République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed El Bachir El Ibrahimi B.B.A.
Faculté de Mathématique et Informatique

Département d'informatique

Mémoire

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

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : Réseau et Multimédia

Thème

Les systèmes de détections d'intrusion basés sur L'ensemble machine learning

Présenté par :

- BAYMOUT Mohamed Tayeb
- BENMERIEM Abdelouahab

Devant le jury :

Président : Dr. Naili Makhlouf

Examinateur : Dr. Zaouache Djaafar

Encadrant : Dr. Maza Sofiane

Année universitaire : 2023/2024

Remerciment

En premier lieu, nous remercions Dieu le très haut qui nous a donné le courage et la volonté de réaliser ce modeste travail.

Nous tenons à saisir cette occasion et adresser nos profonds remerciements et nos profondes reconnaissances à toutes personnes qui nous ont aidé de près ou de loin dans la réalisation de ce mémoire.

Nous remercions M. Maza Sofiane, en tant que Directeur de mémoire, pour ses précieux conseils, sa disponibilité, sa compréhension, sa gentillesse, les encouragements et son orientation tout au long de notre recherche.

Nos remerciements s'étendent également à M. Atia Abdelouahab, chef de département d'Informatique.

Un grand merci à tous nos enseignants pour toutes les connaissances qu'ils nous ont inculquées tout au long des cinq années.

Enfin nous exprimons notre profonde reconnaissance à tous responsables de l'université de Bordj Bou Arreridj qui ont contribué à notre formation.

Dédicace

Je dédie ce modeste travail

A tous ceux qui me connaissent, en particulier,

A ma mère et à la mémoire de mon père.

A mes frères et sœurs.

A tous mes amis et collègues.

Sans oublier tous les professeurs qui ont contribué

À ma formation de l'enseignement primaire, moyenne et

Secondaire jusqu'à l'enseignement supérieur.

Abdelouahab

Dédicace

Je dédie ce modeste travail

A tous ceux qui me connaissent, en particulier,

A mon père et ma mère.

A mes frères et ma sœur.

A tous mes amis et collègues.

Sans oublier tous les professeurs qui ont contribué

À ma formation de l'enseignement primaire, moyenne et

Secondaire jusqu'à l'enseignement supérieur.

Tayeb

Résumé

Ce travail explore les diverses techniques employées dans les systèmes de détection d'intrusion, en mettant l'accent sur l'utilisation du Machine Learning. Ensuite, trois méthodes d'ensemble en apprentissage automatique : Bagging, Boosting et Voting sont présentées.

Ces méthodes visent à améliorer l'efficacité des modèles en fusionnant plusieurs modèles individuels. Enfin, une comparaison basée sur le taux de précision est établie entre ces trois méthodes.

Mots-clés: systèmes de détection d'intrusion, Machine Learning, Bagging, Boosting, Voting.

Abstract

This paper examines the different techniques employed in intrusion detection systems, with a focus on Machine Learning. Following this, three ensemble methods in machine learning: Bagging, Boosting, and Voting are introduced.

These methods aim to enhance model efficiency by merging several individual models. Finally, a comparison based on accuracy rate is established among these three methods.

Keywords: Intrusion Detection Systems, Machine Learning, Bagging, Boosting, Voting.

Sommaire

Chapitre 02 : Système de détection d'intrusion

Chapitre 03 : Conception et Réalisation

Liste des Figures

Liste des Tableaux

Introduction Générale

Les systèmes informatiques et les réseaux sont actuellement utilisés dans divers domaines, tels que les écoles, les institutions financières, les compagnies d'assurance et le secteur militaire, et ils jouent un rôle essentiel dans le fonctionnement des organisations. Ces systèmes gèrent une technologie très intéressante qui peut être vulnérable aux attaques exploitant les failles du réseau. La nécessité de détecter les activités malveillantes est devenue cruciale car les mesures de prévention se révèlent souvent insuffisantes, ce qui a conduit à la mise en place de systèmes de détection d'intrusions.

Une intrusion se réfère à toute tentative visant à contourner les mesures de sécurité mises en place sur une machine ou un réseau, ou à porter atteinte à l'intégrité, à la confidentialité ou à la disponibilité du réseau. Ces tentatives d'intrusion peuvent aller d'innocentes à extrêmement dangereuses, entraînant des dommages potentiels pour la société.

Bien que le domaine de la détection des intrusions soit relativement nouveau, il se développe rapidement. Jusqu'à présent, il y an environ cent systèmes de détection d'intrusions (IDS), qu'ils soient du domaine public ou du secteur commercial. Compte tenu de l'augmentation constante du nombre et de la gravité des attaques réseau au fil des années, ces systèmes de surveillance du réseau sont presque nécessaires.

Notre recherche se concentre sur les différentes méthodes utilisées pour détecter les intrusions. En général, ces techniques incluent l'extraction de données, l'apprentissage automatique et les approches statiques et examinent les techniques d'agrégation en apprentissage automatique visant à améliorer les performances des modèles en combinant plusieurs modèles individuels.

Organisation de notre travail :

Chapitre 1 : Ce chapitre s'articule autour de la machine learning, en présentant une étape sur les technologies d’optimisation.

Chapitre 2 : Ce chapitre est traité des différentes facettes des systèmes de détection d'intrusion.

Chapitre 3 : Ce chapitre se concentre sur les résultats de l'expérience.

Enfin, une conclusion globale résume les éléments clés de cette étude et offre des perspectives pour l'avenir.

Chapitre I

Ensemble machine Learning

1.1. Introduction

Dans le domaine de l'apprentissage automatique, l'apprentissage d'ensemble utilise plusieurs algorithmes d'apprentissage afin d'obtenir de meilleurs résultats. Dans les contextes de classification, nous utilisons plusieurs classificateurs et les combinons, ce qui permet d'obtenir de meilleurs résultats.

Il est plus facile de penser de cette manière. Lorsque vous rencontrez un problème, il est souvent préférable de demander à vos amis de vous aider plutôt que de travailler seul. L'intelligence collective permet souvent d'obtenir de meilleures performances. Voyons comment l'intelligence collective fonctionne dans les algorithmes de Machine Learning.

1.2. Machine learning

Dans la machine learning, un modèle est créé par un algorithme basé sur des données. Ce modèle représente ou approxime les données dans le but de les généraliser autant que possible. Il nous aide à mieux comprendre les données déjà disponibles et à faire des prédictions sur des données inconnues en fonction des données d'entrée. L'apprentissage automatique permet de prédire des valeurs financières, d'identifier des failles de sécurité informatique, de confirmer l'identité d'un utilisateur, de personnaliser les moteurs de recherche en fonction des utilisateurs, d'identifier le vol de données, pour lancer des logiciels antivirus et de procéder à une cryptanalyse [1]. Les étapes de l'implémentation du machine learning sont les suivantes :

1. Collecter les informations requises.
2. Préparation et élimination des données.
3. Étape d'apprentissage.
4. Étape de vérification.
5. Étape de réalisation.

1.2.1 **Collecter les données nécessaires**

Tout d'abord, il est nécessaire d'obtenir une quantité adéquate d'informations qui reflètent le problème à résoudre. Il est parfois difficile. Certains éléments sont plus onéreux que d'autres pour les obtenir. Prenons l'exemple d'un paquet réseau qui est chiffré., les informations de l'en-tête sont plus faciles à obtenir que la partie des données. Les données obtenues doivent d'abord être nettoyées ou prétraitées, en les traduisant et en réduisant leur portée à ce qui est purement pertinent. Cette étape a pour but d'augmenter la précision du modèle, d'en minimiser la taille et d'optimiser le temps d'apprentissage et d'exécution.

Ce nettoyage est souvent très difficile à mettre en œuvre et nécessite une connaissance approfondie des données à traiter. Par conséquent, des méthodes automatiques telles que les techniques de sélection de feature ont été étudiées [1].

1.2.2 **Construire le modèle**

La quantité et la qualité des données ont une incidence directe sur l'efficacité du modèle final. Pour que les machines deviennent plus intelligentes et puissent prendre des décisions par elles-mêmes, elles doivent traiter de grandes quantités de données. Les résultats obtenus seront plus précis et mieux adaptés aux exigences de l'organisation si ces informations sont plus nombreuses et plus fiables. Il est donc essentiel de collecter les données en fonction des objectifs définis à l'étape précédente. Vous recueillez des informations provenant de plusieurs sources ? Regroupez-les en combinant des bases de données disparates.

1.2.3 **Préparation et nettoyage des données**

La qualité des données est la condition première de la réussite d'un modèle d'apprentissage. Le prétraitement des données recueillies est donc essentiel pour en maximiser le potentiel. Les données qui ne sont pas correctement marquées, inaccessibles, dupliquées, incohérentes ou inutiles… De nombreux problèmes au sein de votre entrepôt de données peuvent survenir à la suite de l'intégration des données. Afin de rendre les données brutes plus comparables, la deuxième étape vise à les nettoyer et à les standardiser, voire à les enrichir en utilisant des données provenant d'autres sources. L'objectif est de permettre aux algorithmes d'utiliser et de maintenir ce type de données.

1.2.4 **Phase d'apprentissage**

La phase d'apprentissage du modèle commence lorsqu'un certain sous-ensemble de l'ensemble des données nettoyées est utilisé comme donnée d'entraînement. Afin de réduire l'erreur entre les valeurs des données réelles et les prédictions, le modèle modifie ses paramètres au cours de cette étape critique. Les algorithmes d'apprentissage se présentent sous diverses formes, avec des méthodes supervisées et non supervisées. Le modèle peut réduire la différence entre les sorties prévues et les sorties réelles en utilisant des algorithmes supervisés qui obtiennent à la fois les entrées et les sorties souhaitées. Lorsqu'il existe une relation établie entre les entrées et les sorties d'une tâche de reconnaissance, ces techniques sont fréquemment employées. À l'inverse, les algorithmes non supervisés n'ont pas de résultats prédéterminés, ils reçoivent simplement des données d'entrée et doivent trouver des structures ou des modèles dans les données. Pour éviter le surapprentissage (situation dans laquelle le modèle s'habitue trop à l'ensemble de la formation et

devient moins capable de se généraliser à de nouvelles données), il est important de valider le modèle après sa formation initiale [2].

1.2.5 Phase de validation

Dans cette phase, des tests et des validations sont effectués sur le modèle et ses paramètres en fonction de critères prédéterminés. La généralisation des données recueillies pendant la période d'apprentissage permet d'obtenir le modèle optimal. Nous disposons pour cela de deux séries d'exemples : l'une pour les tests et l'autre pour l'apprentissage [2].

1.2.6 Phase d'exécution

Le modèle machine learning est prêt à être déployé dans un environnement de production après une formation réussie et une validation des performances. Il s'agit de le préparer à des utilisations réelles afin qu'il puisse produire des prédictions en cours de route sur la base d'informations fraîches. Par exemple, si l'objectif du modèle est d'identifier les irrégularités dans le trafic réseau TCP/IP, il peut être utilisé pour identifier les activités suspectes dans les systèmes de surveillance du trafic réseau en temps réel tel que le pare-feu et les systèmes de détection d'intrusion.

Il est impératif d'observer en permanence les performances du modèle dans le cadre du TCP/IP. Cela implique de recueillir des données concernant à la fois les résultats réels observés et les prédictions du modèle, telles que les anomalies observées dans le trafic du réseau. Le suivi de ces mesures vous aidera à repérer les performances qui s'écartent des prévisions ou qui montrent des signes de déclin. Le modèle peut nécessiter de nouvelles mises à jour des données ou un examen des performances, par exemple, s'il commence à signaler de fausses alertes ou à ignorer des activités suspectes.

Une fois qu'un modèle a été sélectionné et évalué, il est essentiel de mesurer ses résultats. Pour ce faire, il est d'usage de comparer plusieurs modèles en utilisant le même ensemble de données. Cette étape est importante pour déterminer les performances comparatives de chaque modèle. Des mesures telles que la proportion de vrais positifs, de vrais négatifs, de faux positifs et de faux négatifs peut être utilisées pour évaluer les performances. Toutefois, ces mesures ne tiennent pas compte de facteurs tels que le temps d'exécution, le temps d'apprentissage ou la taille du modèle. La courbe caractéristique d'exploitation du récepteur (ROC) est une méthode couramment utilisée pour évaluer les performances d'un modèle [1].

L'aire sous la courbe ROC offre un indicateur numérique de la performance du modèle et montre le rapport entre les vrais positifs et les faux positifs.

1.2.7 Types de modèle

Différents modèles sont disponibles. En plus du fait qu'ils soient ou non supervisés, ils peuvent également être distingués en fonction de leurs caractéristiques de classification ou de régression. Alors que ces dernières prédisent des valeurs pour chaque élément de données, les premières classifient les données. Comme la classification semble être le principal problème des systèmes de détection d'intrusion (IDS), voici une liste partielle des modèles qui pourraient être utilisés pour mettre en œuvre les IDS.

1.3. Algorithmes d'ensemble de la machine learning

Les techniques Bagging, Boosting, et Voting sont les techniques de base dans lesquelles s'inscrit l'apprentissage collectif traditionnel. Le voting et le baggin utilisent des techniques d'échantillonnage de données différentes, même s'ils utilisent tous deux de nombreux algorithmes pour parvenir à un résultat final. En revanche, le boosting combine les prédictions de différents modèles de base afin d'améliorer la résilience globale du modèle [3].

Le bagging, également appelé Bootstrap Aggregating, améliore les performances des modèles dans les tâches de regroupement, de régression et de classification. Le soft voting, qui est préféré au hard voting, prend des décisions en faisant la moyenne des résultats de plusieurs algorithmes différents. Cette approche globale de l'apprentissage d'ensemble permet d'améliorer la précision et la robustesse des modèles dans toute une série d'applications d'apprentissage automatique [4].

L'un des objectifs majeurs de la mise en place d'une solution commerciale consiste à diminuer la durée d'apprentissage. Propose une méthode pour réduire la durée du processus d'apprentissage et la taille de la structure. Une efficacité encore plus grande peut être obtenue si nous établissons le cadre de chaque approche employée dans l'IDS.

1.3.1 Bagging

Bagging ou "Bootstrap L'agrégation", est une technique utilisée pour accroître la stabilité et la précision d'un modèle. À l'aide d'un échantillonnage aléatoire avec remplacement, plusieurs sous-ensembles de données de formation sont créés et un modèle est ensuite formé sur chaque sous-ensemble. La réduction de la variance et de l'ajustement excessif d'un modèle est l'objectif du bagging.

La moyenne des prédictions de chaque modèle (pour la régression) ou le vote majoritaire (pour la classification) est utilisé pour fusionner les modèles individuels. En ajoutant de la variabilité au processus de formation, le bagging permet d'éviter l'ajustement excessif et de réduire la variance du modèle [5].

Le bagging est important car il peut augmenter la stabilité et la précision d'un modèle, en particulier lorsque l'on travaille avec des échantillons bruyants ou des données de haute dimension.

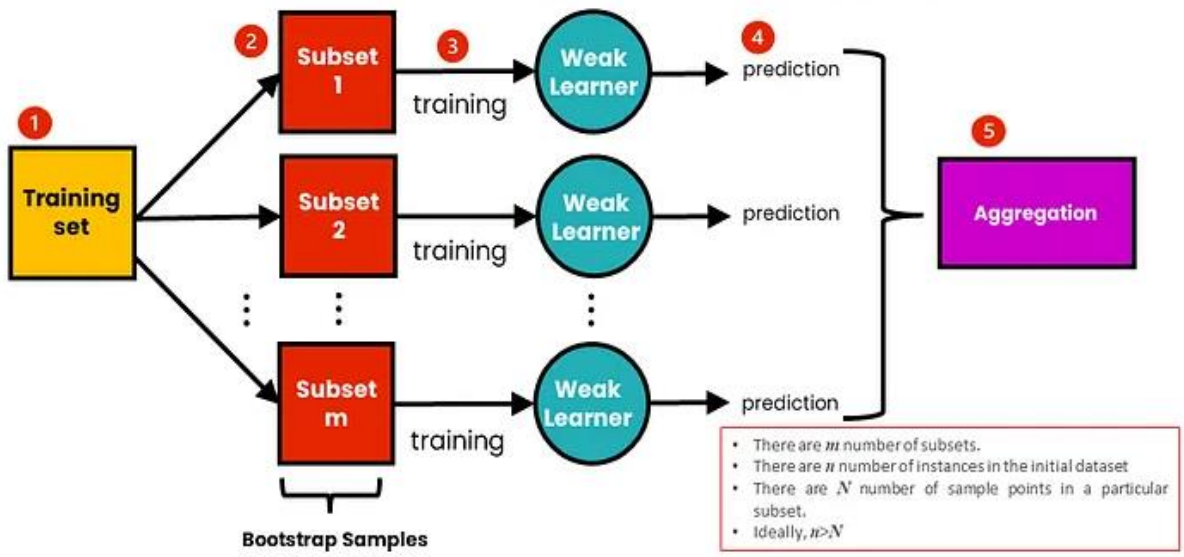

FIGURE 1.1 : Le processus de Baggin.[7].

1.3.1.1 Agrégation Bootstrap

Leo Breiman, statisticien [6], a proposé cette technique d'agrégation pour la première fois en 1996. À l'aide d'un échantillonnage aléatoire avec remplacement, plusieurs sous-ensembles de données d'apprentissage sont créés et un modèle est ensuite formé sur chaque sous-ensemble. Enfin, le vote majoritaire (pour la classification) ou la moyenne des prédictions (pour la régression) est utilisé pour intégrer les différents modèles.

Le mot "bootstrap" décrit un ensemble de techniques permettant d'extrapoler les conclusions statistiques d'un échantillon initial à d'autres échantillons.

L'utilisation de plusieurs ensembles de données rééchantillonnés à partir de l'ensemble des données observées par un tirage aléatoire avec actualisation est la raison d'être de la technique du bootstrap.

Une technique de rééchantillonnage statistique connue sous le nom de « bootstrap » est utilisée depuis longtemps pour estimer des quantités ou des attributs statistiques. Pour s'assurer que chaque classificateur élémentaire de l'ensemble est formé sur un ensemble de formation distinct, chacun est formé sur un échantillon bootstrap. La combinaison de ces classificateurs permet de créer un prédicteur plus puissant [5].

Algorithme 1 : La méthode bagging [8].

Entrée :

B^a : *base d'apprentissage*

T : taille de la base d'apprentissage X forme à reconnaitre

C : classifieur

Début

Pour *j de 1 à C* ***faireDébut***

Générer la sous-base b_j *à partir de* B^a

Construire le classifieur e_j *(x) en utilisant la base* b_j

Fin pour

Combiner les C classifieurs construits par le vote à la majorité afin d'obtenir la décision finale de x.

Fin

1.3.1.2 Types de Baggin

La machine learning utilise une variété de techniques de mise en sac. En voici quelques exemples:

- **Utilisation de Paste**

  Dans cette variante du bagging, les données d'apprentissage sont échantillonnées aléatoirement sans remplacement pour créer de nombreux sous-ensembles, qui sont ensuite utilisés pour former un modèle. Lorsque l'ensemble des données de formation est important, le collage peut être utile, car l'échantillonnage avec remplacement serait coûteux en termes de calcul [11].

- **Méthode du sous-espace aléatoire**

  Au lieu d'utiliser les données d'apprentissage, cette variante du bagging génère plusieurs sous-ensembles de caractéristiques. En utilisant uniquement les caractéristiques choisies, un modèle est formé sur l'ensemble des données d'apprentissage de chaque sous-ensemble. Parce

qu'elle permet aux modèles de se concentrer sur les aspects les plus significatifs, cette technique peut s'avérer utile lorsque l'ensemble des données contient un grand nombre de caractéristiques sans importance [9].

- **Méthode des patchs aléatoires**

Cette variante du bagging crée plusieurs sous-ensembles de caractéristiques et de données d'apprentissage. Ensuite, les caractéristiques et les échantillons choisis dans chaque sous-ensemble sont utilisés pour former un modèle. Cette stratégie peut s'avérer utile lorsque l'ensemble des données contient à la fois des caractéristiques et des données non pertinentes [9].

1.3.2 Boosting

Le boosting est une technique d'apprentissage automatique supervisé qui crée un modèle d'ensemble fort en agrégeant les prédictions de plusieurs modèles faibles ou modèles de base. Le boosting met l'accent sur l'entraînement itératif des modèles de base de manière à mettre en évidence les données qui ont été mal classées lors des itérations précédentes, contrairement aux méthodes d'ensemble traditionnelles telles que le bagging ou le moyennage. Le fait de donner la priorité aux échantillons qui ont été mal classés lors des itérations précédentes a pour but d'aider le modèle à apprendre de ses erreurs et à s'améliorer au fil du temps.

En utilisant une distribution d'exemples de formation, Adaboost sélectionne un classificateur faible de manière itérative. Un poids est attribué en fonction de la difficulté du classificateur actuel pour chaque exemple. Le pseudo-code de cet algorithme est expliqué en détail ci-dessous en raison de sa simplicité et de son efficacité. Le terme Adaboost est un acronyme pour adaptive boosting. Grâce à l'itération adaptative, un vote majoritaire est construit. Une fois l'algorithme terminé, une valeur (AT) calculée dans la boucle de l'algorithme est appliquée à chaque classificateur HT [11].

Ils utilisent un vote majoritaire pondéré pour classer un nouveau exemple.

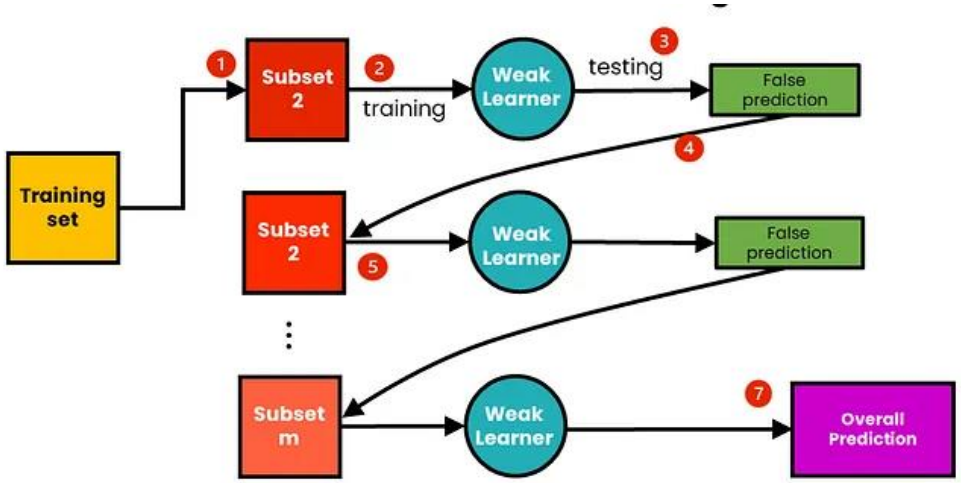

FIGURE 1.2 : Le processus de Boosting [7].

Algorithme 2 : Approche d'Adaboost [10].

Initialisation D1(i) = 1/m, i = 1, . . ., m	// Poids initiaux
Pour t = 1, . . ., T : ht = L (LS, Dt)	// Apprenant avec un poids Dt faible
εbt = P i:h (xi)6=yi Dt(i)	// Erreur de ht pondérée
αt = 1 2 ln 1−εbt εbt	// Coefficient de ht dans l'ensemble
Pour i = 1, . . ., m :	// Renforcement du poids des exemples
Si ht(xi) 6= yi :	// mal classés
De t+1(i) = Dt(i) 1−εbt bt	
Sinon, De t+1(i) = Dt(i)	
Fin	
Dt+1(i) = De t+1(i)/ (P i De t+1(i))	// Normalisation
Fin	
fT (x) = PT t=1 αtht(x)	// Vote pondéré
HT (x) = sign (fT (x))	// Classifieur final

1.3.2.1 Les types de boosting

Examinons les nombreuses techniques de renforcement fréquemment utilisées dans l'apprentissage automatique :

- **Adaptatif Boosting :** Cet algorithme est l'un des plus utilisés. Afin de construire un classificateur fort, AdaBoost génère un certain nombre de classificateurs faibles.

