de voisinage r représente donc la longueur du voisinage d'un neurone i en termes de nombre de neurones. On définit alors une fonction **h(i,k)** égale à **1** pour tous les neurones **k** voisins du neurone **i** compris dans le rayon r et égal à zéro pour tous les autres neurones.

L'algorithme d'apprentissage de la carte de Kohonen se présente comme suit : Après un temps très important de convergence, le réseau évolue de manière à représenter au mieux la topologie

de l'espace de départ. Il faut noter que la notion de conservation de la topologie est en fait abusive puisqu'en général, la taille du vecteur d'entrée est bien supérieure à la dimension de la carte (souvent égale à 2) et il est donc impossible de conserver parfaitement la topologie.

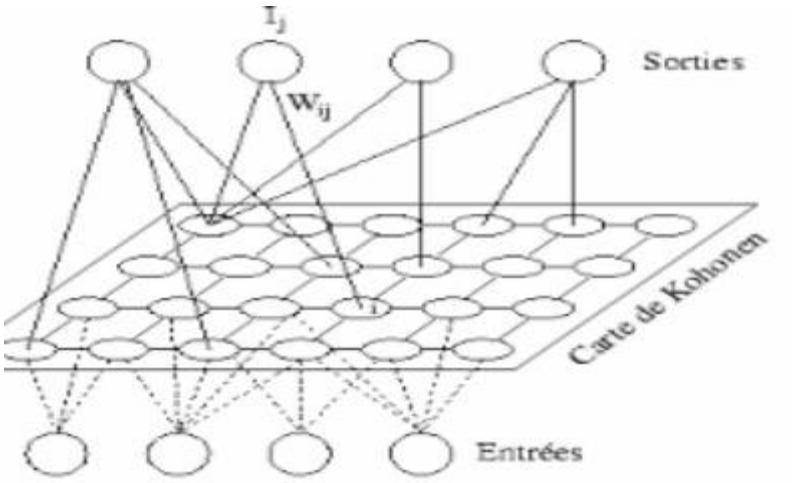


Figure III. 15 : Carte auto-adaptative de kohonen [38]

Les avantages de la carte auto-organisatrice sont :

- L'espace de sortie est un espace de représentation donc on peut visualiser les sorties de la carte.
- Représentation des données de grande dimension. L'un des inconvénients est le temps

de convergence qui est très long. Il n'y a pas de preuve de convergence en multidimensionnel et d'unicité de la représentation

III.2.2.5.3 Perceptron Multicouche (MLP)

Le perceptron multicouches (MLP), avec son algorithme d'apprentissage basé sur la rétropropagation des erreurs, est l'une des architectures de réseaux de neurones les plus couramment utilisées pour résoudre divers problèmes en intelligence artificielle.

Le MLP, ou perceptron multicouche, est un type de réseau de neurones artificiels composé de couches, où l'information circule de manière unidirectionnelle, de la couche d'entrée à la couche de sortie. Dans l'exemple illustré dans la **Figure III.21**, un réseau MLP est constitué d'une couche d'entrée, de deux couches cachées et d'une couche de sortie. La couche d'entrée est toujours accompagnée d'une couche virtuelle associée aux entrées du système.

Dans cet exemple particulier, il y a 3 neurones dans la couche d'entrée, 4 neurones dans la première couche cachée, 3 neurones dans la deuxième couche cachée et 4 neurones dans la couche de sortie. En général, un MLP peut avoir n'importe quel nombre de couches et n'importe quel nombre de neurones (ou d'entrées) par couche. **[40]**

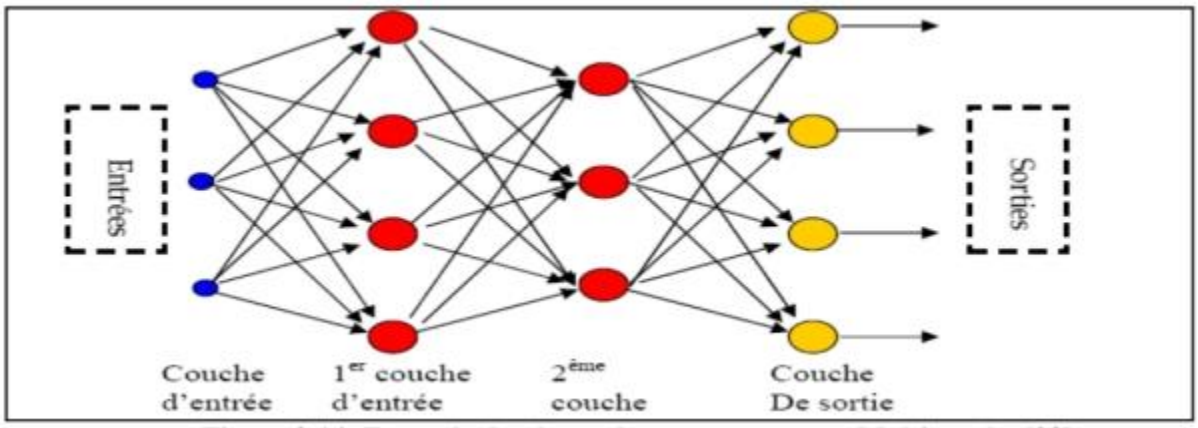
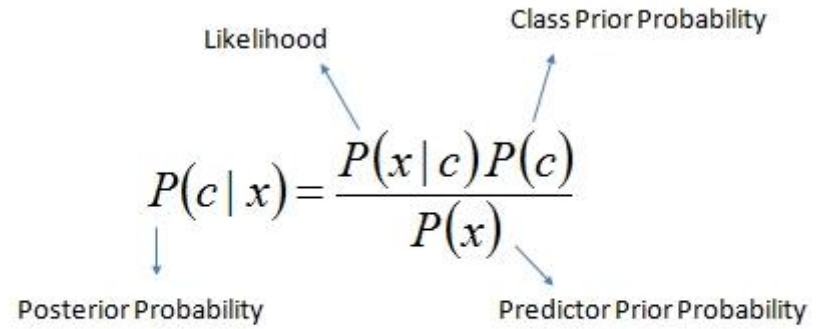


Figure III. 16 : Exemple de réseau de type perceptron Multicouche [38]

III.3 Quelques Algorithmes de Classifications

III.3.1 Naïve Bayes

Naïve Bayes est un algorithme d'apprentissage automatique probabiliste largement utilisé pour la classification, en particulier dans le traitement du langage naturel et la classification de texte. Il se base sur le théorème de Bayes et fait l'hypothèse simplificatrice que les caractéristiques utilisées pour la classification sont indépendantes les unes des autres, d'où le terme "naïf". Pour l'entraînement, il estime les probabilités de chaque classe et des caractéristiques données une classe. Lors de la prédiction, il utilise ces probabilités pour déterminer la classe la plus probable pour de nouvelles données en fonction de leurs caractéristiques observées. C'est un algorithme rapide et efficace, bien qu'il puisse ne pas être adapté à toutes les situations en raison de son hypothèse d'indépendance naïve. Comme la formule qui suit :



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure III. 17 :Formule de naive bayes

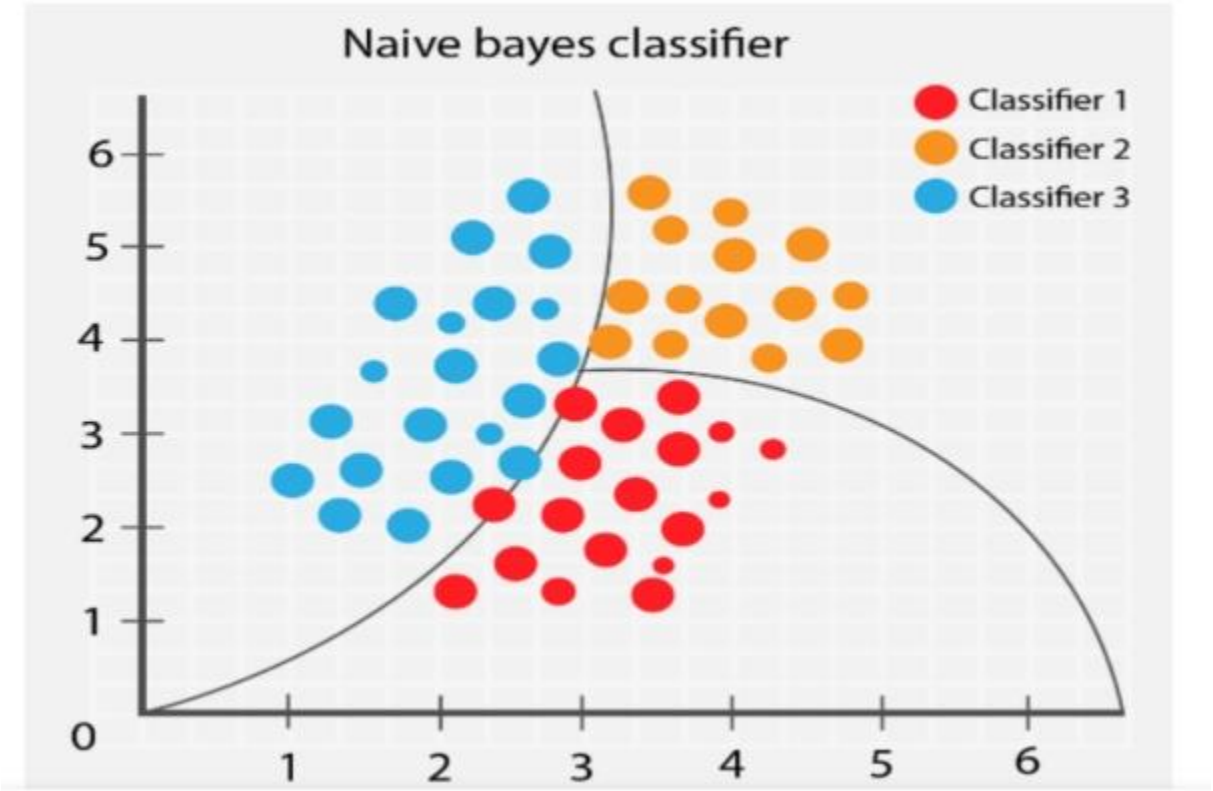


Figure III. 18 : Schéma d'illustration

III.3.2 Arbre de Décision

Un arbre de décision est un modèle d'apprentissage automatique qui prend des décisions en suivant une série de questions basées sur des caractéristiques des données. Il divise progressivement les données en sous-groupes jusqu'à ce qu'une décision soit prise. Les arbres de décision sont utilisés pour la classification et la prédiction, offrent une interprétabilité élevée et sont efficaces pour résoudre une variété de problèmes. Cependant, ils peuvent être sujets au surapprentissage s'ils deviennent trop complexes, ce qui nécessite une régularisation appropriée.

III.3.3 Machine à Vecteur de Support (SVM)

Les machines à vecteur de support (SVM) sont des algorithmes d'apprentissage supervisé utilisés pour la classification et la régression. Leur objectif principal est de trouver un hyperplan de séparation qui maximise la marge entre les classes dans un ensemble de données. Les SVM sont efficaces pour traiter des ensembles de données de grande dimension et peuvent être étendues pour gérer des données non linéaires en utilisant des noyaux. Elles sont appréciées pour leur capacité à trouver des frontières de décision robustes tout en évitant le surapprentissage.

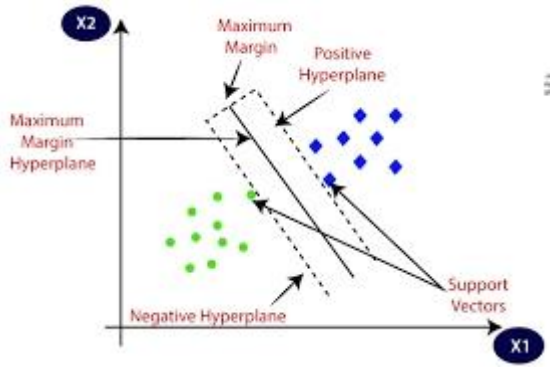


Figure III. 19 : Machine à Vecteurs de support (SVM)

III.3.4 L'algorithme des K plus proche voisin (KNN)

III.3.4.1 Définition et Historique

L'algorithme des k plus proches voisins s'écrit en abrégé k-NN ou KNN, de l'anglais k-nearest neighbors, appartient à la famille des algorithmes d'apprentissage automatique (machine learning). Le terme de machine learning a été utilisé pour la première fois par l'informaticien américain Arthur Samuel en 1959. Les algorithmes d'apprentissage automatique ont connu un fort regain d'intérêt au début des années 2000 notamment grâce à la quantité de données disponibles sur internet. L'algorithme des k plus proches voisins est un algorithme d'apprentissage supervisé, il est nécessaire d'avoir des données labellisées. À partir d'un ensemble E de données labellisées, il sera possible de classer (déterminer le label) d'une nouvelle donnée (donnée n'appartenant pas à E). À noter qu'il est aussi possible d'utiliser l'algorithme des k plus proches voisins à des fins de régression en statistiques (on cherche à déterminer une valeur à la place d'une classe), mais cet aspect des choses ne sera pas abordé en première. De nombreuses sociétés (exemple les GAFAM) utilisent les données concernant leurs utilisateurs afin de "nourrir" des algorithmes de machine learning qui permettront à ces sociétés d'en savoir toujours plus sur nous et ainsi de mieux cerné nos "besoins" en termes de consommation.

III.3.4.2 Principe de l'algorithme

L'algorithme des k-plus proche voisins (KNN) est une méthode d'apprentissage à base d'instances. Il ne comporte pas de phase d'apprentissage en tant que telle. Cette méthode enregistre toutes les classes qui lui sont fournies à l'aide de données d'apprentissage, définit et classe les nouvelles instances en fonction d'une mesure de similarité. Ses k plus proches voisins sont alors considérés : on observe leur classe/catégorie, et celle qui revient le plus (la classe majoritaire) parmi les voisins est affectée aux nouvelles instances à classer.

La distance peut, en général, être n'importe quelle mesure métrique : la distance euclidienne standard est le choix le plus courant. Les méthodes basées sur les voisins sont connues sous le nom de méthodes d'apprentissage automatique non généralisantes, car elles « se souviennent » simplement de toutes ses données d'apprentissage

Données en entrée :

- Un ensemble de données D.
- Une fonction de définition distance d

- Un nombre entier K

Pour une nouvelle observation X dont on veut prédire sa variable de sortie y faire:

1. Calculer toutes les distances de cette observation X avec les autres observations du jeu de données D
2. Retenir les K observations du jeu de données D les proches de X en utilisant la fonction de calcul de distance d
3. Prendre les valeurs de y des K observations retenues :
    a. Si on effectue une régression, calculer la moyenne (ou la médiane) de y retenus
    b. Si on effectue une classification, calculer le mode de y retenues
4. Retourner la valeur calculée dans l'étape 3 comme étant la valeur qui a été prédite par K-NN pour l'observation X. **[38]**

III.3.4.3 Calcul de Similarité dans l'algorithme de K-NN

Il existe plusieurs fonctions de calcul de distance, notamment, la distance euclidienne, la distance de Manhattan, la distance de Minkowski …etc. On choisit la fonction de distance en fonction des types de données qu'on manipule. Ainsi pour les données quantitatives (exemple : poids, salaires, taille, montant de panier électronique etc…) et du même type, la distance euclidienne est un bon candidat. Quant à la distance de Manhattan, elle est une bonne mesure à utiliser quand les données (input variables) ne sont pas du même type (exemple : âge, sexe, longueur, poids etc…).

Il est inutile de coder, soi-même ces distances, généralement, les librairies de machine learning comme scikit learn, effectue ces calculs en interne. Il suffit juste d'indiquer la mesure de distance qu'on souhaite utiliser.

III.3.4.3.1 Calcul de distance

a) La Distance euclidienne

- Distance qui calcule la racine carrée de la somme des différences carrées entre les coordonnées de deux points :

$$D_e(x,y) = \sqrt{\sum_{j=1}^{n}(x_j - y_j)^2}$$

Figure III. 20 : Calcul de la distance euclidienne

b) La distance de Manhattan

- La distance de Manhattan : calcule la somme des valeurs absolues des différences entre les coordonnées de deux points :

$$D_m(x,y) = \sum_{i=1}^{k} | x_i - y_i |$$

Figure III. 21: Distance de Manhattan

III.3.4.3.2 Choix de K

Le choix de la valeur k à utiliser pour effectuer une prédiction avec K-NN, varie en fonction du jeu de données. En règle générale, moins on utilisera de voisins (un nombre K petit) plus on sera sujette aux sous apprentissage (underfitting). Par ailleurs, plus on utilise de voisins (un nombre K grand) plus, sera fiable dans notre prédiction. Toutefois, si on utilise nombre de voisins avec K=N et N étant le nombre d'observations, on risque d'avoir du overfitting et par conséquent un modèle qui se généralise mal sur des observations qu'il n'a pas encore vu.

Exemple : dans l'exemple suivant, si K=3 il (le ?) sera classé avec les croix, si K=5 il sera classé avec les ronds **[41]**

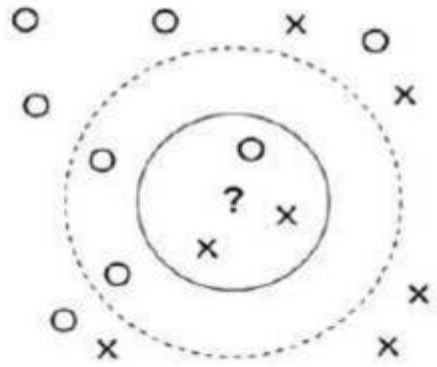


Figure III. 22: Algorithme KNN

Toutes les méthodes discutées sont connues pour leurs traits saillants et leurs inconvénients inhérents. Il faut du temps pour que l'arbre de décision soit construit. La méthode du voisin le plus proche prend énormément de temps lorsque la taille de l'ensemble de l'ensemble de données augmente. Ne fonctionne mieux que sur des données numériques, ce qui nécessite la conversion des données textuelles de l'ensemble de données en une valeur numérique

III.4 Deep Learning de la Sécurité des Réseaux

La sécurité du réseau est très importante dans le domaine des technologies de l'information, car elle vise à protéger la partie matérielle et logicielle du système en utilisant des stratégies de sécurité. Avec la croissance des données dans le secteur des technologies de l'information, qui inclut les machines réseau, les systèmes d'exploitation et les applications, il est essentiel de prévenir les problèmes du système.

Actuellement, la plupart des approches pour assurer la sécurité des réseaux sont manuelles. Cependant, il existe une nouvelle approche appelée "apprentissage en profondeur" qui s'inspire de la manière dont le cerveau humain fonctionne en apprenant des caractéristiques à différents niveaux d'abstraction. Un type particulier d'architecture d'apprentissage en profondeur, appelé réseau de Boltzmann profond (DBN), est capable d'apprendre des fonctions complexes de l'entrée à la sortie sans dépendre de caractéristiques spécifiques créées par l'homme. DBN utilise un algorithme d'apprentissage non supervisé, appelé machine de Boltzmann restreinte (RBM), pour chaque couche de son architecture.

En résumé, l'apprentissage en profondeur, et en particulier les réseaux de Boltzmann profonds, offre une approche automatisée pour renforcer la sécurité des réseaux en apprenant des modèles

complexes directement à partir des données, sans nécessiter une intervention humaine constante. **[41]**

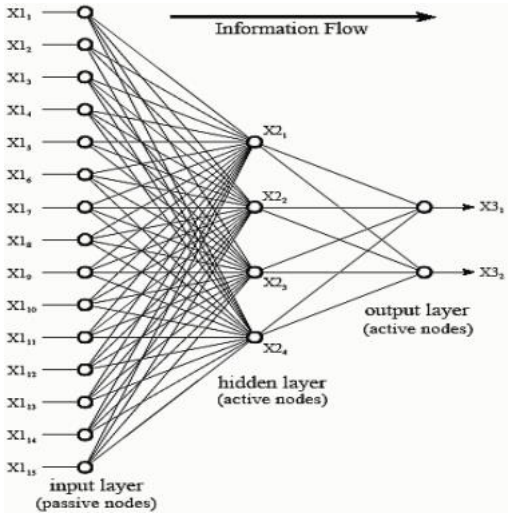


Figure III. 23: Structure of Deep Learning [41]

IV.4.1 Identification de la Circulation

L'identification du trafic est cruciale pour la sécurité du réseau car elle permet de repérer toute intrusion dans le système. Actuellement, les méthodes traditionnelles de détection, comme l'identification du port et le protocole HTTP standard, deviennent moins efficaces en raison de l'augmentation des données. Une approche basée sur la signature, qui repose sur des données spécifiques, est également utilisée.

Les chercheurs ont développé plusieurs algorithmes, tels que **Naive Bayes**, **Random Forests** et **Decision Tree**, pour classer le trafic réseau. Certains ont même combiné **SVM** et RBM pour détecter le trafic réseau.

