الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université 20 août 1955 de Skikda
Faculté des Sciences
Département d'Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de master

Un système de détection d'intrusion pour la Cybersécurité

Filière
Informatique

Spécialité
Réseau et system distribué

Présenté par
Amdjed Djouama

Supervisé par
PR. Smaine Mazouzi

Année Universitaire 2022 – 2023

Résumé

Le domaine en évolution de la cybersécurité présente un champ de bataille dynamique dans lequel s'affronte les cybers criminels et les experts en sécurité. Les intrusions sont devenues une préoccupation majeure dans le cyberespace. Différentes méthodes sont employées pour faire face à ces menaces, mais il y a eu un besoin plus que jamais d'actualiser les méthodes traditionnelles depuis les rudiments approches telles que les listes noires et les listes blanches mises à jour manuellement. Une autre méthode consiste à la création manuelle de règles, il s'agit généralement de l'une des méthodes les plus courantes à ce jour.

De nombreux travaux similaires impliquent l'intégration de l'apprentissage automatique et l'intelligence artificielle dans les systèmes d'intrusion dans les hôtes et au niveau des infrastructures réseaux. Au début, cela présentait des solutions de faible précision, mais la croissance exponentielle dans le domaine de l'apprentissage automatique au cours de la dernière décennie a conduit à de vastes améliorations dans l'apprentissage des algorithmes et leurs exigences.

Dans le présent, travail nous appliquons l'algorithme du k plus proches voisins ainsi que l'algorithme de forêt aléatoire à un système de détection d'intrusion déployé sur le réseau informatique cela est dans le but d'améliorer la précision du système de détection d'intrusion.

ABSTRACT

The evolving field of cybersecurity presents a dynamic battlefield in which cyber criminals and security experts face off. Intrusions have become a major concern in cyberspace. Various methods are employed to deal with these threats, but there has been a need more than ever to update traditional methods from the rudimentary approaches such as manually updated blacklists and whitelists. Another method is to manually create rules, this is usually one of the most common methods today.

Many similar works involve the integration of machine learning and artificial intelligence in intrusion systems in hosts and at the level of network infrastructures. At first, this presented low-precision solutions, but the exponential growth in the field of machine learning over the past decade has led to vast improvements in learning algorithms and their requirements.

In the present work, we apply the k-nearest-neighbor algorithm and the random forest algorithm to an intrusion detection system deployed on the computer network with the aim of improving the accuracy of the detection system. of intrusion.

خلاصة

يقدم مجال الأمن السيبراني المتطور ساحة معركة ديناميكية يواجه فيها مجرمو الإنترنت وخبراء الأمن. أصبحت عمليات الاقتحام مصدر قلق كبير في الفضاء السيبراني. يتم استخدام طرق مختلفة للتعامل مع هذه التهديدات ، ولكن كانت هناك حاجة أكثر من أي وقت مضى لتحديث الأساليب التقليدية من الأساليب البدائية مثل القوائم السوداء والقوائم البيضاء المحدثة يدويًا. هناك طريقة أخرى تتمثل في إنشاء القواعد يدويًا ، وعادةً ما تكون هذه إحدى أكثر الطرق شيوعًا اليوم.

تتضمن العديد من الأعمال المماثلة تكامل التعلم الآلي والذكاء الاصطناعي في أنظمة الاقتحام في المضيفين وعلى مستوى البنى التحتية للشبكة. في البداية ، قدم هذا حلولًا منخفضة الدقة ، لكن النمو الهائل في مجال التعلم الآلي على مدار العقد الماضي أدى إلى تحسينات هائلة في خوارزميات التعلم ومتطلباتها.

في العمل الحالي ، قمنا بتطبيق خوارزمية k- أقرب الجيران وخوارزمية الغابة العشوائية على نظام كشف التسلل المنشور على شبكة الكمبيوتر بهدف تحسين دقة نظام الكشف عن التطفل.

Remerciements

Je tiens à remercier tout d'abord ALLAH le tout puissant de m'avoir donné le courage, la patience et la santé pour réaliser ce modeste travail.

Je suis infiniment reconnaissant envers mon encadreur, le Professeur Mazouzi Smaine, pour son accompagnement tout au long de mon travail de recherche. Sa patience, sa disponibilité et ses conseils avisés ont été d'une aide inestimable pour me permettre de mener à bien notre projet. Je suis honoré d'avoir eu la chance de travailler avec lui et suis très reconnaissants pour la richesse de ses enseignements et sa contribution à notre réflexion. Je garderai en mémoire les leçons précieuses qu'il nous a transmises et sa générosité en tant que mentor. Enfin, je suis fier de pouvoir présenter ce travail sous sa direction.

Je suis reconnaissant envers les membres du jury d'avoir évalué mon travail de recherche.

J'exprime ma gratitude envers ma famille et amis pour leur soutien et son encouragement infaillible. Enfin, un grand merci à tous ceux qui m'ont aidés directement ou indirectement à mener à bien ce travail.

Dédicace

A ma famille, mes parents, mon frère et mes sœurs,

Votre aide a été un facteur clé de mon succès, et je ne pourrai jamais vous remercier assez pour votre générosité et votre soutien. J’espère avoir l'occasion de vous rendre la pareille dans le futur.

À mes oncles surtout Smaine et Rabah.

Je tiens à vous remercier sincèrement pour votre précieuse aide et votre soutien tout au long de mon parcours éducatif. Votre contribution a été inestimable pour la réalisation de ce mémoire de fin d'études.

Grâce à votre soutien, j'ai pu surmonter les défis et les obstacles, et réaliser cette grande réussite. Je suis reconnaissant(e) pour vos conseils, votre expertise, votre encouragement et votre patience.

Je vous adresse mes plus sincères remerciements et mes meilleurs vœux pour l'avenir.

Cordialement,

Amdjed.

Tables des matières

Liste des figures:

Liste des tableaux :

Liste des abréviations :

APWG : Groupe de travail anti-hameçonnage (Anti-Phishing working group)
URL : Localisateur universel de ressources (Universal Resource Locator)
DOS : Déni de service (Denial of Service)
R2L : Distance à l'utilisateur (Remote to User)
U2R : Utilisateur vers Racine (User to Root)
SVM : Machine à vecteur de support (Support Vector Machine)
KNN : K Voisin le plus proche (K Nearest Neighbor)
ROC : Caractéristiques de fonctionnement du récepteur (Reciever Operating Characteristic)

Introduction générale

La cybersécurité est un problème croissant à l'époque moderne en raison de la croissance rapide et le progrès technologique. Internet fournit toutes les connaissances qui ont été accumulé par l'homme et avec l'avènement de l'informatique mobile à la portée de tous, les cyberattaques et les cybercrimes sont devenus trop populaires. Un rapport d'un groupe de travail anti phishing a montré qu'environ 227 000 détections de logiciels malveillants se produisent quotidiennement qui est lié à plus de 20 millions de nouveaux malwares par jour.

Les malwares peuvent être simplement définis en tant que programme informatique créé pour nuire à un système informatique (Kaspersky 2017).

Dans le passé, il existait une méthode simple pour lutter contre les logiciels malveillants, mais au cours des deux dernières décennies, il y a eu une évolution des cyberattaques et comment les failles sont exploitées, en parallèle les techniques de cybersécurité subissent également une évolution vers des approches plus intelligentes.

Le principal problème qui découle de la croissance de la technologie et de l'internet est le niveau de compétence faible requis pour mener une attaque sur un ordinateur cible à son insu. Des scripts automatisés et des programmes sophistiqués capables de contourner les mesures de sécurité sont facilement accessibles à quiconque souhaite mener une attaque, et les attaques de cybercriminels peu qualifiés sont en augmentation (Aliyev, 2010).

Un rapport de 2016 de l'APWG a montré que des milliards de dollars américains ont été perdus à cause d'attaques par hameçonnage, et que 42,71 % de ces attaques visaient le secteur du commerce de détail. Des attaques de telle ampleur contre les infrastructures commerciales d'un pays peuvent considérablement paralyser la croissance des entreprises. Ce rapport a également montré que les États-Unis hébergeaient le plus grand nombre de sites d'hameçonnage et que la Chine était la plus touchée par l'hameçonnage.