AdaBoost entraîne ensuite un nouveau classificateur sur l'ensemble de données révisé en donnant aux cas mal classés à chaque itération des poids plus importants [11].

- **Gradient Boosting :** L'une des approches d'amplification les plus appréciées est l'amplification par gradient. Dans cette approche, chaque prédiction corrige l'erreur de celle qui la précède. Contrairement à Adaboost, chaque prédicteur est formé en utilisant les erreurs résiduelles de son prédécesseur comme étiquettes plutôt que d'ajuster les poids des cas de formation [11].
- **XGBoost :** Un algorithme est développé pour créer des arbres de décision de manière séquentielle. En termes de XGBoost, les poids sont importants. Après l'attribution de poids à chaque variable indépendante, l'arbre de décision est utilisé pour prévoir les résultats. Le deuxième arbre de décision reçoit les facteurs que le premier arbre de décision a mal prédits et augmente leur poids. Ensuite, un modèle puissant et plus précis est produit en combinant ces classificateurs et prédicteurs distincts. Il est applicable aux problèmes de prédiction définis par l'utilisateur, à la régression, à la classification et au classement [11].

1.3.3 Voting

Un classificateur de vote est un type de modèle d'apprentissage automatique qui apprend à partir d'une large collection de modèles et prévoit un résultat (classe) en fonction du modèle qui a le plus de chances d'être sélectionné comme résultat.

Supposons que vous ayez formé quelques classificateurs, chacun atteignant environ 80% de précision. Vous pouvez avoir un classificateur de régression logistique, un classificateur SVM, un classificateur de forêt aléatoire, un classificateur de k plus proches voisins, et peut-être quelques autres [12].

Un moyen très simple de créer un classificateur encore meilleur est d'agréger les prédictions de chaque classificateur et de prédire la classe qui obtient le plus de votes. Ce classificateur de vote majoritaire est appelé un classificateur de vote dur.

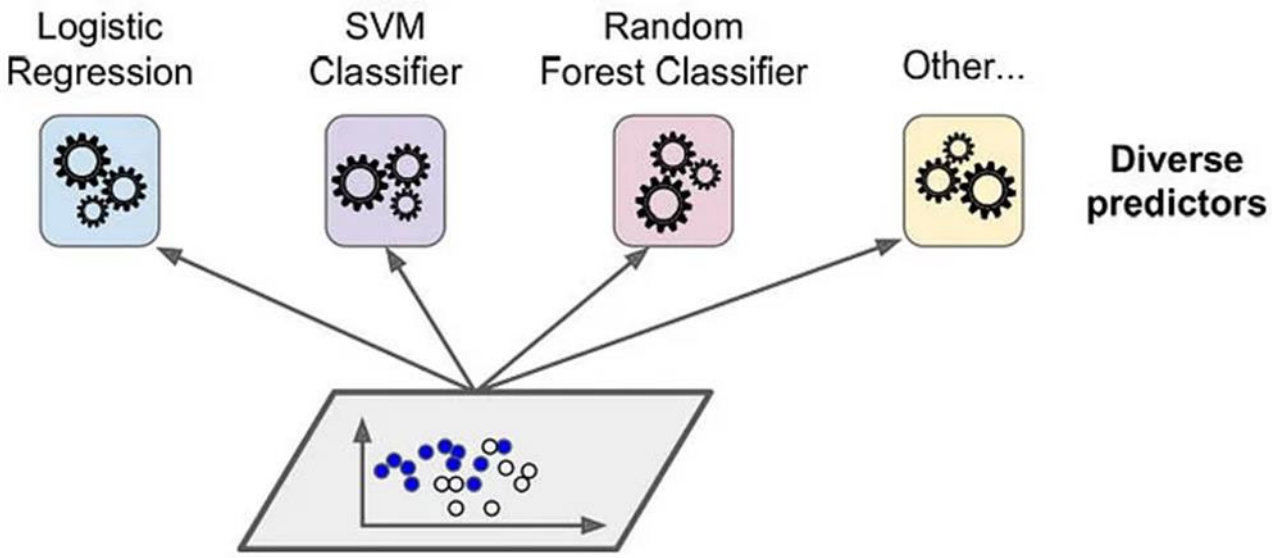

FIGURE 1.3 : Entraînement de classificateurs diversifiés [13].

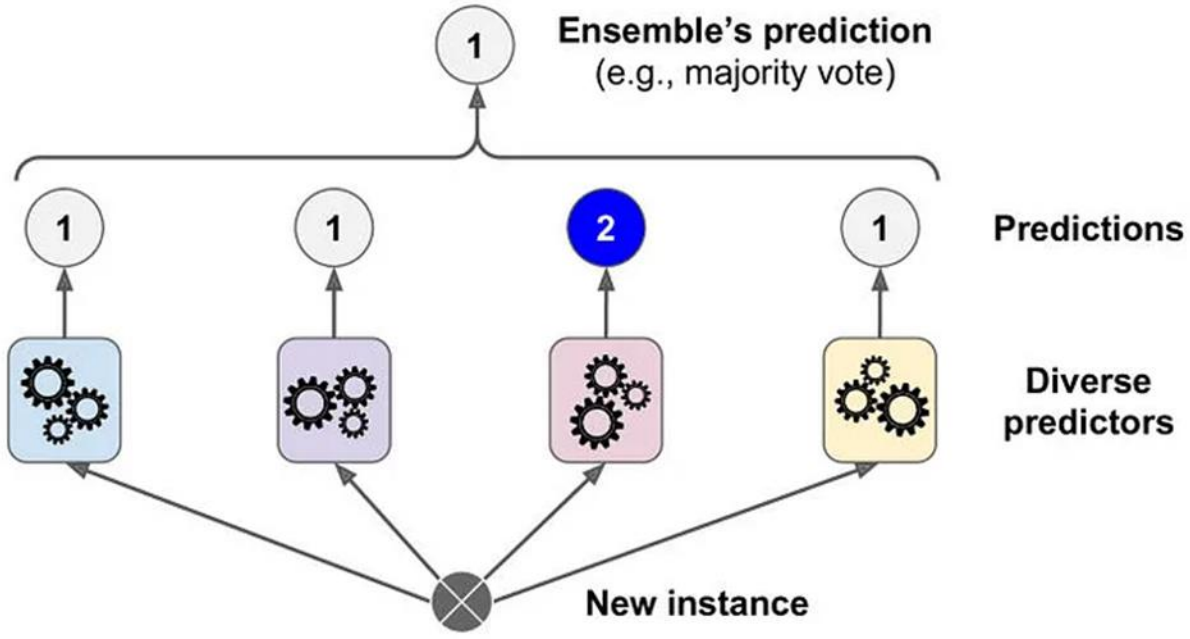

FIGURE 1.4 : Prédictions du classificateur de voting [13].

De manière assez surprenante, ce classificateur de vote atteint souvent une précision plus élevée que le meilleur classificateur de l'ensemble. En fait, même si chaque classificateur est un apprenant faible (ce qui signifie qu'il fait seulement légèrement mieux que le hasard), l'ensemble peut toujours être un apprenant fort (atteignant une haute précision), à condition qu'il y ait un nombre suffisant d'apprenants faibles et qu'ils soient suffisamment diversifiés.

Comment est-ce possible ? L'analogie suivante peut aider à éclairer ce mystère. Supposons que vous ayez une pièce légèrement biaisée qui a 51% de chances de tomber sur pile et 49% de chances de tomber sur face. Si vous la lancez 1 000 fois, vous obtiendrez généralement plus ou moins 510 piles et 490 faces, et donc une majorité de piles. Si vous faites le calcul, vous constaterez que la probabilité d'obtenir une majorité de piles après 1 000 lancers est proche de 75%. Plus vous lancez la pièce, plus la probabilité est élevée (par exemple, avec 10 000 lancers, la probabilité dépasse 97%). Cela est dû à la loi des grands nombres : plus vous lancez la pièce, plus le ratio de faces se rapproche de la probabilité de faces (51%). La figure 7-3 montre 10 séries de lancers de pièces biaisées. Vous pouvez voir qu'à mesure que le nombre de lancers augmente, le ratio de faces se rapproche de 51%. Finalement, les 10 séries finissent toutes si proches de 51% qu'elles sont constamment au-dessus de 50% [13].

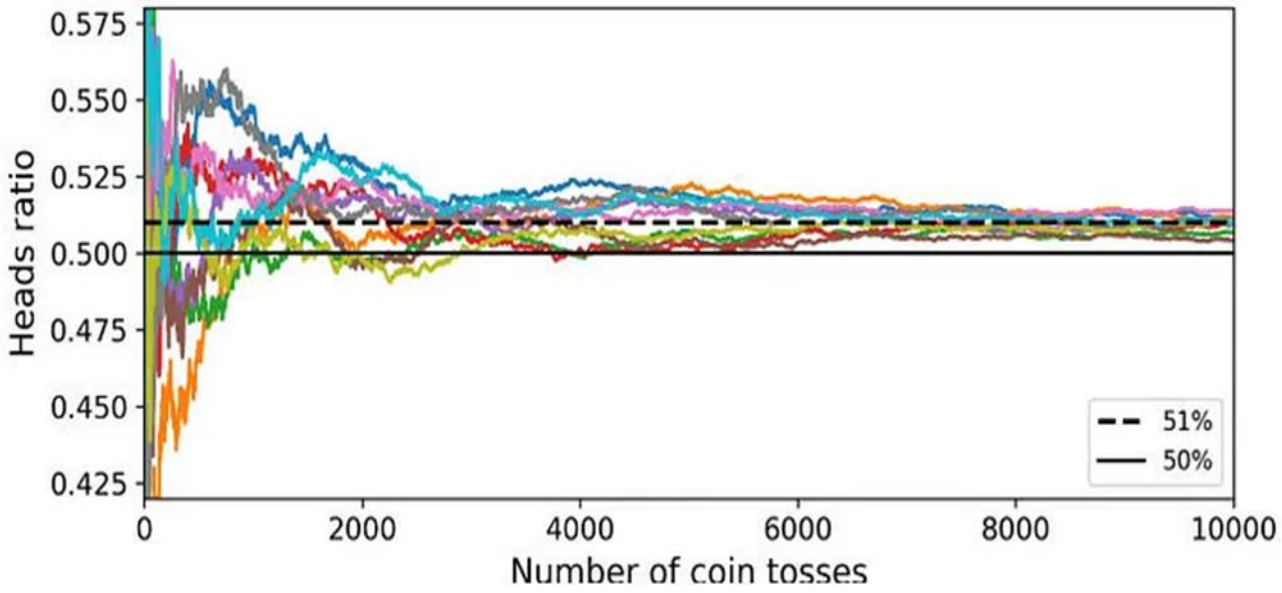

FIGURE 1.5: La loi des grands nombres [13].

De même, supposez que vous construisez un ensemble contenant 1 000 classificateurs qui sont individuellement corrects seulement 51 % du temps (à peine mieux que le hasard). Si vous prédisez la classe majoritaire votée, vous pouvez espérer jusqu'à 75 % de précision ! Cependant, cela n'est vrai que si tous les classificateurs sont parfaitement indépendants, commettant des erreurs non corrélées, ce qui n'est clairement pas le cas puisqu'ils sont entraînés sur les mêmes données. Ils

sont susceptibles de commettre les mêmes types d'erreurs, il y aura donc de nombreux votes majoritaires pour la mauvaise classe, réduisant la précision de l'ensemble. Les méthodes d'ensemble fonctionnent mieux lorsque les prédicteurs sont aussi indépendants les uns des autres que possible. Une façon d'obtenir des classificateurs divers est de les entraîner en utilisant des algorithmes très différents. Cela augmente la probabilité qu'ils commettent des erreurs de types très différents, améliorant ainsi la précision de l'ensemble [13].

Si tous les classificateurs sont capables d'estimer les probabilités de classe (c'est-à-dire, ils ont une méthode predict_proba ()), alors vous pouvez dire à Scikit-Learn de prédire la classe avec la probabilité de classe la plus élevée, moyennée sur tous les classificateurs individuels. Cela s'appelle donc le vote. Il atteint souvent de meilleures performances que le vote dur car il donne plus de poids aux votes très confiants. Tout ce que vous avez à faire est de remplacer voting="hard" par voting="soft" et de vous assurer que tous les classificateurs peuvent estimer les probabilités de classe. Ce n'est pas le cas de la classe SVC par défaut, donc vous devez définir son hyperparamètre de probabilité sur True (cela fera en sorte que la classe SVC utilise la validation croisée pour estimer les probabilités de classe, ralentissant l'apprentissage, et cela ajoutera une méthode predict_proba ()). Si vous modifiez le code précédent pour utiliser le vote mou, vous verrez que le classificateur de vote atteint une précision de plus de 91,2 % [13].

1.3.4 Étude comparative entre Bagging, Boosting et Voting

1.3.4.1 boosting vs bagging

Les techniques d'ensemble avancées dans le domaine de l'apprentissage automatique comprennent le boosting et le bagging. En combinant plusieurs modèles de base avec des apprenants faibles, l'approche d'ensemble est une stratégie d'apprentissage automatique qui produit le meilleur modèle de prédiction. Pour favoriser une prise de décision précise et améliorée, les techniques d'ensemble de l'apprentissage automatique intègrent des informations provenant de plusieurs apprenants faibles [14].

Une étude comparative réalisée par Dietterich [15] montre qu'AdaBoost est plus efficace que le bagging lorsqu'il s'agit de données déséquilibrées et qu'il est moins dangereux lorsqu'il s'agit de sur apprentissage, car il vise à maximiser les votes pondérés. C'est ce que démontre cette théorie :

- L'erreur AdaBoost empirique sur l'échantillon d'apprentissage diminue de manière exponentielle avec le nombre d'itérations.
- L'erreur de généralisation diminue et continue de diminuer même après que l'erreur empirique a atteint son point le plus bas.

L'erreur de généralisation du boosting ne diminue pas une fois que l'erreur d'apprentissage atteint son minimum, au contraire du bagging, qui nécessite un grand nombre d'itérations avant que l'erreur de généralisation ne se stabilise. Étant donné que chaque classificateur doit être conservé pour prédire un nouvel échantillon, le bagging nécessite plus de mémoire que le boosting.

Bagging	**Boosting**
Combine plusieurs modèles qui ont été formés sur différents sous-ensembles de données.	Entraîner les modèles l'un après l'autre, en faisant attention aux erreurs produites par le modèle précédent.
En calculant la moyenne de chaque erreur individuelle du modèle, afin de réduire la variation.	Élimine les biais et la variance en corrigeant les classifications incorrectes du modèle précédent.
Forêt aléatoire	AdaBoost, Gradient Boosting Mechanism, XGBoost,

TABLE 1.1: Bagging vs Boosting

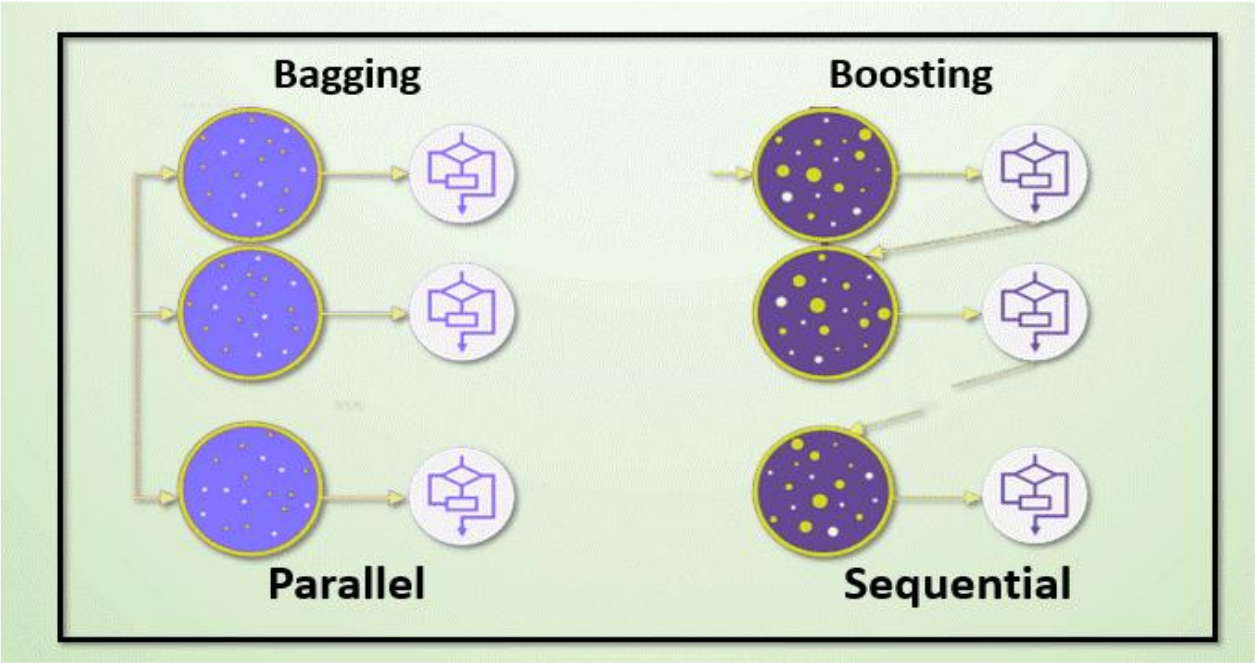

FIGURE 1.6: La difference entre Bagging et Boosting [16].

1.3.4.2 Voting vs Bagging

Comme on peut le voir, les algorithmes dans la méthode de vote sont entraînés avec le même ensemble de données, mais les algorithmes dans la méthode de bagging sont entraînés avec différents ensembles de données échantillonnés qui ont été bootstrapés. En d'autres termes, les algorithmes dans la méthode de bagging sont entraînés avec des ensembles de données qui ont été échantillonnés de manière aléatoire à partir de l'ensemble de données original avec remplacement. De plus, dans la méthode de bagging, les algorithmes sont identiques tandis que différents algorithmes sont combinés dans la méthode de vote. Le classificateur de forêt aléatoire appartient à la méthode de bagging car il est composé de plusieurs classificateurs d'arbres de décision.

Pour la méthode de vote, il existe deux méthodes pour effectuer le vote qui sont le vote dur et le vote doux. Le vote dur est équivalent au vote majoritaire, et le vote doux revient essentiellement à moyenner la sortie de plusieurs algorithmes. Le vote doux est généralement choisi comme méthode de vote à suivre. Le diagramme ci-dessous montre le mécanisme du vote doux [12].

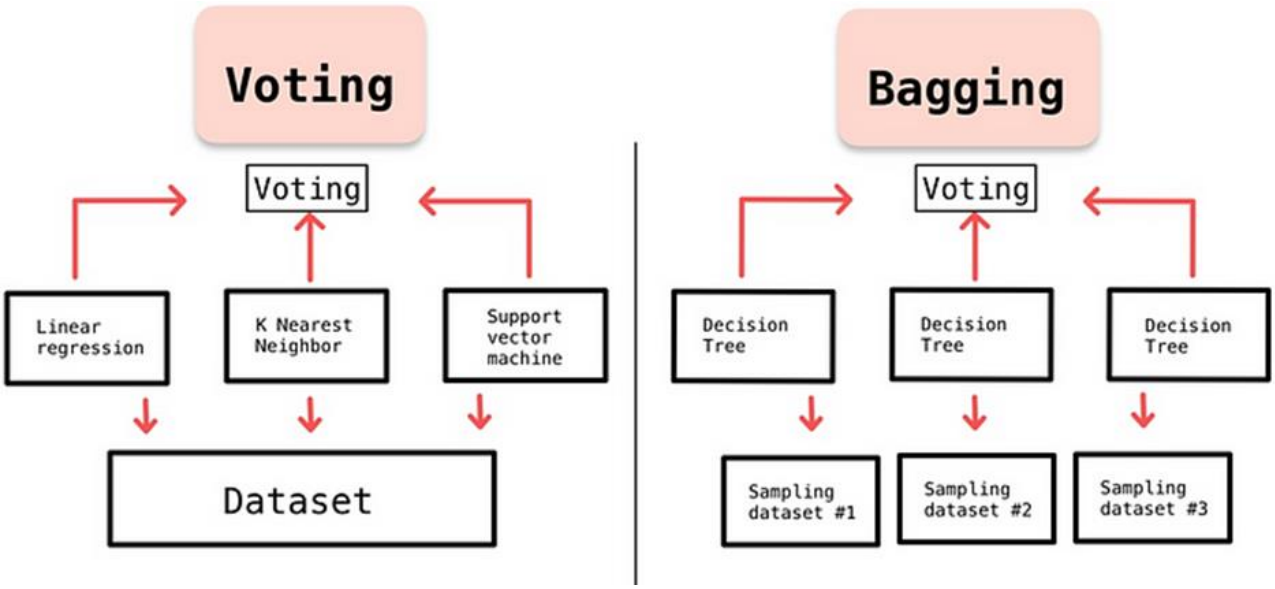

FIGURE 1.7 : Voting vs Bagging [12].

1.3.5 Avantages et inconvénients

Classifieur	Avantages	Inconvénients
Bagging	• La variance des modèles est réduite grâce à la mise en sac. • Cela améliore leur capacité de généralisation. • Les modèles qui utilisent le bagging sont plus résistants au bruit. • Une parallélisation efficace est possible pour des temps d'exécution optimaux. • Le bagging permet une large compatibilité des modèles. • Assure la stabilité des prédictions.	• Multiplicité des modèles créés et entraînés, entraînant une augmentation de la complexité du système. • Perte d'interprétabilité des prédictions du modèle agrégé, en particulier dans le cas de modèles de base compliqués. • Risque persistant de sur-apprentissage des données d'entraînement par les modèles de base. • Coûts de calcul plus élevés en raison de l'utilisation de différents modèles pour l'apprentissage sur différents ensembles de données.
Boosting	• Les performances du modèle s'améliorent à chaque itération. • Identifier et traiter les cas de mauvaise classification pour réduire les préjudices. • Flexibilité pour différents types de données et de modèles. • Moins de risques de sur-apprentissage grâce aux techniques de régularisation intégrées. • La capacité d'enregistrer des liens complexes entre diverses propriétés des données.	• Le Boosting est difficile à mettre en œuvre en temps réel en raison de la complexité accrue de l'algorithme. • La grande flexibilité de ces techniques entraîne de multiples paramètres qui affectent directement le comportement du modèle.

Voting	• Précision améliorée : Les classificateurs par vote offrent une meilleure précision en combinant les prédictions de plusieurs modèles. • Réduction du surajustement : Ils sont moins susceptibles de surajuster grâce à la moyenne des biais et à la réduction de la variance. • Gestion polyvalente des données : Adaptés à divers types de données, y compris catégorielles, numériques et textuelles. • Facilité de mise en œuvre : Relativement simple à mettre en place, notamment avec des bibliothèques comme Scikit-Learn. • Stabilité accrue : Moins sensibles aux variations dans les données d'entraînement, offrant des prédictions plus cohérentes.	• Dépendance aux modèles de base divers : Un classifieur par vote nécessite une diversité de modèles de base pour être efficace. Si les modèles sont trop similaires, cela peut limiter les améliorations de la précision prédictive. • Augmentation des ressources informatiques : Combinaison de prédictions de plusieurs modèles peut être plus coûteuse en termes de calcul, ce qui peut poser problème dans des environnements avec des ressources limitées. • Interprétabilité et explicabilité réduites : Il peut être plus difficile d'interpréter et d'expliquer les prédictions d'un classifieur par vote par rapport à un seul modèle, nécessitant ainsi plus d'efforts pour comprendre les contributions de chaque modèle.