L'apprentissage automatique a apporté des améliorations significatives à l'analyse de la sécurité des réseaux. L'apprentissage en profondeur, en particulier, peut détecter les anomalies en utilisant des données statistiques pour évaluer les problèmes réseau. Pour comprendre ces anomalies, il est crucial d'évaluer divers aspects de la sécurité du réseau. **[40]**

IV.4.2 Facteurs de la Sécurité Réseau

Premièrement, détecter une intrusion dans le système est complexe car il faut distinguer les données normales des données anormales. Pour ce faire, une méthode de détection doit identifier les caractéristiques des données malveillantes et créer un système capable de classifier avec précision les données authentiques et malveillantes. Cette méthode, appelée réduction de dimensionnalité, utilise des approches de codage automatique pour mesurer la distance entre les nœuds du réseau. La cohérence de cette distance indique la normalité des données, et des distances plus longues peuvent signaler la présence de données malveillantes. Deux systèmes de mesure, la distance de Manhattan et la distance euclidienne, sont utilisés à cet effet.

Deuxièmement, un défi de l'apprentissage en profondeur est la qualité des données utilisées pour détecter une anomalie, un phénomène appelé empoisonnement à la normalité. Les caractéristiques utilisées pour extraire les données sont cruciales, surtout dans l'apprentissage en profondeur non supervisé. Il est essentiel que cette approche garantisse que les informations normales ne soient pas altérées. **[41]**

IV.4.3 Technique utilisée pour détecter les anomalies dans la sécurité du réseau

De manière significative, plusieurs approches exploitant l'apprentissage en profondeur peuvent être utilisées pour repérer des anomalies dans la sécurité du réseau. Pour débuter, la réduction de la dimensionnalité de l'encodeur automatique constitue un système reposant sur des composants d'encodeur et de décodeur, comprenant des couches d'entrée, de sortie, et cachées. Ce système suit trois étapes cruciales : la formation préalable, le déroulement, et un ajustement précis.

Le réseau de croyances profondes (DBN) représente également une méthode d'apprentissage en profondeur, utilisant une couche non supervisée de GAR (Gaussian Belief Network) et un niveau supervisé de réseaux de tiers. Le DBN se divise en deux parties, où le RBM (Restricted Boltzmann Machine) est conditionné de manière non supervisée. Le second processus implique l'utilisation de la dernière sortie du RBM comme entrée du nouveau niveau de BP (Backpropagation), qui est ensuite classé par une approche supervisée.

Par conséquent, ces deux techniques se conjuguent pour créer un système de détection hybride des activités malveillantes. Cette approche repose sur la réduction de la dimensionnalité via l'utilisation de la méthode de codage automatique, créant un espace entre les vecteurs. Ces données sont ensuite classifiées à l'aide du système DBN basé sur l'apprentissage en profondeur. Enfin, la précision de la détection est optimisée pour minimiser les complexités temporelles associées au système hybride.

L'identification de protocoles inconnus constitue un défi majeur pour les méthodes de détection traditionnelles en matière de sécurité des réseaux, car environ 17% des flux de trafic sur les réseaux restent inconnus. Cependant, l'apprentissage en profondeur s'efforce d'atténuer ce problème en étant capable d'identifier plus de la moitié du flux de trafic inconnu par rapport aux méthodes traditionnelles.

De manière plus avantageuse, cette approche permet de calculer une probabilité sur les flux inconnus, améliorant ainsi la précision du processus de détection **[41]**

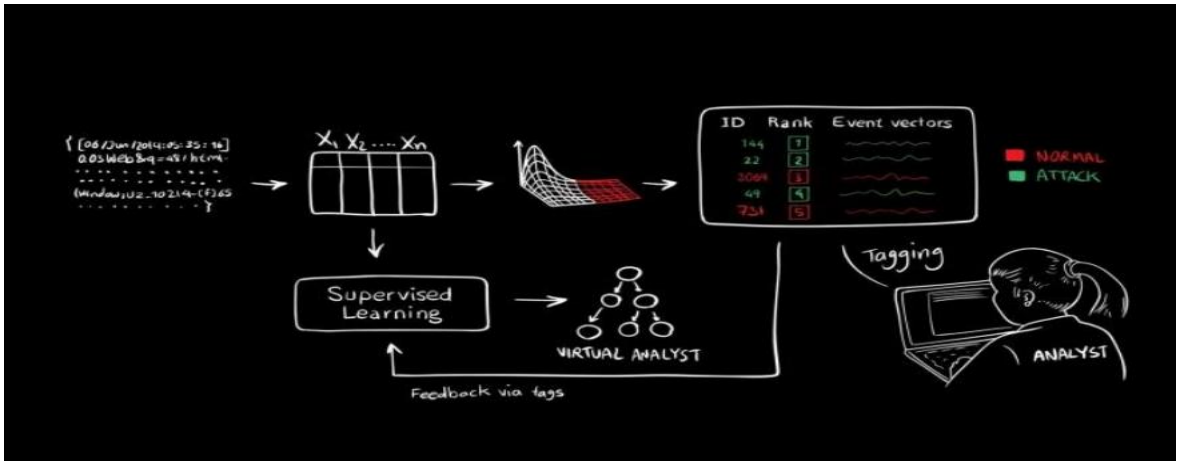


Figure III. 24 : Analyse les données et détecte les activités suspectes à l'aide d'un apprentissage automatique non supervisé.[41]

III.5 CONCLUSION

La mise en œuvre de solutions de machine Learning est grandement facilitée par les implémentation open source des différents algorithmes. Il n'est alors plus nécessaire de connaitre le fonctionnement exact de ces derniers ainsi que les mathématiques poussées dont ils proviennent.

Néanmoins, une bonne connaissance des données sur lesquelles on travaille est nécessaire pour concevoir des modèles efficaces. Avant de considérer faire des prédictions, il faut déjà savoir collecter les données, les prétraiter (car les sources sont bien souvent inconstantes), les stocker, les visualiser pour mieux les connaitre et enfin seulement il est intéressant d'envisager le machine learning. Le prochain chapitre va expliquer l'utilisation du machine learning dans le domaine des détection précoces des attaques e-mail de types phishing.

CHAPITRE IV : MISE EN PLACE D'UNE DETECTION PRECOCE AVEC DEEP LEARNING : CAS DU DDoS

IV.1 INTRODUCTION

Dans le paysage numérique actuel, les attaques par déni de service distribué (DDoS) posent une menace constante aux infrastructures en ligne, mettant en péril la disponibilité des services et la sécurité des données. Face à ces défis, la mise en place d'une détection précoce devient essentielle pour contrer ces attaques de manière proactive.

L'utilisation du Deep Learning offre une approche innovante pour renforcer la détection précoce des attaques DDoS. Les réseaux neuronaux profonds, une branche de l'intelligence artificielle, sont capables d'apprendre des modèles complexes à partir de grandes quantités de données. Dans le contexte des attaques DDoS, cette capacité permet d'analyser le trafic réseau et d'identifier rapidement des schémas suspects.

IV.2 Mise en Place du Détection Précoce avec Deep Learning

IV.2.1 Conception

Le schéma global du fonctionnent de notre système d'alerte précoce par les diagrammes cas d'utilisation suivant : Avec ma solution de détection précoce utilisant l'apprentissage automatique, je suis en mesure d'anticiper et d'identifier rapidement les éléments suspects. En analysant en temps réel des modèles et des tendances dans les données, ma technologie de Machine Learning est votre alliée pour détecter précocement les situations potentiellement problématiques. Cette approche proactive permet d'agir rapidement et de prendre des mesures avant que les problèmes ne s'aggravent, renforçant ainsi la sécurité et l'efficacité de votre système. Schéma de notre solution

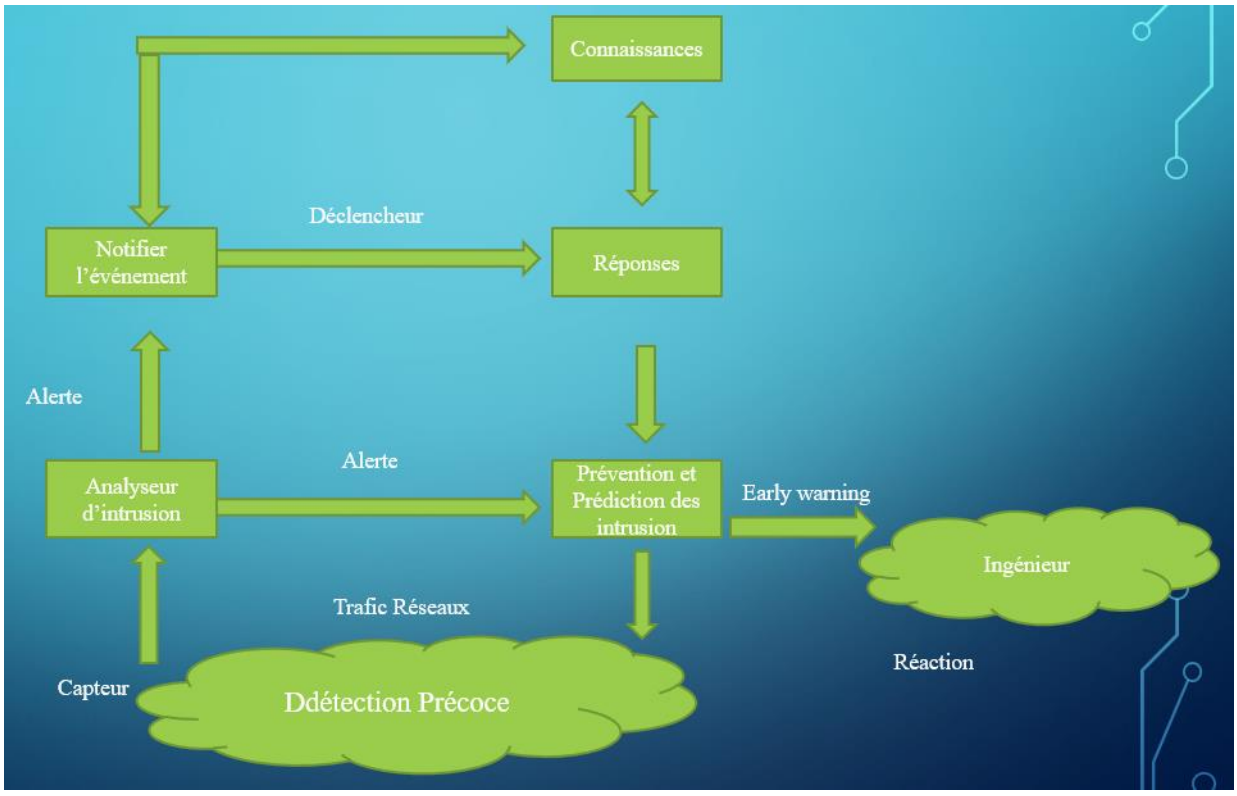


Figure IV. 1: Solution

Dans le domaine de l'apprentissage automatique, la réussite d'un projet repose sur plusieurs étapes cruciales. Tout commence par la récupération des datasets, la collecte de données qui serviront de base à notre modèle. Ensuite, lors de la phase de pré-processing, nous nettoyons et préparons les données pour les rendre compatibles avec notre modèle.

Vient ensuite la phase d'apprentissage, où notre modèle apprend à partir des données, en identifiant des schémas et des relations. La classification, une étape importante, permet au modèle de catégoriser les informations en fonction des caractéristiques définies.

La phase de validation intervient pour évaluer la performance du modèle. C'est là que nous calculons des métriques pour mesurer son efficacité et sa précision. Cette étape critique nous aide à ajuster notre modèle, le rendant plus robuste et fiable.

En fin de compte, l'application des cinq étapes du machine learning garantit un processus systématique, de la récupération des datasets à la classification, en passant par la phase d'apprentissage, de validation, et de pré-processing. Chaque étape est cruciale pour le succès global de notre projet d'apprentissage automatique.

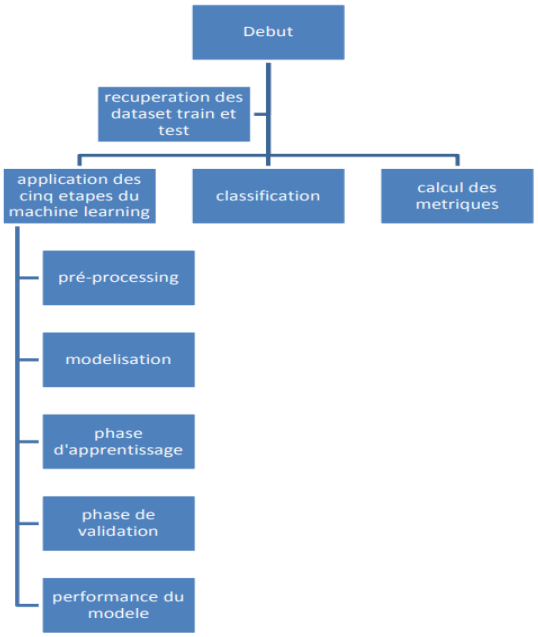


Figure IV. 2 : Phase de l'Algorithme de Classification

IV.2.2 Outils de réalisation

Dans cette partie nous allons présenter les principaux outils que nous avons utilisés pour la mise en place de notre projet

IV.2.2.1 Choix du Langage de Programmation

Les modèles que nous avons conçus ont été mis en œuvre dans l'environnement Python 3.10.1 (développé par la Python Software Foundation depuis 2001). Leurs principales caractéristiques sont résumées dans les points suivants :

- Le langage python est portable : l'alerte ne doit dépendre de l'architecture matérielle ni du système d'exploitation

- Est un langage de programmation interprété
- Peu créer des interfaces graphiques
- Peut faire circuler des informations à travers un réseau
- Est un langage utilisé pour l'analyse de données et en Machine Learning(apprentissage du machine Learning)

Nous avons utilisé scikit-learn qui est une bibliothèque d'apprentissage statique en python, c'est le moteur de beaucoup d'application de l'intelligence artificielle et de la science des données, elle est construite a base de :

- Numpy(version utilisé 1.21.4) : extension de python, elle permet de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des matrices et polynomes.
- Matplotlib(version utilisé 3.5.0) : est une biblothèque de tracage de graphique python 2D capable de générer des histogrammes, des spectres de puissance, des graphiques a barre, diffèrents diagrammes avec seulement quelques ligne de code

IV.2.3 Méthodologies de recherche

Les Etapes suivies /outils utilisés dans le cadre de la méthodologie de recherche sont les suivantes :

- Un ordinateur Dell lattitude 7490 avec un Ram de 16Go et disque dur 500Go
- Python 3.10.1 est utilisée comme langage de programmation
- le jeu de données Kaggle est utilisée
- Scikit-learn est utilisé pour la simulation (Pandas : , Numpy : 1.21.4 , Matplotlib : 3.5.0 , Sklearn : 0.24.2 ,)
- svm, naive bayes, random forest, xgboost, Email.mime.multipart, sont utilisée comme classifier dans notre projet **et** classent les instances en attaque ou en normal.

IV.2.4 Présentation et Exploration du Dataset

Il s'agit d'un jeu de données spécifique au SDN généré à l'aide d'un émulateur Mininet, utilisé pour la classification du trafic à l'aide d'algorithmes d'apprentissage automatique et d'apprentissage profond. Le projet démarre en créant dix topologies dans Mininet, où les commutateurs sont connectés à un seul contrôleur Ryu. La simulation du réseau inclut du trafic **TCP, UDP** et **ICMP** normal, ainsi que du trafic malveillant comprenant des attaques **TCP Syn,** des attaques **UDP Flood** et des attaques **ICMP**.

Au total, l'ensemble de données comprend 23 fonctionnalités, certaines extraites des commutateurs et d'autres calculées. Les fonctionnalités extraites comprennent des éléments tels que l'identifiant du commutateur, le nombre de paquets, le nombre d'octets, la durée en secondes, **la durée en nanosecondes (la durée totale étant la somme des deux), les adresses IP source et de destination, le numéro de port**, les octets transmis et reçus sur le port du commutateur. Le champ "**dt**" affiche la date et l'heure converties en nombre, et un flux est surveillé à des intervalles de 30 secondes.

Les fonctionnalités calculées comprennent le nombre de paquets par flux, le nombre d'octets par flux, le débit de paquets (calculé en divisant le nombre de paquets par flux par l'intervalle de surveillance), le nombre de messages "**Packet_ins**", les entrées de flux totales dans le commutateur, les taux de transfert **(tx_kbps, rx_kbps),** et la bande passante du port (somme de **tx_kbps et rx_kbps).**

La dernière colonne indique l'étiquette de classe, distinguant si le type de trafic est bénin (étiquette 0) ou malveillant (étiquette 1). La simulation du réseau s'étend sur 250 minutes,

collectant un total de 104 345 lignes de données. La simulation peut être répétée à des intervalles définis pour recueillir davantage de données.

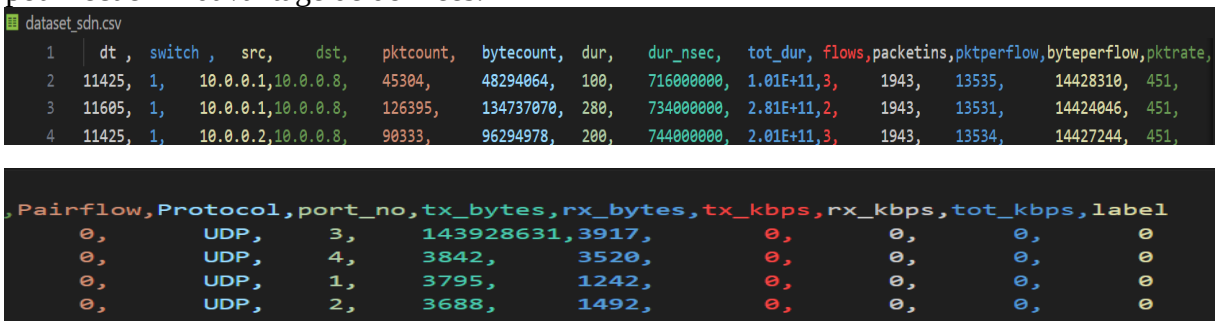


Figure IV. 3 : Ensemble de données du Dataset

IV.2.5 Importation des modules d'apprentissage

Cette partie du code importe plusieurs bibliothèques et modules Python qui sont nécessaires pour effectuer différentes tâches dans l'ensemble du script. Voici une explication brève de chaque importation :

- **Pandas (pd)** : Bibliothèque pour la manipulation et l'analyse de données tabulaires.
- **Mumpy (np)** : Bibliothèque pour la manipulation de tableaux et matrices multidimensionnels, ainsi que des fonctions mathématiques.
- **Matplotlib.pyplot (plt, figure)** : Bibliothèque pour la création de graphiques et de visualisations.
- **Seaborn (sns)** : Bibliothèque de visualisation basée sur Matplotlib, permettant de créer des graphiques statistiques attractifs.
- **Sklearn.metrics** : Fournit des métriques pour évaluer les performances des modèles d'apprentissage automatique.
- **train_test_split** : Fonction pour diviser les données en ensembles d'entraînement et de test.
- **cross_val_score, cross_val_predict** : Fonctions pour la validation croisée des modèles.
- **GridSearchCV** : Utilisé pour effectuer une recherche sur la grille des hyperparamètres pour trouver les meilleurs paramètres d'un modèle.
- **time** : Module pour mesurer le temps d'exécution du code.
- **DecisionTreeClassifier, LogisticRegression, svm, KNeighborsClassifier, RandomForestClassifier** : Implémentations de différents modèles d'apprentissage automatique.
- **smtplib, MIMEMultipart, MIMEText** : Utilisés pour envoyer des e-mails via un serveur SMTP. Cela pourrait être utilisé pour notifier quelqu'un lorsque le script est terminé.

Ces imports sont essentiels pour le reste du code, qui semble être un script d'analyse de données et d'entraînement de modèles d'apprentissage automatique, probablement dans le domaine de la détection d'anomalies réseau.