Cette situation a motivé la recherche d'une amélioration des applications de cybersécurité afin d'atténuer la perte de données personnelles et de réduire les dommages causés par les

cyberattaques, car une cyberattaque bien coordonnée peut causer des dommages considérables à une entreprise. Les méthodes traditionnelles en place ne peuvent pas suivre les innovations rapides qui se produisent dans le domaine de la cybercriminalité. Un exemple de méthode traditionnelle pour atténuer les attaques par hameçonnage est l'utilisation de listes noires. Une liste noire est une liste de d'URL nuisibles qui sont organisés et mises à jour par une société de sécurité telle que : Avast. Un réseau bloque toutes les URL figurant sur la liste noire et autorise toutes les autres URL à circuler sur le réseau. En 2016, 300.000 sites web d'hameçonnage ont été signalés chaque mois, ce qui pose un problème à l'entreprise de sécurité qui crée ces listes noires généralement en deux phases : premièrement, les sites d'hameçonnage qui sont susceptible de réussir à attaquer un système avant d'être signalés comme illégitimes et inscrits sur la liste noire, car tous les sites d'hameçonnage sont créés pour imiter des sites légitimes, deuxièmement, le nombre de nouveaux sites confirmé comme étant des sites d'hameçonnage en règle générale il est très difficile de suivre l'évolution du nombre de nouveaux sites d'hameçonnage.

Le moyen le plus efficace de s'attaquer à ce problème croissant consiste à utiliser des algorithmes d'apprentissage automatique afin de pouvoir détecter les attaques avant qu'elles ne se produisent. Il existe déjà un grand nombre de journaux (logs) des réseaux contenant des traces d'attaques passées, et ceux-ci peuvent être introduits dans l'algorithme pour l'entraîner à reconnaître les attaques et à alerter le système de sécurité pour qu'il prenne des mesures de prévention.

L'objectif de ce travail est d'améliorer le taux de détection d'un ensemble de cyberattaques en tirant parti de la sélection des caractéristiques de ces derniers et en évaluant quelle combinaison de caractéristiques fournirait la meilleure précision de classification en utilisant la méthode de validation croisée en K fois et l'algorithme de la forêt aléatoire.

CHAPITRE 1 : Contexte théorique

Introduction

Dans ce chapitre, un aperçu général sur la cybersécurité sera présenté. Ca consiste à un ensemble de définitions des notions et des techniques du domaine, ainsi que des explications nécessaires pour comprendre la détection d'intrusion et les outils associés. Il sera question également de présenter dans ce chapitre les types d'attaques et intrusions ainsi que les systèmes de détection d'intrusions (IDS : Intrusion Detection Systems) et les travaux connexes.

1. Types d'attaques

Afin de comprendre la raison d'un multitude de types d'attaques, il est nécessaire savoir que la plateforme cible et l'intention derrière l'attaque sont les principaux facteurs décisifs à l'origine du type d'attaque auquel un système serait exposé à un moment donné, par exemple par exemple, une attaque de phishing (type d'attaque) serait effectuée sur le Web ou par email (plateforme d'attaque) et est destiné à voler des informations personnelles telles que des mots de passe et des informations de carte crédit bancaire. Certaines autres attaques visent simplement à perturber le flux du réseau et constituer une nuisance ou faire de la publicité de marchandises.

Voici quelques-unes de ces classes d'attaques :

- Attaques par déni de service (DOS). L'attaquant occupe les ressources du système (mémoire et CPU) et les engage en permanence pour empêcher les utilisateurs disposant d'un accès légitime les utiliser (Abliz, 2011).

  Types d'attaque : Dos, Terre, Neptune, Pod, Schtroumpf.

- Attaques à distance vers l'utilisateur (R2L). L'attaquant envoie des paquets sur le Web (à distance) pour exposer les vulnérabilités d'un réseau ou d'une machine et tente d'exploiter les privilèges d'un système auquel il n'a pas physiquement accès (S.Paliwal, 2012).

  Types d'attaque : FTP_write, guess_passwd, imap, multihop, phf, espion,client warez, maître warez

- Utilisateur vers super-utilisateur (U2R). Ceci est généralement effectué par un utilisateur légitime ou un attaquant qui a accédé à un réseau ou à un système de bas

niveau ou accès et privilèges standard. L'attaque ici implique l'attaquant essayant illégalement d'élever leurs privilèges d'utilisateur et d'obtenir un super utilisateur privilèges en recherchant les vulnérabilités du système. C'est parfois en raison d'une négligence de la part de l'administrateur du système, comme la non modification des informations de connexion administrateur par défaut sur un nouvel équipement (S.Paliwal, 2012).

Types d'attaque: Buffer_overload, Loadmodule, Perl, Rootkit

- Sonde : Une attaque par sonde implique simplement que l'attaquant analyse le système pour les vulnérabilités à exploiter. Ces vulnérabilités incluent les ports ouverts, retour portes et mauvais mots de passe (V. Shmatikov, 2007). C'est une attaque facile pour les attaquants de bas niveau qui utilisent simplement un logiciel de piratage créé par d'autres, ces types d'attaquants sont connus sous le nom de Kiddies.

  Types d'attaque. IPsweep, Nmap, Satan

- Attaques d'hameçonnage : Les attaques de phishing sont effectuées en créant des clones de sites Web légitimes afin d'inciter les utilisateurs d'un site Web légitime à entrer leurs identifiants de connexion, informations bancaire ou privées telles que leur numéro de sécurité sociale. Il s'agit d'un type d'attaque qui exploite plusieurs vulnérabilités en une seule attaque comme les erreurs dans une URL, la plupart des utilisateurs ne remarqueraient pas si le moteur de recherche populaire "www.google.com" a été remplacé par "www.goggle.com". Les URL similaires et l'apparence du site Web sont utilisés en tandem pour tromper les cibles. Ces URL peuvent être jointes dans un email.

Types d'attaque : Spear phishing, Clone Phishing, Chasse à la baleine

Tableau 1 : classes d'attaques et types observés

Classe d'attaque	Nom d'attaque
Déni de service (DOS)	Dos
	Atterrir
	Neptune
	Sous
	Smurf
	Larme
Attaques à distance vers l'utilisateur (R2L)	FTP_ecriture
	Geuss_passwd
	Imap
	Multi-sauts
	Phf
	Espionner
	Warezclient
	Warezmaster
Utilisateur vers super-utilisateur (U2R)	Débordement de tompon
	Module de charge
	Perle
	Rootkit
Sonde	Lpsbalayage
	Nmap
	Satan

2. Types de systèmes de détection d'intrusion

Les systèmes de détection d'intrusion sont des méthodes et des applications créées pour évaluer et protéger les ordinateurs, les réseaux, les programmes et les données contre les attaques, les accès non autorisés, les lecture/écriture et suppression/corruption. Ceux ci peuvent être divisés en hôte et en réseau selon où et comment il est déployé. Les systèmes de détection d'intrusion ne doivent pas être confondu avec d'autres mesures de sécurité pouvant exister dans un réseau ou un système, telles que pare feux et antivirus. Les systèmes de détection d'intrusion fonctionnent en combinaison avec ceux-ci comme ils sont plus étroitement liés à la détection et au déclenchement des alarmes.

Les intrusions peuvent se produire à la fois en interne ou en externe en raison de cela les système de détection utilisent une configuration de détection d'intrusion hybride.

Les systèmes d'intrusion basés sur l'hôte et sur le réseau détectent les intrusions à l'aide d'un de deux méthodes ou parfois une combinaison des deux :

2.1 Basé sur l'hôte

Ce type de système de détection d'intrusion est principalement utilisé pour surveiller l'environnement d'un hôte ; ressources, système de fichiers et applications de l'hôte. Un système basé hôte surveille à la fois l'état et le comportement de son hôte. Grâce à cela, il peut détecter le comportement inhabituel des programmes et des fichiers (M. Gupta, 2015).

Même si les systèmes de détection d'intrusion basés sur l'hôte ont été principalement créés pour surveillance interne il existe quelques variantes du système de détection d'intrusion basé sur l'hôte qui peut également détecter les intrusions sur le réseau (AP Singh, 2016).