TABLE 1.2 : Avantages et inconvénients [16] [12].

1.4. Conclusion

Dans ce chapitre, on aborde les principes fondamentaux de l'apprentissage automatique. Tout d'abord, la définition et les différents types ont été discutés afin de fournir une vision claire. Ensuite, les techniques d'ensemble en apprentissage automatique ont été expliqués en détail. Dans le prochain chapitre, nous allons exposer les diverses méthodes employées pour la détection des intrusions.

Chapitre II

Système de détection d'intrusion

2.1. Introduction

Un grand nombre d'entreprises bénéficient des progrès de la technologie Internet, ce qui peut contribuer à la croissance de l'industrie, mais il existe naturellement des risques de sécurité lorsque toutes les procédures sont transférées en ligne. Les L'attaques informatiques se sont multipliées récemment, ciblant en particulier les entreprises et les secteurs qui fournissent des services en ligne. Il s'agit d'un problème important de sécurité des données, car les organisations dépendent fortement de la sécurité des données pour protéger leurs ressources et les informations de leurs clients. Les entreprises doivent garantir la disponibilité, la confidentialité et l'intégrité des données, en d'autres termes, elles doivent gérer, stocker et préserver les données de manière appropriée afin d'empêcher tout accès illégal.

L'attaque informatique peut être évitée de plusieurs façons, notamment par l'utilisation de systèmes de détection d'intrusion (IDS), qui font partie de la sécurité des réseaux et qui protègent la sécurité des données et des informations en surveillant le trafic d'un paquet de données afin de détecter une intrusion. Un certain nombre d'études ont été menées sur les systèmes de détection d'intrusion basés sur les anomalies, qui utilisent une variété d'algorithmes, y compris des algorithmes d'apprentissage automatique et d'exploration de données.

2.2. La sécurité informatique

2.2.1. Définition

Le terme « sécurité informatique » désigne une vaste catégorie de tactiques, procédures, méthodes, produits et équipements utilisés pour protéger la disponibilité, la confidentialité et l'intégrité des ressources numériques et des données au sein d'une organisation.

Afin de prévenir, d'identifier et de neutraliser un large éventail de cybermenaces et de cyberattaques, une stratégie de sécurité informatique rigoureuse associe des ressources humaines à des technologies de pointe. Outre le réseau et ses nombreux composants, tels que les centres de données physiques ou en nuage, elle comprend également la protection de tous les points finaux, des logiciels et des systèmes matériels. [17]

2.2.2. La différence entre la Sécurité informatique et cybersécurité :

Il est également essentiel de faire la distinction entre la cybersécurité et la sécurité informatique. La cybersécurité est la défense d'une entreprise contre les intrusions malveillantes et les accès extérieurs.

En revanche, le champ d'application de la sécurité informatique est plus large. Elle comprend,

en particulier, toute fonction de protection contre les menaces en ligne et de maintien de la disponibilité, de la confidentialité et de l'intégrité des données. Il peut s'agir d'une défense contre des problèmes de sécurité courants et non malveillants, tels que des systèmes mal configurés ou du matériel défectueux. [17]

2.2.3. Les Techniques de la sécurité

- **La protection physique** : Pour garantir la disponibilité, la confidentialité et l'intégrité des informations, la sécurité physique est une composante essentielle de toutes les formes de sécurité. Une personne ayant accès au système informatique d'une entreprise a la possibilité de le compromettre, voire de le détruire. [18]
- **Chiffrement :** Les algorithmes de chiffrement sont utilisés pour s'assurer que les données cryptées peuvent être décryptées en texte clair à l'aide de la clé de décryptage, un algorithme utilise la clé de cryptage pour modifier les données de manière prévisible. [19]
- **Signature numérique :** La signature numérique « empreinte numérique » est le plus haut niveau de sécurité et d'authentification disponible dans les signatures électroniques. La signature numérique est constituée de deux clés. Deux clés sont impliquées : une clé publique générée à partir de la clé privée et utilisée pour confirmer la légitimité de la signature ; la clé privée est cryptée et utilisée pour identifier le propriétaire du fichier. [20]
- **Antivirus :** Les logiciels antivirus analysent les fichiers et la mémoire vive de l'ordinateur à la recherche de modèles qui pourraient indiquer l'existence de logiciels malveillants ou dangereux. En utilisant les signatures ou les descriptions des logiciels malveillants connus, les logiciels antivirus parfois plus communément appelés logiciels anti-malveillants analysent des modèles. Il est essentiel que vous disposiez des mises à jour antivirus les plus récentes sur votre ordinateur, car les fournisseurs de logiciels antivirus découvrent et mettent à jour des virus quotidiennement. [21]
- **Le pare-feu :** Un pare-feu est un type de dispositif de sécurité réseau qui surveille l'ensemble du trafic réseau entrant et sortant et utilise des règles de sécurité préétablies pour déterminer s'il convient d'autoriser ou d'interdire certains types de données. [22]
- **Analyse des vulnérabilités** : L'analyse de la vulnérabilité permet de localiser, de classer et de hiérarchiser les failles de sécurité dans les programmes informatiques, l'infrastructure du réseau et les applications. Une entreprise peut être vulnérable à des risques ou à des cybermenaces en raison de failles de sécurité. Les scanners de sécurité réseau et d'autres outils de test automatisés sont fréquemment utilisés dans les évaluations de vulnérabilité et les résultats sont présentés dans un rapport d'évaluation de vulnérabilité. [23]

2.3. Les attaques informatiques

2.3.1. Définition

Une attaque informatique, est une activité criminelle qui vise les systèmes informatiques et/ou les utilisateurs de ces systèmes dans le but d'obtenir un accès non autorisé à ces systèmes et aux données qu'ils contiennent.

En général, l'objectif d'une attaque criminelle est d'en tirer profit, mais il arrive aussi qu'elle ait pour but de perturber les opérations en refusant l'accès aux systèmes informatiques. Les acteurs de la menace peuvent être n'importe qui, qu'il s'agisse d'un individu isolé qui tente de s'emparer d'informations d'identification qu'il a volées et de demander une rançon ou d'une entité parrainée par un État qui tente de saboter des activités à l'étranger. Quelle que soit la raison, la majorité des réseaux informatiques. Ainsi que les personnes qui les gèrent seront à un moment ou à un autre confrontés à une attaque quelconque, à laquelle ils doivent être prêts. [24]

2.3.2. Les Classes d'attaques :

Il existe 4 classes d'attaque : [25]

- L'attaque DOS (denial of service)
- Probe (Probing)
- U2R (User to Root)
- R2L (Remote to Local)

La classe d'attaques	Type D'attaques
DoS	Back, Land, Smurf, Teardrop, Neptune, worm, Udpstorm, Processtable, Pod.
Probe	Satan, Ipsweep, Namp, Portsweep, Mscan, Saint.
U2R	Eject, Loadmodule, Perl, Xterm, Ps.
R2L	Buffer_Password, Ftp_write, Imap, Phf, Multihop, warezmaster, warezclient, spy, Xlock, Xnsoop, Snmpguess, snmpgetattack, Httptunnel, Sendmail, Named.

Table 2.1 : Les classes d'attaques. [25]

2.3.2.1. L'attaque DOS (denial of service) :

Une attaque par déni de service (DoS) est une classe d'attaque informatique à mettre hors service, à désactiver ou à perturber un site Web, un réseau ou un service. Les logiciels malveillants visent généralement à interférer avec un système ou à empêcher les données d'y entrer ou d'en sortir, afin de rendre la cible temporairement inopérante ou inutilisable. Lorsqu'un site Web est régulièrement et massivement consulté à partir de plusieurs endroits, il peut empêcher les visiteurs légitimes d'y accéder. C'est un exemple d'attaque par déni de service.

Le fait de lancer une attaque par déni de service à partir de différents endroits et de manière coordonnée est souvent appelé attaque par déni de service distribuée (DDoS). [27]

2.3.2.2. Probe

L'attaquant de cette classe débute en interrogeant la future victime, ce qu'on nomme un scan. Cette enquête examinera chaque port IP pour déterminer les services proposés par le système Après avoir terminé cette analyse, Ainsi, la machine de l'intrus tente de détecter le système d'exploitation utilisé par cette victime et d'utiliser les informations qu'elle a collectées. La plus grande catégorie d'attaques nécessite une expertise technique minimale. Les attaques telles que Ipsweep, Mscan, Nmap, Saint et Satan sont des exemples de ce genre. [27]

Type d'attaque	Service	Mécanisme	L'effet de l'attaque
Satan	Icmp	Abus de Propiétés	Identification des machines atives
Ipsweep	Plusieurs	Abus de Propiétés	Recherche des vulnérabilités
Namp	Plusieurs	Abus de Propiétés	Identification des ports actifs sur une machine
Mscan	Plusieurs	Abus de Propiétés	Recherche des vulnérabilités
Saint	Plusieurs	Abus de Propiétés	Recherche des vulnérabilités

Table 2.2: Attaque de Probe. [26]

2.3.2.3. U2R (User to Root)

Ces attaques sont des exploitations dans lesquelles le pirate démarre sur le système avec un compte d'utilisateur normal et tente d'abuser des vulnérabilités du système afin d'obtenir des privilèges de super-utilisateur, par exemple perl, xterm. [28]

Type d'attaque	Service	Mécanisme	Effet de l'attaque
Loadmodule	Session utilisateur	Un mauvais système d'installation	Gagne le shell root
Perl	Session utilisateur	Un mauvais système d'installation	Gagne le shell root
Xterm	Session utilisateur	Débordement du buffer	Gagne le shell root
Ps	Session utilisateur	Une mauvaise gestion des fichiers temporels	Gagne le shell root

Table 2.3: Attaque U2R. [26]

2.3.2.4. R2L (**Remote to Local)**

Dans cette classe R2L, ou "Remote to Local", désigne un type de cyberattaque dans lequel un pirate accède à un système distant, tel qu'un serveur ou un ordinateur, puis pivote à partir de cet accès à distance pour accéder à d'autres systèmes ou ressources au sein du réseau local. Ce type d'attaque peut être particulièrement préoccupant, car une fois que le pirate a pénétré dans un système, il peut s'en servir comme point d'appui pour explorer et exploiter d'autres systèmes au sein du réseau, ce qui peut entraîner des dommages étendus ou le vol de données. [27]

Type d'attaque	Service	Mécanisme	Effet de l'attaque
ftp-write	ftp	Mauvaise configuration	Gagne un accès utilisateur
Imap	Imap	Bug	Gagne un accès root
Phf	HTTP	Bug	Exécute des commandes autant qu'utilisateur HTTP
Xlock	Smtp	Configuration	Mystifie un utilisateur pour obtenir le mot de passe
Xnsoop	Smtp	Configuration	Contrôle le stockage des clés à distance
Sendmail	Smtp	Bug	Exécute des commandes autant que root
Named	Dns	Bug	Gagne un accès root

Table 2.4 : Attaque R2L. [26]

2.3.3. Exemples des attaques :

- **Trojans ou chevaux de Troie :**

Le logiciel du cheval de Troie peut être utilisé pour ouvrir une porte dérobée dans le système qui permet aux pirates d'accéder à un ordinateur ou à un réseau lorsque l'utilisateur exécute le programme qui semble inoffensif. La légende des guerriers grecs qui se sont faufilés dans Troie et ont gagné la guerre en se cachant à l'intérieur d'un cheval est à l'origine de ce danger. Les guerriers grecs sont sortis et ont lancé un assaut dès que le "cadeau" a été reconnu et présenté dans les murs de Troie. De même, un utilisateur imprudent peut installer un programme qui semble inoffensif sur sa machine, mais qui ajoute involontairement un danger caché. [29]

- **Attaques de phishing :**

Aussi appelé hameçonnage, le phishing est un type de cyberattaque où le pirate utilise l'usurpation d'identité pour obtenir les informations personnelles de l'utilisateur. Par conséquent, le pirate n'aurait pas eu besoin de forcer l'entrée d'un système informatique, car la victime lui aurait fourni directement ces informations. [30]

- **Attaque de malware :**

Les logiciels malveillants sont généralement appelés malwares. Lorsqu'un logiciel malveillant envahit un système, il peut en modifier le fonctionnement, effacer des données ou espionner l'activité du réseau ou de l'utilisateur. Les logiciels malveillants peuvent se déplacer d'un appareil à l'autre ou rester stationnaires, n'ayant d'impact que sur l'appareil hôte. [29]

- **Attaque par mot de passe :**

Comme la plupart des gens utilisent des mots de passe pour vérifier leur accès, il est logique qu'un pirate informatique cherche à découvrir le mot de passe d'une cible. Pour ce faire, il utilise diverses techniques. Les particuliers conservent souvent des copies de leurs mots de passe sur des bouts de papier ou des notes autocollantes sur leur bureau. Un pirate a deux options : il peut payer quelqu'un à l'intérieur pour obtenir le mot de passe, ou il peut le découvrir par lui-même. [29]

- **Ransomware:**

Un ransomware est un type de logiciel malveillant qui s'introduit dans votre système informatique et crypte vos données ou vous empêche d'y accéder, à moins qu'une rançon ne soit versée. Le ransomware peut arrêter complètement le système informatique une fois que la personne s'en est emparée. [30]

2.4. Les systèmes de détection d'intrusion

2.4.1. Définition

Un composant de la sécurité des réseaux appelé système de détection d'intrusion est utilisé pour surveiller le flux de données et identifier toute action spécifique qui pourrait être interprétée comme une intrusion [31].

En fonction des sources de données et des techniques de détection, les systèmes de détection d'intrusion peuvent être classés de manière générale. Il existe trois types d'IDS basés sur des sources de données : les IDS basés sur le réseau, les IDS basés sur l'hôte et les IDS hybrides [28]. Un type d'IDS appelé IDS basé sur l'hôte (HIDS) est installé sur l'hôte, qui peut être l'utilisateur ou la zone démilitarisée. Il implique des méthodes de travail qui examinent les paquets envoyés à des hôtes particuliers, comme un serveur web, afin de les étudier ultérieurement sous la forme de journaux d'hôtes et de trafic. Un système de détection des intrusions (IDS) installé à la porte du canal internet ou à l'entrée du trafic réseau est connu sous le nom d'IDS basé sur le réseau (NIDS).

Les paquets de données du trafic réseau (TCP/IP) sont analysés par les IDS. Les deux modules qui composent les systèmes de détection d'intrusion basés sur des approches de détection sont la détection des abus (basée sur les signatures) et la détection des anomalies (basée sur les signatures) [29]. En comparant des modèles ou des signatures qui existent déjà dans la base de données avec l'événement (une incursion ou une attaque) qui s'est produit, les approches de détection basées sur les signatures identifient une intrusion. L'idée est que le même modèle sera utilisé pour détecter de futures intrusions. Un profil normal pour l'IDS doit d'abord être créé pour définir l'activité normale sur l'hôte ou le réseau. De cette manière, si les circonstances étaient différentes, le profil créé sur l'IDS serait reconnu comme une intrusion. Cette technique est connue sous le nom de détection basée sur les anomalies et elle est utilisée pour détecter une intrusion ou une anomalie basée sur un événement qui se produit sur le réseau ou l'hôte. [31]

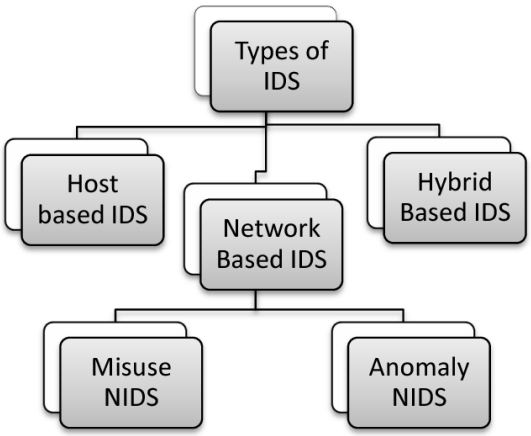

Figure 2.1 : Les types d'IDS. [33]

2.4.2. Les Différents types d'IDS :

Les HIDS et les NIDS (systèmes de détection d'intrusion basés sur l'hôte et le réseau) sont des éléments essentiels de la sécurité informatique. Tandis que le NIDS analyse le trafic réseau à la recherche de toute menace circulant sur le réseau, le HIDS garde un œil sur les appareils individuels, analysant l'activité locale à la recherche de schémas suspects et d'attaques. La protection des données sensibles et des actifs numériques est l'une des principales responsabilités de ces systèmes. [34]

2.4.2.1. Système L'IDS basé hôte (host- based IDS) :

HIDS est un type de logiciel de sécurité destiné à suivre et à examiner l'activité d'un hôte ou d'un point de terminaison spécifique afin d'identifier et de traiter d'éventuelles failles de sécurité. Afin de trouver des activités suspectes ou des indications d'accès non autorisé ou d'altération, il examine les journaux du système, l'intégrité des fichiers, les actions de l'utilisateur et les connexions au réseau.

Pour que le HIDS fonctionne, des agents ou des capteurs doivent être installés sur chaque hôte afin de surveiller en permanence les événements et les activités du système. Il compare ces événements à un modèle d'attaque et à une base de données des comportements anormaux. Le système avertit les administrateurs par le biais d'alertes ou de notifications dès qu'il détecte une activité inhabituelle, leur donnant ainsi la possibilité de l'examiner et de prendre les mesures qui s'imposent. [34]

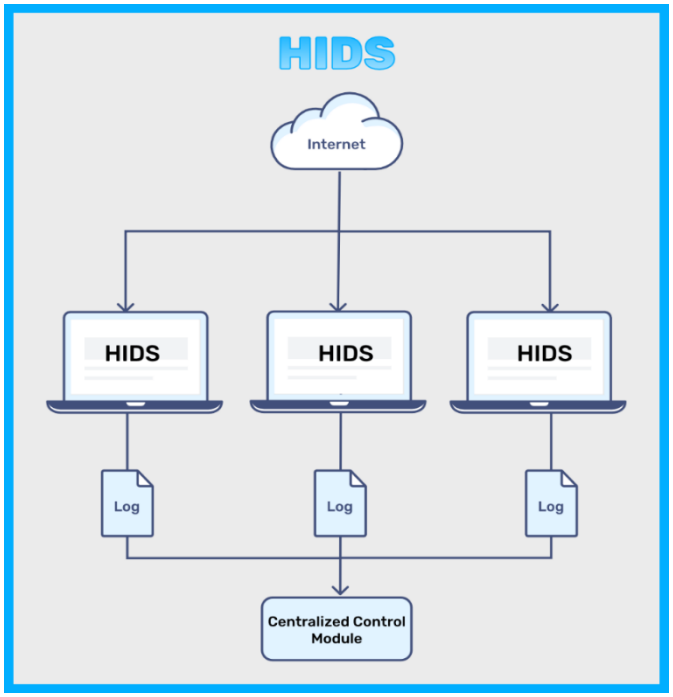

Figure 2.2 : Système L'IDS basé hôte. [35]

2.4.2.2. Système L'IDS basé réseau (Network- based IDS) :

NIDS est un système de sécurité conçu pour surveiller et analyser le trafic réseau afin de détecter d'éventuelles menaces ou failles de sécurité. Il surveille en temps réel les paquets de données circulant sur le réseau et fonctionne comme un système passif. Pour améliorer la sécurité globale du réseau, le NIDS aide à identifier et à répondre à une variété de cybermenaces, y compris les logiciels malveillants, les tentatives d'accès non autorisés et les schémas suspects.

Le NIDS utilise diverses méthodes, telles que la détection basée sur les anomalies et les signatures, pour examiner le trafic du réseau. Il examine les charges utiles et les en-têtes des paquets et les compare à un fichier de signatures d'attaques bien connues. Une alerte est déclenchée en cas de correspondance. En détectant les écarts par rapport au comportement habituel d'un réseau, la détection basée sur les anomalies met en évidence toute action atypique susceptible d'indiquer une intrusion. En bloquant le trafic nuisible ou en avertissant les administrateurs de menaces imminentes, les NIDS peuvent également stopper les attaques. [34]

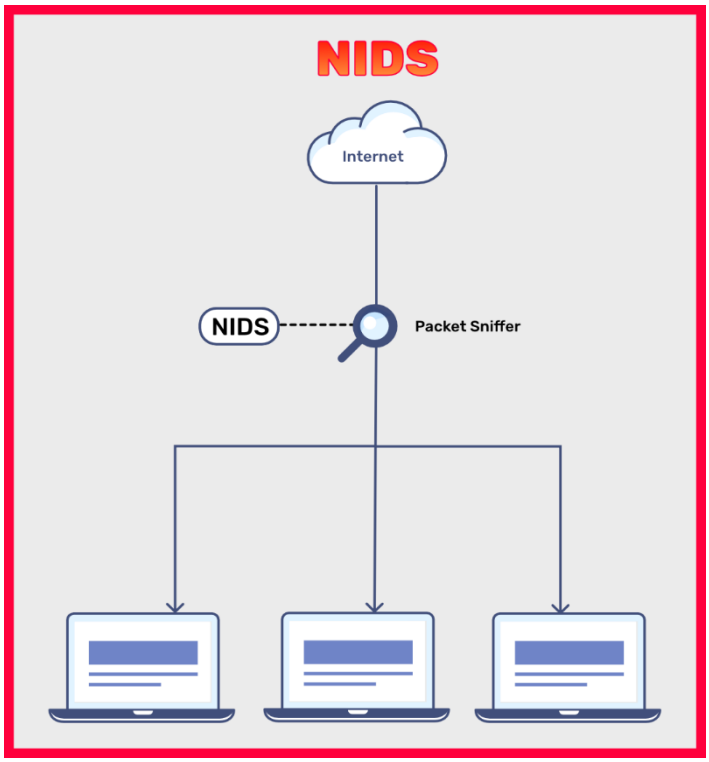

Figure 2.3 : Système L'IDS basé réseau. [35]

Il existe deux types de NIDS : [36]

- **NIDS en ligne :** dans ce type NIDS effectue une analyse du réseau en temps réel.
- **NIDS hors ligne (mode tap) :** dans ce type le NIDS traite les paquets qui ont été recueillis dans ce cas. Il les soumet donc à plusieurs étapes avant de déterminer s'il convient ou non de lancer une attaque.

2.4.2.3. Système de détection d'intrusion Hybrides :

Un système hybride de détection des intrusions intègre deux ou plusieurs méthodes de détection des intrusions. Il combine les données du réseau avec les données du système ou de l'agent hôte afin d'obtenir une image complète du système. Par rapport aux autres systèmes, le système hybride de détection des intrusions est plus puissant. [37]

2.4.3. Comparaison entre les différents HIDS et NIDS :

Les différences sont indiquées dans le tableau suivant :

Base de la différence	HIDS	NIDS
Définition	HIDS est un identifiant unique attribué à un dispositif matériel informatique.	NIDS est un identifiant unique attribué à un réseau.
Type	Les HIDS sont généralement spécifiques à un seul appareil.	Les NIDS sont régulièrement attribués à un réseau et peuvent être utilisés par de nombreux appareils connectés à ce réseau.
Localisation	Stockés localement sur les appareils.	Rangé sur le réseau
Format	En général, des chaînes alphanumériques.	Généralement numériques.
Utilisation	Utilisé pour l'établissement du pilote de l'appareil, le dépannage et les demandes de garantie.	Utilisé pour l'administration du réseau, la sécurité et la communication entre les appareils du réseau.
Objectif	La raison d'être d'un HIDS est de distinguer un dispositif matériel spécifique.	L'objectif d'un NIDS est de distinguer un réseau spécifique.