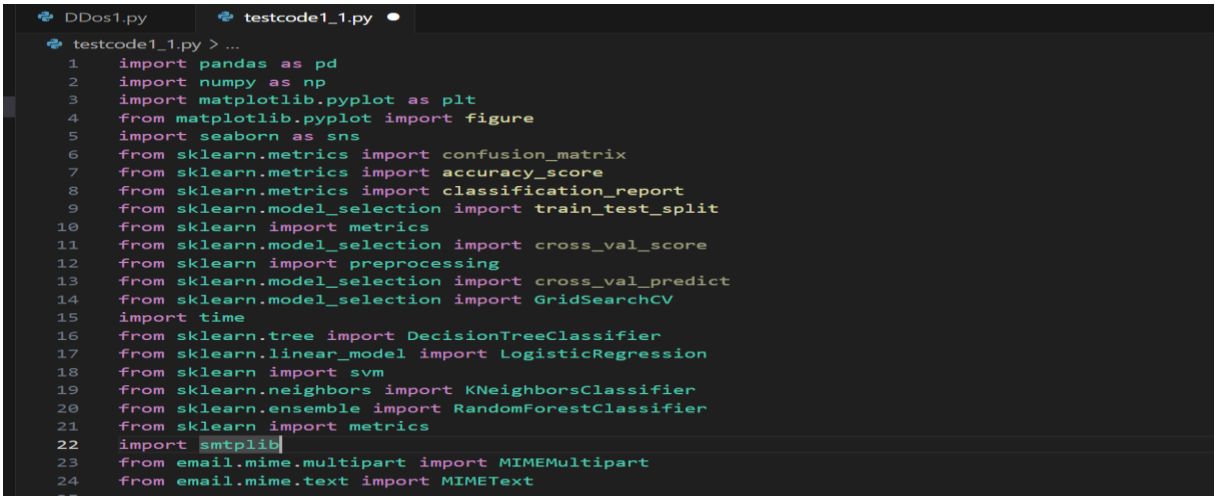


Figure IV. 4 : Module utilisé

IV.2.6 Chargement des Données du dataset

Lecture des données :

Les données sont lues à partir du fichier CSV et stockées dans le DataFrame **data**.

Affichage des premières lignes :

Les cinq premières lignes du DataFrame sont affichées à l'aide de la méthode **head()**. Cela permet de donner un aperçu initial des données.

Affichage de la forme des données :

La forme du DataFrame (nombre de lignes et de colonnes) est affichée à l'aide de la méthode **shape**. Cela donne des informations sur la dimensionnalité des données.

Affichage des informations sur les données :

La méthode **info()** est utilisée pour afficher des informations détaillées sur le DataFrame, y compris les types de données, le nombre de valeurs non nulles et l'utilisation de la mémoire.

Affichage des valeurs uniques de la colonne 'label' :

La méthode **unique()** est appliquée à la colonne 'label' pour afficher les valeurs uniques présentes dans cette colonne. Cela peut être utile pour comprendre les catégories ou les classes présentes.

Affichage du décompte des valeurs dans la colonne 'label' :

La méthode **value_counts()** est utilisée pour compter le nombre d'occurrences de chaque valeur unique dans la colonne 'label'. Cela fournit une distribution des classes ou catégories dans le jeu de données.

En résumé, cette partie du code permet d'explorer rapidement les données en affichant les premières lignes, la forme, les informations détaillées, les valeurs uniques et la distribution des classes dans la colonne 'label'. Cela offre une première compréhension du contenu du jeu de données.



Figure IV. 5: Lecture et Affichage des données

IV.2.7 Modèles D'apprentissage Et Métriques

Ce code définit une classe nommée "Model" qui semble être utilisée pour construire des modèles d'apprentissage automatique. Voici une explication et un commentaire sur chaque partie du code :

class Model : Définit une classe appelée "**Model**".

global y : Déclare la variable y comme globale. Cela suggère que y est utilisé dans d'autres parties du code ou qu'il est défini en dehors de cette classe.

def __init__(self, data) : Définit la méthode spéciale _ _init_ _, qui est le constructeur de la classe. Cette méthode est appelée lorsqu'une instance de la classe est créée. Elle prend comme paramètre **data**.

self.data = data : Attribue le paramètre **data** à l'attribut **self.data** de l'instance.

X = preprocessing.StandardScaler().fit(self.data).transform(self.data) : Standardise les données en utilisant le **StandardScaler** de **scikit-learn.** Cela transforme les données pour qu'elles aient une moyenne nulle et une variance unitaire.

self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(X, y, random_state=42, test_size=0.3) : Divise les données standardisées en ensembles d'entraînement **(X_train, y_train)** et de test **(X_test, y_test).** Les données sont divisées de manière aléatoire avec une proportion de 70% pour l'ensemble d'entraînement et 30% pour l'ensemble de test. Le paramètre **random_state=42** assure la reproductibilité des résultats.

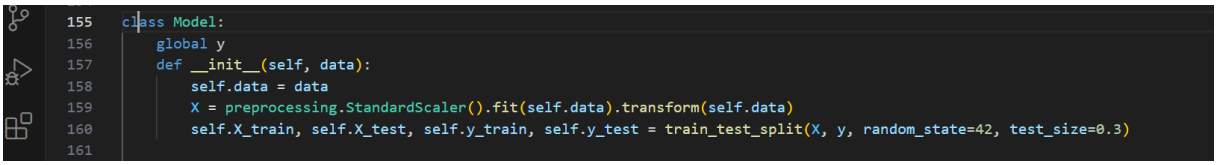


Figure IV. 6: Définition de notre Modèle

IV.2.7.1 LogisticRegression

Cette partie du code implémente un processus d'ajustement d'un modèle de régression logistique avec différentes méthodes de résolution (solveurs) pour trouver celle qui donne la meilleure performance.

solvers = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'] : Définit une liste de solveurs que la régression logistique va utiliser pour résoudre le problème d'optimisation associé à l'ajustement du modèle. Chaque solveur utilise une méthode différente pour trouver les coefficients optimaux.

start_time = time.time() : Enregistre le temps de début de l'exécution du code pour mesurer le temps nécessaire pour ajuster le modèle.
results_lr = [] : Initialise une liste vide pour stocker les résultats de chaque tentative d'ajustement du modèle avec un solveur spécifique.
accuracy_list = [] : Initialise une liste vide pour stocker les précisions (accuracy) associées à chaque solveur.
for solver in solvers: : Débute une boucle qui itère sur chaque solveur.

- **LR = LogisticRegression(C=0.03, solver=solver).fit(self.X_train, self.y_train)** : Ajuste un modèle de régression logistique avec le solveur actuel, une régularisation (C=0.03), sur les données d'entraînement.
- **predicted_lr = LR.predict(self.X_test) :** Prédit les valeurs sur l'ensemble de test à l'aide du modèle ajusté.
- **accuracy_lr = accuracy_score(self.y_test, predicted_lr)** : Calcule la précision du modèle sur l'ensemble de test.
- **results_lr.append(...) :** Ajoute les résultats de l'ajustement actuel à la liste des résultats.
- **accuracy_list.append(accuracy_lr) :** Ajoute la précision associée au solveur actuel à la liste des précisions.

solver_name = solvers[accuracy_list.index(max(accuracy_list))] : Identifie le solveur qui a donné la meilleure précision en consultant la liste des précisions.
LR = LogisticRegression(C=0.03, solver=solver_name).fit(self.X_train,self.y_train) : Ajuste à nouveau le modèle de régression logistique avec le meilleur solveur sur les données d'entraînement.
predicted_lr = LR.predict(self.X_test) : Prédit à nouveau les valeurs sur l'ensemble de test avec le meilleur modèle ajusté.
accuracy_lr = accuracy_score(self.y_test, predicted_lr) : Calcule la précision finale du modèle sur l'ensemble de test.

```
162     def LogisticRegression(self):
163         solvers = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
164 
165         start_time = time.time()
166         results_lr = []
167         accuracy_list = []
168         for solver in solvers:
169             LR = LogisticRegression(C=0.03, solver=solver).fit(self.X_train, self.y_train)
170             predicted_lr = LR.predict(self.X_test)
171             accuracy_lr = accuracy_score(self.y_test, predicted_lr)
172             results_lr.append({'solver' : solver, 'accuracy': str(round(accuracy_lr * 100, 2)) + "%",
173                                'Coefficients': {'W' : LR.coef_, 'b': LR.intercept_}})
174 
175             accuracy_list.append(accuracy_lr)
176 
177         solver_name = solvers[accuracy_list.index(max(accuracy_list))]
178         LR = LogisticRegression(C=0.03, solver=solver_name).fit(self.X_train,self.y_train)
179         predicted_lr = LR.predict(self.X_test)
180         accuracy_lr = accuracy_score(self.y_test, predicted_lr)
```

Figure IV. 7: LogisticRégression

IV.2.7.2 Machine à Vecteur de Support (SVM)

Ce code implémente un processus d'ajustement d'un modèle de **Support Vector Machine (SVM)** avec différents noyaux (kernels) et mesure la précision du modèle pour chaque noyau.

- **start_time = time.time() :** Enregistre le temps de début de l'exécution du code pour mesurer le temps nécessaire pour ajuster le modèle SVM.
- **accuracy_list = [] :** Initialise une liste vide pour stocker les précisions associées à chaque noyau SVM.
- **result_svm = [] :** Initialise une liste vide pour stocker les résultats (noyau et précision) de chaque tentative d'ajustement du modèle SVM.

- **kernels = ['linear', 'poly','rbf', 'sigmoid'] :** Définit une liste de noyaux SVM à essayer, notamment linéaire, polynomial, radial (rbf), et sigmoïdal.
- **for kernel in kernels :** Débute une boucle qui itère sur chaque noyau SVM.

SVM = svm.SVC(kernel=kernel).fit(self.X_train, self.y_train) : Ajuste un modèle SVM avec le noyau actuel sur les données d'entraînement.
predicted_svm = SVM.predict(self.X_test) : Prédit les valeurs sur l'ensemble de test à l'aide du modèle SVM ajusté.
accuracy_svm = accuracy_score(self.y_test, predicted_svm) : Calcule la précision du modèle SVM sur l'ensemble de test.
result_svm.append({"kernel" : kernel, "accuracy": f"{round(accuracy_svm*100,2)}%"}) : Ajoute les résultats (noyau et précision) à la liste des résultats.
print("Accuracy: %.2f%%" % round((accuracy_svm * 100.0),2)) : Affiche la précision associée au noyau actuel.
accuracy_list.append(accuracy_svm) : Ajoute la précision associée au noyau actuel à la liste des précisions.
kernel_name = kernels[accuracy_list.index(max(accuracy_list))] : Identifie le noyau qui a donné la meilleure précision en consultant la liste des précisions.
SVM = svm.SVC(kernel=kernel_name).fit(self.X_train, self.y_train) : Ajuste à nouveau le modèle SVM avec le meilleur noyau sur les données d'entraînement.
predicted_svm = SVM.predict(self.X_test) : Prédit à nouveau les valeurs sur l'ensemble de test avec le meilleur modèle SVM ajusté.
accuracy_svm = accuracy_score(self.y_test, predicted_svm) : Calcule la précision finale du modèle SVM sur l'ensemble de test.
print(f"Accuracy of SVM model {round(accuracy_svm,2)*100}%") : Affiche la précision finale du modèle SVM.
print('best kernel is : ', kernel_name) : Affiche le meilleur noyau trouvé.
print(classification_report(predicted_svm, self.y_test)) : Affiche le rapport de classification du modèle SVM.
print("--- %s seconds ---" % (time.time() - start_time)) : Affiche le temps total nécessaire pour ajuster le modèle SVM.
Ce code teste différents noyaux SVM, enregistre la précision associée à chacun, sélectionne le meilleur noyau, ajuste le modèle SVM avec ce noyau, et affiche la précision finale ainsi que d'autres informations utiles.

```
189     def SupportVectorMachine(self):
190         start_time = time.time()
191         accuracy_list = []
192         result_svm = []
193         kernels = ['linear', 'poly','rbf', 'sigmoid']
194         #kernels = ['rbf']
195         for kernel in kernels:
196             SVM = svm.SVC(kernel=kernel).fit(self.X_train, self.y_train)
197             predicted_svm = SVM.predict(self.X_test)
198             accuracy_svm = accuracy_score(self.y_test, predicted_svm)
199             result_svm.append({"kernel" : kernel, "accuracy": f"{round(accuracy_svm*100,2)}%"})
200             print("Accuracy: %.2f%%" % round((accuracy_svm * 100.0),2))
201             print('######################################################################')
202             accuracy_list.append(accuracy_svm)
203         kernel_name = kernels[accuracy_list.index(max(accuracy_list))]
204         SVM = svm.SVC(kernel=kernel_name).fit(self.X_train, self.y_train)
205         predicted_svm = SVM.predict(self.X_test)
206         accuracy_svm = accuracy_score(self.y_test, predicted_svm)
```

Figure IV. 8: Code Support Vecteur Machine

IV.2.7.3 K plus Proche Voisin

Ce code implémente un processus d'ajustement d'un modèle de classification des k-plus proches voisins (KNN) avec plusieurs voisins (K) en utilisant différentes valeurs pour K.
start_time = time.time() : Enregistre le temps de début de l'exécution du code pour mesurer le temps nécessaire pour ajuster le modèle KNN.

Ks = 12 : Définit le nombre maximum de voisins (K) à tester, ici 12.
accuracy_knn = np.zeros((Ks-1)) : Initialise un tableau de zéros pour stocker les précisions associées à chaque valeur de K. Le "-1" est utilisé car la numérotation des indices commence à zéro.
std_acc = np.zeros((Ks-1)) : Initialise un tableau de zéros pour stocker les écarts-types associés à chaque valeur de K.
for n in range(1,Ks): : Débute une boucle qui itère sur chaque valeur de K de 1 à 11.
a. **neigh = KNeighborsClassifier(n_neighbors = n).fit(self.X_train,self.y_train) :** Ajuste un modèle KNN avec le nombre de voisins (K) actuel sur les données d'entraînement.
b. **yhat = neigh.predict(self.X_test) :** Prédit les valeurs sur l'ensemble de test à l'aide du modèle KNN ajusté.
c. **accuracy_knn[n-1] = metrics.accuracy_score(self.y_test, yhat) :** Stocke la précision associée à la valeur de K actuelle dans le tableau des précisions.
d. **std_acc[n-1] = np.std(yhat==self.y_test)/np.sqrt(yhat.shape[0]) :** Calcule l'écart-type associé à la valeur de K actuelle et le stocke dans le tableau des écarts-types.
plt.figure(figsize=(10,6)) : Initialise une figure pour afficher les résultats.
plt.plot(range(1,Ks),accuracy_knn,'g') : Trace la précision en fonction du nombre de voisins (K).
plt.fill_between(range(1,Ks),accuracy_knn - 1 * std_acc,accuracy_knn + 1 * std_acc, alpha=0.10) : Remplit la zone entre la précision moins un écart-type et la précision plus un écart-type.
plt.fill_between(range(1,Ks),accuracy_knn - 3 * std_acc,accuracy_knn + 3 * std_acc, alpha=0.10,color="green") : Remplit la zone entre la précision moins trois écart-types et la précision plus trois écart-types en vert.
plt.legend(('Accuracy ', '+/- 1xstd','+/- 3xstd')) : Ajoute une légende à la figure.
plt.ylabel('Accuracy ') : Étiquette l'axe des ordonnées de la figure.
plt.xlabel('Number of Neighbors (K)') : Étiquette l'axe des abscisses de la figure.
plt.tight_layout() : Ajuste automatiquement la disposition de la figure pour éviter le chevauchement.
plt.show() : Affiche la figure.
knnc = KNeighborsClassifier() : Initialise un modèle KNN.
knnc_search = GridSearchCV(...) : Utilise la validation croisée avec grille pour rechercher les meilleurs hyperparamètres **(n_neighbors, weights, metric)** pour le modèle KNN sur les données d'entraînement.
knnc_search.fit(self.X_train, self.y_train) : Ajuste le modèle KNN avec les meilleurs hyperparamètres identifiés.
n_neighbors = knnc_search.best_params_['n_neighbors'] : Récupère le nombre optimal de voisins identifié par la recherche sur la grille.
weights = knnc_search.best_params_['weights'] : Récupère la méthode de pondération optimale identifiée par la recherche sur la grille.
metric = knnc_search.best_params_['metric'] : Récupère la mesure de distance optimale identifiée par la recherche sur la grille.
KNN = KNeighborsClassifier(n_neighbors=n_neighbors, metric=metric, weights=weights).fit(self.X_train,self.y_train) : Ajuste le modèle KNN avec les meilleurs hyperparamètres sur les données d'entraînement.
predicted_knn = KNN.predict(self.X_test) : Prédit les valeurs sur l'ensemble de test avec le modèle KNN ajusté.
accuracy_knn = metrics.accuracy_score(self.y_test, predicted_knn) : Calcule la précision finale du modèle KNN sur l'ensemble de test.
print(f"Accuracy of KNN model {round(accuracy_knn,2)*100}%") : Affiche la précision finale du modèle KNN.

```
214
215     def KNearetsNeighbor(self):
216         start_time = time.time()
217         Ks = 12
218         accuracy_knn = np.zeros((Ks-1))
219         std_acc = np.zeros((Ks-1))
220         #print(accuracy_knn)
221         for n in range(1,Ks):
222 
223             #Train Model and Predict
224             neigh = KNeighborsClassifier(n_neighbors = n).fit(self.X_train,self.y_train)
225             yhat=neigh.predict(self.X_test)
226             accuracy_knn[n-1] = metrics.accuracy_score(self.y_test, yhat)
227             std_acc[n-1]=np.std(yhat==self.y_test)/np.sqrt(yhat.shape[0])
228 
229         #print(accuracy_knn,'\n\n') # courseranyn ozinde tek osy gana jazylyp turdy
230         #print(std_acc)
231         #accuracy_knn[0] = 0
232         plt.figure(figsize=(10,6))
233         plt.plot(range(1,Ks),accuracy_knn,'g')
234         plt.fill_between(range(1,Ks),accuracy_knn - 1 * std_acc,accuracy_knn + 1 * std_acc, alpha=0.10)
235         plt.fill_between(range(1,Ks),accuracy_knn - 3 * std_acc,accuracy_knn + 3 * std_acc, alpha=0.10,color="green")
236         plt.legend(('Accuracy ', '+/- 1xstd','+/- 3xstd'))
237         plt.ylabel('Accuracy ')
```

Figure IV. 9: Code Métrique Pour KNN

IV.2.7.4 Arbre de Décision

Ce code met en œuvre le processus d'ajustement d'un modèle d'arbre de décision en utilisant la validation croisée avec grille pour rechercher les meilleurs hyperparamètres **(criterion, max_depth, max_leaf_nodes).** Voici une explication claire et précise du code :

- **start_time = time.time() :** Enregistre le temps de début de l'exécution du code pour mesurer le temps nécessaire pour ajuster le modèle d'arbre de décision.
- **tree = DecisionTreeClassifier() :** Initialise un modèle d'arbre de décision.
- **dt_search = GridSearchCV(...) :** Utilise la validation croisée avec grille pour rechercher les meilleurs hyperparamètres **(criterion, max_depth, max_leaf_nodes)** pour le modèle d'arbre de décision sur les données d'entraînement.
- **param_grid={'criterion' : ['gini', 'entropy'], 'max_depth' : [2,3,4,5,6,7,8, 9, 10], 'max_leaf_nodes' : [2,3,4,5,6,7,8,9,10, 11]}** : Définit les valeurs à tester pour chaque hyperparamètre.
- **n_jobs=-1, cv=5, scoring='accuracy', verbose=2 :** Utilise tous les cœurs disponibles, utilise une validation croisée à 5 plis, évalue les performances en termes de précision, et affiche des informations détaillées pendant l'exécution.
- **dt_search.fit(self.X_train, self.y_train) :** Ajuste le modèle d'arbre de décision avec les meilleurs hyperparamètres identifiés par la recherche sur la grille.
- **criterion = dt_search.best_params_['criterion'] :** Récupère le critère optimal identifié par la recherche sur la grille.
- **max_depth = dt_search.best_params_['max_depth'] :** Récupère la profondeur maximale optimale identifiée par la recherche sur la grille.
- **max_leaf_nodes = dt_search.best_params_['max_leaf_nodes'] :** Récupère le nombre maximal de feuilles optimal identifié par la recherche sur la grille.
- **dtree = DecisionTreeClassifier(...)** : Initialise un nouveau modèle d'arbre de décision avec les meilleurs hyperparamètres identifiés.
- **dtree.fit(self.X_train, self.y_train) :** Ajuste le modèle d'arbre de décision avec les meilleurs hyperparamètres sur les données d'entraînement.

predicted_dt = dtree.predict(self.X_test) : Prédit les valeurs sur l'ensemble de test avec le modèle d'arbre de décision ajusté.

accuracy_dt = metrics.accuracy_score(self.y_test, predicted_dt) : Calcule la précision finale du modèle d'arbre de décision sur l'ensemble de test.