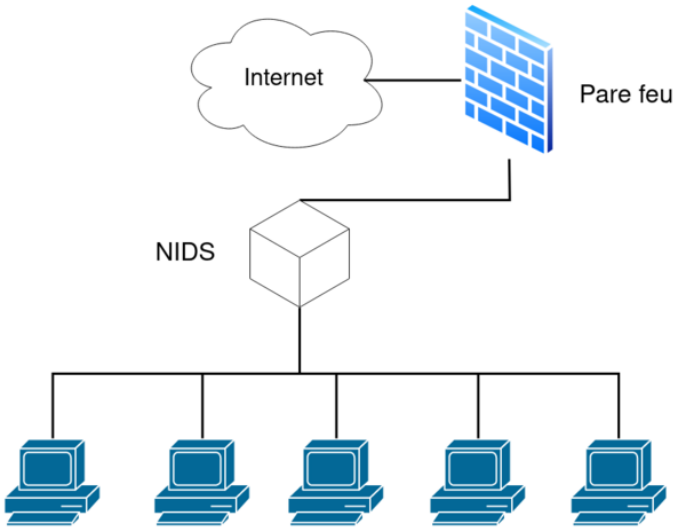


Figure 1.1 Configuration d'un système de détection d'intrusion basé sur l'hôte

2.2 Basé sur le réseau

La principale différence entre le système basé sur le réseau et le système basé sur l'hôte réside dans la manière dont il est déployé. Il existe en tant que partie du réseau pour détecter les tentatives d'intrusion sur le réseau. Le système de détection des intrusions basé sur le réseau analyse le trafic entrant et sortant à la recherche de modèles en dehors du comportement attendu (J. G. Noraini, 2011).

Il est à noter qu'un système de détection des intrusions basé sur le réseau ne remplace pas un pare-feu dans un système.

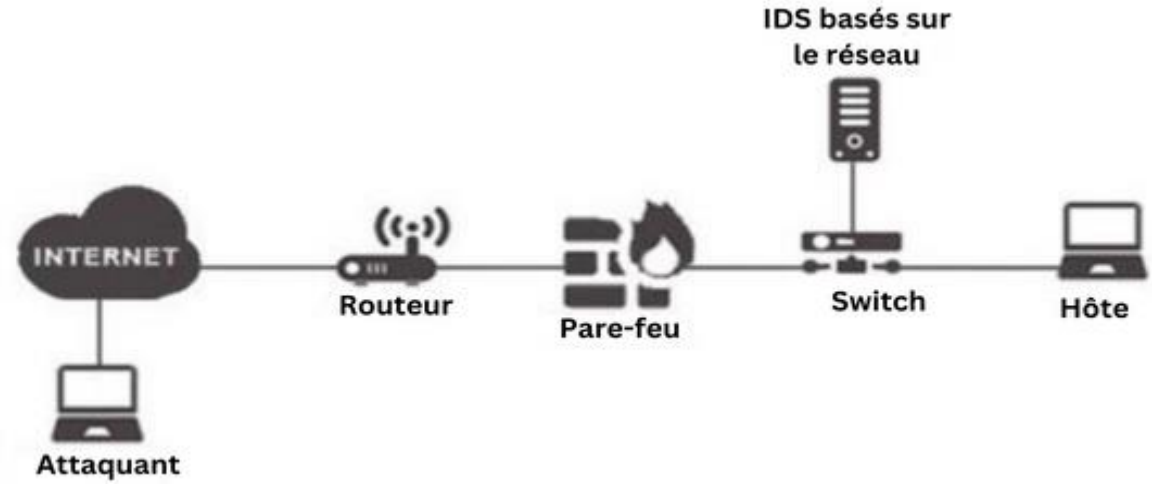


Figure 1.2 Configuration d'un système de détection d'intrusion basé sur le réseau.

3. Types d'approche de détection d'intrusion

Les systèmes de détection d'intrusion abordent le problème de la détection de deux manières fondamentales quel que soit le type, basé sur l'hôte ou sur le réseau. L'approche est soit basée sur la signature ou basée sur l'anomalie. Dans d'autres cas, il est possible de trouver des systèmes utilisant un système hybride.

3.1 Basé sur l'utilisation abusive (basé sur la signature)

Les attaques ont généralement des signatures telles que des métadonnées et des empreintes de fichiers (MD5 ou hachage SHA1), ces signatures peuvent être utilisées pour déterminer si une attaque est en cours ou le comportement observé du système est normal. Ceci est accompli en comparant les signatures aux signatures des attaques précédentes (AP Singh, 2016). Cette méthode de la détection est très efficace car le taux de faux positifs dans la détection est faible. La mise en garde c'est que le système a besoin d'être mis à jour fréquemment sinon, des attaques dont les signatures n'existent pas dans le catalogue ne seront probablement pas détectées. Cela signifie que les systèmes basés signature ne sont pas efficaces contre les attaques du jour zéro (nouvelles attaques qui n'ont pas encore été recensé)

3.2 Basé sur les anomalies

La détection basée sur les anomalies recherche à travers les réseaux un comportement anormal, il suspecte tout comportement du réseau qui s'écarte de la ligne de base de fonctionnement habituelle. Cette méthode est préférée -sous certaines conditions- à la signature car elle a beaucoup plus de chances de détecter de nouvelles attaques (zero day). Chaque réseau a une base de référence unique en tant que telle, il est difficile pour les attaquants de réussir à une attaque non détectée par le système de détection d'anomalies. Malgré les avantages déclarés de cette méthode, elle a une forte propension à produire de fréquents faux positifs, car certains des changements détectés sont des modifications légitimes du système.

Les systèmes de détection basés sur les anomalies peuvent être divisés en deux catégories : les systèmes basés sur les statistiques et les systèmes basés sur les connaissances.

La méthode statistique envisage la détection d'anomalies à partir du hasard, tandis que la méthode basée sur la connaissance implique la capture du comportement revendiqué à partir d'instances de trafic réseau et d'autres données système pertinentes.

3.3 Base sur l'hybride

Cette méthode combine les méthodes de signature et d'anomalie afin de maximiser les avantages et de minimiser les inconvénients des deux méthodes, ce qui signifie que le taux de détection des attaques du jour zéro et des attaques nouvelles/inconnues augmente tout en parvenant à réduire le nombre de faux positifs. Une étude réalisée par (A. Buczak) a révélé qu'aucun système n'était purement fondé sur la signature ou l'anomalie, les systèmes de détection d'intrusion étant généralement déployés sous la forme d'une configuration hybride.

3.4 Une taxonomie des systèmes de détection d'intrusion

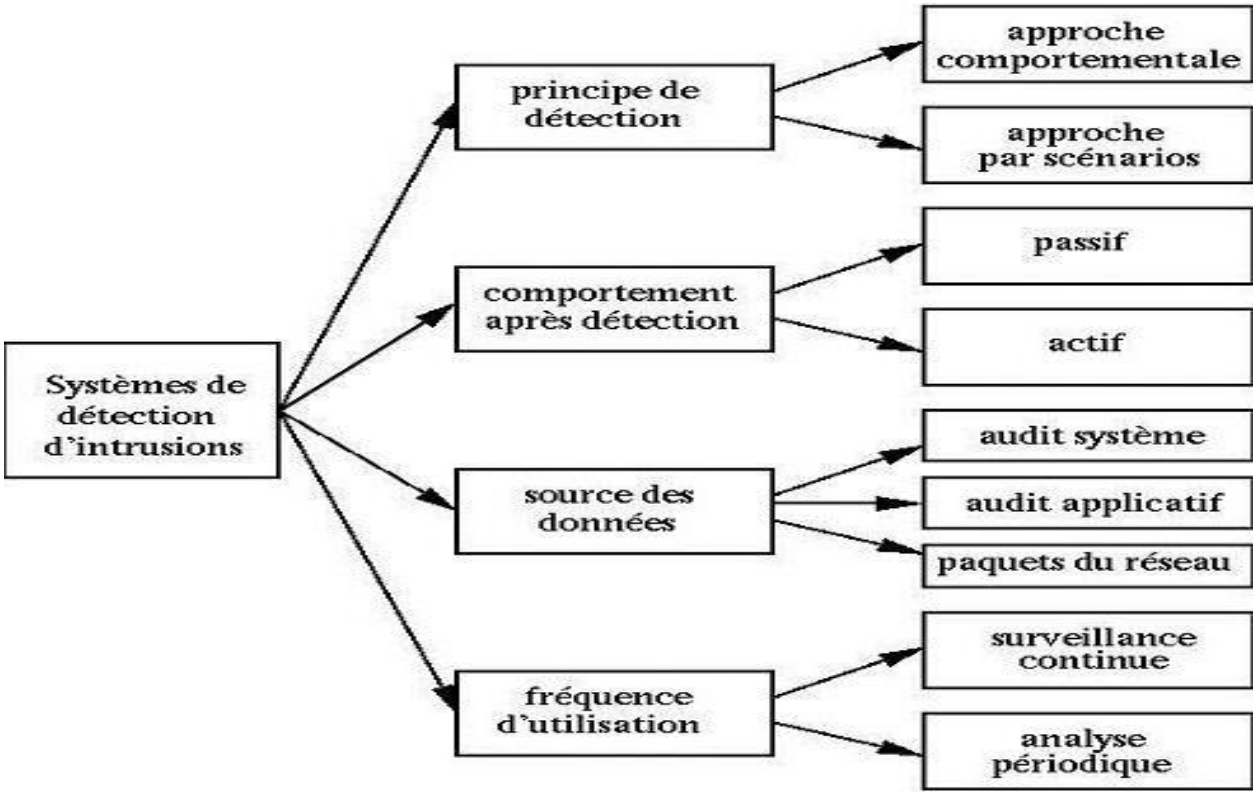


Figure 1.3 Diagramme montrant les catégories de détection d'intrusion.