Table 2.5: Comparaison entre les différents HIDS et NIDS. [38]

2.4.4. Classification des systèmes de détection d'intrusions :

Les systèmes de détection d'intrusion peuvent être classés en cinq catégories : [39]

- Les méthodes de détection d'intrusion
- Comportement après détection
- Localisation de l'analyse des données
- Sources des données.
- Fréquence de l'analyse.

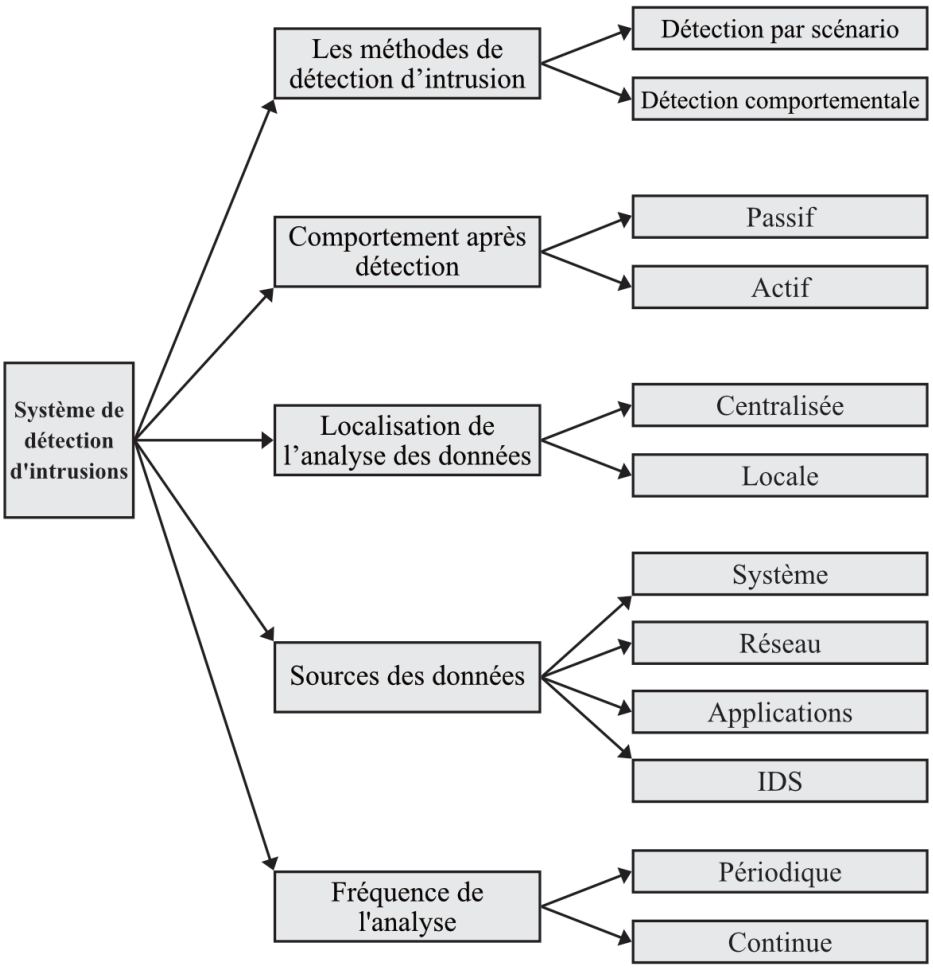

Figure 2.4 : Classification des systèmes de détection d'intrusions**.** [39]

2.4.4.1. Les méthodes de détection d'intrusion :

Le système de détection d'intrusion basé sur deux méthodes de détection :

- La détection par scénario (Signature-based).
- La détection comportementale (Anomaly-Based).

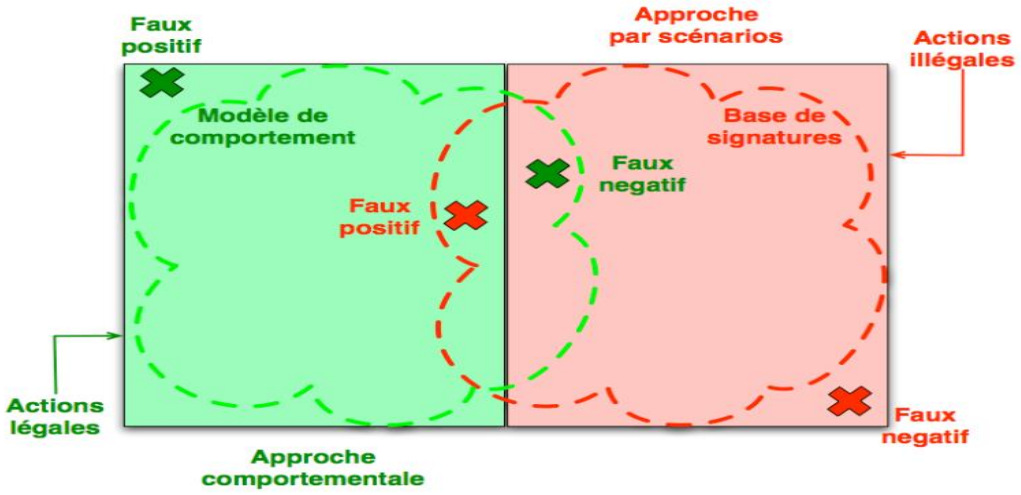

Figure 2.5 : Les erreurs des méthodes d'un IDS. [40]

2.4.4.1.1. La détection par scénario (Signature-based) :

La détection basée sur les signatures est la technique IDS la plus utilisée et la plus conventionnelle. Elle consiste à comparer le trafic réseau entrant à une base de données de modèles d'attaque également appelés signatures dont il a été prouvé qu'ils représentent une activité malveillante. L'IDS avertit l'utilisateur ou arrête l'attaque si une correspondance est trouvée. Lorsqu'il s'agit d'attaques connues et courantes telles que les logiciels malveillants, le phishing et le déni de service, la détection basée sur les signatures fonctionne bien. Parce qu'elle dépend des fournisseurs ou des spécialistes de la sécurité pour mettre à jour régulièrement la base de données des signatures, elle est également simple à mettre en place et à maintenir.

La détection basée sur les signatures n'est cependant pas sans limites. Sans correspondance avec une signature dans la base de données, elle est incapable d'identifier des versions nouvelles, non identifiées ou modifiées d'attaques déjà existantes. Elle présente également un taux élevé de faux positifs, c'est-à-dire le nombre de rapports de trafic authentiques qui sont confondus avec des activités malveillantes. En outre, comme la détection basée sur les signatures doit vérifier chaque paquet de données par rapport à un grand nombre de signatures, elle peut être gourmande en ressources et dégrader la vitesse du réseau. [41]

2.4.4.1.2. La détection comportementale (Anomaly-Based) :

La détection basée sur les anomalies est une technique IDS plus récente et plus sophistiquée. Elle consiste à créer une base de référence de l'activité typique du réseau à l'aide de l'intelligence artificielle, de l'apprentissage automatique ou de l'analyse statistique. Ensuite, il surveille le trafic réseau à la recherche d'anomalies – des différences par rapport à la norme – qui pourraient indiquer un comportement malveillant. Lorsqu'une anomalie est détectée, l'IDS avertit l'attaquant ou met

fin à l'attaque. Lorsqu'il s'agit d'attaques non identifiées ou nouvelles, ainsi que de variantes d'attaques connues, la détection basée sur les anomalies fonctionne bien. En outre, comme elle peut ajuster la ligne de base en fonction de ce qu'elle apprend du comportement du réseau, elle est dynamique et adaptative.

La détection basée sur les anomalies présente toutefois certaines limites. En particulier pour les réseaux complexes et hétérogènes, il peut être difficile de créer et de préserver une base de référence cohérente du comportement normal. En outre, lorsque le trafic hostile se fond dans le trafic authentique ou imite un comportement typique, il y a un risque élevé de faux négatifs ou d'attaques manquées. En outre, parce qu'elle nécessite une technologie et une expertise sophistiquées, la détection basée sur les anomalies peut être coûteuse et difficile à développer et à mettre en œuvre. [41]

2.4.4.1.3. Hybride détection :

L'approche hybride combinant des techniques de détection basées sur les signatures et sur les anomalies est employée par plusieurs solutions IDS en raison de leurs avantages et inconvénients respectifs. En combinant les avantages de la détection basée sur les anomalies et de la détection basée sur les signatures, la détection hybride peut offrir une défense plus complète et plus précise contre une plus grande variété de menaces. Par exemple, la détection hybride utilise la détection basée sur les anomalies pour trouver des attaques nouvelles ou non identifiées qui échappent à la détection basée sur les signatures, tout en utilisant la détection basée sur les signatures pour identifier rapidement et efficacement les attaques existantes. Comme la détection hybride valide et met en corrélation les alertes provenant des deux sources, elle peut également réduire les faux positifs et les faux négatifs des deux méthodes.

La détection hybride n'est pas non plus une solution idéale. Elle peut encore présenter certains des inconvénients de chaque approche, notamment en termes de maintenance, de complexité et d'utilisation des ressources. En outre, elle peut entraîner de nouvelles difficultés, notamment en ce qui concerne la coordination des politiques, la gestion des alertes et l'intégration des données. Ainsi, pour obtenir les meilleurs résultats de la détection hybride, une planification, une configuration et une optimisation adéquates sont nécessaires. [41]

2.4.4.2. Comportement après détection :

La catégorisation des systèmes de détection d'intrusions sont classés en passifs ou actifs selon leur réaction à une attaque détectée. [39]

- **Passif :** Les systèmes passifs se contentent de générer des alarmes et de notifier l'administrateur, qui doit prendre des mesures appropriées. En revanche.
- **Actif :** les systèmes actifs peuvent prendre des mesures automatiques pour arrêter l'attaque, comme couper les connexions suspectes ou reconfigurer les pares-feux. Cette réaction automatique peut être dangereuse, car elle pourrait causer des dénis de service en raison d'erreurs d'identification. Il est recommandé d'offrir Une réaction optionnelle à un acteur humain pour une décision définitive.

2.4.4.3. Localisation de l'analyse des données :

Les IDS peuvent être différenciés par leur localisation d'analyse des données : soit centralisée sur une seule machine pour une vue globale mais avec une charge système et réseau élevée, soit locale sur chaque machine pour minimiser le trafic réseau mais risquant de manquer des attaques distribuées. [39]

2.4.4.4. Sources des données

2.4.4.4.1. Source d'information système

Les sources d'information système comprennent les commandes système pour surveiller les activités en temps réel et les fonctionnalités telles que l'accounting et l'audit de sécurité, fournissant des données sur l'utilisation des ressources et les actions des utilisateurs pour garantir la sécurité et la traçabilité des activités. [39]

2.4.4.4.2. Source d'information réseau

Les dispositifs de capture de trafic réseau, tels que les snifers, permettent de détecter les attaques en déni de service et les tentatives de pénétration à distance, tout en restant discrets grâce à une configuration dédiée. Cependant, l'origine réelle des attaques détectées peut être difficile à déterminer en raison de la facilité de modification des paquets réseau. [39]

2.4.4.4.3. Source d'information applicative

Les applications peuvent être des sources d'information pour les IDS avec des capteurs internes et externes.

Les capteurs internes exécutent le filtrage directement dans le code de l'application, tandis que les capteurs externes le font en dehors de celle-ci, par exemple en filtrant les logs ou en interceptant son exécution. Les données interceptées au niveau de l'application sont fiables et offrent une sémantique plus riche que les sources système et réseau. Par exemple, un capteur applicatif peut analyser le texte en clair des requêtes HTTPS, tandis qu'un capteur réseau ne verrait que des données chiffrées.

Cependant, il est important de noter que les capteurs réseau peuvent subir des désynchronisations, contrairement aux capteurs applicatifs. [39]

2.4.4.4.4. Source d'information basée IDS

Les alertes d'un IDS fournissent une source d'information synthétisée sur des événements de sécurité, déclenchant des analyses plus approfondies en cas d'attaque potentielle. En corrélant ces alertes, des intrusions complexes peuvent être détectées, générant ainsi des méta-alertes de haut niveau. Ce processus permet une meilleure compréhension des menaces et une réponse plus efficace aux incidents de sécurité. [39]

2.4.4.5. Fréquence de l'analyse

Les systèmes de détection d'intrusions peuvent être périodiques ou continus. [39]

- **Périodiques :** Les périodiques analysent régulièrement les fichiers d'audit à la recherche d'activités suspectes, adaptés à des environnements moins sensibles.
- **Continus :** Les systèmes continus, plus courants, surveillent en temps réel les fichiers d'audit ou les paquets réseau pour une détection immédiate, essentielle pour des contextes sensibles ou commerciaux. Cette méthode est gourmande en ressources, nécessitant une analyse constante des activités du système.

2.4.5. Dataset

Dataset est une collection structurée d'informations. Les données sont un ensemble d'informations obtenues par le biais d'observations, de mesures, d'études ou d'analyses. Il peut s'agir de faits, de statistiques, de chiffres, de noms ou même de simples descriptions d'objets. Les données de notre étude peuvent être présentées sous forme de graphiques, de diagrammes ou de tableaux. Les data scientists aident à analyser les données obtenues par le biais du data mining. [42]

2.4.5.1. DARPA 98

Le DARPA 1998 IDS dataset est l'un des premiers ensembles de données utilisés dans le domaine de la cybersécurité. Sous la direction conjointe du laboratoire de recherche de l'armée de l'air américaine et de l'Agence pour les projets de recherche avancée de la défense (DARPA), la collection a été créée en mars 1998 au laboratoire Lincoln du MIT. De nouvelles données ont été ajoutées à l'ensemble en 1999 et 2000.

Toutes les activités du réseau et tous les paquets de données capturés au format TCPdump peuvent être trouvés dans l'ensemble des données DARPA IDS.

Les captures de trafic, les fichiers d'audit Windows NT (pour DARPA 1999), les fichiers Solaris BSM et les fichiers contenant les processus en cours d'exécution figurent parmi les formats dans lesquels les données évaluées sont envoyées.

Des systèmes réels et simulés coexistaient dans le réseau cible, qui produisait également un trafic de fond. Seuls les systèmes réels ont été la cible des attaques. [43]

2.4.5.2. KDD Cup 99

Le dataset KDD Cup99 constitue une amélioration, mais elle présente toujours des problèmes similaires à ceux de l'ensemble des données DARPA IDS. Cet ensemble a été créé spécifiquement pour être utilisé dans la troisième itération du Concours international d'outils de découverte de connaissances et d'exploration de données, qui est une composante de la cinquième conférence internationale sur la découverte de connaissances et l'exploration de données (International Conference on Knowledge Discovery and Data Mining).

La collection se compose d'une liste exhaustive et normalisée de vecteurs de caractéristiques qui illustrent les attaques de quatre catégories principales documentées dans un réseau d'architecture militaire : sondages, attaques d'utilisateur à racine, d'utilisateur à distance et de déni de service. [43]

2.4.5.3. NSL-KDD

NSL-KDD Dataset contient les mêmes catégories primaires d'attaques que KDD 99. Le nombre total d'attaques individuelles est comparable au nombre des incidents réguliers.

Bien que cette version actualisée de l'ensemble KDD présente plusieurs problèmes et ne reproduise pas exactement un réseau réel, NSLKDD est l'un des ensembles de données les plus populaires pour l'entraînement des applications ML-IDS. Cet ensemble a été utilisé par les chercheurs en 2009, principalement en raison du grand nombre d'enregistrements provenant à la fois de la formation et des tests, ainsi que de la rareté des données accessibles au public.

Cette fonctionnalité permet au chercheur d'exécuter des expériences sur l'ensemble de la collection plutôt que sur une partie aléatoire de celle-ci, ce qui garantit la cohérence des résultats obtenus. [43]

2.5. Conclusion

Dans ce deuxième chapitre nous présentons trois parties, en commençant par les définitions et les différentes classes d'attaques informatique et quelques exemples d'attaques. Puis présente la sécurité informatique et la différence entre la Sécurité informatique et cybersécurité et quelques techniques de la sécurité. Puis la présentation des systèmes de détection d'intrusion, définition, et les différents types d'IDS et Ses critères de classification et dataset. Dans le chapitre suivant, nous exposerons l'approche proposée de notre IDS et nous présenterons les résultats obtenus.

Chapitre III

Conception et Réalisation

3.1. Introduction

Notre application se base sur les méthodes Booting, Bagging et Voting déjà décrient dans les chapitres précédents pour construire un système de détection d'intrusion basé sur l'analyse du comportement de ces connexions en effectuant une classification supervisée des connexions TCP/IP dans la base de données NSL_KDD et en les classant en deux groupes (attaque et normal). Nous commençons ce chapitre par une description de la base de données NSL-KDD et des processus de prétraitement que nous y avons appliqués avant de détailler les différentes étapes de la création de notre application. Ensuite, nous discutons des résultats et présentons notre modèle de classification.

3.2. Approche Proposé

3.2.1. Architecture du modèle

Sur la base de l'ensemble de données NSL-KDD, le système suggéré pour la conception d'un système de détection d'intrusion dans le réseau (Network Intrusion Detection System) a été mis au point système de détection d'intrusion dans les réseaux. Après avoir appliqué un prétraitement à l'ensemble de données brutes. La normalisation est appliquée pour que les valeurs de toutes les caractéristiques soient comprises entre 0 et 1. L'ensemble de formation a été utilisé pour former le modèle et l'ensemble de test a été utilisé pour évaluer le modèle formé.

L'ensemble d'apprentissage a été utilisé pour former le modèle et l'ensemble de test a été utilisé pour évaluer le modèle formé. Et ces caractéristiques importantes seront utilisées avec l'ensemble de test dans la partie prédiction. Après avoir sélectionné les caractéristiques pour l'ensemble de formation, une classification est mise en œuvre sur l'ensemble de formation à l'aide des algorithmes de l'ensemble de Machine Learning : Bagging, Boosting et Voting.

Enfin, le modèle est évalué par la prédiction avec l'ensemble de test et en comparant les résultats. Le système proposé est illustré à la figure.

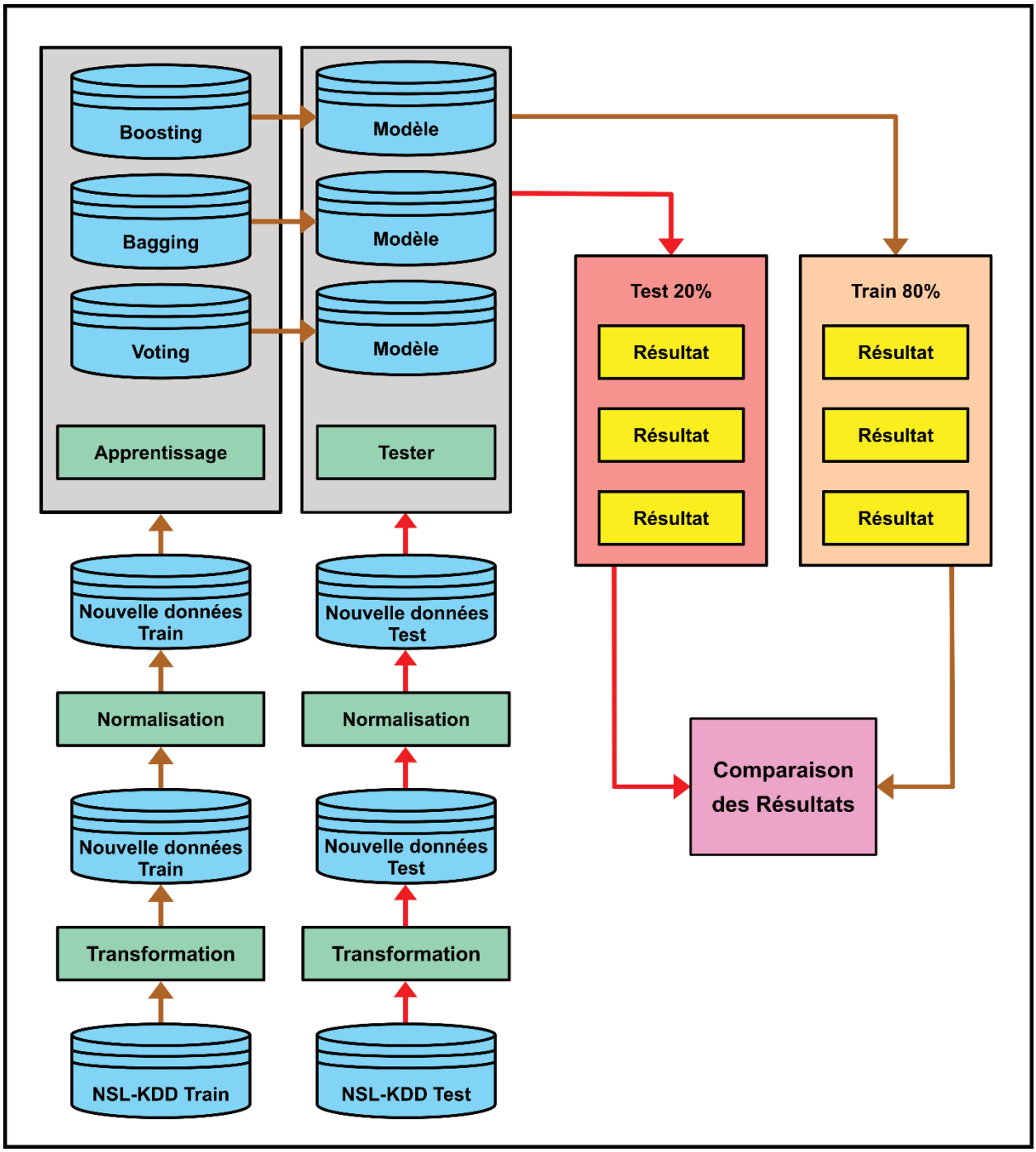

Figure 3.1 : Schéma représente l'architecture de fonctionnement du modèle IDS.

3.2.2. Description du Dataset

3.2.2.1. Aperçu sur dataset NSL-KDD

La couche de sécurité du réseau découverte de connaissances dans les bases de données, ou NSL-KDD, repose sur l'ensemble de données KDD99, qui est une collection de connexions TCP/IP extraites de l'ensemble de données d'évaluation des systèmes de détection d'intrusion. Afin d'évaluer les systèmes de détection des intrusions dans les réseaux informatiques, les Lincoln Labs du MIT ont rassemblé et diffusé les ensembles de données développés en 1998

par la Defense Advanced Research Projects Agency (DARPA) et l'Air Force Research Laboratory (AFRL). [45]

Un ensemble de données connu sous le nom de NSL-KDD, qui est une version abrégée de KDD99 créée pour résoudre certains problèmes liés à la version initiale du protocole, a été dévoilé par les chercheurs en détection d'intrusion dans les réseaux en 2010. Considéré comme un ensemble de données de référence, NSL-KDD aide les chercheurs à comparer diverses méthodes de détection des intrusions. Les changements entre KDD 99 et NSL-KDD sont les suivants : [45]

- La performance de la classification est améliorée parce que les enregistrements redondants sont exclus des données d'apprentissage.
- Les taux de détection améliorés sont facilités par l'absence d'enregistrements en double dans les ensembles de test suggérés.
- On peut soutenir qu'il est raisonnable d'effectuer les expériences sur l'ensemble du jeu sans avoir à en sélectionner aléatoirement une infime partie, car il y a un nombre raisonnable d'enregistrements à la fois dans les données d'apprentissage et dans les jeux de test. Ainsi, les résultats de l'évaluation de l'outil seront cohérents et comparables.