- **print(f"criterion: {criterion}, max depth: {max_depth}, max_leaf: {max_leaf_nodes}") :** Affiche les meilleurs hyperparamètres identifiés par la recherche sur la grille.

print(f"The Accuracy is : {round(accuracy_dt * 100,2)}%") : Affiche la précision finale du modèle d'arbre de décision sur l'ensemble de test.

```
264     def DecisionTree(self):
265         start_time = time.time()
266         tree = DecisionTreeClassifier()
267         dt_search = GridSearchCV(tree, param_grid={'criterion' : ['gini', 'entropy'],
268                                                    'max_depth' : [2,3,4,5,6,7,8, 9, 10],
269                                                    'max_leaf_nodes' : [2,3,4,5,6,7,8,9,10, 11]},
270                                  n_jobs=-1, cv=5, scoring='accuracy', verbose=2)
271
272         dt_search.fit(self.X_train, self.y_train)
273
274         criterion = dt_search.best_params_['criterion']
275         max_depth = dt_search.best_params_['max_depth']
276         max_leaf_nodes = dt_search.best_params_['max_leaf_nodes']
277         dtree = DecisionTreeClassifier(criterion=criterion,
278                                        max_depth=max_depth,
279                                        max_leaf_nodes=max_leaf_nodes).fit(self.X_train, self.y_train)
280         predicted_dt = dtree.predict(self.X_test)
281         accuracy_dt = metrics.accuracy_score(self.y_test, predicted_dt)
```

Figure IV. 10: Arbre de Décision

IV.2.7.5 RandomForest (Foret Aléatoire)

Cette fonction **RandomForest** implémente un modèle d'ensemble appelé **Random Forest**, qui est une méthode d'apprentissage supervisé utilisée pour la classification et la régression.

- ✓ **Initialisation** :

RandomForestClassifier est instancié avec plusieurs paramètres qui définissent le comportement du modèle.

criterion='gini' : indique que l'indice de Gini est utilisé comme mesure de qualité pour diviser les nœuds de l'arbre.

n_estimators=500 spécifie le nombre d'arbres dans la forêt.

min_samples_split=10 détermine le nombre minimal d'échantillons requis pour diviser un nœud interne.

max_features='auto' indique que le nombre de caractéristiques à considérer lors de la recherche de la meilleure division est égal à la racine carrée du nombre total de caractéristiques.

oob_score=True active le calcul du score "out-of-bag".

random_state=1 fixe la graine aléatoire pour assurer la reproductibilité.

n_jobs=-1 permet d'utiliser tous les cœurs disponibles pour l'ajustement du modèle.

Enfin, le modèle est ajusté aux données d'entraînement (self.X_train et self.y_train).

- ✓ **Prédiction et Évaluation**

Les données de test **(self.X_test)** sont utilisées pour faire des prédictions avec le modèle entraîné.

La précision des prédictions est calculée à l'aide de la fonction **accuracy_score**.

- ✓ **Affichage des Résultats**:

La précision du modèle **Random Forest** est affichée.

Un rapport de classification détaillé est imprimé, montrant des mesures telles que **la précision**, **le rappel** et **la f1-score** pour chaque classe.

Le temps écoulé pour entraîner le modèle est également affiché.

Cette fonction entraîne un modèle **Random Forest**, l'utilise pour prédire les classes des données de test, évalue la précision et affiche diverses métriques de performance du modèle.

```python
def RandomForest(self):
    start_time = time.time()
    RF = RandomForestClassifier(criterion='gini',
                                n_estimators=500,
                                min_samples_split=10,
                                #min_samples_leaf=1,
                                max_features='auto',
                                oob_score=True,
                                random_state=1,
                                n_jobs=-1).fit(self.X_train, self.y_train)

    predicted_rf = RF.predict(self.X_test)
    svm_accuracy = accuracy_score(self.y_test, predicted_rf)
    print(f"Accuracy of RF is : {round(svm_accuracy*100,2)}%", '\n')
    print("########################################################################")
    print(classification_report(predicted_rf, self.y_test))
    print("########################################################################")
    print("--- %s seconds ---" % (time.time() - start_time))
```

Figure IV. 11: Code de RandomForest

IV.2.7.6 SERVEUR GMAIL SMTP

gmail_cfg agit comme un ensemble de paramètres de configuration pour l'envoi d'e-mails via Gmail. Il contient des informations essentielles telles que l'adresse du serveur, le port, le nom d'utilisateur et le mot de passe, permettant ainsi au programme d'établir une connexion sécurisée avec le serveur SMTP Gmail pour l'envoi d'e-mails.

```python
gmail_cfg = {
    "server": "smtp.gmail.com",
    "port": 465,  # Utiliser le port 465 pour SMTP_SSL
    "username": "testpythonmemo@gmail.com",
    "password": "vtfmtfgjvboqouxo"
}
```

Figure IV. 12: Fonction d'envoi

Cette fonction encapsule le processus d'envoi d'e-mails via **SMTP** en utilisant une connexion sécurisée. Elle prend en charge la configuration du serveur, la création et l'envoi du message, ainsi que l'affichage d'un message de confirmation. L'exemple d'utilisation démontre comment utiliser cette fonction pour envoyer un e-mail spécifique

```python
def mail_alert(destinataire, sujet, corps):
    with smtplib.SMTP_SSL(gmail_cfg["server"], gmail_cfg["port"]) as smtp:
        smtp.login(gmail_cfg["username"], gmail_cfg["password"])

        message = MIMEMultipart()
        message['from'] = gmail_cfg["username"]
        message['to'] = "bba3579@gmail.com"
        message['subject'] = sujet
        message.attach(MIMEText(corps, 'plain'))
```

Figure IV. 13: Envoi Avec SMTP_SSL

IV.2.8 Fonction de Détection précoce

Le code que vous avez partagé semble être une implémentation en Python d'une méthode de détection précoce des attaques DDoS (Distributed Denial of Service) utilisant la régression logistique.

Fonction detect_early_ddos(self, threshold=0.1):

Cette fonction prend en paramètre un seuil **(threshold)** qui est par défaut fixé à 0.1. Elle effectue les étapes suivantes :

Calcul de la variation du nombre de paquets (pktcount_variation**) :**Utilisation de la bibliothèque NumPy **(np)** pour calculer la différence absolue entre les valeurs consécutives du nombre de paquets (**pktcount)** dans l'ensemble de test **(self.X_test).**

Détection précoce :

- Création d'un tableau de booléens indiquant si la variation du nombre de paquets dépasse le seuil spécifié.
- La conversion du tableau de booléens en entiers **(0 pour False et 1 pour True).**

Renvoi du résultat :
La fonction renvoie le tableau résultant, indiquant les périodes où une attaque DDoS est détectée.

Fonction LogisticRegression_with_early_detection_ddos(self, threshold=0.5, ddos_threshold=0.1):
Cette fonction utilise la régression logistique pour la classification, tout en intégrant une détection précoce des attaques DDoS. Voici son fonctionnement détaillé :

Liste des solveurs : Une liste de solveurs pour la régression logistique est définie (ex. 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga').

Boucle sur les solveurs :

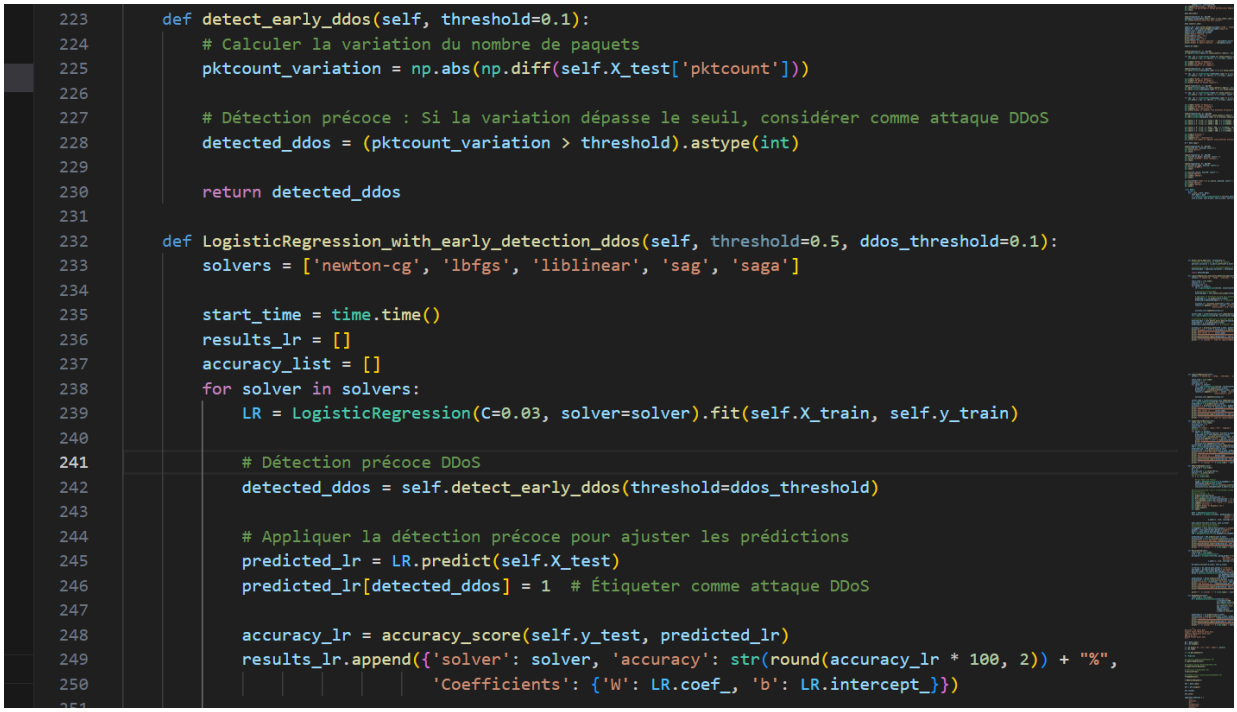
- Une boucle est effectuée sur chaque solveur, où un modèle de régression logistique est entraîné avec le solveur actuel.
- La détection précoce DDoS est effectuée en appelant la fonction detect_early_ddos avec un seuil spécifié (ddos_threshold).
- Les prédictions du modèle de régression logistique sont ajustées en marquant comme attaque DDoS les périodes détectées précocement.
- La précision du modèle est calculée et enregistrée dans une liste (accuracy_list).
- Les résultats, y compris le solveur utilisé, la précision, et les coefficients du modèle, sont stockés dans une liste (results_lr).

Sélection du meilleur solveur : Le solveur avec la plus haute précision est sélectionnée.
Entraînement du meilleur modèle : Un nouveau modèle de régression logistique est entraîné avec le meilleur solveur sélectionné.

Détection précoce DDoS pour le meilleur modèle :

- La détection précoce DDoS est effectuée à nouveau sur le meilleur modèle.
- Les prédictions du modèle sont ajustées en marquant comme attaque DDoS les périodes détectées précocement.

Affichage des résultats : La précision du meilleur modèle est affichée, ainsi que des informations supplémentaires telles que le solveur utilisé, le rapport de classification, et le temps d'exécution.
Cette implémentation combine la régression logistique pour la classification avec une stratégie de détection précoce des attaques DDoS basée sur la variation du nombre de paquets. Elle explore différents solveurs pour la régression logistique et sélectionne le modèle avec la meilleure performance.



```python
def detect_early_ddos(self, threshold=0.1):
    # Calculer la variation du nombre de paquets
    pktcount_variation = np.abs(np.diff(self.X_test['pktcount']))

    # Détection précoce : Si la variation dépasse le seuil, considérer comme attaque DDoS
    detected_ddos = (pktcount_variation > threshold).astype(int)

    return detected_ddos

def LogisticRegression_with_early_detection_ddos(self, threshold=0.5, ddos_threshold=0.1):
    solvers = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']

    start_time = time.time()
    results_lr = []
    accuracy_list = []
    for solver in solvers:
        LR = LogisticRegression(C=0.03, solver=solver).fit(self.X_train, self.y_train)

        # Détection précoce DDoS
        detected_ddos = self.detect_early_ddos(threshold=ddos_threshold)

        # Appliquer la détection précoce pour ajuster les prédictions
        predicted_lr = LR.predict(self.X_test)
        predicted_lr[detected_ddos] = 1  # Étiqueter comme attaque DDoS

        accuracy_lr = accuracy_score(self.y_test, predicted_lr)
        results_lr.append({'solver': solver, 'accuracy': str(round(accuracy_lr * 100, 2)) + "%",
                           'Coefficients': {'W': LR.coef_, 'b': LR.intercept_}})
```

Figure IV. 14: Détection Précoce

IV.3 Analyse des données Obtenus

L'analyse des résultats fournis pour le DataFrame montre plusieurs informations importantes sur le jeu de données. Voici une analyse détaillée :

Structure des Données : Le DataFrame comprend **104 345 entrées (lignes)** et **23 colonnes (variables**).

Il existe différents types de données dans les colonnes : **int64, float64,** et **object**.

Présence de Valeurs Null : Aucune valeur null n'est présente dans la plupart des colonnes, sauf dans les colonnes **rx_kbps** et **tot_kbps,** qui ont des valeurs **null** pour certaines entrées.

La présence de valeurs **null** dans ces deux colonnes peut nécessiter une attention particulière lors de l'analyse et du traitement des données.

Informations sur les Colonnes Numériques : Des colonnes telles que **pktcount, bytecount**, **dur**, **flows,** etc., représentent des données numériques, souvent associées au trafic réseau.

Des statistiques descriptives (moyenne, écart type, minimum, 25e percentile, médiane, 75e percentile, maximum) pour ces colonnes peuvent fournir des informations sur la distribution des données.

Informations sur les Colonnes de Texte (object) : Les colonnes **src**, **dst,** et **Protocol** sont de **type object,** indiquant probablement des données catégorielles ou des chaînes de caractères.

La colonne Protocol peut nécessiter un encodage pour être utilisée dans des modèles de machine learning, car de nombreux algorithmes exigent des données numériques.

Consommation de Mémoire : La mémoire utilisée par le DataFrame est d'environ 18,3 Mo, ce qui donne une idée de la taille du jeu de données. Cela peut être important en fonction des ressources disponibles.

Colonnes avec Valeurs Numériques et Flottantes :

Les colonnes **rx_kbps**, **tot_kbps**, et **dur_nsec** sont de type flottant **(float64)**, ce qui indique probablement des mesures continues ou des fractions.

Colonnes de Classification : La colonne label semble être la variable cible de classification, avec des valeurs binaires **(0 ou 1),** probablement indiquant des catégories bénignes ou malveillantes.
Corrélation Entre Colonnes : L'absence de valeurs nulles dans la plupart des colonnes peut indiquer une corrélation forte entre certaines d'entre elles. L'utilisation d'outils tels que les matrices de corrélation peut aider à identifier ces relations.
Colonnes Potentiellement Redondantes : Il est possible que certaines colonnes, comme dur et **tot_dur**, ou **pktperflow** et **pktrate,** puissent être redondantes. Une analyse plus approfondie peut être nécessaire pour déterminer si elles peuvent être combinées ou si l'une d'entre elles est plus pertinente pour la modélisation.
Ces résultats fournissent une base solide pour comprendre la structure du jeu de données, mais une analyse plus approfondie, y compris des statistiques descriptives et une exploration visuelle, pourrait être nécessaire pour obtenir une compréhension plus complète du contenu des données et de leur pertinence pour la modélisation pour la suite du projet.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104345 entries, 0 to 104344
Data columns (total 23 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   dt           104345 non-null  int64
 1   switch       104345 non-null  int64
 2   src          104345 non-null  object
 3   dst          104345 non-null  object
 4   pktcount     104345 non-null  int64
 5   bytecount    104345 non-null  int64
 6   dur          104345 non-null  int64
 7   dur_nsec     104345 non-null  int64
 8   tot_dur      104345 non-null  float64
 9   flows        104345 non-null  int64
 10  packetins    104345 non-null  int64
 11  pktperflow   104345 non-null  int64
 12  byteperflow  104345 non-null  int64
 13  pktrate      104345 non-null  int64
 14  Pairflow     104345 non-null  int64
 15  Protocol     104345 non-null  object
 16  port_no      104345 non-null  int64
 17  tx_bytes     104345 non-null  int64
 18  rx_bytes     104345 non-null  int64
 19  tx_kbps      104345 non-null  int64
 20  rx_kbps      103839 non-null  float64
 21  tot_kbps     103839 non-null  float64
 22  label        104345 non-null  int64
dtypes: float64(3), int64(17), object(3)
memory usage: 18.3+ MB
```

Figure IV. 15: Présentation du Dataset

Et le classement des données sont fait :

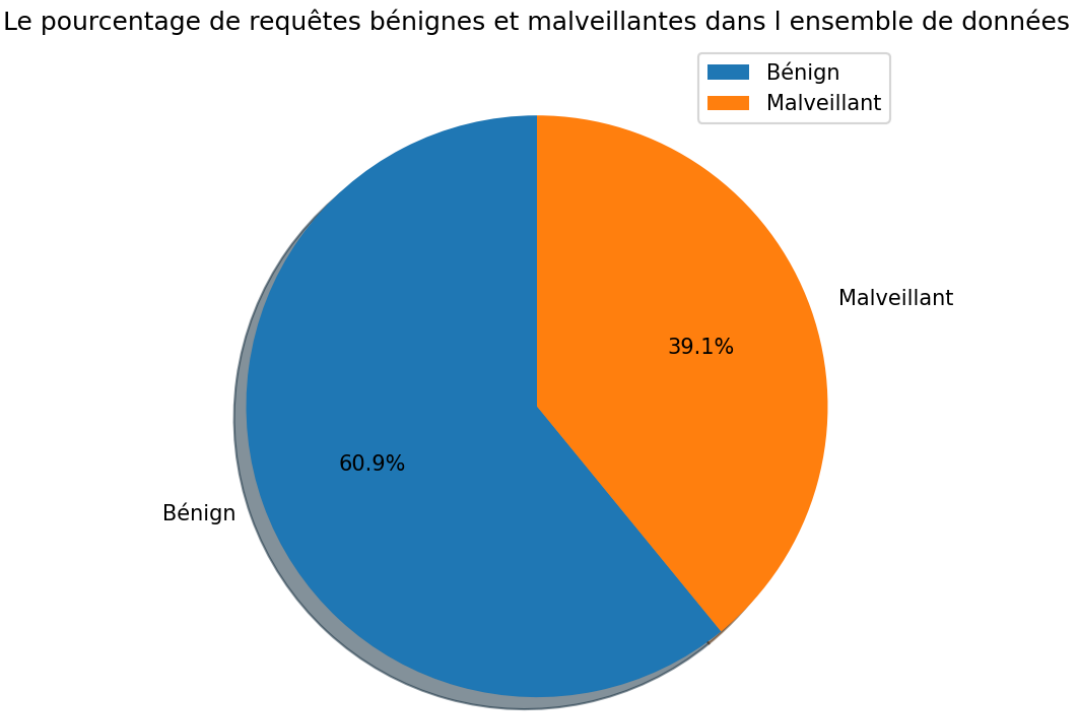


Figure IV. 16: Classification des requêtes bon et mauvais

***Caractéristiques des valeurs NULL :**

-**<u>Valeur manquante</u> :** Nous avons un diagramme à barres qui offre une vue visuelle du nombre de valeurs manquantes dans chaque colonne du DataFrame **data**. Cela peut être utile pour identifier rapidement les colonnes avec des données manquantes et orienter le processus de nettoyage des données.

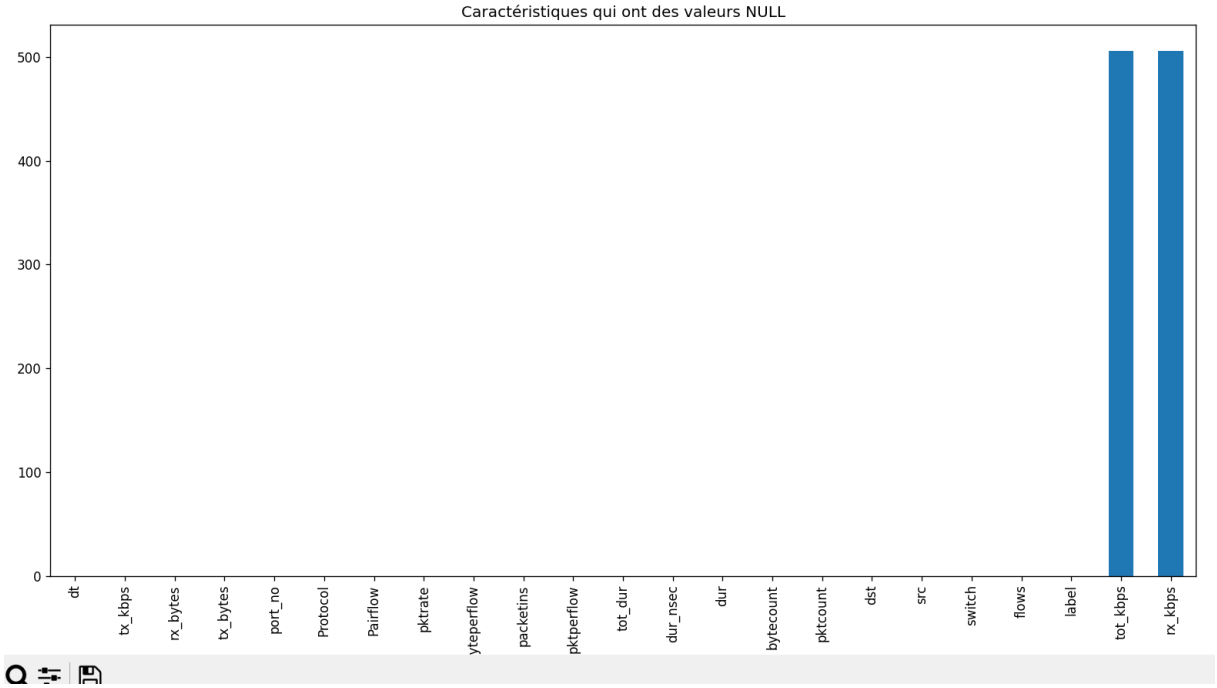


Figure IV. 17: Valeurs Manquantes

***Préparation des Données pour la Classification : Vue d'Ensemble**

Les résultats que vous avez fournis montrent une analyse des types de colonnes dans votre ensemble de données. Voici une interprétation précise et claire :

Colonnes Numériques (**Numeric Columns**):

Vous avez un total de **20 colonnes numériques** dans votre ensemble de données, qui sont répertoriées ci-dessous :'**dt**', '**switch**', '**pktcount**', '**bytecount**', '**dur**', '**dur_nsec**', '**tot_dur**', '**flows**', '**packetins**', '**pktperflow**', '**byteperflow**', '**pktrate**', '**Pairflow**', '**port_no**', '**tx_bytes**', '**rx_bytes**', '**tx_kbps**', '**rx_kbps**', '**tot_kbps**', '**label**'.

Ces colonnes contiennent des données numériques telles que des entiers (int64) et des nombres à virgule flottante (float64).

Colonnes d'Objets (Object Columns): Vous avez un total de 3 colonnes d'objets (c'est-à-dire non numériques) dans votre ensemble de données, qui sont répertoriées ci-dessous : '**src**', '**dst**', '**Protocol**'.

Ces colonnes contiennent des données de type objet, ce qui peut indiquer généralement des chaînes de caractères ou des catégories.

Nombre de Caractéristiques Numériques (Number of Numeric Features): Vous avez un total de 20 caractéristiques numériques dans votre ensemble de données.

Nombre de Caractéristiques d'Objets (Number of Object Features): Vous avez un total de 3 caractéristiques d'objets dans votre ensemble de données.

Ces informations sont utiles pour comprendre la nature des données dans notre ensemble de données, ce qui peut orienter vos choix d'analyses et de modélisation. Les caractéristiques numériques sont généralement utilisées pour des opérations mathématiques et statistiques, tandis que les caractéristiques d'objets peuvent nécessiter un traitement spécial, tel que l'encodage catégorique, lors de la modélisation.