4. Apprentissage automatique

Traditionnellement, les systèmes de détection d'intrusion dépendent fortement de l'intervention humaine pour rester à jour. Cette dépendance comprend l'ajout de nouvelles URL d'hameçonnage aux listes noires, l'ajout de nouvelles règles aux systèmes basés sur des règles,

la création d'exceptions et la gestion des listes blanches. Avec la croissance de l'internet, la taille même des réseaux existants et le taux de création rapide d'attaques "zero-day", il est facile de voir les lacunes d'un système de détection d'intrusion basé sur l'homme.

L'apprentissage automatique est un moyen de lutter contre cet inconvénient. Tout d'abord, l'apprentissage automatique est quelque chose qui s'améliore avec une grande échelle, un système d'apprentissage automatique est simplement un système qui s'appuie sur l'apprentissage des occurrences passées d'attaques et de tentatives d'intrusion et sur l'apprentissage des modèles qui les accompagnent et de la façon dont ils diffèrent du comportement normal.

Une fois qu'un algorithme d'apprentissage automatique a découvert ces schémas, la reconnaissance et la classification s'effectuent à un rythme plus rapide et à plus grande échelle que n'importe quel système centré sur l'homme.

Les algorithmes d'apprentissage automatique sont globalement regroupés en trois groupes, en se basant sur la manière dont l'algorithme est entraîné. Ces groupes sont les suivants :

- **Apprentissage supervisé :** Cette classe d'algorithmes nécessite des données d'entraînement qui ont déjà été étiquetées. Les algorithmes d'apprentissage supervisé sont conçus pour la prédiction. Un exemple d'algorithme d'apprentissage supervisé utilisé dans la détection des intrusions est la machine à vecteur de support (SVM) et la forêt aléatoire. Le SVM est particulièrement utilisé dans les systèmes de détection d'intrusion basés sur les réseaux en raison de sa facilité de calcul (A. Elike Hodo, 2017).
- **Apprentissage semi-supervisé :** les algorithmes d'apprentissage automatique diffèrent du modèle supervisé par leur capacité à utiliser des données non étiquetées pour s'entraîner, c'est-à-dire qu'ils sont capables de construire des modèles sans aide. Cette capacité est utile lorsque de grandes quantités de données de formation étiquetées sont rares ou indisponibles. La machine à vecteurs de support semi-supervisée est un exemple d'algorithme d'apprentissage automatique semi-supervisé. Cet algorithme est également utilisé dans les systèmes de détection d'intrusion basées sur les réseaux (A.Elike Hodo, 2017).
- **Apprentissage non supervisé :** Il s'agit également d'un algorithme de classification

avec des données non étiquetées

4.1 Processus d'apprentissage automatique

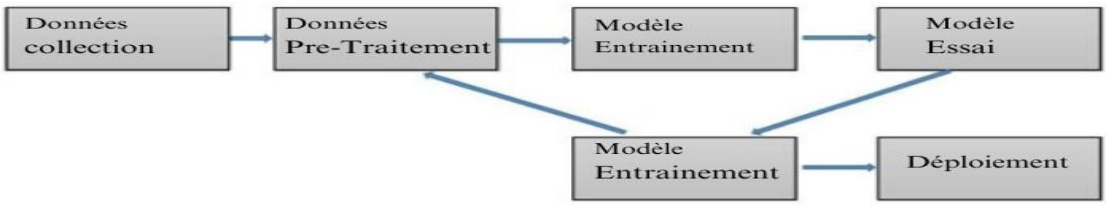


Figure 1.4 Flux de travail de base d'un algorithme d'apprentissage automatique.

1. **Collecte de données :** Selon le projet, des données pertinentes spécifiques à la problématique traitée sont collectées et stockées (Chumachenko, 2017).

2. **Prétraitement des données :** À ce stade, les données collectées sont organisées et transformées en un format pouvant être introduit dans l'algorithme d'apprentissage automatique. Dans la plupart des cas, cela est stocké sous forme de tableau stocké dans des fichier *CSV* par exemple et chargés en mémoire dans des tableaux *numPy*. L'extraction de caractéristiques se produit également à ce stade (toutes les informations issues de la phase de collecte des données ne sont pas pertinentes, en tant que telles, certaines choses sont entièrement à ignorées). Les données prétraitées sont ensuite, divisées en ensembles de données d'entraînement et de test.

   a. **Extraction de caractéristiques :** L'extraction de caractéristiques est une tâche de prétraitement qui consiste à sélectionner des attributs pertinents spécifiques pour créer les ensembles de données d'entrainement et de test qui seraient introduits dans l'algorithme. Ceci peut aussi améliorer certains aspects : il réduit la probabilité de sur-apprentissage, il rend l'interprétation plus facile et améliore la généralisation. Une extraction incorrecte des

caractéristiques peut entraîner l'exécution du modèle plus longtemps que nécessaire car il doit parcourir plus de données s'entrainer et tester (Oscar JimenezdelToro, 2017).

3. **L'apprentissage :** À ce stade, l'ensemble de données d'apprentissage issu de l'étape de prétraitement des données est introduit dans l'algorithme d'apprentissage automatique choisi et un modèle est construit. Cette étape peut être faite une fois ou peut être répété, cela dépend de l'algorithme.

4. **Test :** Une fois le modèle construit, le jeu de données de test issu également de l'étape de traitement est introduit dans le modèle exactement dans le même format que la pour l'entrainement. La précision de la classification ou de la prédiction est calculée. Plus on se rapproche d'un cent pour cent de précision, plus notre modèle est meilleur. Bien sûr, à ce stade, son statistiquement impossible de construire un modèle avec une précision de cent pour cent, au fur et à mesure que la taille des données augmente et que des ajustements sont apportés au modèle, les étapes 2 à 4 sont répétées.

5. **Déploiement** : À ce stade, le modèle avec la meilleure prédiction ou classification, en fonction des besoins est choisi en fonction des résultats obtenus. (Chumachenko, 2017).

4.2 Apprentissage supervisé : classification et régression

La classification et la régression relèvent de l'apprentissage supervisé où les données sont étiquetées et les résultats sont connus et peuvent être cartographiés en conséquence. Cela signifie que le modèle est créé à partir d'ensembles de données dont le résultat est connu dans les deux cas.

- **Classification** : Un problème de classification est un problème où le modèle essaie de trouver une catégorie (classe) pour un élément dans le jeu de données. Par exemple, si certaines caractéristiques (attributs) sont présentées en fonction de la similarité avec une classe dans l'ensemble de données d'apprentissage à quelle classe une instance doit-elle appartenir (google developers, 2019).

- **Régression** : Les problèmes de régression sont ceux où le modèle tente de prédire quelle valeur est proche d'un attribut en comparant à des échantillons similaires dans un ensemble de données. Les problèmes de régression ont tendance à être davantage utilisés sur données continues par opposition aux tâches de classification. (google developers, 2019).

Conclusion

Dans ce chapitre, nous avons survolé le domaine de la cybersécurité et plus particulièrement les intrusions, avec leurs différents types, et leurs méthodes de détections. Nous avons également présenté un aperçu des techniques d'apprentissage automatique, notamment les techniques supervisées, telles que celles de la classification et de la régression. Ce type de techniques sont les plus utilisées dans la détection d'intrusion.

CHAPITRE 2 : Combinaison des Techniques d'Apprentissage en détection d'intrusion

Introduction

L'objectif principal de ce projet consistait à sélectionner une bonne combinaison de fonctionnalités pour augmenter la précision de la classification de l'algorithme K plus proche voisins « K Nearest Neighbor » sur le jeu de données NSL issu de KDD99. Quelques combinaisons ont été itérées jusqu'à ce que la combinaison avec la plus grande précision de classification a été trouvée (Özgür A).