Les enregistrements de connexion TCP/IP sont inclus dans la collection de données NSL-KDD. Les 41 attributs qui composent chaque enregistrement de connexion expliquent la connexion et sont associés à 5 classes (quatre types d'attaques et une classe normale). Les principaux types d'attaques trouvés dans l'ensemble des données NSL-KDD, comme détaillé dans le premier chapitre de cette thèse, sont des attaques par déni de service, l'attaque d'utilisateur à racine, l'attaque d'utilisateur à distance et l'exploration. [44] [45]

Nom du fichier	Normal	DoS	Probe	R2L	U2R
KDDTrain+ 20%	13449 (53.39 %)	9234 (36.65 %)	2289 (9.09%)	209 (0.83 %)	11 (0.04 %)
KDDTrain+	67343 (53.46 %)	45927 (36.46 %)	11656 (9.25 %)	995 (0.79%)	52 (0.04 %)
KDDTest+	9711 (43.08 %)	7458 (33.08 %)	2421 (10.74 %)	2754 (12.22 %)	200 (0.89 %)
KDDTest-21	2152 (18.16 %)	4342 (36.64 %)	2402 (20.27 %)	2754 (23.24 %)	200 (1.69 %)

Tableau 3.1 : Répartition des fichiers et les classes de NSL-KDD.[46]

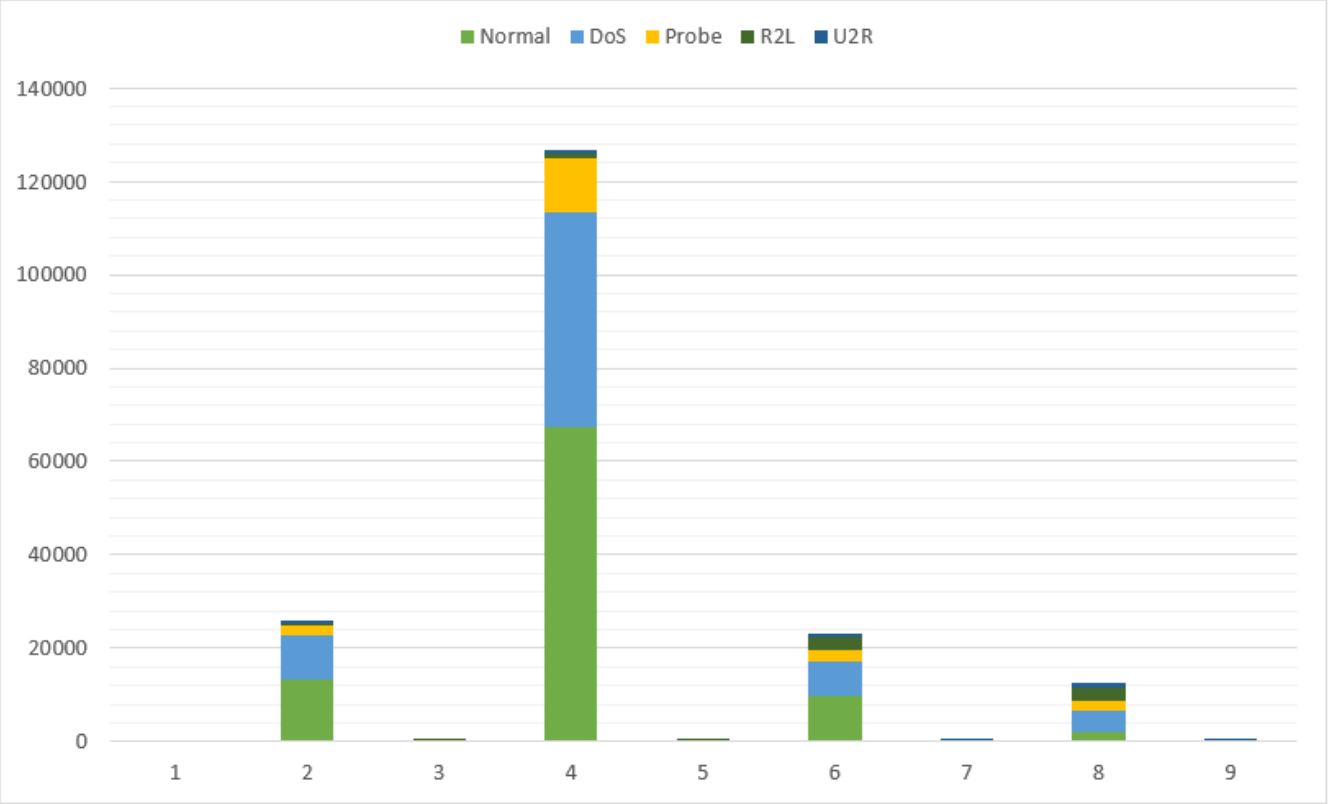

Figure 3.2 : Répartition des fichiers et les classes de NSL-KDD.

3.2.2.2. Le contenu de le dataset NSL-KDD [45]

KDDTrain +. ARFF : Ensemble complet d'étiquettes NSL-KDD au format ARFF avec des étiquettes binaires.

KDDTrain + .TXT : Ensemble complet d'entraînements NSL-KDD au format CSV avec étiquettes d'attaque et niveaux de difficulté inclus.

KDDTrain + _20Percent.ARFF: 20% du fichier KDDTrain + arff est extrait.

KDDTrain + _20Percent.TXT : 20% du fichier KDDTrain + txt est extrait.

KDDTest +. ARFF : Le test complet NSL-KDD avec des étiquettes binaires au format ARFF.

KDDTest + .TXT : Ensemble complet de tests NSL-KDD dans un fichier CSV, avec les étiquettes d'attaque et le niveau de difficulté.

KDDTest-21. ARFF : Une sélection d'enregistrements avec un niveau de difficulté de 21 sur 21 qui ne sont pas inclus dans le fichier KDDTest +.arff.

KDDTest-21. TXT : Une sélection d'enregistrements avec un niveau de difficulté de 21 sur 21 qui ne sont pas inclus dans le fichier KDDTest +.txt.

3.2.2.3. Attributs du dataset NSL-KDD

Les 41 attributs de la base de données NSL-KDD et les types de données correspondants sont compilés dans le tableau (1) de l'annexe. Trois groupes de ces caractéristiques peuvent être identifiés :

- **Les attributs de base :** Ces attributs, qui comprennent la durée, les hôtes de destination et de source, le port et le drapeau, définissent les composants fondamentaux d'une connexion.
- **Les attributs du trafic :** Ces attributs, comme le nombre de connexions à une même machine, sont basées sur des statistiques.
- **Les caractéristiques du contenu :** Ces attributs dépendent de la charge utile (données) des paquets de trafic et comprennent le nombre de connexions et de tentatives d'accès aux fichiers de contrôle qui ont échoué.

Feature number	Type	Feature name
1	Basic Features	Duration
2		Protocol_type
3		Service
4		Flag
5		Src_bytes
6		Dst_bytes
7		Land
8		Wrong_fragment
9		Urgent
10		Hot
11	Content features	Num_failed _logins
12		Logged_in
13		Num_compromised
14		Root_shell
15		Su_attempted
16		Num_root
17		Num_file_creations
18		Num_shells
19		Num_access_files
20		Num_outbound_cmds
21		Is_hot_login
22		Is_guest_login

23	Traffic features	Same service features	Count
24			Srv_count
25			Serror_rate
26			Srv_serror_rate
27			Rerror_rate
28			Srv_rerror_rate
29			Same_srv_rate
30			Diff_srv_rate
31			Srv_diff_host_rate
32		Same host features	Dst_host_count
33			Dst_host_srv_count
34			Dst_host_same_srv_rate
35			Dst_host_diff_srv_rate
36			Dst_host_same_src_port_rate
37			Dst_host_srv_diff_host_rate
38			Dst_host_serror_rate
39			Dst_host_srv_serror_rate
40			Dst_host_rerror_rate
41			Dst_host_srv_rerror_rate

Figure 3.3 : Types de caractéristiques présentes dans l'ensemble de données NSL-KDD. [47]

3.2.3. Réalisation de classification du modèle

Notre sujet est construit en trois étapes, comme c'est le cas pour d'autres modèles de classification : le prétraitement, l'apprentissage et le test. Le sujet est la classification des communications TCP/IP à des fins malveillantes. Comme le montre l'image ci-dessous :

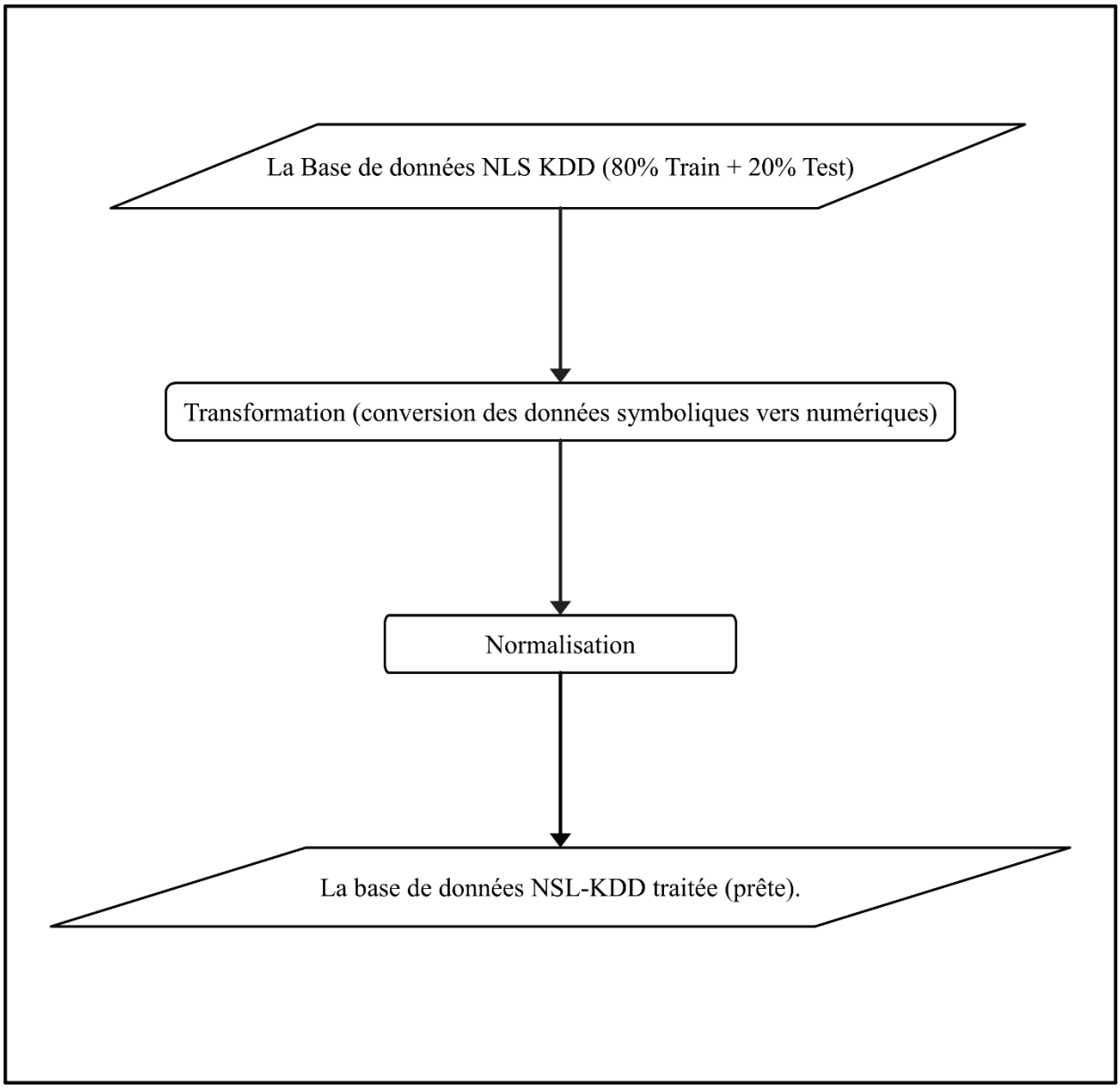

Figure 3.4 : Diagramme Représente le Processus du prétraitement de NSL-KDD.

3.2.4. Prétraitement du dataset NSL-KDD

Cette étape a pour but de préparer les données à une utilisation directe par le traitement des différents modules. Les caractéristiques de l'ensemble de données NSL-KDD sont une combinaison de caractéristiques continues, discrètes et symboliques avec une large gamme de valeurs, et l'ensemble de données de traitement est excessivement grand et prend du temps. Un prétraitement est donc nécessaire.

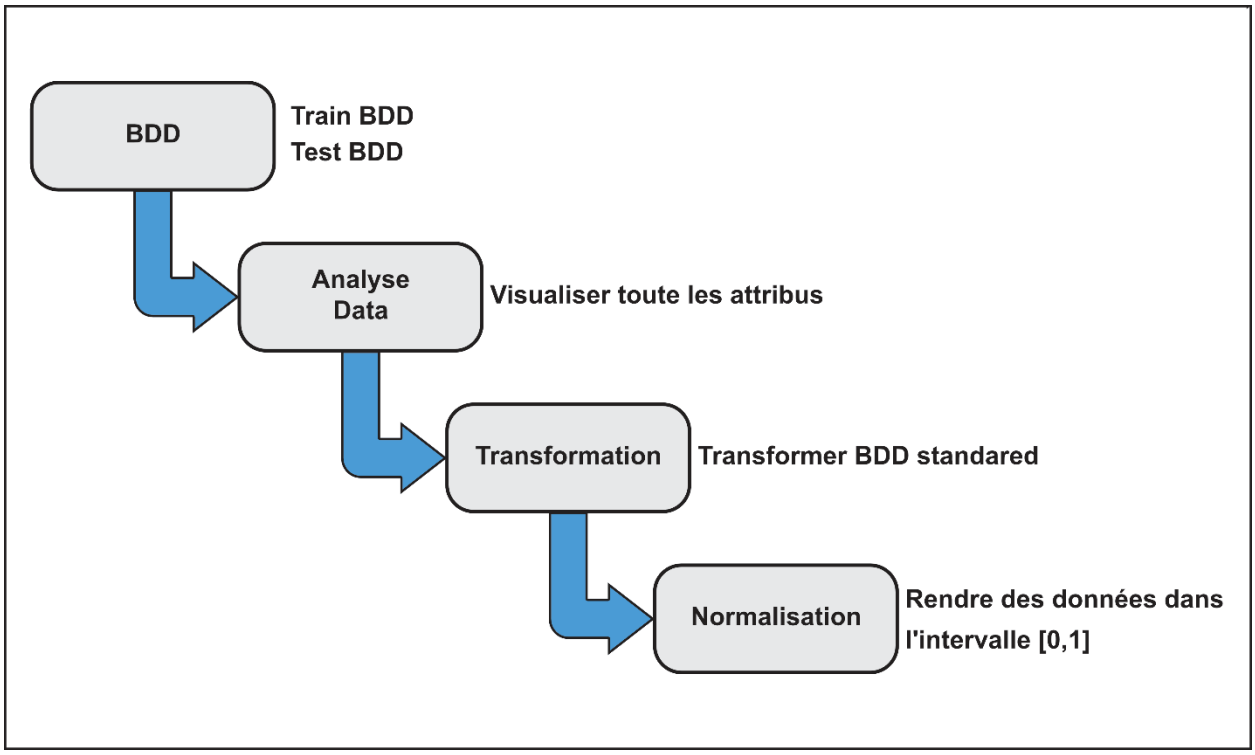

Figure 3.5 : Les Phases de Prétraitement.

Il existe trois types de données NSL-KDD : binaires, nominales et numériques. Comme l'indique la figure 3.2, les attributs 2, 3 et 4 sont nominales, tandis que les attributs 7, 12, 14, 15, 21 et 22 sont binaires. Les autres attributs sont numériques. L'ensemble de données NSL-KDD a fait l'objet d'une conversion de type d'attribut et d'une opération de prétraitement des données, comme indiqué ci-après, avant le début des travaux expérimentaux.

No.	1: duration	2: protocol_type	3: service	4: flag	5: src_bytes	6: dst_bytes	7: land	8: wrong_fragment	9: urgent	10: hot	11: num_failed_logins	12: logged_
	Numeric	Nominal	Nominal	Nominal	Numeric	Numeric	Nominal	Numeric	Numeric	Numeric	Numeric	Nominal
1	0.0	tcp	ftp_data	SF	491.0	0.0	0	0.0	0.0	0.0	0.0	0
2	0.0	udp	other	SF	146.0	0.0	0	0.0	0.0	0.0	0.0	0
3	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
4	0.0	tcp	http	SF	232.0	8153.0	0	0.0	0.0	0.0	0.0	1
5	0.0	tcp	http	SF	199.0	420.0	0	0.0	0.0	0.0	0.0	1
6	0.0	tcp	private	REJ	0.0	0.0	0	0.0	0.0	0.0	0.0	0
7	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
8	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
9	0.0	tcp	remot...	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
10	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
11	0.0	tcp	private	REJ	0.0	0.0	0	0.0	0.0	0.0	0.0	0
12	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
13	0.0	tcp	http	SF	287.0	2251.0	0	0.0	0.0	0.0	0.0	1
14	0.0	tcp	ftp_data	SF	334.0	0.0	0	0.0	0.0	0.0	0.0	1
15	0.0	tcp	name	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
16	0.0	tcp	netbio...	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
17	0.0	tcp	http	SF	300.0	13788.0	0	0.0	0.0	0.0	0.0	1
18	0.0	icmp	eco_i	SF	18.0	0.0	0	0.0	0.0	0.0	0.0	0
19	0.0	tcp	http	SF	233.0	616.0	0	0.0	0.0	0.0	0.0	1
20	0.0	tcp	http	SF	343.0	1178.0	0	0.0	0.0	0.0	0.0	1
21	0.0	tcp	mtp	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
22	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
23	0.0	tcp	http	SF	253.0	11905.0	0	0.0	0.0	0.0	0.0	1
24	5607.0	udp	other	SF	147.0	105.0	0	0.0	0.0	0.0	0.0	0
25	0.0	tcp	mtp	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
26	507.0	tcp	telnet	SF	437.0	14421.0	0	0.0	0.0	0.0	0.0	1
27	0.0	tcp	private	S0	0.0	0.0	0	0.0	0.0	0.0	0.0	0
28	0.0	tcp	http	SF	227.0	6588.0	0	0.0	0.0	0.0	0.0	1

Figure 3.6 : Capture de l'enregistrement NSL-KDD Train.

3.2.4.1. Transformation

Cette étape consiste à transformer les attributs symboliques en valeurs numériques. Dans NSL-KDD, il existe trois attributs non numériques : le type de protocole (3 symboles différents), le service (70 symboles différents) et le drapeau (11 symboles différents). Nous avons ensuite procédé à une transformation en valeurs numériques, comme le montrent les tableaux suivants :

Conversion Originale	Conversion
Tcp	0
Icmp	1
udp	2

Tableau 3.2 : Transforamtion de l'attribut Protocol_type. [54]

Les valeurs « aol », « auth » et « bgp » seront remplacées respectivement par « 0 », « 1 » et « 2 » pour l'attribut « service », qui comporte 70 catégories de valeurs différentes. La conversion se poursuivra par ordre alphabétique une fois que ces catégories auront été classées et les données de l'attribut Flag sont converties en utilisant le même principe.

Avant conversion	**Après Conversion**	**Avant conversion**	**Après Conversion**
OTH	0	S1	6
REJ	1	S2	7
RSTO	2	S3	8
RSTOS0	3	SF	9
RSTR	4	SH	10
S0		5	

Tableau 3.3 : La Transformation alphabétique des valeurs de l'attribut "flag"

3.2.4.2. Normalisation

Après l'opération de transformation, les valeurs obtenues sont extrêmement diverses et représentent un large éventail. Certains paramètres prennent des valeurs importantes (src_bytes, dst_bytes, etc.), tandis que d'autres ne prennent que des valeurs modérées (serror_rate, same_srvrate, etc.), ce qui peut compromettre la rentabilité du modèle de détection d'intrusions. Pour prévenir ce souci et assurer l'efficacité du modèle généré, il est nécessaire d'ajuster ou de normaliser les valeurs de la base de données. Dans notre situation, les données de la base NSL-KDD sont normalisées dans l'intervalle de [0, 1].

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_diff_srv_rate	dst_host_same_src_port_rat
0	0	2	44	9	146	0	0	0	0	0	...	0.60	0.8
1	0	1	49	5	0	0	0	0	0	0	...	0.05	0.0
2	0	1	24	9	232	8153	0	0	0	0	...	0.00	0.0
3	0	1	24	9	199	420	0	0	0	0	...	0.00	0.0
4	0	1	49	1	0	0	0	0	0	0	...	0.07	0.0
5	0	1	49	5	0	0	0	0	0	0	...	0.05	0.0
6	0	1	49	5	0	0	0	0	0	0	...	0.07	0.0
7	0	1	51	5	0	0	0	0	0	0	...	0.05	0.0
8	0	1	49	5	0	0	0	0	0	0	...	0.06	0.0
9	0	1	49	1	0	0	0	0	0	0	...	0.07	0.0
10	0	1	49	5	0	0	0	0	0	0	...	0.07	0.0
11	0	1	24	9	287	2251	0	0	0	0	...	0.00	0.1
12	0	1	20	9	334	0	0	0	0	0	...	0.00	1.0
13	0	1	36	5	0	0	0	0	0	0	...	0.07	0.0
14	0	1	38	5	0	0	0	0	0	0	...	0.06	0.0
15	0	1	24	9	300	13788	0	0	0	0	...	0.00	0.0
16	0	0	14	9	18	0	0	0	0	0	...	0.00	1.0
17	0	1	24	9	233	616	0	0	0	0	...	0.00	0.0
18	0	1	24	9	343	1178	0	0	0	0	...	0.00	0.0
19	0	1	35	5	0	0	0	0	0	0	...	0.05	0.0
20	0	1	49	5	0	0	0	0	0	0	...	0.06	0.0

Figure 3.7 : Normalisation de l'enregistrement NSL-KDD Train.

3.3. Implémentation et discussion des résultats

3.3.1. Définition d'environnement de programmation

a. Google Colab

Ce produit est développé par Google Research. Colab offre à tout le monde la possibilité de créer et d'exécuter du code Python de manière indépendante via le navigateur, et est particulièrement adapté à l'apprentissage automatique, à l'analyse de données et à l'enseignement.

Cet outil nous offre la possibilité de concevoir rapidement des applications de Machine Learning en Python. Dans un premier temps, il suffit d'avoir un compte Gmail pour commencer. [48]

Figure 3.8 : Logo de google colab. [53]

b. Python

Python est un langage open source de programmation orienté objet et multiplateforme. Les bibliothèques spécialisées de Python permettent d'utiliser Python dans de multiples domaines tels que le développement logiciel, l'analyse de données ou la gestion d'infrastructures. Il ne se concentre donc pas exclusivement sur la programmation web, contrairement au langage HTML par exemple. [49]

Figure 3.9 : Logo de language python. [53]

c. Sklearn

Nous avons opté pour l'utilisation de Sklearn, une bibliothèque d'apprentissage statique en Python qui regroupe toutes les fonctionnalités clés du Machine Learning. Il renferme les algorithmes les plus cruciaux ainsi que différentes fonctions de pré-processus. [49]

Les avantages offerts par Scikit-Learn comprennent :

- La régression (linéaire et logistique).
- Les voisins les plus proches sont classés.
- Différenciation, (K-Means et K-Means++).
- Choix de la conception.
- Première manipulation, (la normalisation Min-Max).