```
Numeric Columns:
Index(['dt', 'switch', 'pktcount', 'bytecount', 'dur', 'dur_nsec', 'tot_dur',
       'flows', 'packetins', 'pktperflow', 'byteperflow', 'pktrate',
       'Pairflow', 'port_no', 'tx_bytes', 'rx_bytes', 'tx_kbps', 'rx_kbps',
       'tot_kbps', 'label'],
      dtype='object')

Object Columns:
Index(['src', 'dst', 'Protocol'], dtype='object')

Number of Numeric Features:  20
Number of Object Features:  3
```

Figure IV. 18: Analyse Exploratoire de la Base de Données Réseau

*Présentation du Nombre Total de Requêtes :

Nombre total de requête un diagramme à barres horizontales qui permet de visualiser le nombre total de requêtes pour chaque adresse IP source dans le DataFrame. Cela peut être utile pour identifier les adresses IP les plus actives ou les plus sollicitées dans le jeu de données.

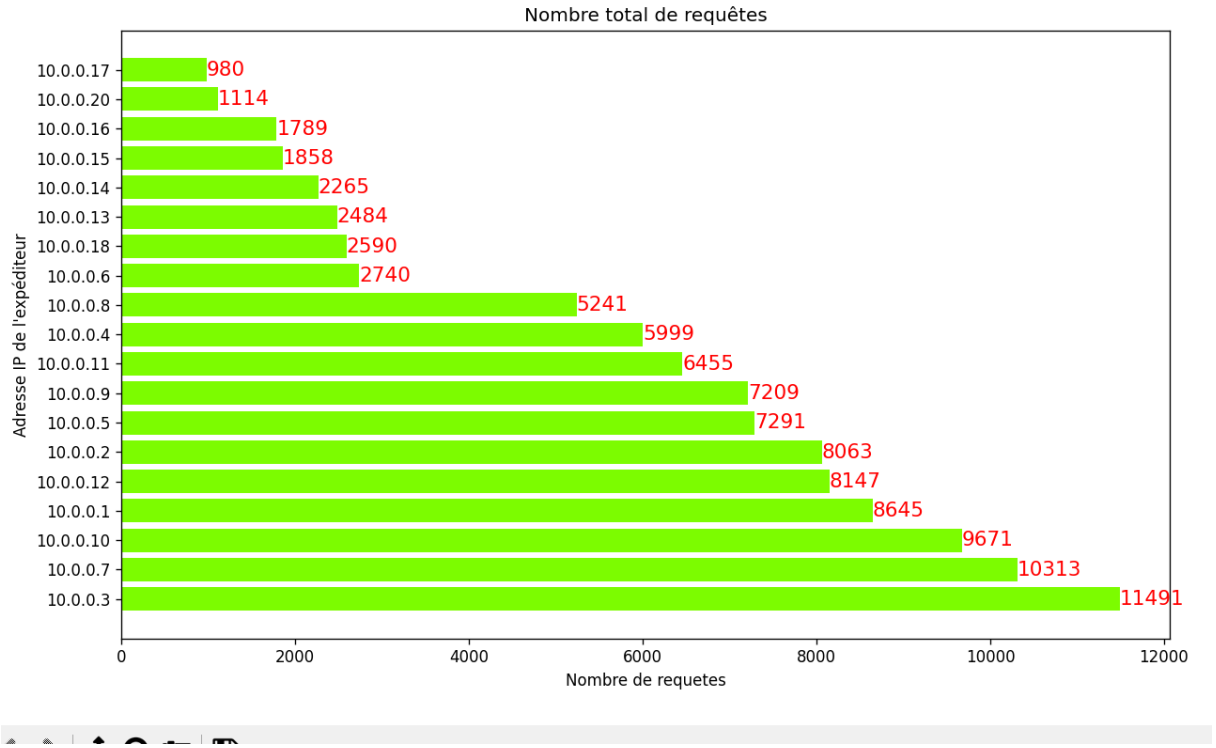


Figure IV. 19: Nombre Total de Requêtes

*Nombre de requêtes lancé par l'attaquant :

Nous sommes en présence d'un diagramme à barres horizontales qui permet de visualiser le nombre total de requêtes pour chaque adresse IP source, mais uniquement pour les entrées du DataFrame associées à des attaques. Cette visualisation peut être utile pour identifier les adresses IP source les plus fréquemment impliquées dans des activités malveillantes.

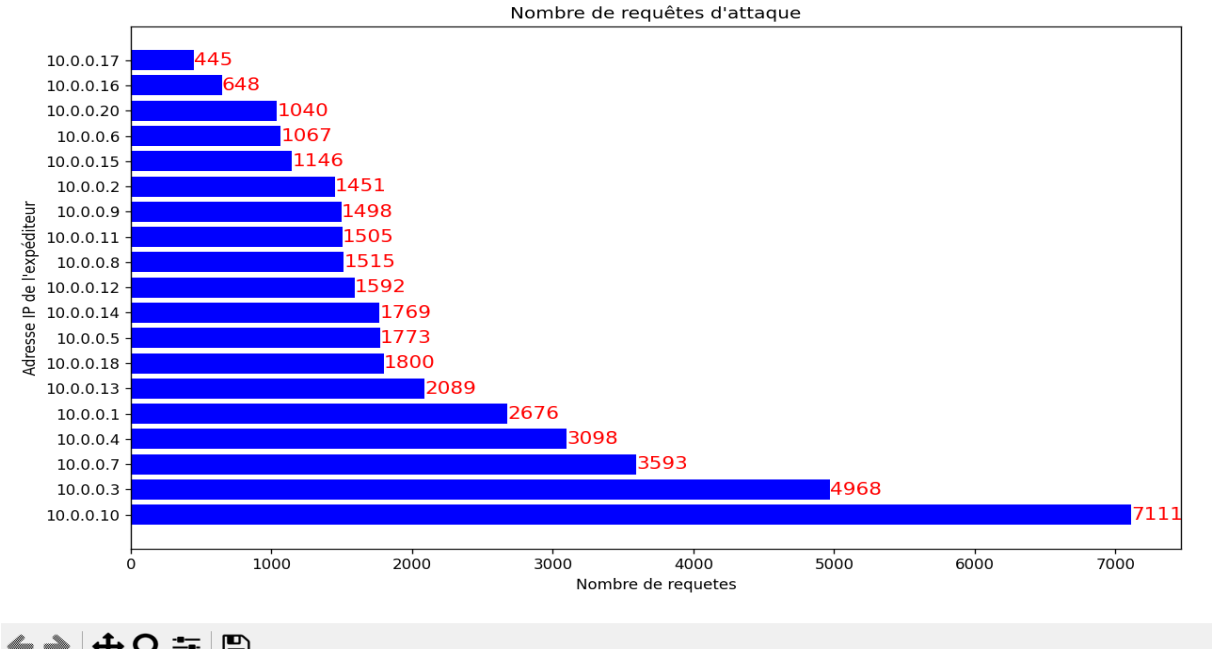


Figure IV. 20: Nombre de Requêtes d'attaques

***Requêtes Provenant d'adresses IP différents**

Ici nous avons un diagramme à barres horizontales pour représenter le nombre de requêtes provenant de différentes adresses IP, en distinguant les requêtes malveillantes de l'ensemble total. Les adresses IP sont affichées sur l'axe vertical, le nombre de requêtes sur l'axe horizontal, et deux séries de barres de couleurs différentes représentent les requêtes totales et malveillantes.

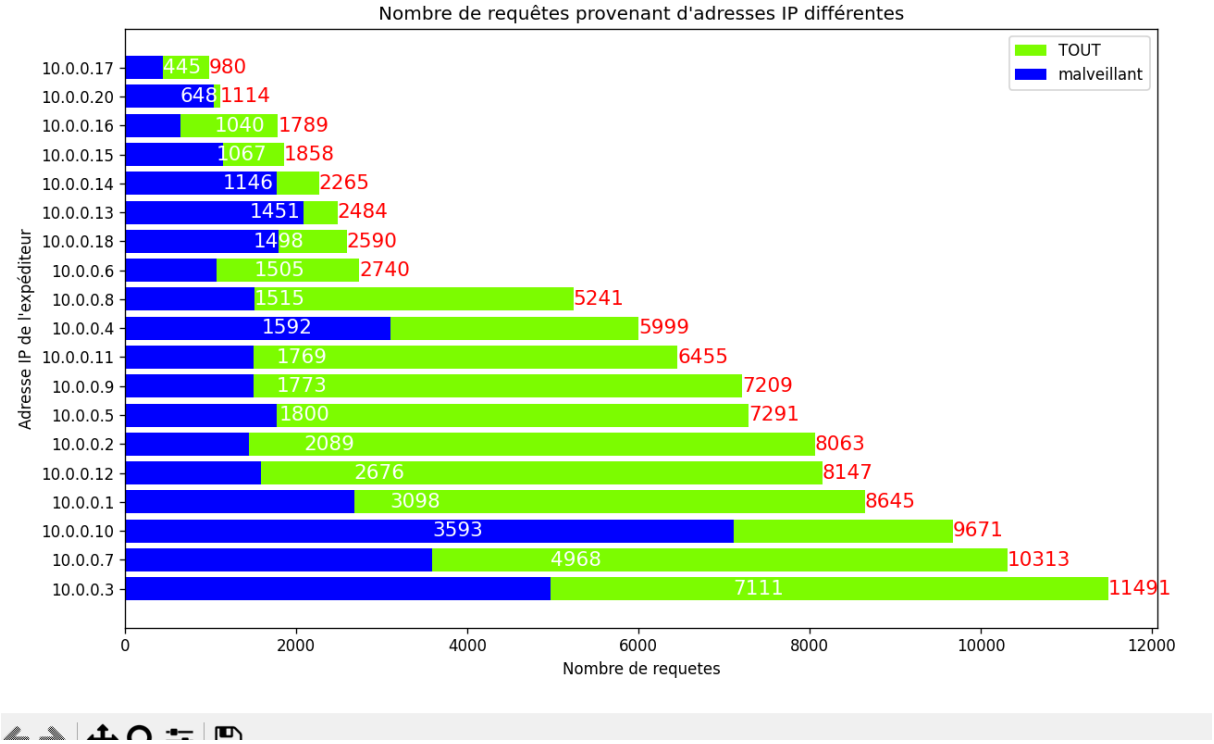


Figure IV. 21: Nombre de requêtes

***Nombre de Requêtes par Protocol :**

Ici nous visualisons le nombre de requêtes pour chaque protocole, en différenciant les requêtes malveillantes de l'ensemble total, à l'aide d'un diagramme à barres coloré avec des annotations de comptage.

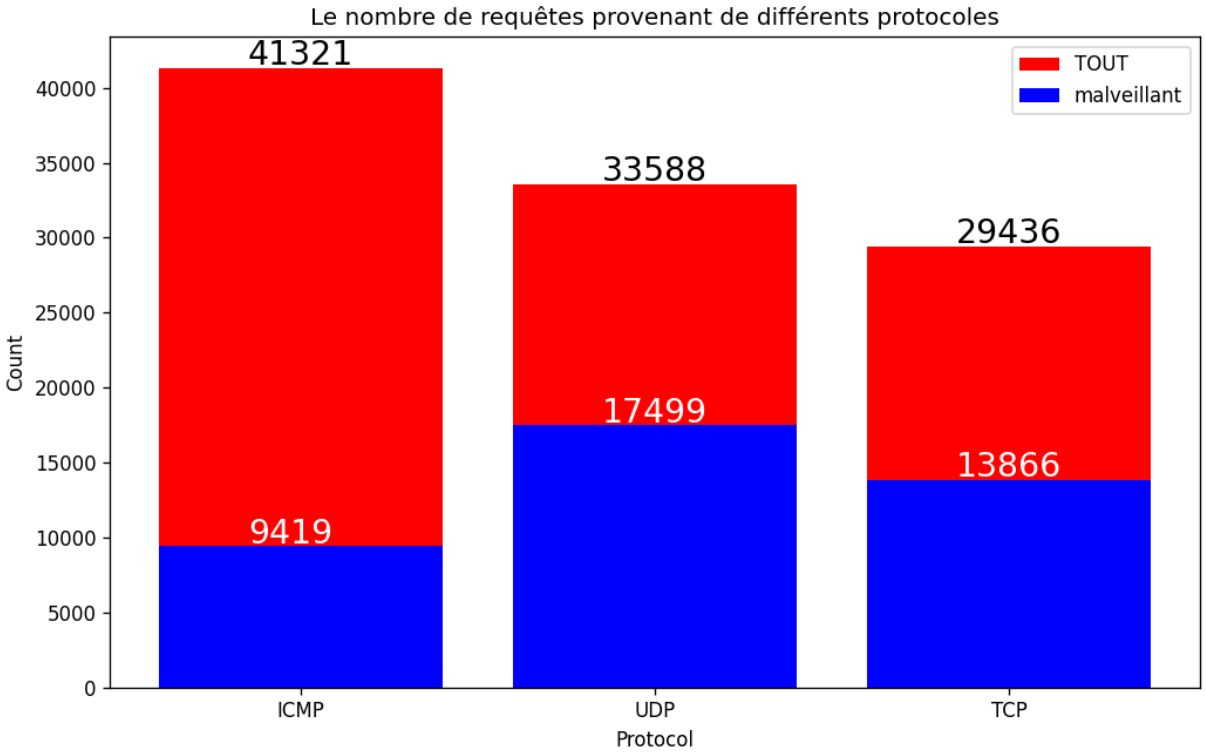


Figure IV. 22: Requêtes Des Différents Protocoles

***Distribution du temps :**

L'histogramme généré c'est pour visualiser la distribution des valeurs de la variable "durée" dans le DataFrame. Chaque barre de l'histogramme représente la fréquence des valeurs de durée tombant dans une plage particulière, et la couleur des barres est définie en bleu.

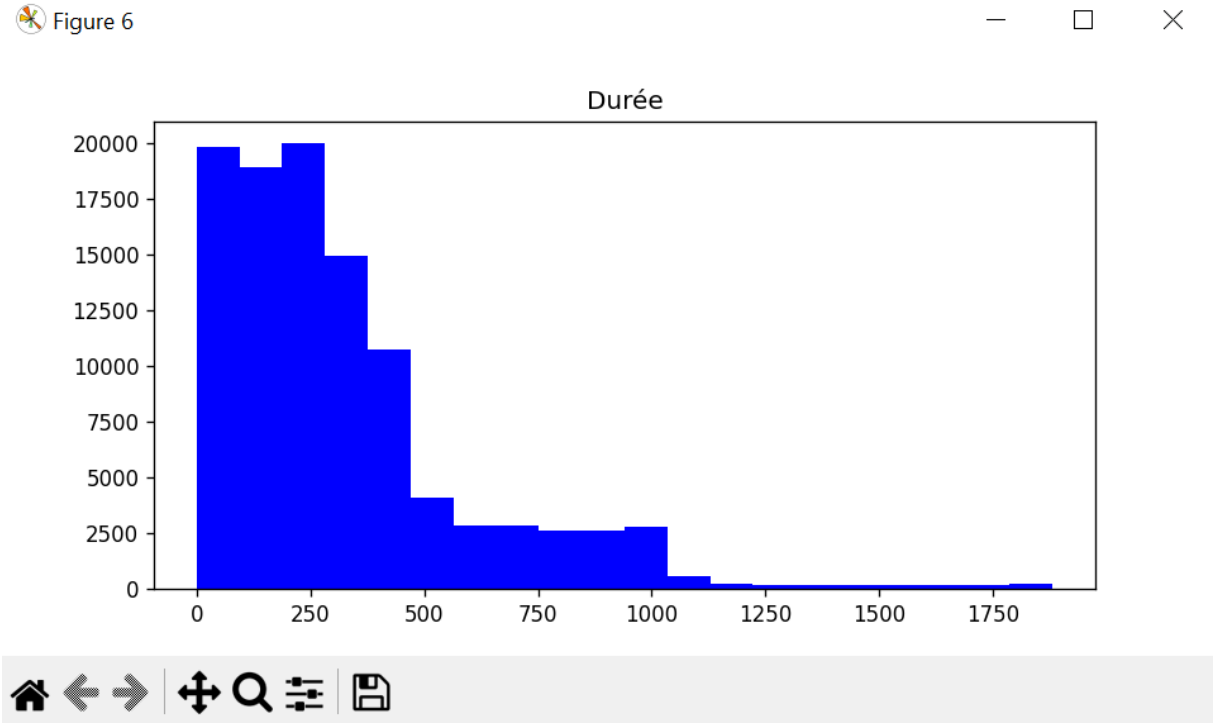


Figure IV. 23: Durée

***Les Octets Envoyée :**

C’est un histogramme pour visualiser la distribution des valeurs de la variable **"tx_bytes"** dans le DataFrame. La couleur rouge des barres de l'histogramme aide à identifier visuellement la distribution des octets transmis, et le titre du graphique fournit une information contextuelle sur la variable étudiée. Chaque barre de l'histogramme représente la fréquence des observations tombant dans une plage particulière de valeurs de **"tx_bytes"**

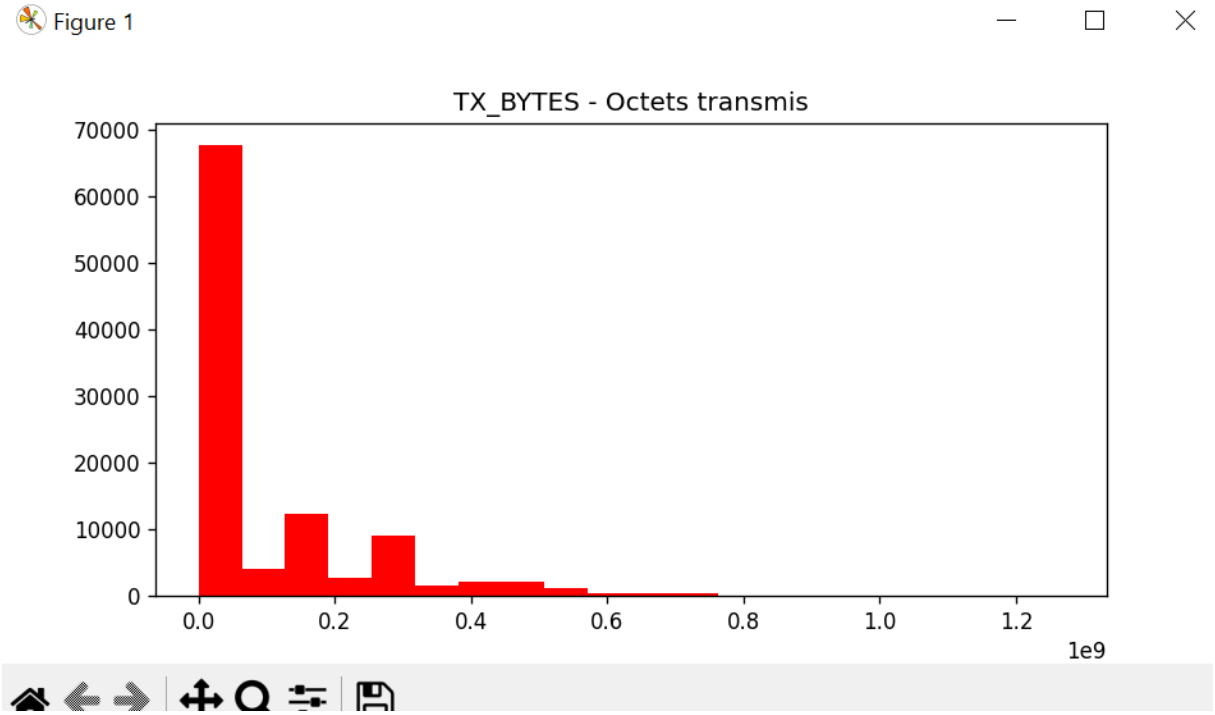


Figure IV. 24: Octets Transmis

***les valeurs de tx_kbps :**

Ici nous avons un histogramme pour visualiser la distribution des valeurs de la variable "tx_kbps" dans le DataFrame df. Les barres de l'histogramme, colorées en vert, représentent la fréquence des observations tombant dans chaque intervalle de valeurs de "tx_kbps". Le titre du graphique fournit une information contextuelle sur la variable étudiée, à savoir les kilo-octets transmis par seconde

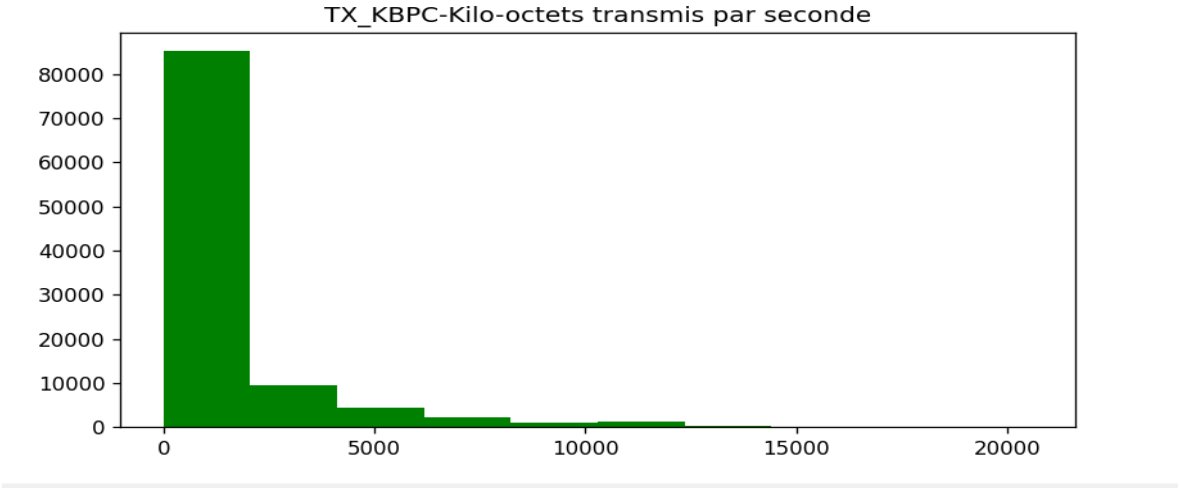


Figure IV. 25: Octets envoyé par Seconde

***Fréquence des différents états de SWITCH**

La figure ci-dessous est un histogramme pour visualiser la distribution des valeurs de la variable "switch" dans le DataFrame. Les barres de l'histogramme, colorées en rouge, représentent la fréquence des observations tombant dans chaque intervalle de valeurs de "switch". Le titre du graphique fournit une information contextuelle sur la variable étudiée, à savoir "SWITCH". L'étiquette de l'axe des x indique que l'axe représente la variable "switch".

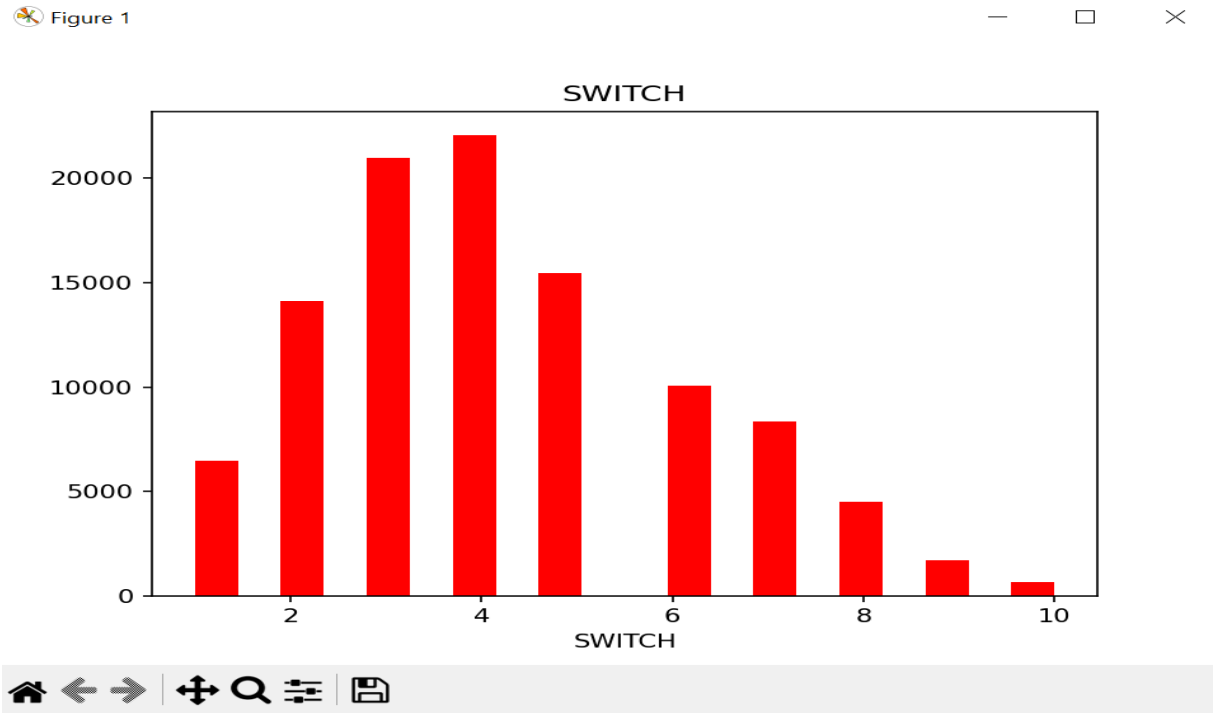


Figure IV. 26: Répartitions des valeurs de SWITCH

***Analyse de la variable 'SWITCH' pour les données étiquetées comme malveillantes** :

Cet histogramme permet une analyse plus approfondie de la variable "switch" dans le contexte des observations malveillantes, aidant ainsi à mieux comprendre son comportement dans ces scénarios

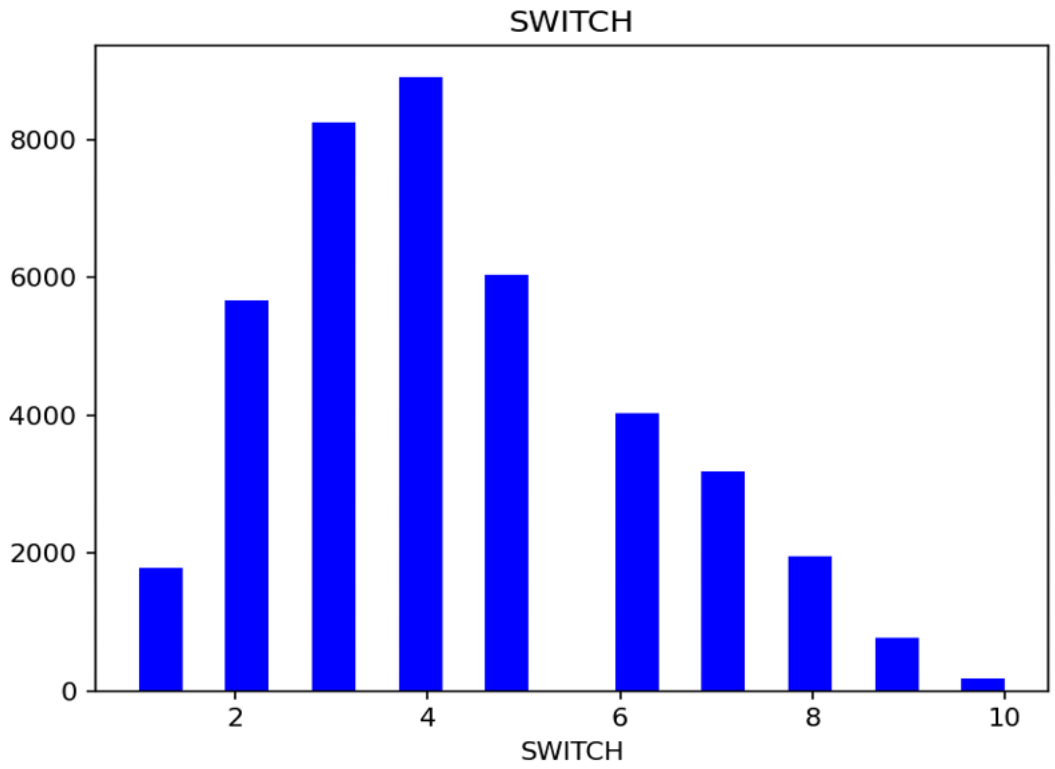


Figure IV. 27: Caractéristiques de 'SWITCH' lors d'événements malveillants

IV.4 Les Modèles classiques d'apprentissage automatique : Métriques & Interprétation des résultats obtenus