1. Ensemble de données

L'ensemble de données NSL-KDD a été utilisé pour ce projet. Il s'agit d'une version allégée de l'Ensemble de données DARPA 99 et il a 42 caractéristiques distinctes : les classes cibles de ce jeu de données peuvent être vu de deux manières.

- **Comme classe binaire** : Ceci est idéal pour un jeu de données comme NSL-KDD avec plusieurs caractéristiques, afin de garder la complexité sous contrôle. La classes cible est simplement '1 ou 0'. '0' pour une activité réseau normale, '1' pour des attaques. Cela généralise tous les différents types d'attaques à la valeur cible de '1'.

- **Comme multi classes :** Pour les cibles multi-classes, au lieu d'avoir une seule réponse cible de valeurs binaires, vous aurez plusieurs réponses cibles. Dans cet ensemble de données, il y a 22 classes distinctes d'attaques, signifierait que le modèle essaierait de classer en 22 réponses cibles possibles. Au cours de l'expérimentation, il a été rapidement observé, à partir des résultats du processus initial d'apprentissage et de test de l'algorithme utilisé, qu'il n'est pas idéal d'utiliser l'ensemble de données pour une classification multi classe.

1.1 Caractéristiques de l'ensemble de données NSL-KDD

Tableau 2 Noms des fonctionnalités et types de données

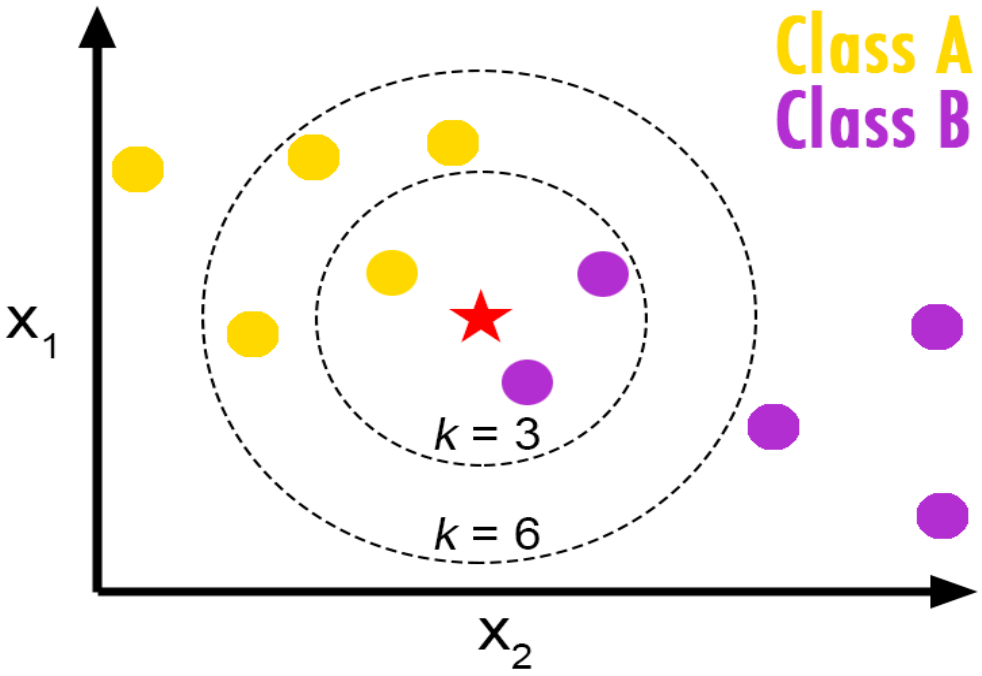
Nom et fonctionnalité	Type de données
duration	Continu
src_bytes	Continu
dts_bytes	Continu
land	Continu
incorrect fragment	Continu
urgent	Continu
hot	Continu
num_failed_logins	Continu
logged_in	Continu
num_compromised	Continu
root_shell	Continu
su_attempted	Continu
num_root	Continu
num_file_creations	Continu
num_shells	Continu
num_access_files	Continu
num_outband_cmds	Continu
is_host_login	Continu
is_guest_login	Continu
count	Continu
srv_count	Continu
serror_rate	Continu
srv_serror_rate	Continu
rerror_rate	Continu
srv_rerror_rate	Continu
same_srv_rate	Continu
diff_srv_rate	Continu
srv_diff_host_rate	Continu

dst_host_count	Continu
dst_host_srv_count	Continu
dst_host_same_srv_rate	Continu
dst_host_diff_srv_rate	Continu
dst_host_same_src_port_rate	Continu
dst_host_srv_diff_host _rate	Continu
dst_host_serror_rate	Continu
dst_host_srv_serror_rate	Continu
Dst_host_rerror_rate	Continu
Dst_host_srv_rerror_rate	Continu

1.2 K-plus proches voisins

L'algorithme du K-plus proches voisins « K-Nearest Neighbors (KNN) » est un algorithme d'apprentissage automatique simple et précis qui classe les échantillons en fonction des échantillons d'entraînement les plus proches dans un espace de caractéristiques. Il peut être utilisé à la fois pour la régression et la classification. KNN est un algorithme qui ne fait aucune hypothèse sur la structure des données et la distribution, ce qui signifie qu'il s'agit d'un algorithme non-paramétrique. C'est un avantage car les données réelles n'obéissent guère règles théoriques. Techniquement, puisque l'ensemble de données d'entraînement sont stockés par le KNN, l'étape d'apprentissage est n'est plus une exigence (Chumachenko, 2017).

KNN fonctionne en classant ou en prédisant sur la base d'un nombre fixe (K) d'échantillons d'entrainement les plus proches de l'échantillon d'entrée. Cela signifie que pour une valeur choisie de K, un échantillon serait classé ou prédit appartenir à la même classe majoritaire parmi les classes des K échantillons les plus proches.



CLASSE A CLASSE B CLASSE INCONNUE

Si K = 3 Alors Inconnu = VIOLET Sinon Si K = 6 Alors Inconnu = JAUNE

Figure 2.1 Exemple de classification par KNN

La distance de l'inconnu aux voisins les plus proches est mesurée à l'aide de différentes méthodes. La plus populaire est la distance euclidienne, d'autres méthodes populaires incluent la Distance de Manhattan, la distance de Hamming, la distance de Minkowski (Chumachenko, 2017).

Mathématiquement, ils peuvent être représentés par :

Distance euclidienne= d(p,q)= d(q,p)= $\sqrt{}$ ($(q1 - p1)^2 + (q2 - p2)^2 + \ldots + (qn - pn)^2$)

$$= \sqrt{} \left(\sum_{i=1}^{n} (q_i + p_i)^2\right) \quad [1]$$

Distance de Manhattan = $d1\ (p, q) = ||p - q||_1 = \sum_{i=1}^{n} |p_i - q_i|$ [2]

Distance de Hamming = $d_{ij} \sum_{k=1}^{p} |x_{ik} - x_{jk}|$ [3]

Distance de Minkowski = $\sum_{i=1}^{n} (|x_i - y_i|^p)^{\frac{1}{p}}$ [4]

Chaque type de distance est idéal dans certains cas d'utilisation. La distance de Manhattan est la mieux adaptée pour les problèmes KNN où les caractéristiques sont de types différents. La distance euclidienne est adaptée au cas contraire où les éléments sont du même type.

La valeur de 'k' est une partie très importante de l'algorithme KNN en raison de la précision de prédiction. La sélection d'une valeur pour 'k' est une tâche qui ne doit pas être considérée comme acquise. Sélection de petites valeurs pour k entraîneront plus que probablement une précision moindre si l'algorithme s'est entrainé à partir d'un ensemble de données bruité, et si la valeur de 'k' est trop élevée, le modèle peut surappris entraînant une faible précision. Un nombre impair 'k' doit également être choisi si le nombre des classes est pair pour éviter les égalités lors du vote à la majorité.

1.3 Validation croisée

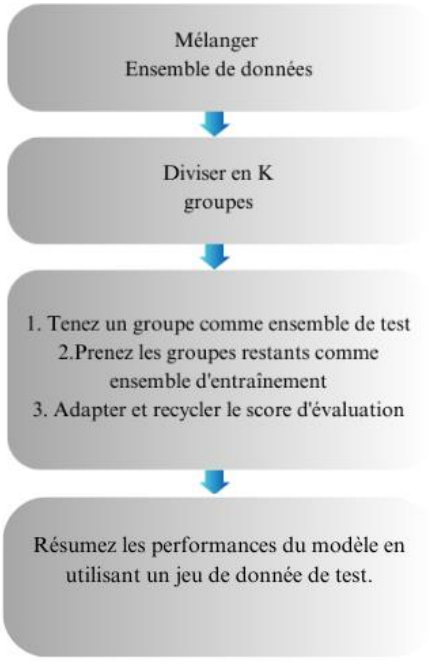


Figure 2.2 Organigramme de validation croisée.