Figure 3.10 : Logo de Sklearn. [53]

d. Pandas

Pandas constitue une bibliothèque très efficace pour importer vos tableaux Excel (et d'autres formats) dans Python afin de générer des statistiques et de charger votre Dataset. [50]

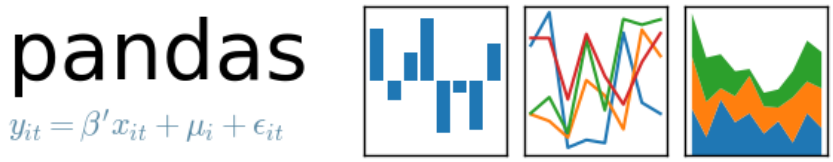

Figure 3.11 : Logo de Pandas. [53]

e. Matplotlib

La bibliothèque Matplotlib est l'outil qui permet de représenter nos Datasets, nos fonctions et nos résultats en utilisant des graphes, des courbes et des nuages de points. [51]

Figure 3.12 : Logo de Matplotlib. [53]

d. Seaborn

La bibliothèque Seaborn est utilisée pour générer des graphiques statistiques en Python. Elle repose sur la technologie Matplotlib et s'intègre parfaitement aux structures Pandas. [52]

Figure 3.13 : Logo de Seaborn. [53]

3.3.2. Test et Evaluation des résultats

Avant de classifier les données en ensembles de test et de formation. La performance de la précision est améliorée par l'application d'approches de sélection des caractéristiques. Dans chaque mélange, le classificateur est entraîné sur 80 % des données afin de refléter la véritable relation contrainte-déformation, et le modèle est validé à l'aide des 20 % de données restantes.

3.3.2.1. Les mésures d'évaluation

Afin d'évaluer l'efficacité du modèle de détection d'intrusions, on comparera les résultats obtenus avec les données réelles (données marquées). Toutes les données de Test NSL-KDD ont été étiquetées, ce qui signifie que la classe de chaque instance est connue. Chaque situation est considérée comme normale ou anomalie. [54]

- **La précision :** Cette mesure donne des informations sur la probabilité qu'une prévision pour une certaine catégorie soit exacte et s'applique également à chaque catégorie.

$$\textbf{Précision} = \frac{\text{TP}}{\text{TP} + \text{FB}} \times 100\%$$

- **Rappel :** Il s'agit de la corrélation entre le nombre total d'intrusions et le nombre d'intrusions identifiées avec précision. Elle peut être définie à l'aide de la formule suivante :

$$\textbf{Rappel} = \frac{\text{TP}}{\text{FN} + \text{TP}} \times 100\%$$

- **FP :** Le rapport entre les numéros de trafic réguliers qui sont classés par erreur comme des intrusions et le nombre total de numéros de trafic standard sert de base au calcul.

$$\textbf{FP} = \frac{\text{FP}}{\text{TN} + \text{FP}} \times 100\%$$

- **Accuracy :** indique l'efficacité de la méthode de détection. Il s'agit d'une mesure qui montre également la corrélation entre le nombre total d'objets détectés et les objets exacts.

$$\textbf{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{FP} + \text{FN} + \text{TP} + \text{TN}} \times 100\%$$

Où :

TP (Vrai positif) : Une attaque a été correctement repérée par l'analyse.

FP (Faux positif) : Le test a détecté une activité normale comme une attaque.

TN (Vrai négatif) : Le test a correctement détecté une activité normale.

FN (Faux négatif) : Une attaque a été identifiée comme une activité normale lors du test.

3.3.3. Définition du Modèle

Nous avons développé un modèle de détection d'intrusion basé sur les **Classificateurs Boosting, Bagging et Voting.** Ces méthodes globales combinent les prédictions de divers modèles de base afin d'améliorer les performances de classification.

Le boosting est une approche globale qui consiste à combiner plusieurs modèles faibles afin de créer un modèle robuste. La construction de chaque modèle se fait de manière chronologique en se focalisant sur les exemples mal classés par les modèles précédents. Ceci facilite la correction des erreurs et l'amélioration progressive des performances du modèle.

Le Bagging consiste à entraîner plusieurs modèles de base sur des échantillons aléatoires provenant du jeu de données d'entraînement. En intégrant de manière adéquate les prédictions de ces modèles, notre objectif est de diminuer la variance et d'améliorer la solidité du modèle final.

Le Voting est une méthode qui combine les prédictions de divers modèles de base en utilisant soit un vote majoritaire (Hard Voting) soit une moyenne pondérée des probabilités prédites. Grâce à cette méthode, nous pouvons exploiter les avantages de divers modèles de classification afin d'obtenir une décision finale plus précise et fiable.

Une fois que nos modèles auront été entraînés avec le jeu de données NSL-KDD train, nous les testerons sur le jeu de données NSL-KDD test 20 afin d'évaluer leur efficacité. Après avoir comparé les résultats obtenus avec chaque méthode globale, nous pourrons identifier celle qui offre les meilleures performances en matière de détection des intrusions dans les réseaux internet.

3.3.4. Chargements des données du dataset

Au début, les données du fichier CSV doivent être lues et chargées. La bibliothèque Pandas de Python contient des méthodes et des classes permettant de lire plusieurs formats de fichiers. Une fonction Pandas cruciale pour la lecture et la manipulation des fichiers CSV est read_csv.

```
[ ] import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    from pandas.api.types import is_numeric_dtype
    import warnings
    import sklearn
    from sklearn.ensemble import BaggingClassifier
    from sklearn import tree
    from sklearn.model_selection import train_test_split
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.linear_model import LogisticRegression
    from sklearn.ensemble import BaggingClassifier
    from sklearn.preprocessing import StandardScaler, LabelEncoder
    from sklearn.tree  import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, VotingClassifier, GradientBoostingClassifier
    from sklearn.svm import SVC, LinearSVC
    from sklearn.naive_bayes import BernoulliNB
    from lightgbm import LGBMClassifier
    from sklearn.feature_selection import RFE
    import itertools
    from xgboost import XGBClassifier
    from tabulate import tabulate
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
[ ] train=pd.read_csv('/content/drive/MyDrive/KDD/kddnsl80.csv')
```

```
[ ] test=pd.read_csv('/content/drive/MyDrive/KDD/kddnsl20.csv')
```

Figure 3.14 : Chargements du dataset sur google colab.

La Méthode Head :

La méthode head() retourne un nombre spécifié de colonnes, une string à partir du sommet, et retourne les 5 premières rows si un nombre n'est pas indiqué.

```
train.head()
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V32	V33	V34	V35	V36	V37	V38	V39	V40	V41
0	1	2	18	10	44	1	0	0	0	0	...	22	17	3	17	0	0	0	5	0	5
1	1	3	43	10	29	1	0	0	0	0	...	1	0	60	88	0	0	0	0	0	5
2	1	2	48	6	1	1	0	0	0	0	...	22	10	5	0	0	100	100	0	0	1
3	1	2	22	10	35	48	0	0	0	0	...	36	100	0	3	4	3	1	0	1	5
4	1	2	22	10	31	37	0	0	0	0	...	36	100	0	0	0	0	0	0	0	5

5 rows × 41 columns

Figure 3.15 : Exécution de la Méthode Head.

La Méthode SeabornCountplot :

On utilise la méthode seaborn.countplot() pour représenter les nombres d'observations dans chaque binage à l'aide de barres.

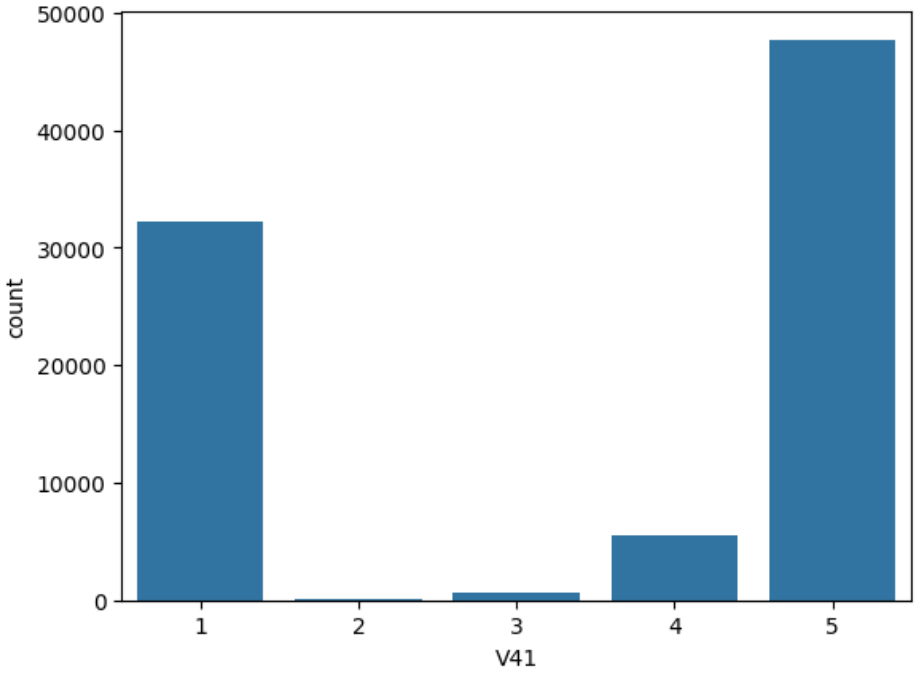

Figure 3.16 : Ensembles des attaques après Regroupement.

Regroupement des attaques en 5 classes principales :

Les Classes D'attaque				
Classe 1	Classe 2	Classe 3	Classe 4	Classe 5
Dos	U2R	R2L	Probe	Normal

Tableau 3.4 : Nouveaux classes d'attaque**.**

3.3.5. La mise en œuvre des Algorithmes

Cette partie présente une comparaison entre les méthodes les plus récentes employées pour la classification des données.

Nous posons la question : Qu'est-ce qui est la meilleur Méthode ?

Algorithme Boosting Classifier « AdaBoostClassifier »

```
models = []
models.append(('AdaBoostClassifier',AdaBoostClassifier()))

for name,model in models:

    # Adaptation de Naive Bayes à l'ensemble de formation
    classifier = model
    classifier.fit(x_train, y_train)

    # Prédire les résultats de l'ensemble de tests
    y_train_pred = classifier.predict(x_train)
    y_test_pred = classifier.predict(x_test)

    # Créer la matrice de confusion
    cm_train = confusion_matrix(y_train, y_train_pred)
    cm_test = confusion_matrix(y_test, y_test_pred)

    # Application de la validation croisée k-Fold
    accuracies_train = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv = 10)
    accuracies_test = cross_val_score(estimator = classifier, X = x_test, y = y_test, cv = 10)
    accuracies_train.mean()
    accuracies_train.std()
    accuracies_test.mean()
    accuracies_test.std()

    print()
    print("Pour {0} La Performance du Modèle est: ".format(name))
    print()

    #La performance de  classification model
    print(" Accuracy est: "+ str((cm_train[0,0]+cm_train[1,1])/(cm_train[0,0]+cm_train[0,1]+cm_train[1,0]+cm_train[1,1])))
    recall_train = cm_train[1,1]/(cm_train[0,1]+cm_train[1,1])
    print("Recall est : "+ str(recall_train))
    print("False Positive rate: "+ str(cm_train[1,0]/(cm_train[0,0]+cm_train[1,0])))
    precision_train = cm_train[1,1]/(cm_train[1,0]+cm_train[1,1])
    print("Precision est: "+ str(precision_train))
    print("F-measure est: "+ str(2*((precision_train*recall_train)/(precision_train+recall_train))))
    print("Entropy est: "+ str(-precision_train*log(precision_train)))

    print()
    print("La Performance du Modèle avec (Test20%) : ".format(name))
    print()

    print(" Accuracy est: "+ str((cm_test[0,0]+cm_test[1,1])/(cm_test[0,0]+cm_test[0,1]+cm_test[1,0]+cm_test[1,1])))
    recall_test = cm_test[1,1]/(cm_test[0,1]+cm_test[1,1])
    print("Recall est : "+ str(recall_test))
    print("False Positive rate: "+ str(cm_test[1,0]/(cm_test[0,0]+cm_test[1,0])))
    precision_test = cm_test[1,1]/(cm_test[1,0]+cm_test[1,1])
    print("Precision est: "+ str(precision_test))
    print("F-measure est: "+ str(2*((precision_test*recall_test)/(precision_test+recall_test))))
    print("Entropy est: "+ str(-precision_test*log(precision_test)))
```

```
Pour AdaBoostClassifier La Performance du Modèle est:

 Accuracy est: 0.9800385618691165
Recall est : 0.9754814269178155
False Positive rate: 0.014635346205878736
Precision est: 0.9873255937799552
F-measure est: 0.9813677747194579
Entropy est: 0.01259374442927411

La Performance du Modèle avec (Test20%) :

 Accuracy est: 0.9790949986768986
Recall est : 0.9784207945071113
False Positive rate: 0.020114942528735632
Precision est: 0.9827586206896551
F-measure est: 0.9805849102973704
Entropy est: 0.01709188507890596
```

Figure 3.17 : Exécution de la Méthode Boosting.

➢ **Les Résultats de la Méthode AdaBoostClassifier**

AdaBoostClassifier	Training (80%)	Test (20%)
Accuracy (%)	98.0039	97.9095
Recall (%)	97.5481	97.8421
Précision (%)	98.7326	98.2759
False Positive rate (%)	1.46353	2.01149
F-Measure (%)	98.1367	98.0584
Entropy (%)	1.25937	1.70919

Tableau 3.5 : Résultats de la Méthode Boosting.

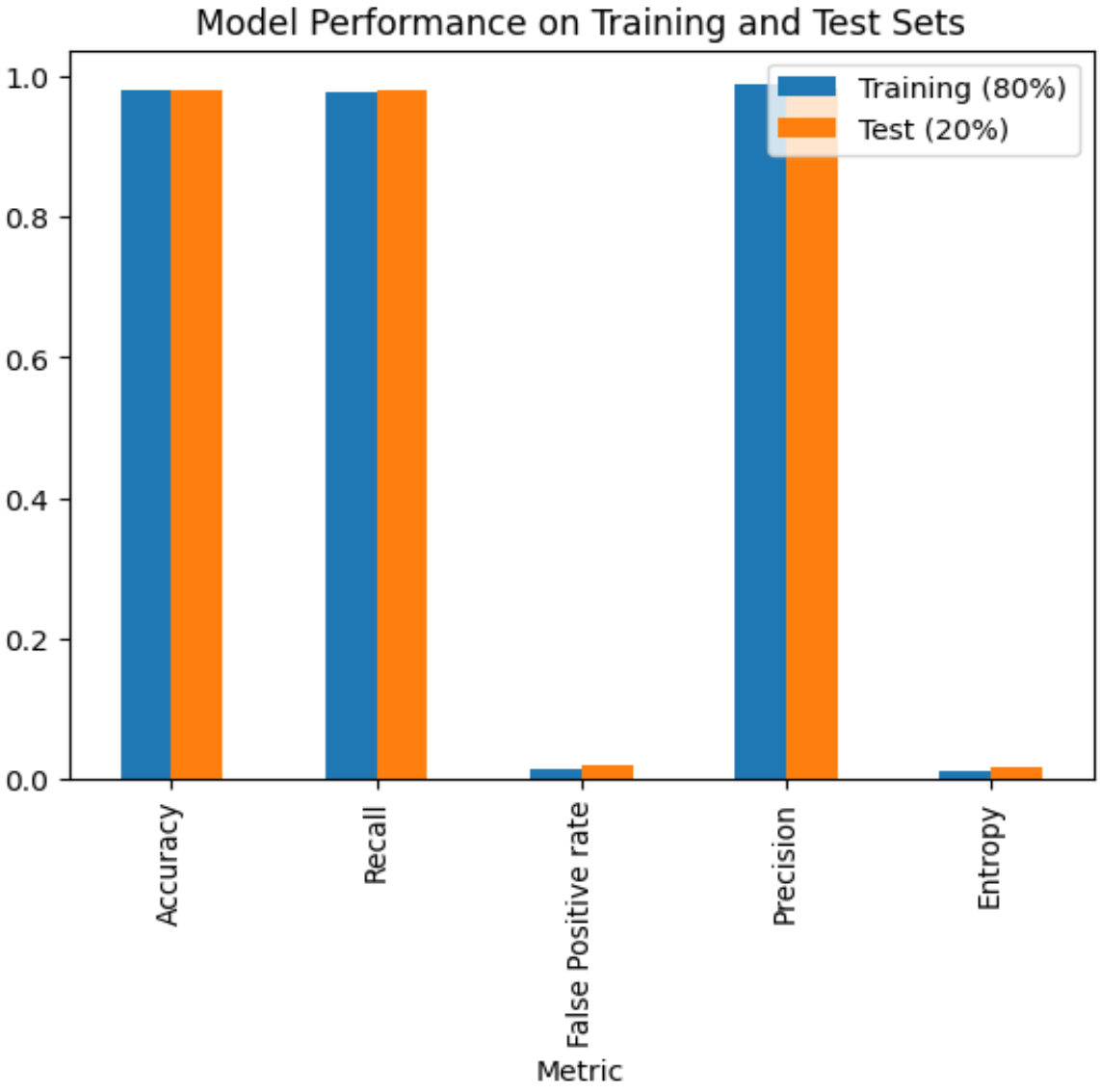

Figure 3.18 : Diagramme du Résultats de la Méthode Boosting.

Interprétation des Résultat de la Méthode Boosting

Accuracy : La proportion de prédictions correctes. Dans ce cas, le modèle a une précision de 98% sur l'ensemble d'entraînement et de 97% sur l'ensemble de test.

Recall : La proportion de cas positifs correctement identifiés. Dans ce cas, le modèle a un rappel de 97.54% sur l'ensemble d'entraînement et de 97.84% sur l'ensemble de test.

False Positive rate : La proportion de cas négatifs incorrectement identifiés comme positifs. Dans ce cas, le modèle a un taux de faux positifs de 1.46% sur l'ensemble d'entraînement et de 2% sur l'ensemble de test.

Précision : La proportion de prédictions positives qui sont correctes. Dans ce cas, le modèle a une précision de 98.73% sur l'ensemble d'entraînement et de 98.27% sur l'ensemble de test.

Entropie : Une mesure de l'incertitude d'un modèle. Dans ce cas, le modèle a une entropie de 1.25% sur l'ensemble d'entraînement et de 1.7% sur l'ensemble de test.

En général, ces résultats suggèrent que le modèle AdaBoostClassifier fonctionne bien sur les ensembles d'entraînement et de test, avec une exactitude, un rappel et une précision élevés. Cependant, le modèle a un taux de faux positifs relativement élevé, ce qui signifie qu'il est plus susceptible d'identifier incorrectement des cas négatifs comme positifs.

Algorithme Bagging Classifier

```
[ ]
    models = []
    models.append(('BaggingClassifier',BaggingClassifier()))


    for name,model in models:

        # Adaptation de Naive Bayes à l'ensemble de formation
        classifier = model
        classifier.fit(x_train, y_train)

        # Prédire les résultats de l'ensemble de tests
        y_train_pred = classifier.predict(x_train)
        y_test_pred = classifier.predict(x_test)

        # Créer la matrice de confusion
        cm_train = confusion_matrix(y_train, y_train_pred)
        cm_test = confusion_matrix(y_test, y_test_pred)

        # Application de la validation croisée k-Fold
        accuracies_train = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv = 10)
        accuracies_test = cross_val_score(estimator = classifier, X = x_test, y = y_test, cv = 10)
        accuracies_train.mean()
        accuracies_train.std()
        accuracies_test.mean()
        accuracies_test.std()

        print()
        print("Pour {0} La Performance du Modèle est: ".format(name))
        print()

        #La performance de  classification model
        print(" Accuracy est: "+ str((cm_train[0,0]+cm_train[1,1])/(cm_train[0,0]+cm_train[0,1]+cm_train[1,0]+cm_train[1,1])))
        recall_train = cm_train[1,1]/(cm_train[0,1]+cm_train[1,1])
        print("Recall est : "+ str(recall_train))
        print("False Positive rate: "+ str(cm_train[1,0]/(cm_train[0,0]+cm_train[1,0])))
        precision_train = cm_train[1,1]/(cm_train[1,0]+cm_train[1,1])
        print("Precision est: "+ str(precision_train))
        print("F-measure est: "+ str(2*((precision_train*recall_train)/(precision_train+recall_train))))
        print("Entropy est: "+ str(-precision_train*log(precision_train)))

        print()
        print("La Performance du Modèle avec (Test20%) : ".format(name))
        print()

        print(" Accuracy est: "+ str((cm_test[0,0]+cm_test[1,1])/(cm_test[0,0]+cm_test[0,1]+cm_test[1,0]+cm_test[1,1])))
        recall_test = cm_test[1,1]/(cm_test[0,1]+cm_test[1,1])
        print("Recall est : "+ str(recall_test))
        print("False Positive rate: "+ str(cm_test[1,0]/(cm_test[0,0]+cm_test[1,0])))
        precision_test = cm_test[1,1]/(cm_test[1,0]+cm_test[1,1])
        print("Precision est: "+ str(precision_test))
        print("F-measure est: "+ str(2*((precision_test*recall_test)/(precision_test+recall_test))))
        print("Entropy est: "+ str(-precision_test*log(precision_test)))
```

```
Pour BaggingClassifier La Performance du Modèle est:

 Accuracy est: 0.9996597482136781
Recall est : 0.9996804771541166
False Positive rate: 0.0003638568829593693
Precision est: 0.9996804771541166
F-measure est: 0.9996804771541166
Entropy est: 0.00031947179302110646

La Performance du Modèle avec (Test20%) :