I.V.4.1Analyse des performances du modèle de Régression Logistique : Précision, Choix du Solveur, et Temps d'Exécution

Les résultats indiquent que le modèle de régression logistique a atteint une précision globale de **76,64%.** Cela signifie que le modèle a correctement prédit la classe correcte pour environ **76,64%** de l'ensemble de données de test.

En ce qui concerne le choix du solveur, le modèle a sélectionné **"liblinear"** comme le meilleur solveur pour cette tâche particulière.

En analysant la matrice de confusion (**precision, recall, f1-score),** on peut voir que le modèle a bien performé pour la classe 0 (inoffensif) avec une précision de **84%,** un rappel de **79%** et un **score F1 de 81**%. Cependant, pour la classe 1 (malveillant), les performances sont légèrement inférieures avec une **précision de 66%**, un **rappel de 72%** et un **score F1 de 69%.** En résumé, bien que le modèle ait une précision globale de **76,64%,** il est important de noter que les performances peuvent varier selon les classes, et des ajustements ou des améliorations peuvent être nécessaires pour améliorer la prédiction des instances malveillantes.

Le **temps d'exécution** du modèle a été de 4,31 secondes. Ce paramètre peut également être pris en compte en fonction des exigences de performances spécifiques.

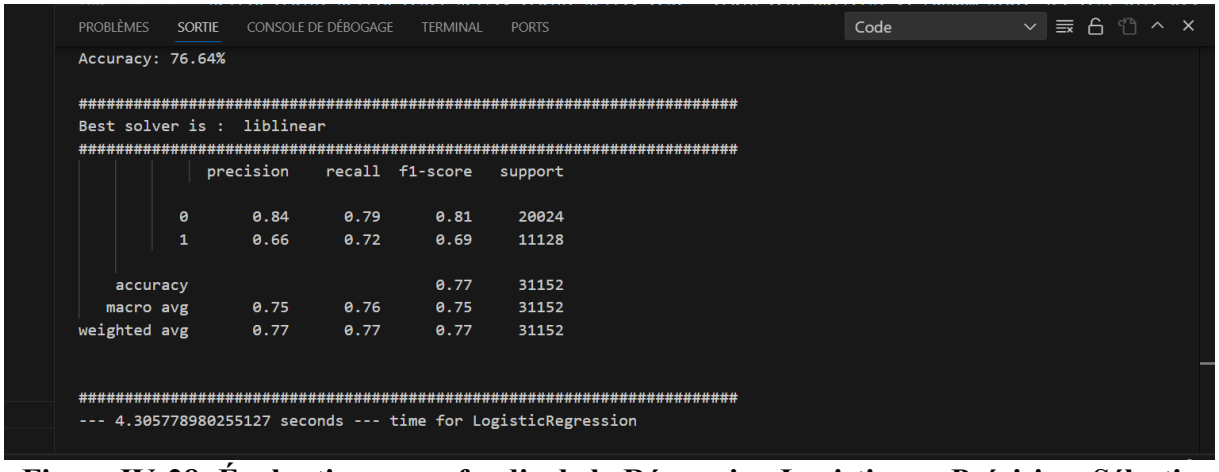


Figure IV. 28: Évaluation approfondie de la Régression Logistique : Précision, Sélection du Solveur et Durée d'Exécution

IV.4.2 Performance Comparative des Noyaux SVM dans la Classification (SVM)

Ces résultats concernent l'entraînement et l'évaluation d'un modèle SVM avec différents noyaux. Analysons et interprétons chaque partie des résultats :

Précisions par Noyau :

- **Linear Kernel** : Accuracy: 78.40%
- **Polynomial Kernel** : Accuracy: 96.53%
- **RBF Kernel** : Accuracy: 96.67%
- **Sigmoid Kernel** : Accuracy: 54.52%

Le noyau RBF a la précision la plus élevée **(96.67%)**, suivi par le noyau polynomial **(96.53%)**. Le noyau linéaire est moins performant (78.40%), et le noyau sigmoid a la performance la plus basse (54.52%).

Meilleur Noyau et Performance Globale :

Accuracy of SVM model 97.0%

Best Kernel is : **rbf**

Le modèle SVM avec le noyau RBF a été sélectionné comme le meilleur, atteignant une précision globale de 97.0%. Cela suggère que le modèle avec le noyau RBF généralise bien aux données de test.

Rapport de Classification :

Précision, Rappel, F1-score pour chaque classe (0 et 1), ainsi que des métriques agrégées.

Les métriques de **précision, rappel** et **f1-score** montrent une performance équilibrée pour les deux classes **(0 et 1)**. La performance est généralement élevée, indiquant une capacité du modèle à bien identifier les deux classes.

Temps d'Exécution : le temps est de **1065.0534799098969 seconds**

Le modèle a pris environ 1065 secondes (environ 17 minutes) pour s'entraîner et évaluer avec le meilleur noyau **(RBF).** Cela peut être important pour évaluer l'efficacité du modèle dans un contexte d'application spécifique.

En définitif, le modèle **SVM** avec le **noyau RBF** semble bien performant sur ces données, offrant une précision globale élevée et de bonnes performances sur les métriques de classification. Cependant, il est important de prendre en compte d'autres facteurs tels que la sensibilité aux temps d'exécution et la capacité à généraliser à d'autres ensembles de données.

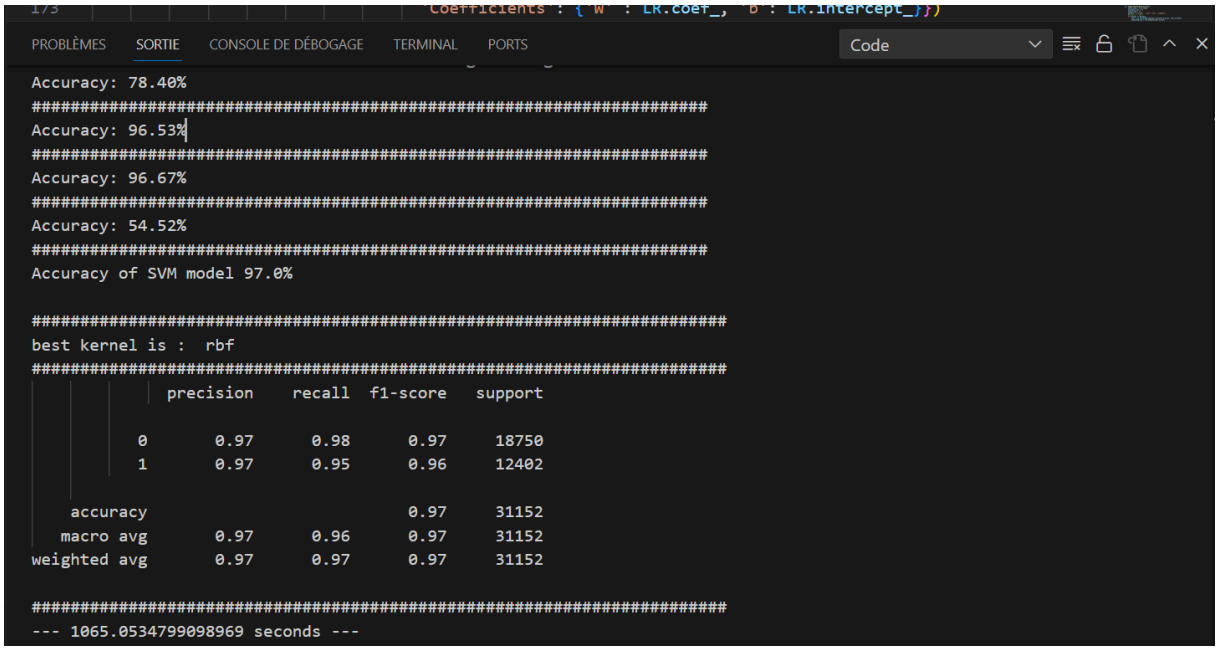


Figure IV. 29: Évaluation et Sélection de Noyaux SVM pour la Prédiction

IV.4.3 Optimisation des Hyperparamètres pour un Modèle d'Arbre de Décision

Ce message indique qu'une opération de recherche des meilleurs hyperparamètres pour un modèle de classification basé sur un arbre de décision est en cours. Voici une explication :

Fitting 5 folds for each of 180 candidates : Le modèle est entraîné (ajusté) sur plusieurs combinaisons d'hyperparamètres (180 au total) en utilisant une validation croisée à 5 plis. Cela signifie que le jeu de données est divisé en 5 parties (plis), et le modèle est entraîné 5 fois, chaque fois avec un ensemble différent comme ensemble de validation.

- **criterion:** Il s'agit d'un critère de qualité pour mesurer la pureté des nœuds dans l'arbre de décision. Ici, "gini" est utilisé, ce qui signifie que l'indice de Gini est utilisé comme critère.
- **max depth:** C'est la profondeur maximale de l'arbre de décision. Ici, la profondeur maximale est fixée à 8.
- **max_leaf:** Il s'agit du nombre maximum de feuilles (terminaisons) dans l'arbre. Ici, le nombre maximum de feuilles est fixé à 11.

Le message indique que le modèle est en train d'expérimenter différentes combinaisons de critères, de profondeurs et de nombre de feuilles pour trouver les hyperparamètres qui fournissent les meilleures performances sur le jeu de données, en utilisant une approche de validation croisée.