La plupart des méthodes d'évaluation de la précision souffrent de l'incapacité à prédire la manière dont ils se comporteront les modèles sur de nouvelles données. L'algorithme des K plus proches voisins souffre également de cet inconvénient. La validation croisée est une méthode qui permet de surmonter ce problème. D'après l'organigramme de la figure 2.2, l'ensemble de données est d'abord mélangé, puis divisé en k groupes (un grand groupe pour la l'entrainement, un groupe plus petit pour le test). Cette opération est répétée k fois jusqu'à ce que chaque partie des données divisées ait été utilisée comme données d'entrainement et comme données de test (José, 2018).

Il existe trois méthodes de validation croisée :

1. **Méthode d'éclatement en K groupes :** L'ensemble de données d'origine est éclaté en K groupes d'échantillons de tailles égales. L'un des K groupe est retenu pour être utilisé comme jeu de test du modèle. Les groupes restants sont utilisés pour entrainer

le modèle. Ce processus est alors répété K fois avec chacun des K – 1 autres groupes. Chaque itération du processus de la validation croisée a une valeur d'erreur. L'erreur moyenne est ensuite calculée pour tous les K modèles. Cela réduit la variance et maintient la précision parmi les modèles. L'inconvénient de cette méthode est son temps d'exécution (Saxena, 2016).

2. **Méthode d'exclusion :** Pour la méthode d'exclusion, le groupe d'échantillons est divisé en deux parties, une partie pour l'entraînement et l'autre pour le test. La taille de cette division est généralement telle que le l'ensemble d'entrainement est plus grand que l'ensemble de test. Le plus grand groupe est utilisé pour ajuster le modèle, puis la deuxième division est ensuite utilisée pour tester le modèle. C'est la plus simple méthode de validation croisée car il s'agit essentiellement d'une validation croisée unique (Saxena, 2016). L'avantage de la méthode d'exclusion « holdout » par rapport à la méthode K- pliage « K-fold » est le temps d'exécution.
3. **Méthode de l'omission** : Cette méthode est similaire à la méthode k-éclatement, avec une très grande valeur pour K, généralement aussi grande que le nombre des échantillons. Le modèle est entraîné sur chaque valeur de l'ensemble de données sauf une, qui est laissée de côté pour le test. Cela réduit la variance mais demande beaucoup de ressources et de temps (Saxena, 2016).

1.4 Forêt aléatoire

La forêt aléatoire est un algorithme d'apprentissage automatique très populaire. C'est un algorithme d'ensemble d'apprentissage, ce qui signifie qu'il crée de nombreux arbres de décision (c'est la raison pour laquelle il est appelé une forêt) pendant la phase d'apprentissage de l'algorithme, puis produit en sortie la classe majoritaire. Il est également utilisé pour les problèmes de régression ou de prédiction.

L'un des principaux avantages de la forêt aléatoire est la vitesse, la classification s'effectue rapidement (Ho, 1995).

La forêt aléatoire crée une forêt de sous-ensembles indépendants d'échantillons de l'ensemble de données. La meilleure répartition est trouvée en sélectionnant au hasard *n* variables à chaque nœud individuel.

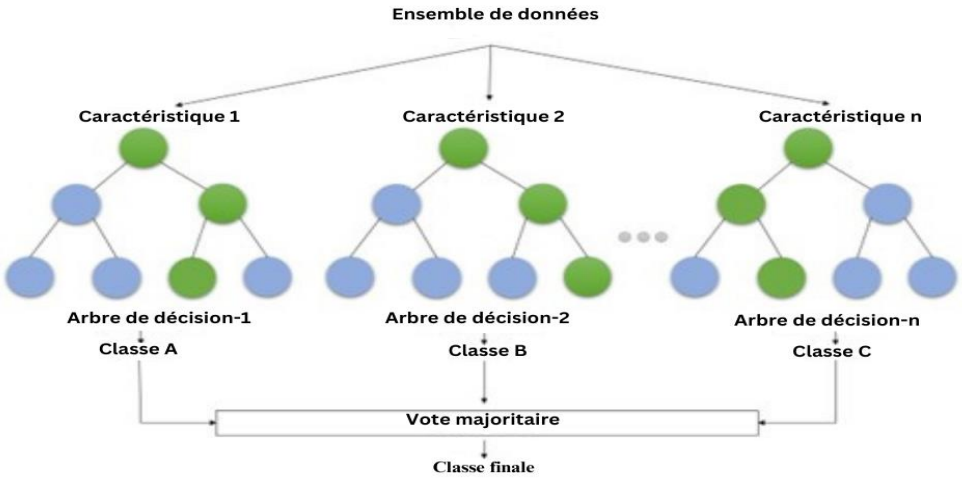


Figure 2.3 Diagramme de flux montrant l'utilisation de l'algorithme de la forêt aléatoire.

CHAPITRE 3 : Implémentation et résultats

Introduction

Dans ce chapitre, nous montrons la mise en œuvre de notre approche de combinaison de classification pour la détection d'intrusions par des techniques d'apprentissage automatique, à savoir l'algorithme KNN et l'algorithme des forets aléatoires. Avant de présenter nos résultats, nous montrons dans la première partie de ce chapitre, les outils logiciels utilisés. Il s'agit du langage de programmation Python, langage par excellence en apprentissage automatique et l'environnement de développement Visual Studio.

1. Le langage Python

1.1 Introduction

Python est devenu l'un des langages de programmation les plus populaires au monde ces dernières années. Il est utilisé dans tous les domaines, de l'apprentissage automatique à la création de sites web en passant par les tests de logiciels. Il peut être utilisé aussi bien par les développeurs que par les non-développeurs.

Python, l'un des langages de programmation les plus populaires au monde.

Python est un langage à usage général, ce qui signifie qu'il est conçu pour être utilisé dans un éventail d'applications, y compris la science des données, le développement de logiciels et de sites Web, l'automatisation, et généralement obtenir des choses faites.

1.2 Qu'est-ce que Python ?

Python est un langage de programmation informatique souvent utilisé pour créer des sites web et des logiciels, automatiser des tâches et effectuer des analyses de données. Python est un langage

polyvalent, ce qui signifie qu'il peut être utilisé pour créer un grand nombre de programmes différents et qu'il n'est pas spécialisé dans des problèmes spécifiques. Cette polyvalence, ainsi que sa convivialité pour les débutants, en ont fait l'un des langages de programmation les plus utilisés aujourd'hui.

L'enquête 2022 de Stack Overflow auprès des développeurs a révélé que Python est le quatrième langage de programmation le plus populaire, les personnes interrogées déclarant qu'elles utilisent Python près de 50 % du temps dans leur travail de développement. Les résultats de l'enquête ont également montré que Python est à égalité avec Rust en tant que technologie la plus recherchée, 18 % des développeurs qui ne l'utilisent pas encore déclarant qu'ils sont intéressés par l'apprentissage de Python

1.3 À quoi sert Python ?

Python est couramment utilisé pour le développement de sites web et de logiciels, l'automatisation des tâches, l'analyse et la visualisation des données. Comme il est relativement facile à apprendre, Python a été adopté par de nombreux non-programmeurs, tels que les comptables et les scientifiques, pour une variété de tâches quotidiennes, comme l'organisation des finances.

"L'écriture de programmes est une activité très créative et gratifiante", explique Charles R Severance, professeur à l'université du Michigan et à Coursera, dans son livre Python for Everybody (Python pour tous). "Vous pouvez écrire des programmes pour de nombreuses raisons, qu'il s'agisse de gagner votre vie, de résoudre un problème difficile d'analyse de données, de vous amuser ou d'aider quelqu'un d'autre à résoudre un problème."

1.4 Que peut-on faire avec Python ? Voici quelques exemples :

1.4.1 Analyse de données et apprentissage automatique

Python est devenu un élément essentiel de la science des données, permettant aux analystes de données et à d'autres professionnels d'utiliser le langage pour effectuer des calculs statistiques complexes, créer des visualisations de données, élaborer des algorithmes d'apprentissage automatique, manipuler et analyser des données, et effectuer d'autres tâches liées aux données.