 Accuracy est: 0.9949722148716592
Recall est : 0.9965432098765432
False Positive rate: 0.0068415051311288486
Precision est: 0.994088669950739
F-measure est: 0.9953144266337854
Entropy est: 0.005893823608255376
```

Figure 3.19 : Exécution de la Méthode Bagging.

➢ **Les Résultats de la Méthode BaggingClassifier**

BaggingClassifier	Training (80%)	Test (20%)
Accuracy (%)	99.966	99.4972
Recall (%)	99.968	99.6543
Précision (%)	99.968	99.4089
False Positive rate (%)	0.0363857	0.06841
F-Measure (%)	99.9680	99.5314
Entropy (%)	0.0319472	0.589382

Tableau 3.6 : Résultats de la Méthode Bagging.

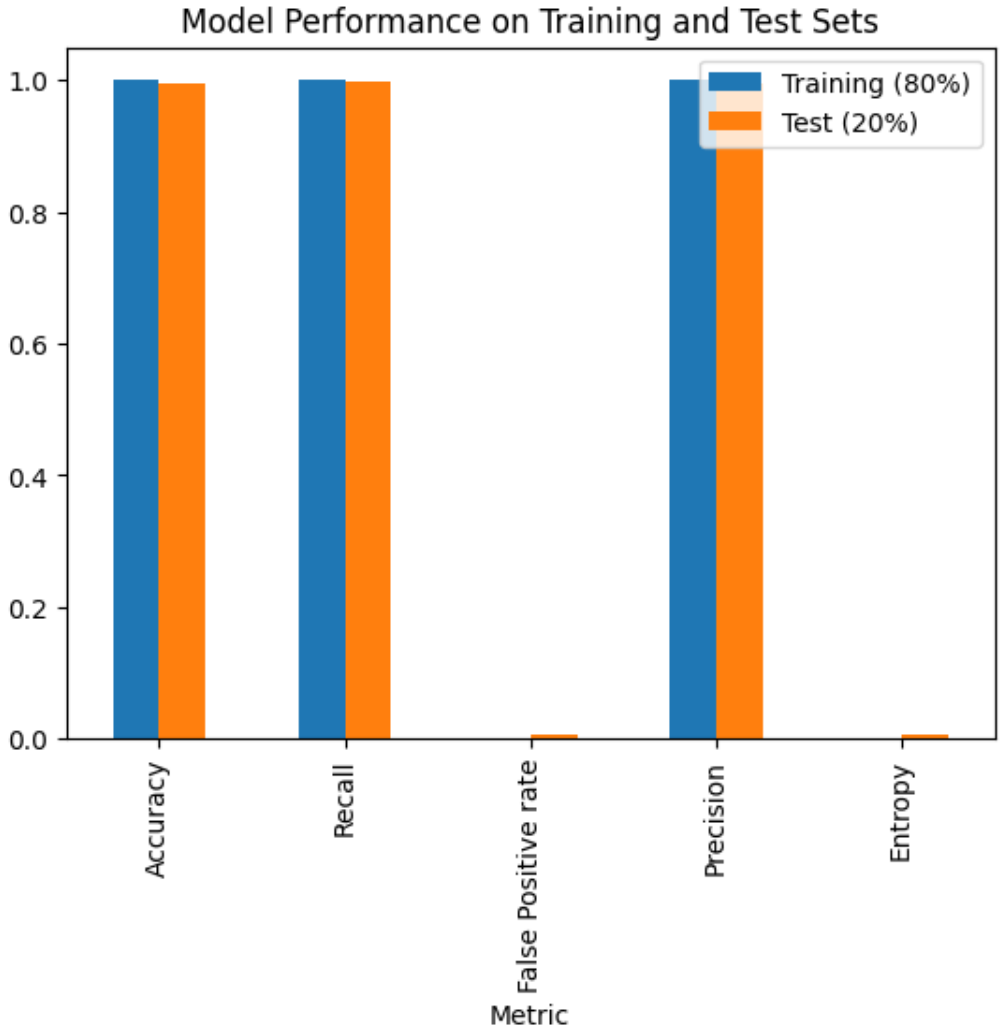

Figure 3.20 : Diagramme du Résultats de la Méthode Bagging.

Interprétation des Résultat de la Méthode Bagging

Accuracy : La proportion de prédictions correctes. Dans ce cas, le modèle BaggingClassifier a une précision de 99.96% sur l'ensemble d'entraînement et de 99.49% sur l'ensemble de test.

Recall : La proportion de cas positifs correctement identifiés. Dans ce cas, le modèle BaggingClassifier a un rappel de 99.96% sur l'ensemble d'entraînement et de 99.65% sur l'ensemble de test.

False Positive rate : La proportion de cas négatifs incorrectement identifiés comme positifs. Dans ce cas, le modèle BaggingClassifier a un taux de faux positifs de 0.036% sur l'ensemble d'entraînement et de 0.068% sur l'ensemble de test.

Précision : La proportion de prédictions positives qui sont correctes. Dans ce cas, le modèle BaggingClassifier a une précision de 99.96% sur l'ensemble d'entraînement et de 99.40% sur l'ensemble de test.

Entropie : Une mesure de l'incertitude d'un modèle. Dans ce cas, le modèle BaggingClassifier a une entropie de 0.031% sur l'ensemble d'entraînement et de 0.58% sur l'ensemble de test.

Ces résultats indiquent généralement que le modèle BaggingClassifier fonctionne efficacement sur les ensembles d'entraînement et de test, offrant une précision, une précision et une exactitude élevées. Toutefois, le modèle présente un taux assez élevé de faux positifs, ce qui implique qu'il est plus probable qu'il identifie incorrectement des cas négatifs comme positifs.

Algorithme VotingClassifier

```
models = []
models.append(('VotingClassifier', VotingClassifier(estimators=[('lr', AdaBoostClassifier()), ('rf', BaggingClassifier()), ('dt', RandomForestClassifier())])))

for name,model in models:

    # Adaptation de Naive Bayes à l'ensemble de formation
    classifier = model
    classifier.fit(x_train, y_train)

    # Prédire les résultats de l'ensemble de tests
    y_train_pred = classifier.predict(x_train)
    y_test_pred = classifier.predict(x_test)

    # Créer la matrice de confusion
    cm_train = confusion_matrix(y_train, y_train_pred)
    cm_test = confusion_matrix(y_test, y_test_pred)

    # Application de la validation croisée k-Fold
    accuracies_train = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv = 10)
    accuracies_test = cross_val_score(estimator = classifier, X = x_test, y = y_test, cv = 10)
    accuracies_train.mean()
    accuracies_train.std()
    accuracies_test.mean()
    accuracies_test.std()

    print()
    print("Pour {0} La Performance du Modèle est: ".format(name))
    print()

    #La performance de  classification model
    print(" Accuracy est: "+ str((cm_train[0,0]+cm_train[1,1])/(cm_train[0,0]+cm_train[0,1]+cm_train[1,0]+cm_train[1,1])))
    recall_train = cm_train[1,1]/(cm_train[0,1]+cm_train[1,1])
    print("Recall est : "+ str(recall_train))
    print("False Positive rate: "+ str(cm_train[1,0]/(cm_train[0,0]+cm_train[1,0])))
    precision_train = cm_train[1,1]/(cm_train[1,0]+cm_train[1,1])
    print("Precision est: "+ str(precision_train))
    print("F-measure est: "+ str(2*((precision_train*recall_train)/(precision_train+recall_train))))
    print("Entropy est: "+ str(-precision_train*log(precision_train)))

    print()
    print("La Performance du Modèle avec (Test20%) : ".format(name))
    print()

    print(" Accuracy est: "+ str((cm_test[0,0]+cm_test[1,1])/(cm_test[0,0]+cm_test[0,1]+cm_test[1,0]+cm_test[1,1])))
    recall_test = cm_test[1,1]/(cm_test[0,1]+cm_test[1,1])
    print("Recall est : "+ str(recall_test))
    print("False Positive rate: "+ str(cm_test[1,0]/(cm_test[0,0]+cm_test[1,0])))
    precision_test = cm_test[1,1]/(cm_test[1,0]+cm_test[1,1])
    print("Precision est: "+ str(precision_test))
    print("F-measure est: "+ str(2*((precision_test*recall_test)/(precision_test+recall_test))))
    print("Entropy est: "+ str(-precision_test*log(precision_test)))
```

```
Pour VotingClassifier La Performance du Modèle est:

 Accuracy est: 0.9997731654757854
Recall est : 0.999786984769411
False Positive rate: 0.00024257125530624622
Precision est: 0.999786984769411
F-measure est: 0.999786984769411
Entropy est: 0.00021299254123368285

La Performance du Modèle avec (Test20%) :

 Accuracy est: 0.9947075946017465
Recall est : 0.9938574938574939
False Positive rate: 0.0043004587155963305
Precision est: 0.9963054187192119
F-measure est: 0.995079950799508
Entropy est: 0.0036877478946769206
```

Figure 3.21 : Exécution de la Méthode Voting.

➢ **Les Résultats de la Méthode VotingClassifier**

VotingClassifier	Training (80%)	Test (20%)
Accuracy (%)	99.9773	99.4708
Recall (%)	99.9787	99.3857
Précision (%)	99.9787	99.6305
False Positive rate (%)	0.0242571	0.430046
F-Measure (%)	99.9786	99.5079
Entropy (%)	0.0212993	0.368775

Tableau 3.7 : Résultats de la Méthode Voting.

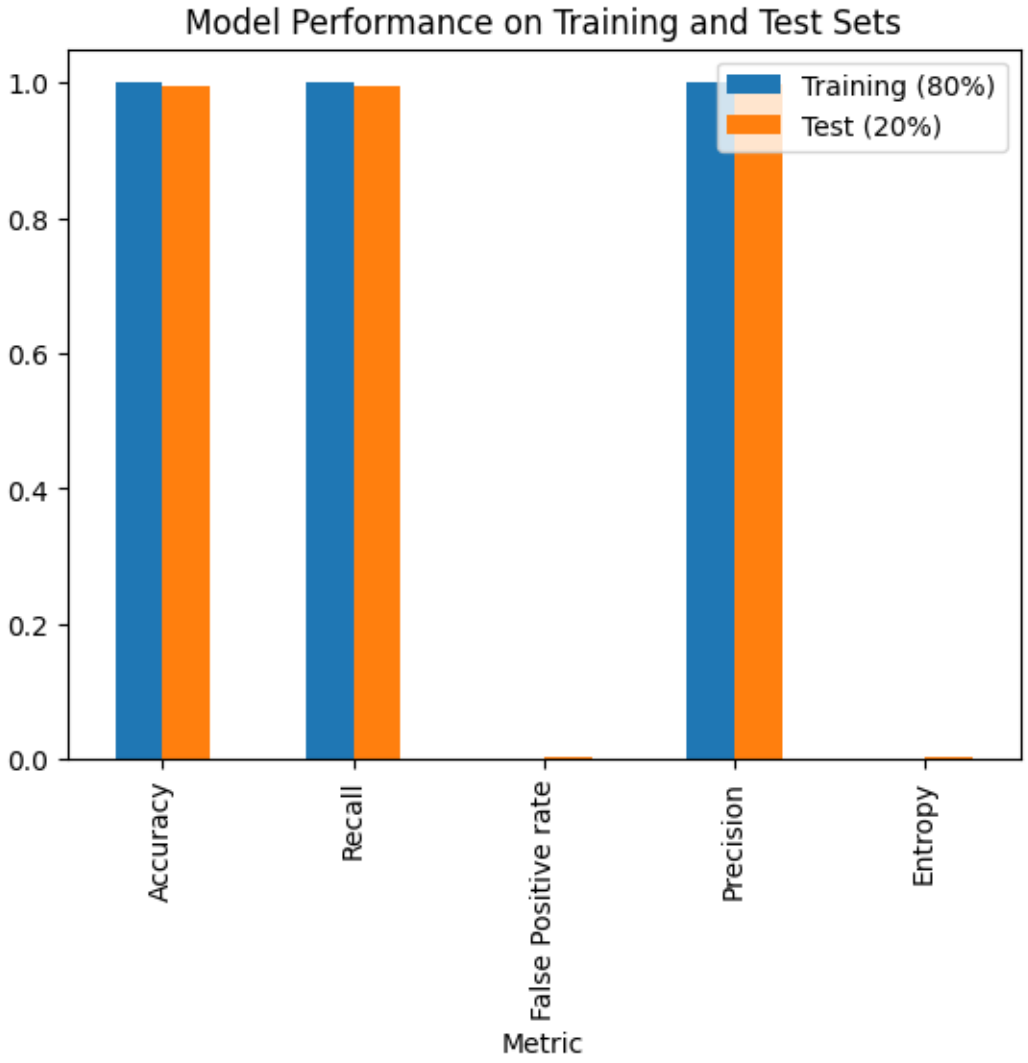

Figure 3.22 : Diagramme du Résultats de la Méthode Bagging.

Interprétation des Résultat de la Méthode Voting

Accuracy : Le taux de précision mesure la proportion de cas correctement classés. Dans le cas du VotingClassifier, l'exactitude sur l'ensemble d'entraînement est de 99.97% et l'exactitude sur l'ensemble de test est de 99.47%. Cela signifie que le modèle a correctement classé 99.97% des cas dans l'ensemble d'entraînement et 99.47 % des cas dans l'ensemble de test.

Recall : Le rappel mesure la proportion de cas positifs correctement classés. Dans le cas du VotingClassifier, le rappel sur l'ensemble d'entraînement est de 99.98% et le rappel sur l'ensemble de test est de 99.38%. Cela signifie que le modèle a correctement classé les cas positifs dans l'ensemble d'entraînement et les cas positifs dans l'ensemble de test.

False Positive rate : Le taux de faux positifs mesure la proportion de cas négatifs incorrectement classés comme positifs. Dans le cas du VotingClassifier, le taux de faux positifs sur l'ensemble d'entraînement est de 0.024% et le taux de faux positifs sur l'ensemble de test est de 0.43%.

Précision : La précision mesure la proportion de cas classés comme positifs qui sont réellement positifs. Dans le cas du VotingClassifier, la précision sur l'ensemble d'entraînement est de 99.97% et la précision sur l'ensemble de test est de 99.63%.

Entropie : L'entropie mesure le degré de désordre ou d'incertitude dans un ensemble de données. Dans le cas du VotingClassifier, l'entropie sur l'ensemble d'entraînement est de 0.21% et l'entropie sur l'ensemble de test est de 0.37%.

- Le VotingClassifier a obtenu des résultats similaires à ceux de l'AdaBoostClassifier et du BaggingClassifier, avec une exactitude et rappel légèrement supérieur.
- Le modèle est relativement certain de ses prédictions et a un taux de faux positifs relativement faible.
- Le VotingClassifier est un modèle d'ensemble qui combine les prédictions de plusieurs modèles de base. Dans ce cas, les modèles de base sont l'AdaBoostClassifier, le BaggingClassifier et le RandomForestClassifier.
- En combinant les prédictions de ces modèles, le VotingClassifier peut obtenir de meilleures performances que chacun des modèles de base pris individuellement.

Comparaison des Résultats avec d'autres Méthodes

Dans cette partie, on fait une comparaison entre les résultats des méthodes Boosting, Baggin et Voting avec les résultats d'autres méthodes telles que : SVM, Décision Tree, KNN, MLP.

```
models = []
models.append(('KNeighborsClassifier', KNeighborsClassifier()))
models.append(('SVM', SVC()))
models.append(('Decison Tree', DecisionTreeClassifier()))
models.append(('MLPForestClassifier', MLPClassifier()))

for name,model in models:

    # Adaptation de Naive Bayes à l'ensemble de formation
    classifier = model
    classifier.fit(x_train, y_train)

    # Prédire les résultats de l'ensemble de tests
    y_train_pred = classifier.predict(x_train)
    y_test_pred = classifier.predict(x_test)

    # Créer la matrice de confusion
    cm_train = confusion_matrix(y_train, y_train_pred)
    cm_test = confusion_matrix(y_test, y_test_pred)

    # Application de la validation croisée k-Fold
    accuracies_train = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv = 10)
    accuracies_test = cross_val_score(estimator = classifier, X = x_test, y = y_test, cv = 10)
    accuracies_train.mean()
    accuracies_train.std()
    accuracies_test.mean()
    accuracies_test.std()

    print()
    print("Pour {0} La Performance du Modèle est: ".format(name))
    print()

    #La performance de  classification model
    print(" Accuracy est: "+ str((cm_train[0,0]+cm_train[1,1])/(cm_train[0,0]+cm_train[0,1]+cm_train[1,0]+cm_train[1,1])))
    recall_train = cm_train[1,1]/(cm_train[0,1]+cm_train[1,1])
    print("Recall est : "+ str(recall_train))
    print("False Positive rate: "+ str(cm_train[1,0]/(cm_train[0,0]+cm_train[1,0])))
    precision_train = cm_train[1,1]/(cm_train[1,0]+cm_train[1,1])
    print("Precision est: "+ str(precision_train))
    print("F-measure est: "+ str(2*((precision_train*recall_train)/(precision_train+recall_train))))
    print("Entropy est: "+ str(-precision_train*log(precision_train)))

    print()
    print("La Performance du Modèle avec (Test20%) : ".format(name))
    print()

    print(" Accuracy est: "+ str((cm_test[0,0]+cm_test[1,1])/(cm_test[0,0]+cm_test[0,1]+cm_test[1,0]+cm_test[1,1])))
    recall_test = cm_test[1,1]/(cm_test[0,1]+cm_test[1,1])
    print("Recall est : "+ str(recall_test))
    print("False Positive rate: "+ str(cm_test[1,0]/(cm_test[0,0]+cm_test[1,0])))
    precision_test = cm_test[1,1]/(cm_test[1,0]+cm_test[1,1])
    print("Precision est: "+ str(precision_test))
    print("F-measure est: "+ str(2*((precision_test*recall_test)/(precision_test+recall_test))))
    print("Entropy est: "+ str(-precision_test*log(precision_test)))
```

Figure 3.23 : Code d'exécution des Méthodes.

```
Pour KNeighborsClassifier La Performance du Modèle est:

 Accuracy est: 0.9872972666439832
Recall est : 0.9884873680844259
False Positive rate: 0.014055494971525506
Precision est: 0.9876451166258388
F-measure est: 0.9880660628662761
Entropy est: 0.012278245531687583

La Performance du Modèle avec (Test20%) :

 Accuracy est: 0.9830643027255888
Recall est : 0.9844751108920651
False Positive rate: 0.018571428571428572
Precision est: 0.9839901477832512
F-measure est: 0.9842325695984232
Entropy est: 0.01588100507626506

Pour SVM La Performance du Modèle est:

 Accuracy est: 0.9683565838720654
Recall est : 0.957991909552951
False Positive rate: 0.0191417490304016
Precision est: 0.9837043348599425
F-measure est: 0.9706778770362586
Entropy est: 0.016162163638994097

La Performance du Modèle avec (Test20%) :

 Accuracy est: 0.9708917703096057
Recall est : 0.9615384615384616
False Positive rate: 0.017657445556209534
Precision est: 0.9852216748768473
F-measure est: 0.9732360097323601
Entropy est: 0.014668583737685356

Pour MLPForestClassifier La Performance du Modèle est:

 Accuracy est: 0.9849722127707837
Recall est : 0.9869769427839453
False Positive rate: 0.017299782240503266
Precision est: 0.9847694110128874
F-measure est: 0.9858719411419736
Entropy est: 0.015114010198971477

La Performance du Modèle avec (Test20%) :

 Accuracy est: 0.9850489547499338
Recall est : 0.9866831072749692
False Positive rate: 0.016842706251784186
Precision est: 0.9854679802955665
F-measure est: 0.98607516943931
Entropy est: 0.014425914679288627

Pour Decison Tree La Performance du Modèle est:

 Accuracy est: 0.9948399047367028
Recall est : 0.996787744007907
False Positive rate: 0.00740529763600114
Precision est: 0.9935960591133005
F-measure est: 0.9951893425434809
Entropy est: 0.006383391745137936

La Performance du Modèle avec (Test20%) :