```
Fitting 5 folds for each of 180 candidates, totalling 900 fits
criterion: gini, max depth: 8, max_leaf: 11
```

Figure IV. 30: Recherche des Meilleurs Paramètres : Modèle d'Arbre de Décision

Ces résultats semblent provenir d'un modèle d'arbre de décision entraîné sur un ensemble de données. Voici une analyse détaillée :

Paramètres du modèle :

- Critère (criterion) utilisé pour mesurer la qualité d'une scission : Gini
- Profondeur maximale de l'arbre (max depth) : 8
- Nombre maximal de feuilles (max leaf) : 11

Ces paramètres décrivent comment l'arbre de décision a été construit.

Précision globale (Accuracy) : 98.22%

L'accuracy mesure la proportion totale d'instances correctement classées. Une valeur élevée, comme celle-ci, suggère une performance globale excellente du modèle.

Métriques de classification :

- **Précision** (Precision) :
    - Pour la classe 0 (étiquette "0"): 98%
    - Pour la classe 1 (étiquette "1"): 99%

La précision mesure la proportion de vrais positifs parmi les exemples prédits comme positifs. Des valeurs élevées indiquent une capacité du modèle à minimiser les faux positifs.

- **Rappel (Recall) :**
    - Pour la classe 0 : 99%
    - Pour la classe 1 : 97%

Le rappel mesure la proportion de vrais positifs parmi les exemples réellement positifs. Des valeurs élevées suggèrent que le modèle est capable de retrouver la plupart des exemples positifs.

- **F1-Score :**
    - **Pour la classe 0 : 99%**
    - **Pour la classe 1 : 98%**

Le F1-Score est une mesure qui combine à la fois la précision et le rappel. Des valeurs élevées indiquent un bon équilibre entre la précision et le rappel.

Moyenne Macro (Macro avg) et Moyenne Pondérée (Weighted avg) :

Ces indicateurs agrègent les performances sur l'ensemble des classes. Dans ce cas, ils sont tous deux élevés, indiquant une bonne performance globale.

Temps d'exécution : Le modèle a pris **62.04 secondes** pour s'entraîner et générer ces résultats.

En conclusion, ces résultats indiquent un modèle d'arbre de décision performant avec une excellente précision, rappel et F1-Score, et ils détaillent également les paramètres spécifiques du modèle utilisé

```
Fitting 5 folds for each of 180 candidates, totalling 900 fits
criterion: gini, max depth: 8, max_leaf: 11
The Accuracy is : 98.22%
##########################################################################
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     18743
           1       0.99      0.97      0.98     12409

    accuracy                           0.98     31152
   macro avg       0.98      0.98      0.98     31152
weighted avg       0.98      0.98      0.98     31152

##########################################################################
--- 62.040077447891235 seconds ---
```

Figure IV. 31: Évaluation Approfondie d'un Modèle avec un Arbre de Décision

***Ajustement du modele :** Ces résultats indiquent une étape spécifique de l'ajustement d'un modèle, en l'occurrence, un modèle d'arbre de décision avec des variations d'hyperparamètres. Analysons-les en détail :

[CV] END ... : Chaque ligne correspond à la fin d'une itération de la validation croisée (cross-validation) avec une combinaison spécifique d'hyperparamètres.

- **criterion=entropy:** Le critère d'impureté utilisé pour mesurer la qualité d'une division dans l'arbre est l'entropie. C'est une mesure de l'incertitude ou du chaos dans les données.
- **max_depth=10:** La profondeur maximale de l'arbre est définie à 10. Cela limite le nombre de niveaux ou de divisions dans l'arbre.
- **max_leaf_nodes=7:** Le nombre maximum de nœuds feuilles (feuilles de l'arbre) est défini à 7. Cela limite le nombre total de feuilles dans l'arbre.
- **total time=0.5s:** Le temps total nécessaire pour ajuster le modèle avec cette configuration particulière d'hyperparamètres est de 0.5 seconde.

Fitting 3 folds for each of 12 candidates, totalling 36 fits : Comme indiqué précédemment, le processus global implique la validation croisée avec **3 folds** (ensembles de données) et la recherche d'hyperparamètres avec **12 combinaisons différentes**. Au total, cela donne **36 ajustements du modèle.**

En résumé, ces lignes de résultats montrent que le modèle d'arbre de décision est ajusté plusieurs fois avec différentes combinaisons d'hyperparamètres, et le temps total requis pour cette étape spécifique de la recherche d'hyperparamètres est indiqué. Cela fait partie du processus visant à trouver les meilleures valeurs d'hyperparamètres pour optimiser les performances du modèle.

```
[CV] END ..criterion=entropy, max_depth=10, max_leaf_nodes=3; total time=   0.3s
[CV] END ..criterion=entropy, max_depth=10, max_leaf_nodes=5; total time=   0.5s
[CV] END ..criterion=entropy, max_depth=10, max_leaf_nodes=7; total time=   0.5s
[CV] END ..criterion=entropy, max_depth=10, max_leaf_nodes=8; total time=   0.5s
[CV] END .criterion=entropy, max_depth=10, max_leaf_nodes=10; total time=   0.5s
[CV] END .criterion=entropy, max_depth=10, max_leaf_nodes=11; total time=   0.4s
Fitting 3 folds for each of 12 candidates, totalling 36 fits
```

Figure IV. 32: Optimisation des hyperparamètres

IV.4.4 Performance Exceptionnelle : Évaluation du Modèle Random Forest

Les résultats indiquent que le modèle Random Forest (RF) a atteint une précision exceptionnelle de 99.99%. Voici une analyse détaillée des mesures de performance :

Précision Globale : La précision globale du modèle, mesurée par l'accuracy, est de 99.99%. Cela signifie que le modèle a correctement classé 99.99% de l'ensemble des échantillons.

Précision, Rappel et F1-Score par Classe :

- **Pour la classe 0 (non-malveillant),** la précision, le rappel et le F1-score sont tous égaux à 1.00. Cela indique que le modèle a correctement identifié tous les exemples de cette classe, sans faux positifs ni faux négatifs.
- **Pour la classe 1 (malveillant),** les mesures de précision, rappel et F1-score sont également tous égaux à 1.00. Cela suggère que le modèle a bien identifié tous les exemples de la classe malveillante, sans erreurs significatives.

Mesures Agrégées (Macro AVG et Weighted AVG) : Les mesures agrégées telles que la macro moyenne (macro avg) et la moyenne pondérée (weighted avg) indiquent également une performance exceptionnelle du modèle. Ces mesures prennent en compte la moyenne des performances par classe, en accordant ou non plus d'importance aux classes selon le cas.

En résumé, les résultats témoignent d'une performance pratiquement parfaite du modèle Random Forest sur cet ensemble de données spécifique. Cependant, il est toujours essentiel de prendre en compte d'autres considérations telles que la possibilité de surajustement ou l'adéquation du modèle aux besoins spécifiques de la tâche

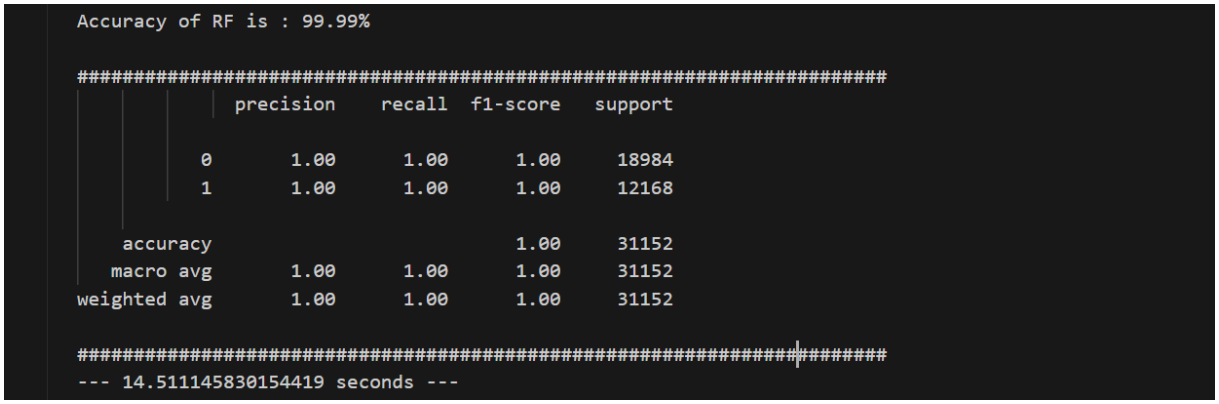


Figure IV. 33: Précision Maximale : Analyse des Résultats du Modèle Random Forest

IV.4.5 Performance Exceptionnelle : Évaluation du Modèle KNN

Le résultat fournit une évaluation de la performance d'un modèle KNN (K-Nearest Neighbors) avec les métriques de précision, rappel, f1-score et l'accuracy, ainsi que les statistiques détaillées pour chaque classe (0 et 1) et une macro moyenne et moyenne pondérée.

Accuracy : L'accuracy du modèle KNN est de 98.0%, ce qui représente la proportion totale de prédictions correctes parmi toutes les prédictions effectuées. C'est une mesure globale de la performance du modèle.

Métriques par classe :

- **Classe 0** : La précision, le rappel et le f1-score pour la classe 0 sont tous très élevés (99%), indiquant que le modèle prédit correctement la classe 0 dans la grande majorité des cas.
- **Classe 1** : Bien que légèrement inférieures à la classe 0, les métriques pour la classe 1 restent élevées, avec une précision, un rappel et un f1-score de 98%.

Moyennes macro et pondérées :

- **Macro avg :** La moyenne des métriques pour chaque classe donne une vue d'ensemble de la performance du modèle sans tenir compte du déséquilibre de classe. Dans ce cas, la moyenne macro est de 98%.
- **Weighted avg :** La moyenne pondérée tient compte du déséquilibre de classe en attribuant des poids proportionnels à la fréquence de chaque classe. La moyenne pondérée est également de 98%.

Durée d'exécution : Le modèle KNN a été entraîné en 3361.55 secondes, ce qui représente le temps total nécessaire pour ajuster le modèle et effectuer les prédictions.

le modèle KNN semble être performant avec une accuracy élevée et des métriques de classe solides. Cependant, il est toujours important de prendre en compte d'autres considérations telles que le temps d'exécution, la sensibilité aux erreurs de type I ou II, et le contexte spécifique de l'application.

```
Accuracy of KNN model 98.0%

###########################################################################
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     19002
           1       0.98      0.98      0.98     12150

    accuracy                           0.98     31152
   macro avg       0.98      0.98      0.98     31152
weighted avg       0.98      0.98      0.98     31152

###########################################################################
--- 3361.546883583069 seconds ---
```

Figure IV. 34: Analyse approfondie des Résultats du Modèle KNN

Graphe des Accuracy du KNN

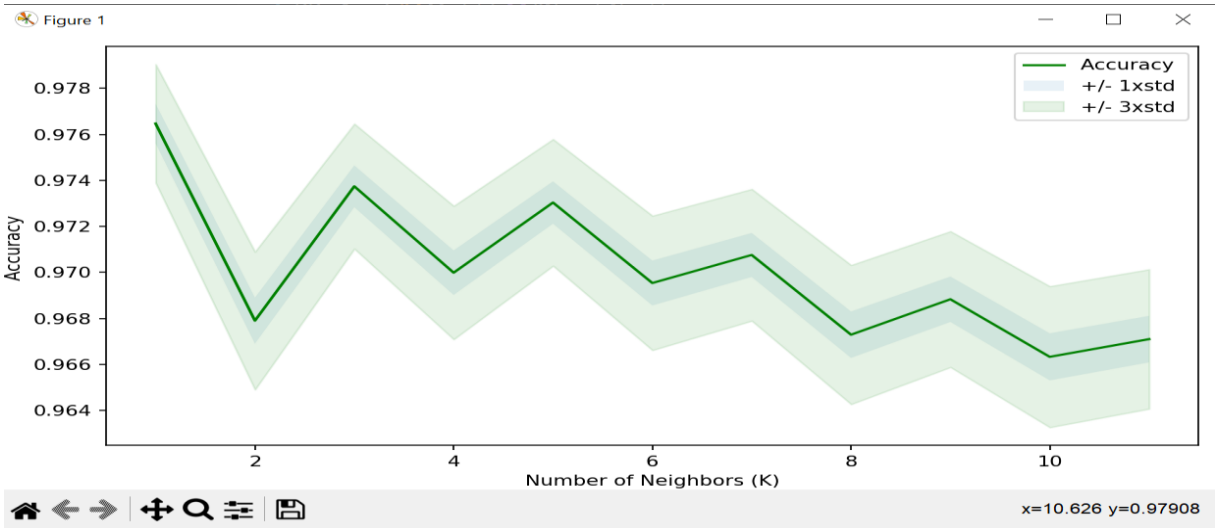


Figure IV. 35: Nombre de K

***Résultat de notre modèle pour la Détection Précoce**

Les résultats de votre modèle de détection précoce axé sur la variation du nombre de paquets montrent les éléments suivants :

Accuracy (Précision) : 76.64%

L'accuracy indique la proportion totale de prédictions correctes réalisées par le modèle. Dans ce cas, elle est de 76.64%, ce qui signifie que le modèle a correctement prédit environ 76.64% des exemples dans l'ensemble de données.

Meilleur solveur : liblinear

Le solveur "liblinear" a été identifié comme le meilleur pour ce modèle particulier. Le choix du solveur peut avoir un impact sur la vitesse de convergence et la performance globale du modèle.

Métriques pour la Classe 0 (Non-DDoS) :

- **Precision (Précision) : 84%**
- **Recall (Rappel) : 79%**
- **F1-score : 81%**

Support : 20024 (nombre d'instances de la classe 0 dans l'ensemble de données)

Métriques pour la Classe 1 (DDoS) :

- **Precision (Précision) : 66%**
- **Recall (Rappel) : 72%**
- **F1-score : 69%**

Support : 11128 (nombre d'instances de la classe 1 dans l'ensemble de données)
Macro avg et weighted avg :
Les moyennes macro et pondérées des métriques donnent une vue d'ensemble de la performance du modèle en considérant l'importance relative des deux classes.
Temps d'exécution : 3.81 secondes
Il s'agit du temps nécessaire pour entraîner le modèle de régression logistique.

L'analyse des métriques montre que le modèle a une précision acceptable, mais il est important de noter que la détection des attaques DDoS, particulièrement basée sur la variation du nombre de paquets, peut être complexe en raison de la diversité des attaques et des méthodes utilisées par

les attaquants. Il pourrait être utile d'explorer davantage la performance du modèle sur des métriques spécifiques aux attaques DDoS, en tenant compte des particularités du trafic réseau.

```
Accuracy: 76.64%

########################################################################
Best solver is :  liblinear
########################################################################
              precision    recall  f1-score   support

           0       0.84      0.79      0.81     20024
           1       0.66      0.72      0.69     11128

    accuracy                           0.77     31152
   macro avg       0.75      0.76      0.75     31152
weighted avg       0.77      0.77      0.77     31152