Python permet de créer un large éventail de visualisations de données différentes, telles que des graphiques linéaires et à barres, des diagrammes circulaires, des histogrammes et des tracés en 3D. Python dispose également d'un certain nombre de bibliothèques qui permettent aux

programmeurs d'écrire des programmes d'analyse de données et d'apprentissage automatique plus rapidement et plus efficacement, comme TensorFlow et Keras.

1.4.2 Développement web

Python est souvent utilisé pour développer l'arrière-plan d'un site web ou d'une application, c'est-à-dire les parties que l'utilisateur ne voit pas. Le rôle de Python dans le développement web peut inclure l'envoi de données vers et depuis des serveurs, le traitement de données et la communication avec des bases de données, le routage d'URL et la garantie de la sécurité. Python propose plusieurs frameworks pour le développement web. Les plus couramment utilisés sont Django et Flask.

Les ingénieurs back-end, les ingénieurs full stack, les développeurs Python, les ingénieurs logiciels et les ingénieurs DevOps font partie des métiers du développement web qui utilisent Python.

1.4.3 Automatisation ou création de scripts

Si vous vous retrouvez à effectuer une tâche de manière répétitive, vous pourriez travailler plus efficacement en l'automatisant avec Python. L'écriture du code utilisé pour construire ces processus automatisés s'appelle l'écriture de scripts. Dans le monde du codage, l'automatisation peut être utilisée pour vérifier les erreurs dans plusieurs fichiers, convertir des fichiers, exécuter des calculs simples et supprimer les doublons dans les données.

Python peut même être utilisé par des débutants pour automatiser des tâches simples sur l'ordinateur, comme renommer des fichiers, trouver et télécharger du contenu en ligne ou envoyer des courriels ou des textes à des intervalles souhaités.

1.4.4 Tests de logiciels et prototypage

Dans le cadre du développement de logiciels, Python peut contribuer à des tâches telles que le contrôle de la construction, le suivi des bogues et les tests. Avec Python, les développeurs de logiciels peuvent automatiser les tests de nouveaux produits ou de nouvelles fonctionnalités. Parmi les outils Python utilisés pour les tests de logiciels, citons Green et Requestium.

1.4.5 Tâches quotidiennes

Python n'est pas réservé aux programmeurs et aux scientifiques des données. L'apprentissage de Python peut ouvrir de nouvelles possibilités à ceux qui exercent des professions moins gourmandes en données, comme les journalistes, les propriétaires de petites entreprises ou les

spécialistes du marketing des médias sociaux. Python peut également permettre aux non-programmeurs de simplifier certaines tâches dans leur vie.

1.5 Voici quelques-unes des tâches que vous pourriez automatiser avec Python :

Suivre les cours de la bourse ou des cryptomonnaies

S'envoyer un texto pour se rappeler de prendre un parapluie dès qu'il pleut.

Mettre à jour votre liste de courses

Renommer de grands lots de fichiers

Convertir des fichiers texte en feuilles de calcul

Assigner aléatoirement des tâches aux membres de la famille

Remplir automatiquement des formulaires en ligne

1.6 Pourquoi Python est-il si populaire ?

Python est populaire pour un certain nombre de raisons. Voici un aperçu plus approfondi de ce qui le rend si polyvalent et si facile à utiliser pour les programmeurs.

Sa syntaxe simple imite le langage naturel, ce qui facilite la lecture et la compréhension. Il est donc plus rapide de construire des projets et de les améliorer.

Il est polyvalent. Python peut être utilisé pour de nombreuses tâches différentes, du développement web à l'apprentissage automatique.

Il est convivial pour les débutants, ce qui le rend populaire auprès des codeurs débutants.

Il s'agit d'un logiciel libre, ce qui signifie que son utilisation et sa distribution sont gratuites, même à des fins commerciales.

Les archives de modules et de bibliothèques de Python - des ensembles de codes créés par des utilisateurs tiers pour étendre les capacités de Python - sont vastes et ne cessent de croître.

Python dispose d'une communauté importante et active qui contribue à l'ensemble des modules et des bibliothèques de Python et agit comme une ressource utile pour les autres programmeurs.

La vaste communauté de soutien signifie que si les programmeurs se heurtent à une pierre d'achoppement, il est relativement facile de trouver une solution ; quelqu'un a forcément déjà rencontré le même problème.

2 Les outils d'apprentissage automatique

2.1 Google Colab (Collaboratory)

Google Colab, abréviation de Google Colaboratory, est une plateforme basée sur le cloud qui fournit un environnement gratuit basé sur un navigateur pour coder, exécuter et collaborer sur des projets Python. Développé par Google, Colab offre aux chercheurs, aux scientifiques des données et aux développeurs un moyen pratique de travailler sur l'apprentissage automatique, l'analyse des données et d'autres tâches informatiques sans avoir besoin d'installer du matériel ou des logiciels au niveau local.

Colab s'appuie sur Jupyter Notebook, une application web open-source qui permet aux utilisateurs de créer et de partager des documents contenant du code en direct, des visualisations et des textes explicatifs. Colab pousse Jupyter Notebook un peu plus loin en fournissant un environnement hébergé dans le nuage avec une puissante accélération matérielle, permettant aux utilisateurs d'exploiter l'infrastructure de Google pour les tâches à forte intensité de calcul.

L'un des principaux avantages de Colab est son intégration avec Google Drive. Les utilisateurs peuvent créer des blocs-notes Colab directement dans Google Drive, et toutes les modifications apportées au bloc-notes sont automatiquement enregistrées dans le compte

Drive de l'utilisateur. Cette intégration transparente garantit que le travail est stocké en toute sécurité et facilement accessible sur tous les appareils.

Colab propose des bibliothèques Python préinstallées et préconfigurées, notamment des bibliothèques populaires telles que NumPy, Pandas, Matplotlib et TensorFlow. Les utilisateurs n'ont donc pas besoin de configurer manuellement leur environnement et peuvent commencer à coder immédiatement. De plus, Colab permet aux utilisateurs d'installer des bibliothèques supplémentaires à l'aide du gestionnaire de paquets intégré, ce qui en fait une plateforme flexible et personnalisable.

Colab se distingue par sa capacité à accéder aux puissants accélérateurs matériels de Google, tels

que les unités de traitement graphique (GPU) et les unités de traitement tensoriel (TPU), et à les utiliser. Ces accélérateurs accélèrent considérablement les calculs, en particulier pour les tâches d'apprentissage automatique, ce qui permet aux utilisateurs d'entraîner des modèles complexes et de traiter des ensembles de données volumineux de manière plus efficace.
Les carnets de notes Colab sont hautement interactifs et prennent en charge à la fois les cellules de code et les cellules markdown. Les cellules de code peuvent contenir du code Python qui peut être exécuté directement dans le bloc-notes. Les cellules Markdown permettent aux utilisateurs d'ajouter du texte formaté, des équations, des images et même du contenu HTML pour fournir des explications et de la documentation sur leur code.

Colab favorise la collaboration grâce à ses capacités de partage. Les utilisateurs peuvent partager leurs carnets avec d'autres personnes en générant un lien partageable ou en fournissant un accès direct à des personnes spécifiques. Les collaborateurs peuvent visualiser, exécuter et modifier le bloc-notes en temps réel, ce qui en fait un excellent outil pour les projets d'équipe ou les révisions de code. De plus, Colab prend en charge l'intégration avec des systèmes de contrôle de version tels que Git, ce qui permet aux utilisateurs de gérer efficacement leurs modifications de code.

En plus de ses fonctionnalités de base, Colab offre un accès facile à diverses sources de données et API. Il permet aux utilisateurs d'importer des données depuis Google Drive, Google Sheets et BigQuery, entre autres, ce qui simplifie l'acquisition et le prétraitement des données. Les utilisateurs peuvent également interagir avec des API externes, telles que celles de reconnaissance d'images ou de traitement du langage naturel, directement depuis leurs carnets Colab.

La nature de Colab, basée sur le cloud, permet également une évolutivité transparente. Les utilisateurs peuvent tirer parti de l'infrastructure de Google pour faire évoluer leurs calculs, en accédant à des ressources plus puissantes en cas de besoin. Cette évolutivité est particulièrement précieuse pour les projets impliquant de grands ensembles de données, des modèles complexes ou des tâches à forte intensité de calcul.