 Accuracy est: 0.9945752844667901
Recall est : 0.9967861557478368
False Positive rate: 0.007970395673213778
Precision est: 0.993103448275862
F-measure est: 0.9949413942011105
Entropy est: 0.006872715652542252
```

Figure 3.24 : Les Résultats d'exécution des Méthodes.

Model (Train 80%).	Accuracy	Recall	Précision	False Positif	Entropie
AdaBoostClassifier (%)	98.00	97.54	98.73	1.46	1.25
BaggingClassifier (%)	99.96	99.96	99.96	0.03	0.031
VotingClassifier (%)	99.97	99.97	99.97	0.02	0.021
KNN (%)	98.72	98.84	98.76	98.80	1.22
SVM (%)	96.83	95.79	98.37	97.06	1.61
MLP (%)	98.44	98.62	98.45	98.53	1.51
Decision Tree (%)	99.5	99.4	99.7	99.5	0.6

Tableau 3.8 : Les Résultats d'exécution des Méthodes avec NSL-KDD Train 80%.

Model (Test 20%).	Accuracy	Recall	Précision	False Positif	Entropie
AdaBoostClassifier (%)	97.90	97.84	98.27	2.01	1.70
BaggingClassifier (%)	99.49	99.65	99.40	0.06	0.58
VotingClassifier (%)	99.47	99.38	99.63	0.43	0.36
KNN (%)	98.30	98.44	98.39	98.42	1.58
SVM (%)	97.08	96.15	98.52	97.32	1.46
MLP (%)	98.46	98.54	98.59	98.57	1.44
Decision Tree (%)	99.35	99.2	99.38	99.26	0.8

Tableau 3.9 : Les Résultats d'exécution des Méthodes avec NSL-KDD Test 20%.

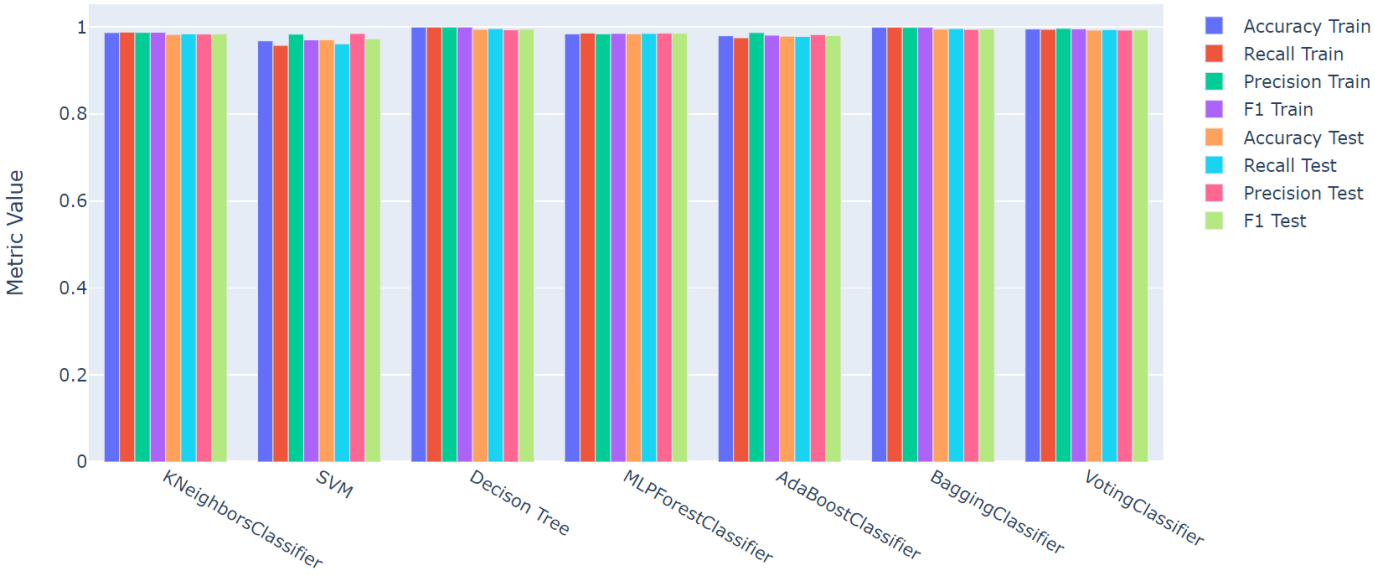

Figure 3.25 : Diagramme Représente Les Résultats d'exécution des Méthodes.

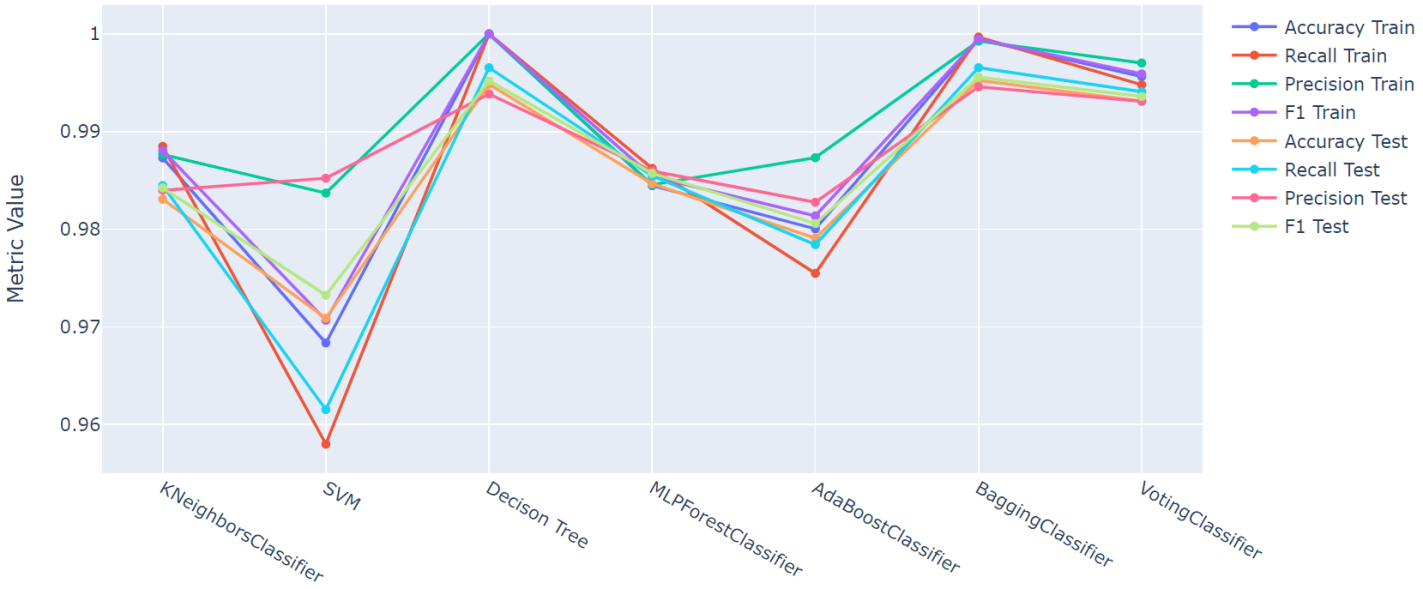

Figure 3.26 : Diagramme Représente Les Résultats d'exécution des Méthodes.

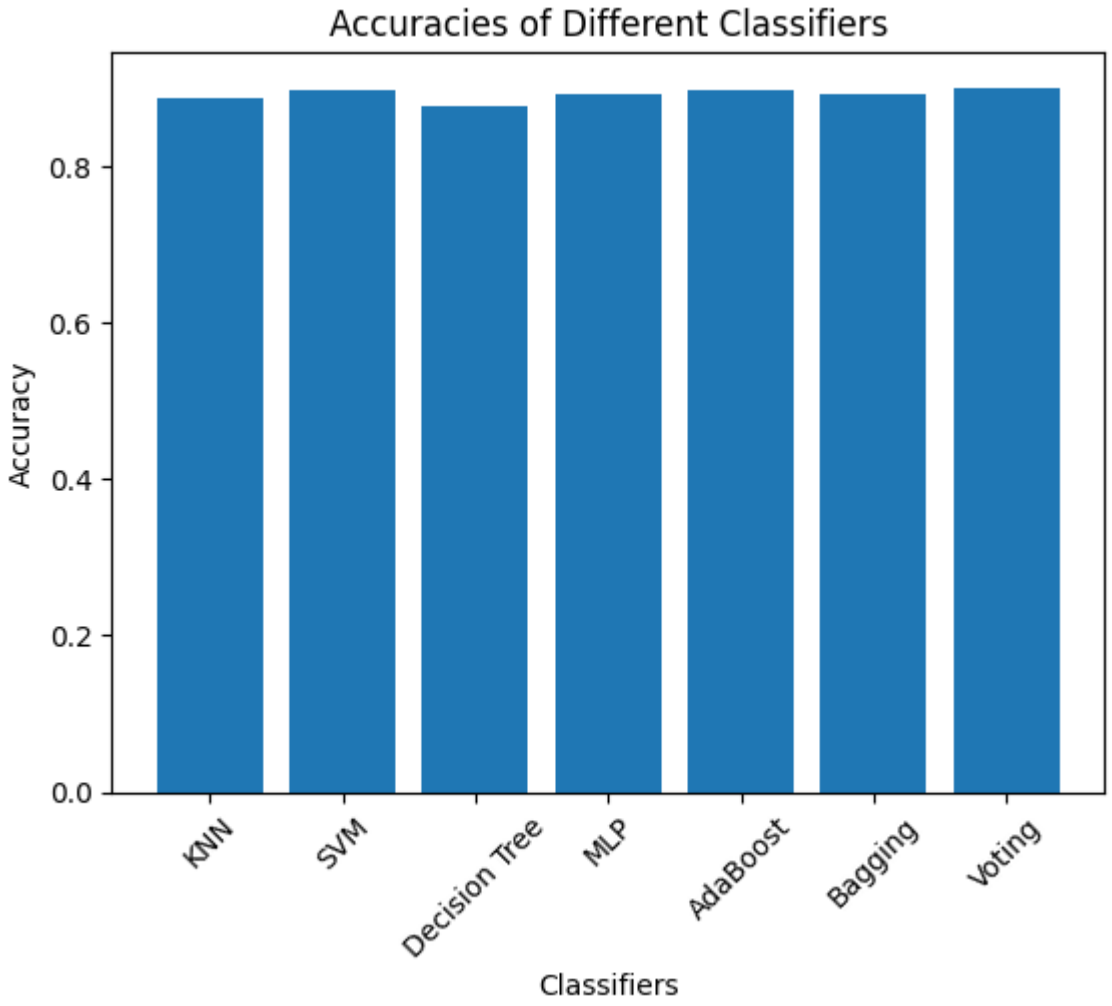

Figure 3.27 : Diagramme Représente Accuracy.

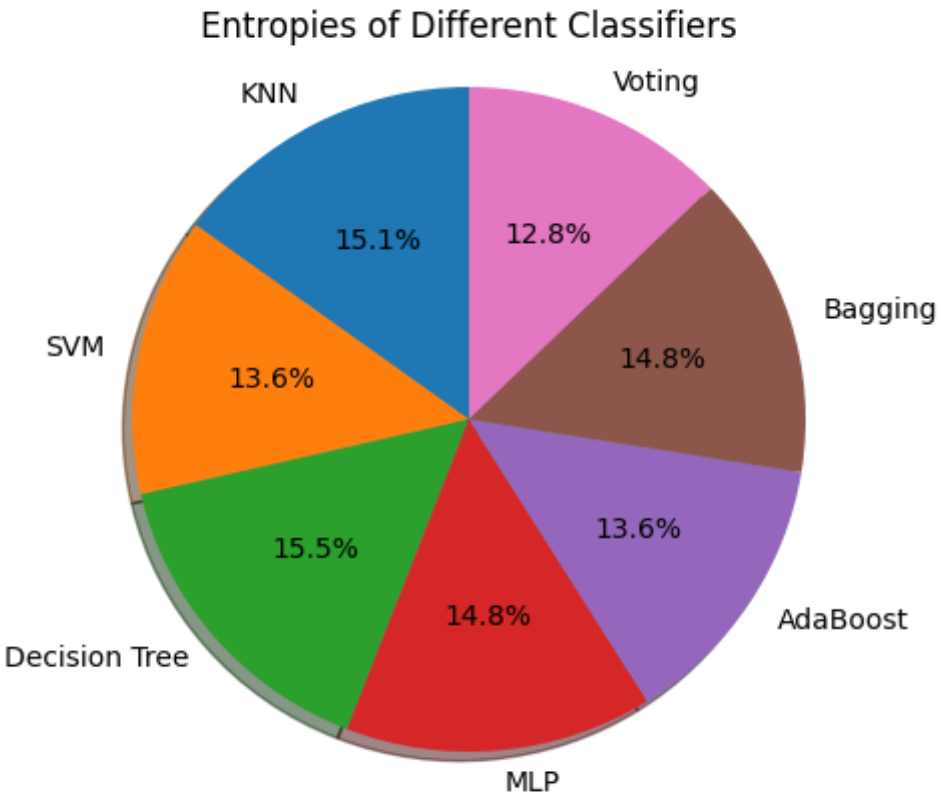

Figure 3.28 : Diagramme circulaire Représente Entropies.

Interprétation Générale des Résultat

Les performances de différents modèles de classification sur un ensemble de données sont présentées par sept modèles de classification différents :

AdaBoost, Bagging, VotingClassifier, Decision Tree, KNN, MLP et SVM., avec une répartition de 80% pour l'entraînement et 20% pour les tests. Les critères de performance incluent l'exactitude, le rappel, la précision, le taux de faux positifs et l'entropie.

Sur l'ensemble d'entraînement (80%) :

AdaBoostClassifier : Avec une précision de 98%, ce modèle présente de très bonnes performances dans l'ensemble. Sa capacité de rappel et sa précision sont également élevées, atteignant respectivement 97,54% et 98,73%. Néanmoins, il présente un taux de faux positifs de 1,46%, ce qui suggère qu'il commet quelques erreurs en classant des échantillons négatifs comme positifs.

BaggingClassifier : Ce modèle présente une précision remarquable de 99,96 %. Il possède également un rappel et une précision très élevés, avec des valeurs identiques de 99,96%. En outre, il présente un taux très bas de faux positifs, à seulement 0.03 %.

VotingClassifier : Ce modèle utilise différents modèles pour prendre des décisions, ce qui lui permet d'atteindre une précision de 99,97%. Il présente également une excellente capacité de rappel, de précision et de taux de faux positifs, tous inférieurs à 0,03%.

KNN (K-Nearest Neighbors) : Ce modèle est fiable avec une précision de 98,72%. Le taux de rappel, de précision et de faux positifs est proche, ce qui témoigne d'un bon équilibre entre le rappel et la précision.

SVM (Machine Vector Support) : Ce modèle présente une précision de 96,83%, ce qui est légèrement moins élevé que les autres modèles. Il possède une grande capacité de rappel et de précision, cependant, son taux de faux positifs est assez élevé, atteignant 1,61%.

MLP (Perceptron Multilayer) : Avec une précision de 98,44%, ce modèle démontre des performances solides. Sa capacité de mémorisation et de précision sont élevées, mais son taux de faux positifs est légèrement supérieur à celui des autres modèles.

Decision Tree : Ce modèle présente une précision de 99,5%, avec des performances élevées en matière de rappel et de précision. Le pourcentage de faux positifs est également très bas, avec seulement 0,6 %.

Sur l'ensemble de test (20%) :

Les résultats des modèles sur l'ensemble de test sont en général similaires à ceux de l'ensemble d'entraînement, même s'ils sont légèrement moins bons que prévu. Les performances élevées sont maintenues par les modèles BaggingClassifier et VotingClassifier, suivis de près par le Decision Tree.

Les classificateurs KNeighborsClassifier, SVM, Decision Tree, MLPClassifier, AdaBoostClassifier et BaggingClassifier ont tous obtenu des performances similaires en termes d'exactitude, de rappel, de précision et d'entropie. Cependant, le VotingClassifier a pu obtenir des résultats légèrement meilleurs en combinant les prédictions de ces classificateurs individuels.

Les résultats montrent que le VotingClassifier a obtenu les meilleures performances en termes d'exactitude, de rappel, de précision et d'entropie. Cela suggère que la combinaison de plusieurs classificateurs peut conduire à de meilleurs résultats que l'utilisation d'un seul classificateur.

Le VotingClassifier est un méta-classificateur qui combine les prédictions de plusieurs classificateurs de base. Cela peut contribuer à améliorer les performances globales du classificateur en réduisant le risque de surajustement et en améliorant la généralisation du modèle.

Dans le cas présent, le classificateur de vote a obtenu de meilleurs résultats que n'importe lequel des classificateurs de base individuels. Cela suggère que la combinaison de ces classificateurs a été en mesure de capturer différents aspects des données et de faire des prédictions plus précises.

Dans l'ensemble, les résultats de cette étude suggèrent que la combinaison de plusieurs classificateurs peut conduire à de meilleurs résultats que l'utilisation d'un seul classificateur. Cela est particulièrement vrai lorsque les classificateurs individuels sont capables de saisir différents aspects des données.

On conclut donc que :

- Les classificateurs classiques sont généralement plus simples à comprendre et à interpréter que les classificateurs d'ensemble.
- Les classificateurs classiques peuvent être moins précis que les classificateurs d'ensemble, en particulier sur des données complexes ou bruitées.
- Les classificateurs d'ensemble combinent les prédictions de plusieurs classificateurs classiques pour obtenir une prédiction plus précise.
- Les classificateurs d'ensemble sont généralement plus précis que les classificateurs classiques, en particulier sur des données complexes ou bruitées.
- Les classificateurs d'ensemble peuvent être plus difficiles à comprendre et à interpréter que les classificateurs classiques.

Dans l'ensemble, le choix du meilleur classificateur dépend des données spécifiques et de la tâche à accomplir. Si les données sont simples et non bruitées, un classificateur classique peut être suffisant. Si les données sont complexes ou bruitées, un classificateur d'ensemble peut être nécessaire pour obtenir une meilleure précision.

3.4. Conclusion

Ce dernier chapitre examine la base de données NSL-KDD qui a été utilisée pour tester et former le modèle de détection des intrusions qui a été développé via des méthodes d'optimisation de type boosting, voting et bagging afin d'augmenter les valeurs du modèle.

Nous avons également présenté les langages et les outils que nous avons utilisés pour mettre notre modèle en pratique. Des expériences ont été menées afin de justifier les performances de ces algorithmes, ainsi que VotingClassifier, qui a été correctement adaptée et a obtenu des résultats améliorés. De plus, les résultats ont démontré l'importance des algorithmes d'ensembles dans l'amélioration de performance du modèle généré.

Conclusion Générale

Ces dernières années, les attaques informatiques se sont multipliées et représentent désormais une menace sérieuse pour les systèmes d'information, les réseaux informatiques et les applications des entreprises. En d'autres termes, en mettant en place les mesures préventives nécessaires, la détection rapide, La répétition et la permanence de ces attaques peuvent favoriser leur diminution. C'est ce qui nous a incité à créer dans ce projet un modèle de système de détection d'intrusion afin de reconnaître les attaques et d'en prévenir les dommages.

Le système de détection d'intrusion basé sur la machine learning développé dans ce projet s'est avéré efficace pour identifier les attaques réseau avec une précision élevée. Les modèles de classification utilisés, notamment le VotingClassifier, le BaggingClassifier, ont démontré leur capacité à apprendre des caractéristiques des données et à faire des prédictions précises.

L'utilisation de techniques d'apprentissage automatique a permis au système de s'adapter aux changements dans les données et d'améliorer sa performance au fil du temps. Cela est particulièrement important dans le contexte de la sécurité réseau, où les attaques évoluent constamment.

Il est important de souligner que notre travail vise à étudier et à concevoir un modèle pour un IDS comportemental. En utilisant les méthodes d'apprentissage automatique Boosting, Baggin et Voting. Nous avons créé un modèle de classification multi-classes (avec cinq classes : Normal, DoS, Probe, U2R et R2L) pour catégoriser les connexions TCP/IP en utilisant les données de la base de données de référence NSL-KDD.

Grâce à ce projet, nous avons pu approfondir notre compréhension du domaine de l'apprentissage automatique. De plus, grâce à cela, nous avons pu explorer et tester de nouvelles bibliothèques Python conçues spécifiquement pour ce domaine. Il est important de comprendre le fonctionnement de divers algorithmes d'apprentissage automatique, tels que les diverses architectures du Machine Learning. Il est cependant important de noter que le système n'est pas parfait et qu'il peut arriver que des attaques ne soient pas repérées ou que des fausses alarmes soient produites. Il est donc crucial de persévérer dans l'amélioration du système et de le surveiller attentivement afin d'assurer sa fiabilité et son efficacité à long terme.

Bien que la plupart des objectifs fixés dans cette étude ont été réalisés, mais il reste encore des perspectives et des améliorations potentielles qui peuvent encore être accomplies à l'avenir, telles que :

- Évaluer l'efficacité de méthodes de classification alternatives, telles que LightGBM, XGBoost et Gradient Boosting.
- Pensez à améliorer la précision de la détection en utilisant des techniques d'apprentissage profond.
- Développer des modèles capables de s'adapter en permanence aux nouvelles informations et menaces.
- Utiliser des techniques d'apprentissage en ligne pour mettre à jour les modèles en temps réel.
- Développer des interfaces et des outils pour intégrer les modèles d'apprentissage automatique dans les systèmes de sécurité existants.

Bibliographie

[1] Cedric Michel and Ludovic Me. ADeLe: an Attack Description Language for Knowledgebased Intrusion Detection. In Proceedings of the 16th IFIP International Conference on Information Security (IFIP/SEC 2001), pages 353–365, June 2001.

[2] Marcus A. Maloof (2005), _Machine Learning and Data Mining for Computer Security_, Springer London Ltd, ISBN-10 184628029X; ISBN-13 978-1846280290**.**

[3] Skurichina, M. (2001). Stabilizing weak classi_ers. PhD thesis, Delft University of Technology, Delft, The Netherlands…. Page 2 star 2.

[4] Avnimelech, R. and Intrator, N. (1999). Boosted mixture of experts: an ensemble learning scheme. Neural Compu-tation, 11:475{490}.

[5] Efron, B. and Tibshirani, R. (1993). An intro- duction to the bootstrap. Chaman and Hall.

[6] Leo Breiman and Leo Breiman. Bagging predictors. In Machine Learning, pages 123–140, 1996.

[7] Brijesh, Soni. Boost-your-machine-learning-models-with-bagging-a-powerful-ensemble-learning-technique.

[8] Derkaoui, Amina et Derkaoui, Soumia.Etude comparative des méthodes ensemblistes de classification des données médicales, Université Tlemcen.

[9] Kosha, Mehta. Professionnel Dynamique. Statistics guides, Bagging.

[10] Gilles Blanchard. « Théorie du Boosting - partie II ». Fraunhofer FIRST.IDA Berlin, Allemagne. 28 Mai 2008.

[11] Sunil, Ray. Quick Introduction to Boosting Algorithms in Machine Learning, 2022.

[12] Changhyun Kim. Korea Advanced Institute of Science and Technology — Business & Technology Management (Ph.D), Ensemble Learning — Voting and Bagging, 2022.

[13] Imamitsingh. voting-classifiers-in-machine-learning-a532935fe592.

[14] Emna BAHRI. Amélioration des méthodes adaptatives pour l'apprentissage supervisé des données réelles. PhD thesis, Université lyon lumière 2, Faculté des Sciences Economiques et de Gestion, 2010.

[15] Thomas G Dietterich. Ensemble methods in machine learning. In Multiple classifier systems, pages 1–15. Springer, 2000.

[16] Priya Pedamkar. Ensemble Learning: Bagging vs Boosting, 2023.

[19] S. Zaafouri. "La sécurité du matériel informatique". Juin 2014

[25] Md. Al Mehedi Hasan et Biprodip Pal et Mohammed Nasser et Shamim Ahmad "Intrusion Detection Using Combination of Various Kernels Based Support Vector Machine", International Journal of Scientific & Engineering Research, Volume 4, Issue 9, September-2013.

[26] Sapna S. Kaushik, Dr. Prof.P.R. Deshmukh," Detection of Attacks in an Intrusion Detection System", International Journal of Computer Science and Information Technologies, Vol. 2 (3), 2011, 982-986.

[27] A. Imad. " Les systèmes de détections d'intrusion basés sur machine learning". Mémoire Master, Université de Bejaïa. (2018).

[28] Sapna S. Kaushik, Dr. Prof.P. R. Deshmukh," Detection of Attacks in an Intrusion Detection System", International Journal of Computer Science and Information Technologies, Vol. 2 (3), 2011, 982-986.

[29] S. Paliwal and R. Gupta, "Denial-of-service, probing \& remote to user (R2L) attack detection using genetic algorithm," International Journal of Computer Applications, vol. 60, no. 19, pp. 57--62, 2012.

[31] S. Sharma and R.K. Gupta, Int. J. Secur. Its Appl. 9, 69 (2015).

[32] Aditya Nur Cahyo; Risanuri Hidayat; Dani Adhipta. "Performance comparison of intrusion detection system-based anomaly detection using artificial neural network and support vector machine ". AIP Conf. Proc. 1755, 070011 (2016).

[33] Ahmad, H., Uppal, M., & Javed, M. (2014). An Overview of Intrusion Detection System (IDS) along with its Commonly Used Techniques and Classifications.

[34] Chokshi, H. "HIDS vs NIDS: Unravelling the Differences in Intrusion Detection Systems. " Neumetric. (2023, August 17).

[36] Dr. N. KHERNANE. "Les Systèmes de détection/prévention d'Intrusion (IDS/IPS)". Cours : Cyber Sécurité 1. Université Batna 2. (2022)

[39] T**.** Azeddine. "Détection d'intrusions dans les réseaux LAN :
Installation et configuration de l'IDS-SNORT". Mémoire Master, Université de Bejaïa. (2016).

[40] G. Hiet, "Détection d'intrusions paramètre par la politique de sécurité grâce au contrôle collaboratif des flux d'informations au sein du système d'exploitation et des applications : mise en œuvre sous Linux pour les programmes Java", Décembre 2008.

[43] Nilă.C et Patriciu.V et Bica.I "MACHINE LEARNING DATASETS FOR CYBER SECURITY APPLICATIONS" YEAR III, ISSUE 3, P.P. 109-112 (2019)

[44] Pierre Borneet all, Les réseaux de neuronesprésentation et applications, Edition TECHNIP, France 2007.

[45] B. Mouloud, K. Zakaria "Détection d'intrusions à base des réseaux de neurones et algorithmes génétiques". Mémoire Master. Université Tlemcen 2017.

[46] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1-6). Ieee.
[47] Intrusion detection using deep sparse auto-encoder and self-taught learning - Scientific Figure on ResearchGate. “Voir le 28 Apr, 2024”
[54] Beneddine Mustapha Mohamed Amine, Abid Malika. “Conception d’un IDS basé sur le Deep Learning et RBN”. Mémoire Master. Université Tiaret 2021.

Webographie

[17] https://www.mailinblack.com/ressources/blog/quelles-sont-les-attaques-informatiques-les-plus-frequentes/. Gretten, J., & Gretten, J. (2023, November 8). "Cyberattaques : définition, types et exemples. Mailinblack". Voir le "25/03/2024"

[18] https://www.crowdstrike.fr/cybersecurity-101/it-security/. "Définition de la sécurité informatique – CrowdStrike. (2022, November 15). crowdstrike.fr". Voir le "25/03/2024"

[20] https://www.kaspersky.fr/resource-center/definitions/encryption. (2023, Septembre 25). "Qu'est-ce que le chiffrement des données ? Définition et explication. ". Voir le "25/03/2024"

[21] https://www.cisa.gov/news-events/news/understanding-anti-virus-software. CISA. "Understanding Anti-Virus Software ", 2009. Voir le "26/03/2024"

[22] https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html. (2024, February 22). "What Is a Firewall ? ". Voir le "26/03/2024"

[23] https://www.hackerone.com/knowledge-center/what-vulnerability-assessment-benefits-tools-and-process. "What Is Vulnerability Assessment? Benefits, Tools, and Process | HackerOne". Voir le "27/03/2024"

[24] https://www.rapid7.com/fundamentals/types-of-attacks/. "Types of Cyber Attacks | Hacking Attacks & Techniques". Voir le "27/03/2024"

[30] https://www.fortinet.com/fr/resources/cyberglossary/types-of-cyber-attacks "Les différents types de cyberattaques | Fortinet. (n.d.). Fortinet". Voir le "30/03/2024"

[35] https://bunny.net/academy/security/what-is-network-intrusion-detection-nids/. "What is Network Intrusion Detection System (NIDS)?". Voir le "31/03/2024"

[37] https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-detection-system-ids "What is an Intrusion Detection System? (n.d.). Palo Alto Networks" Voir le "01/04/2024"

[38] https://www.tutorialspoint.com/difference-between-hids-and-nids. "Différences between HIDs and NIDs" Voir le "01/04/2024"

[41] www.linkedin.com. https://www.linkedin.com/advice/0/what-pros-cons-signature-based-vs-anomaly-based. "What are the pros and cons of signature-based vs. anomaly-based detection? ". (2023, March 14). Voir le "03/04/2024"

[42] https://byjus.com/maths/data-sets/). "*Datasets | Definition, Types, Properties and Examples*". A. (2022, December 13. Voir le "25/04/2024"

[48] " *https://research.google.com/colaboratory/faq.html?hl=fr*". Voir le "01/05/2024"

[49] "Python" "https://www.journaldunet.fr/web-tech/dictionnaire-duwebmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/". Voir le "01/05/2024"

[50] " Pandas". "https://pandas.pydata.org/". Voir le "01/05/2024"

[51] "https://cahier-de-prepa.fr/mp1-janson/download?id=526". Voir le "01/05/2024"

[52] "seaborn: statistical data visualization — seaborn 0.13.2 documentation (pydata.org) ". Voir le "01/05/2024"

[53] "https://www.wikipedia.org/". Voir le "01/05/2024".