########################################################################
--- 3.8114712238311768 seconds --- time for LogisticRegression
```

Figure IV. 36: Résultat du Détection précoce DDoS

IV.5 Configuration du Paramètres SMTP Gmail pour l'alerte

Dans notre projet on‘ à créer un email qu'on a synchronisé avec notre code python qui sera le serveur de messagerie qui va envoyer l'alertes à l'ingénieur

Procédé : On va dans notre mail et clique sur **Gérer mon compte** puis la **Figure IV.17** s'affiche

Et nous cliquons sur **Sécurité**

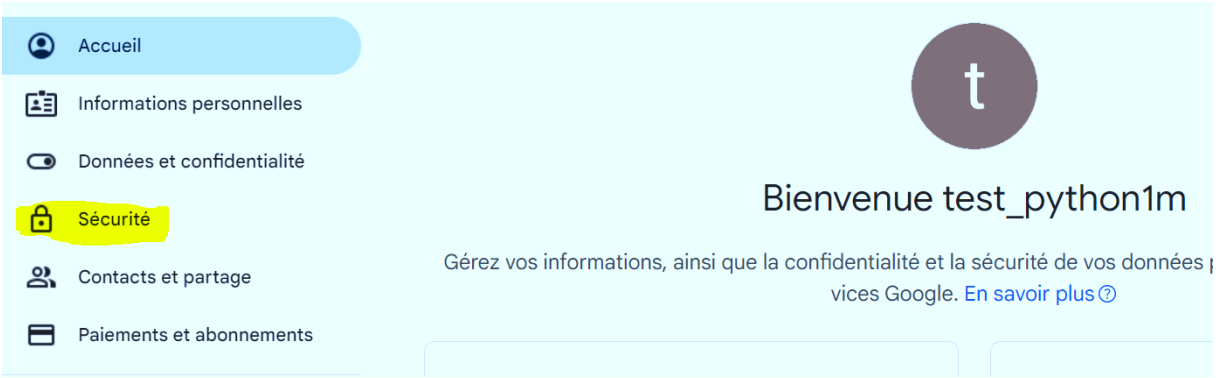


Figure IV. 37: Configuration du Messagerie : Aller sur Sécurité

Après avoir cliqué sur **Sécurité** on descend en bas jusqu'à **validation en deux étapes** puis on une clique

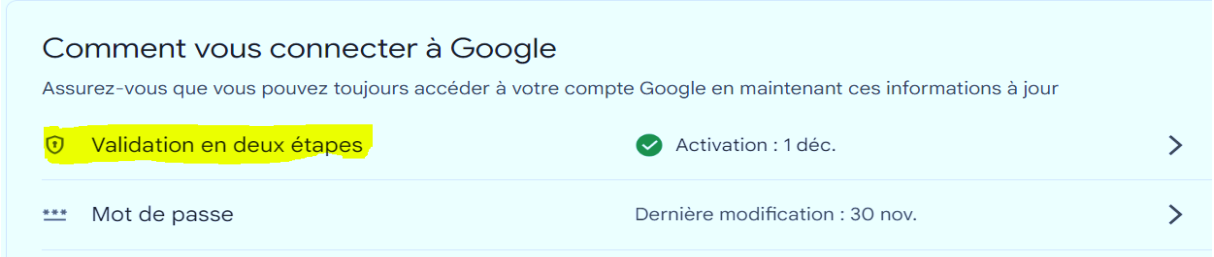


Figure IV. 38: Validation en deux étapes

Cette page qui va s'afficher nous demandant de saisir le mot de passe de notre Email pour avoir accès, après avoir défini notre mot de passe on clique sur **suivant**

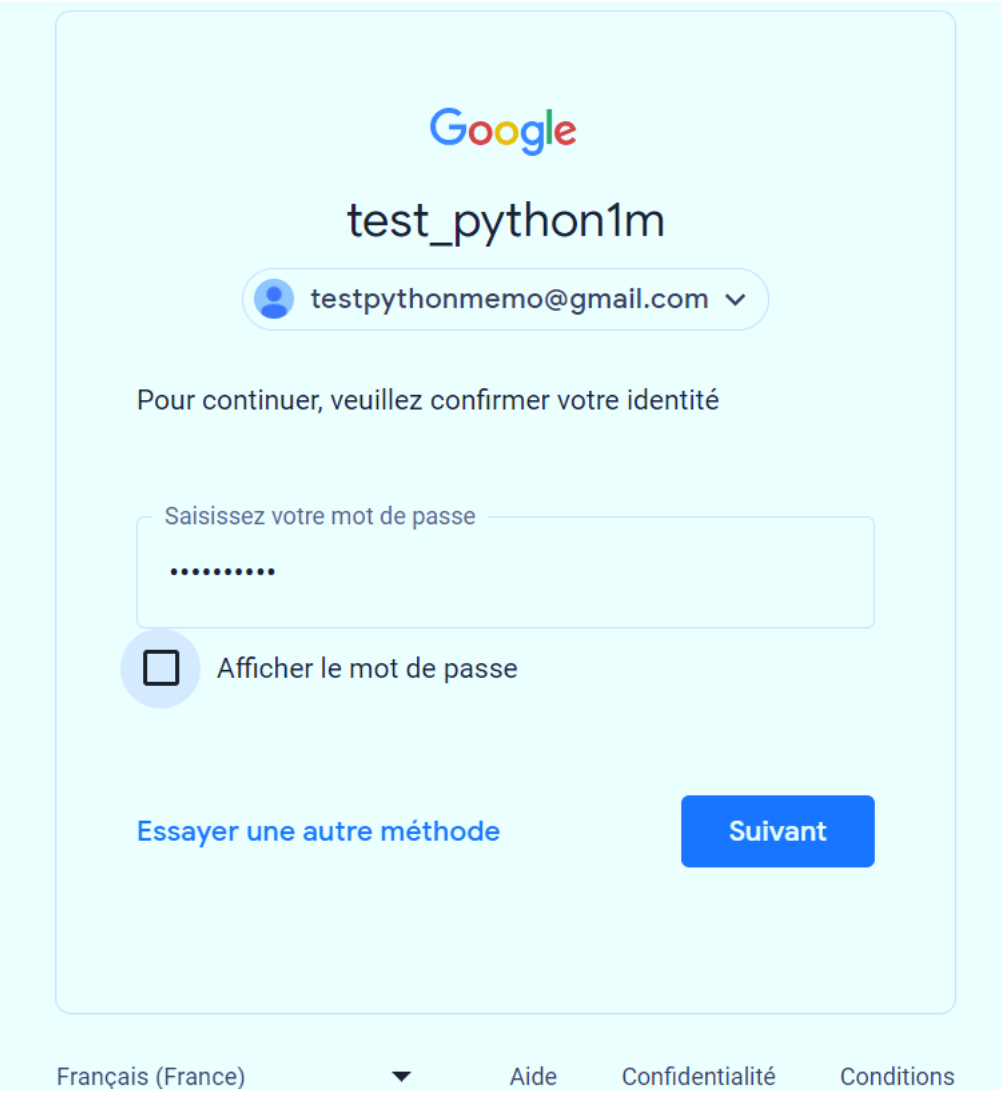


Figure IV. 39: Saisir mot de passe

Puis la fenêtre **valider en 2 étapes** s'affiche et on descend en bas jusqu'à **mot de passe des applications** puis on fait une clique

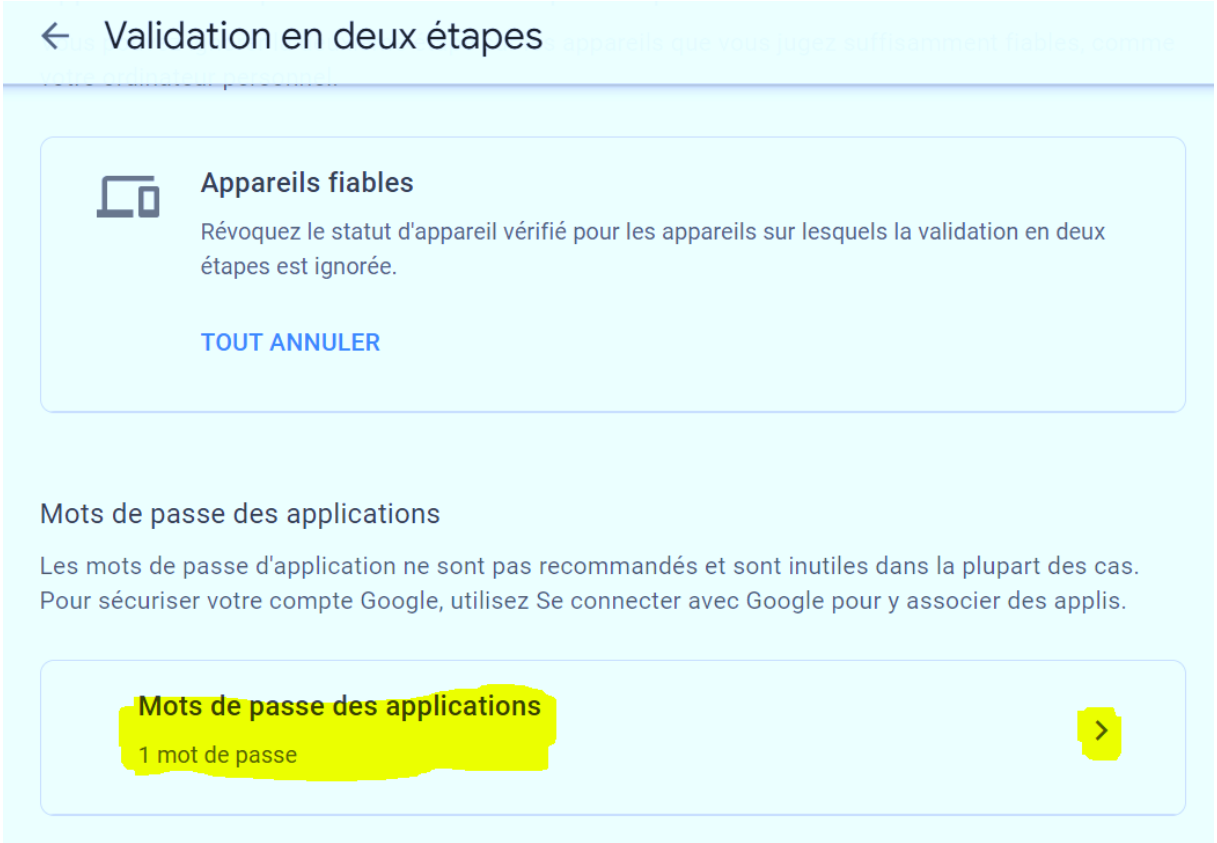


Figure IV. 40: Aller sur mot de passe des applications

Maintenant on va saisir le nom de l'application pour pouvoir créer un mot de passe spécifique

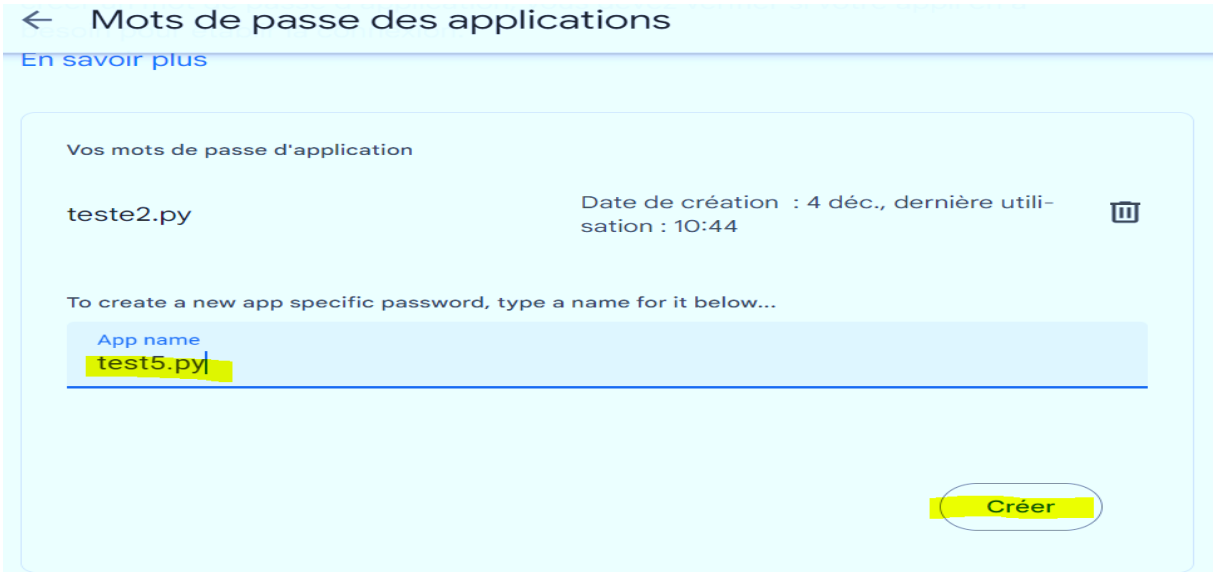


Figure IV. 41: Saisi du nom de applications pour créer mot de passe

Ici Il nous génère un mot de passe de 16chiffres qui doit être secret. Ces ce mot de passe qu'on va utiliser dans notre code pour le lier avec port 465 qui est un port Sécurisé qu'utilise SMTP_SSL

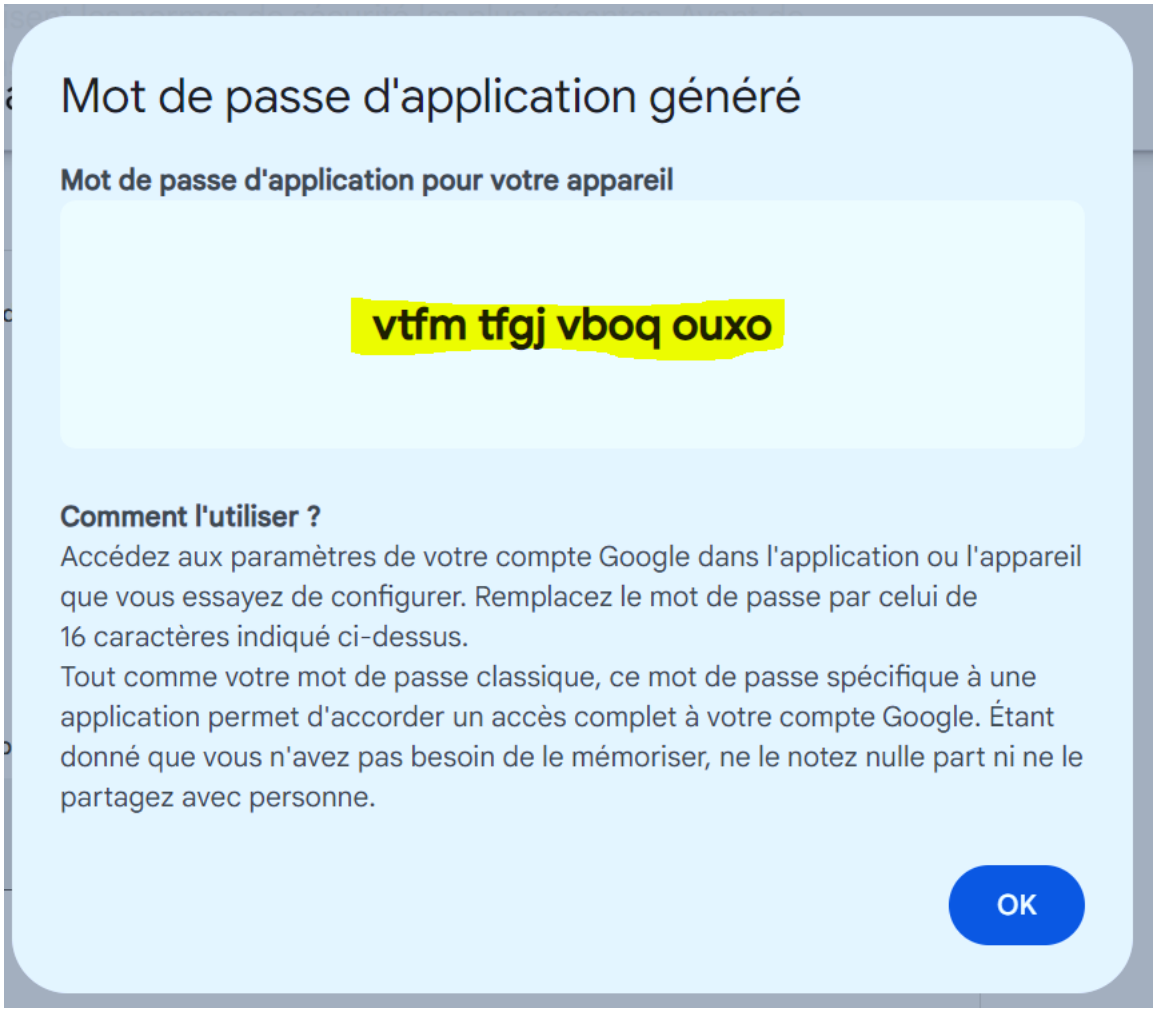


Figure IV. 42 : Mot de passe Générer

Je me dirige maintenant vers mon code pour intégrer la clé secrète générée ainsi que l'adresse e-mail. Ainsi, en cas de détection d'une tentative de phishing, le système enverra un e-mail à l'ingénieur à l'adresse testmemopython@gmail.com, en utilisant la clé secrète générée : vtfm tfgj vboq ouxo."

Voici l'email et le code secret qu'on a intégré dans le code

```
15  gmail_cfg = {
16      "server": "smtp.gmail.com",
17      "port": 465,  # Utiliser le port 465 pour SMTP_SSL
18      "username": "testpythonmemo@gmail.com",
19      "password": "vtfmtfgjvboqouxo"
20  }
```

Maintenant on va inclure le mail de l'ingenieur qui sera le destinataire de ce mail dans ce code le destinataire est : bb3579@gmail.com

```python
def mail_alert(destinataire, sujet, corps):
    with smtplib.SMTP_SSL(gmail_cfg["server"], gmail_cfg["port"]) as smtp:
        smtp.login(gmail_cfg["username"], gmail_cfg["password"])

        message = MIMEMultipart()
        message['from'] = gmail_cfg["username"]
        message['to'] = "bba3579@gmail.com"
        message['subject'] = sujet
        message.attach(MIMEText(corps, 'plain'))
```

Figure IV. 43 : Le protocole SMTP_SSL et le mail destinataire

Maintenant on exécute le code et on vois sa dans notre terminal



Figure IV. 44 : Message d'Alerte envoyé

Maintenant on observe le mail de l'ingénieur

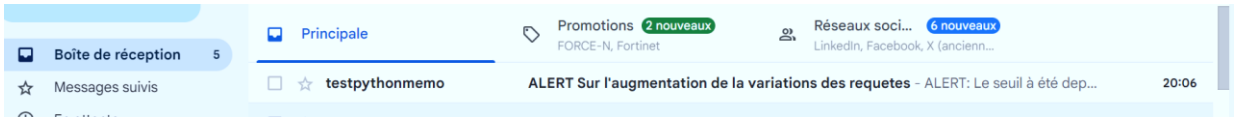


Figure IV. 45 : Affichage du Mail reçu dans la boite l'ingénieur

Quand on ouvre le mail on voix le message suivant :

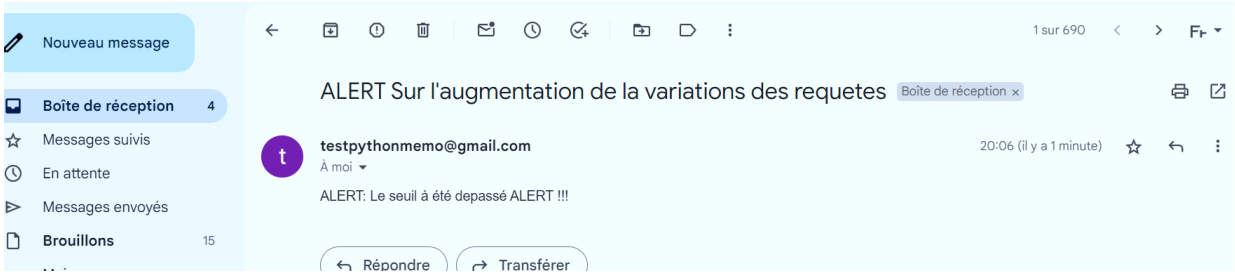


Figure IV. 46 : Message lue par le destinataire

IV.4 Conclusion

En conclusion, l'alerte précoce avec le Deep Learning semble être une approche efficace pour la détection de DDos, avec des résultats encourageants. Cependant, il est toujours recommandé de surveiller et d'ajuster le modèle au fil du temps pour maintenir et améliorer ses performances face à l'évolution des menaces et des schémas d'attaques.

CONCLUSION GENERALE

Cette recherche a exploré de manière approfondie la mise en place d'une détection précoce des attaques DDoS au sein d'un réseau informatique en utilisant des techniques de Deep Learning. Les résultats obtenus à travers l'application d'algorithmes de classification tels que KNN, SVM, RandomForest, et Arbre de Décision ont démontré l'efficacité de ces approches dans le contexte spécifique des menaces DDoS.

L'intégration du Deep Learning a apporté une dimension significative à la détection précoce, permettant une adaptation dynamique aux schémas d'attaques émergents. L'utilisation de données provenant de l'environnement SDN a ajouté une couche de complexité et de contextualisation, renforçant ainsi la pertinence des modèles développés.

Au-delà des résultats, cette étude ouvre la voie à de nouvelles perspectives dans le domaine de la cybersécurité. Les enseignements tirés de cette recherche pourraient orienter le développement futur de systèmes de détection précoce plus sophistiqués et adaptatifs, contribuant ainsi à renforcer la résilience des réseaux informatiques face aux menaces évolutives.

En résumé, cette recherche aspire à être une pierre angulaire dans l'avancement des techniques de détection précoce des attaques DDoS, offrant des pistes concrètes pour améliorer la sécurité des infrastructures informatiques et assurer une cyberdéfense proactive.

REFERENCE ET BIBLIOGRAPHIE

1. H. NSENGE MPIA, « Sécurité Informatique », Institut Supérieur Emmanuel d‟Alzon / Butembo, cours inédit, 2017.
2. LESCOP Yves, « Sécurité Informatique», 2002
3. Laurent BLOCH et Christophe WOLFHUGEL, « Sécurité informatique. Principes et méthode à l'usage des DSI, RSSI et administrateurs », 2e édition, Eyrolles, Paris, 2009.
4. Greg Hoglund et James Butler; « Rootkits : infiltrations du noyau Windows », Paris, Campus Press, 2006 (1re éd. 2005), 338 p.
5. T. NORMAN Alan, 2016. Computer hacking beginners guide. Kindle Edition, 2016. ISBN 9781980390978.
6. JONES Jack, 2017. The complete beginners guide to computer hacking. Paperback, 2017. ISBN 9781548126476.
7. Pattinson, M., Jerram, C., Parsons, K., McCormac, A., & Butavicius, M. (2012). Why do some people manage phishing e-mails better than others ? Information Management and Computer Security, 20(1), 18–28.
8. Sheng, Steve, Brad Wardman, Gary Warner, Lorrie Cranor, Jason Hong et Zhang Chengsan (2009). An Empirical Analysis of Phishing Blacklists
9. Ponemon Institute (2015). The Cost of Phishing and Value of Employee Training. Coming of Age: The Evolving Field of Adventure Therapy Bandoroff and Newes, 2004
10. ANSSI, « Les principales menaces », 2021 https://www.ssi.gouv.fr/entreprise/principales-menaces/cybercriminalite/
11. Abdelhalim Zaidi. Recherche et détection des patterns d'attaques dans les réseaux IP _a hauts débits. Réseaux et télécommunications [cs.NI]. Université d'Evry-Val d'Essonne, 2011.
12. Nathalie Dagorn. Détection et prévention d'intrusion : présentation et limites. [Rapport de recherche], Université de Nancy1, France,2006
13. SLIMANI Ahmed, Application des systèmes immunitaires artificiels à la détection d'intrusion, USTO-MB : 2011
14. David Powell et Robert Stroud: Conceptual Model and Architecture of MAFTIA. Technical Report Series-University of Newcastle Upon Tyne Computing Science, 2003.
15. J. Anderson. Computer security threat monitoring and surveillance. Technical Report 56, Box 40 Fort Washington, pa. 19034, February 26, 1980.
16. D.E. Denning. An intrusion-detection model. IEEE Trans. Softw. Eng, Piscataway, NJ, USA, 13(2):222-232, 1987.
17. P. Biondi. Architecture expérimentale pour la détection d'intrusions dans un système informatique. PhD thesis, ENST Bretagne, 2001.
18. P. Lespérance. Détection des variations d'attaques à l'aide d'une logique temporelle. Master's thesis, Université Laval, 2006.
19. Cédric Michel, Langage de description d'attaques pour la détection d'intrusions par corrélation d'événements ou d'alertes en environnement réseau hétérogène, Université de Rennes 1,2003

20. Yousef Farhaoui, « Evaluation des systèmes de détection et de prévention des intrusions et la conception d'un BIDS », thèse de doctorat, Université Ibn Zohr, 2012
21. D. Burger Meister et J. Krier, « Les systèmes de détection d'intrusion,» 22 Juillet 2006. [En ligne]. Available: http://dbprog.developpez.com.
22. G. Arcas, S. Kadhi, J. Luiggi et F. Debieve, Réunion du groupe SUR/OSSIR : Bro, un NIDS pas comme les autres, 2007.
23. Une source pour la traduction en apprentissage profond Yoshua Bengio, « Introduction aux algorithmes d'apprentissage profonds », sur Université de Montréal.
24. J. Zhou et O. G. Troyanskaya (2015), « Predicting effects of noncoding variants with deep learning-based sequence model », Nature Methods, 12(10), 931-934.
25. Gunadiz Safia, Algorithmes d'intelligence artificielle pour la classification d‟attaques réseaux a partir de données TCP, université de boumerdes,2011
26. Nicolas Monmarché, Algorithmes de fourmis artificielles : applications `a la classification et `a l‟optimisation, 2004.
27. E. Lebarbier, T. Mary-Huard, Classification Non Supervise -AgroParisTech
28. YOUSSEF FATAICHA, RECHERCHE D'INFORMATION DANS LES IMAGES DE DOCUMENTS, MONTRÉAL, LE 27 DÉCEMBRE 2005.
29. Ridha GHAYOULA, Contribution à l'Optimisation de la Synthèse des Antennes Intelligentes par les Réseaux de Neurones,2008]
30. Patrice Wira, Réseaux de neurones artificiels : Architecture et applications
31. Claude Touzet, LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME, 2016
32. OTMANI Imene, L‟ANALYSE DISCRIMINANTE ET LE PERCEPTRON MULTICOUCHE PRESENTE PAR:2011
33. Maria Güell i Pons, Algorithmes d'optimisation, université de lausanne 2009.
34. MAHADOUI RAFIK, DIAGNOSTIC INDUSTRIEL PAR NEURO-FLOU-APPLICATION A UN SYSTEME DE PRODUCTION,2008]
35. Philippe Jauffret, Introduction aux réseaux de neurones ,2002
36. Thanh Ha Dang, 2007, P103 : Mesures de discrimination et leurs applications en apprentissage inductif, l‟Université Paris 6
37. Mr. Mint. Machine Learning made easy. Disponible sur :< https://mrmint.fr/introduction-k-nearest-neighbors> (11 juillet 2019)
38. G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," Neural Computation, vol. 18, July 2006, pp. 1527-1554, doi10.1162/neco.2006.18.7.1527.
39. C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," Proc. 19th Annual Computer Security Applications Conference, Dec. 2003, pp. 14-23, doi10.1109/CSAC.2003.1254306
40. J. Yang, J. Deng, S. Li, and Y. Hao, "Improved traffic detection with support vector machine based on restricted Boltzmann machine," Soft Computing, vol. 19, Dec. 2015, pp. 1-12, doi10.1007/s00500-015-1994-9
41. C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," IEEE Proc. 15th Annual Computer Security Applications Conf., pp. 371- 377, 1999.