Colab prend en charge un large éventail de langages de programmation et offre un support de noyau pour différents temps d'exécution, y compris Python 2 et 3. Les utilisateurs peuvent choisir le noyau désiré pour leurs ordinateurs portables, ce qui permet une flexibilité dans le choix du langage et une compatibilité avec les bases de code existantes.

Enfin, Colab fournit une documentation complète, des tutoriels et des carnets d'exemples pour

aider les utilisateurs à démarrer et à explorer ses capacités. La communauté en ligne autour de Colab est dynamique, avec des forums et des discussions actives où les utilisateurs peuvent chercher de l'aide, partager des idées et collaborer avec d'autres.

En résumé, Google Colab est une plateforme basée sur le cloud qui offre un environnement gratuit, basé sur un navigateur, pour coder et exécuter des projets Python. Elle s'intègre parfaitement à Google Drive, fournit des bibliothèques préinstallées, prend en charge les accélérateurs matériels et encourage la collaboration. Grâce à ses puissantes fonctionnalités, à son évolutivité et au soutien étendu de la communauté, Colab est un outil précieux pour les chercheurs, les scientifiques des données et les développeurs qui cherchent à exploiter les ressources informatiques en nuage et à collaborer à des tâches de calcul.

2.2 Jupyter Notebook

Le bloc-notes Jupyter est un environnement qui permet de programmer en python dans le navigateur Web. Il est constitué simplement d'un document JSON qui contient une liste ordonnée de cellules d'E/S pouvant contenir du code, de formules mathématiques, du texte, des graphiques/tracés et autres types de médias.

Le bloc-notes Jupyter permet à l'utilisateur d'expérimenter du code dans un navigateur, sans avoir de réexécuter le code entier chaque fois que le codeur souhaite expérimenter sur un petit bloc du code. Cela permet une grande flexibilité.

2.3 Scikit-learn

Scikit-learn est une bibliothèque gratuite pour le calcul scientifique et l'apprentissage automatique en langage python. Il contient divers algorithmes de régression, de classification et de clustering.

2.4 NumPy

Numpy est une bibliothèque python qui prend en charge les grands tableaux multidimensionnels et matrices. Il contient également des collections de fonctions mathématiques

de haut niveau pour opérer sur tableaux.

2.5 Matplotlib

Il s'agit d'une bibliothèque de visualisation de données en Python et NumPy. Il permet de produire différents types de représentation graphique des données.

2.6 Pandas

Pandas est une autre bibliothèque python utilisée pour la manipulation et l'analyse de données. Pandas fonctionne avec des Dataframes. Il a de nombreux outils intégrés qui pourraient être utilisés pour une multitude de fonctions. Une fonction importante est le nettoyage des données. Selon les analyses d'IBM, environ 80 % du temps consacré à un projet d'apprentissage automatique est consacré au prétraitement des données. Beaucoup de jeux de données disponibles ont des champs vides et cela peut grandement affecter négativement un modèle. Un exemple d'une fonction dans pandas pour gérer cela est la fonction '. isnull()'.

2.7 Configuration du système utilisé

Cette recherche a été implémentée sur un système avec la configuration suivante :

- 10 cœurs de processeur 4,4 GHz
- 16 Go de RAM
- Système d'exploitation Windows 10 Pro.
- Python 3.10

3 Etapes, outils et environnement

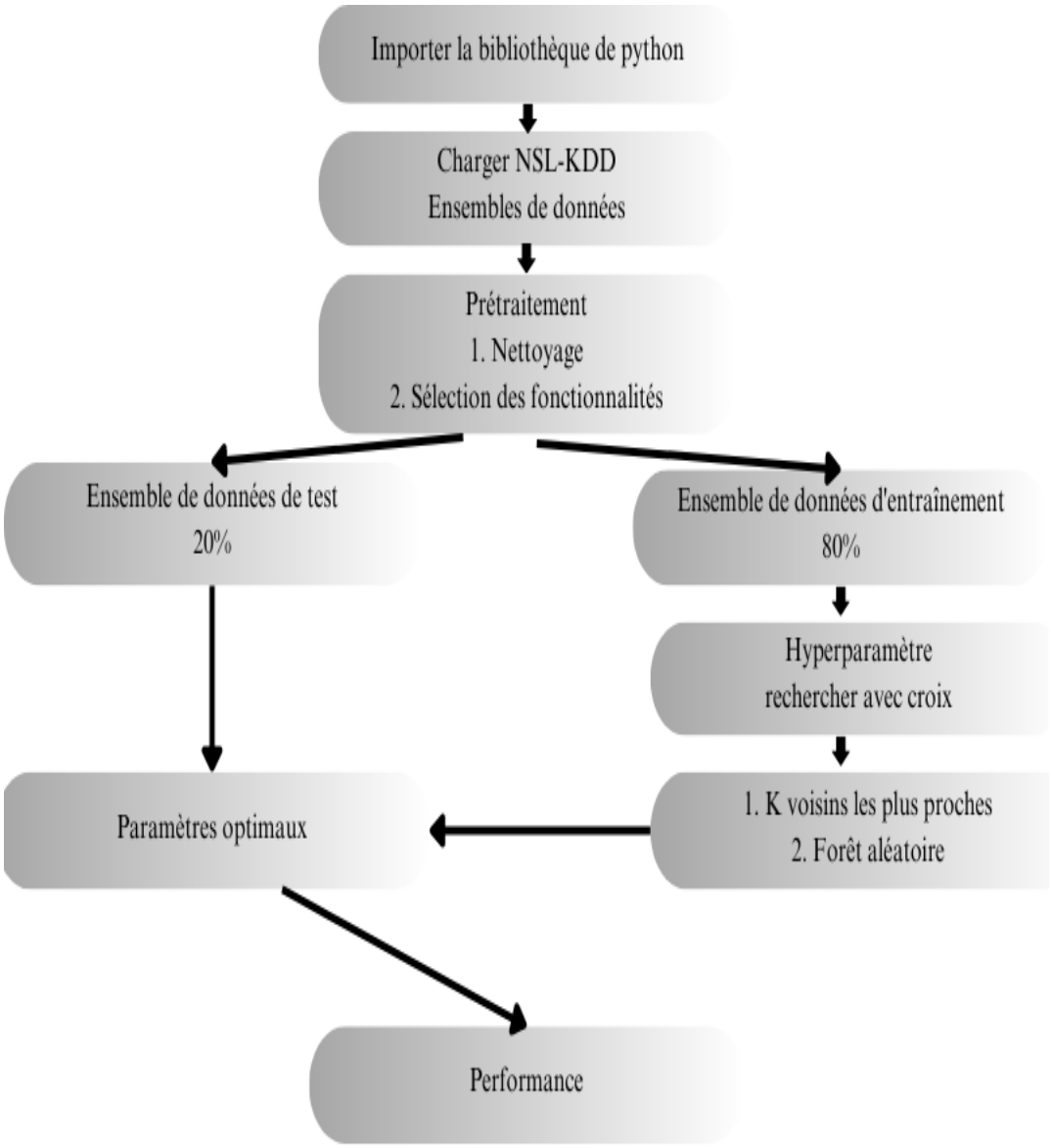


Figure 3.1 ORGANIGRAMME MONTRANT L'ENCHAINEMENT DES ETAPES DE LA METHODE PROPOSEE.

Résultats

Dans ce qui suit on décrit les résultats des modèles et les paramètres utilisés pour évaluer l'algorithme implémenté. Afin de mieux comprendre cette section, les termes suivants serait défini.

- Vrai positif (TP). Un vrai positif est le cas où le modèle prédit qu'un échantillon comme étant une attaque et il s'agit vraiment d'une attaque.
- Faux positif (FP). Un échantillon est faussement positif est celui où le modèle prédit comme étant une attaque alors qu'il ne l'est pas.
- Vrai négatif (TN). Un échantillon vrai négatif est celui où le modèle prédit (correctement) comme étant une connexion normale (pas une attaque) et il l'est vraiment.
- Faux négatif (FN). Un résultat faux négatif est celui où le modèle prédit comme une connexion normale alors qu'il s'agit d'une attaque. (Développeurs Google, 2019)
- Exactitude (Accuracy) : La précision est simplement la mesure de la justesse avec laquelle le modèle prédit la classe d'un échantillon donnée (Shung, 2018).

$$Exactitude = \frac{\text{Total nombres de prédictions correctes}}{\text{Total nombres de prédictions}}$$

$$= \frac{TP+TN}{TP+TN+FP+FN}$$

- **Précision.** La précision est la proportion de vrais positifs (attaques) par rapport au nombre total de positifs (attaque et connexion normale dans le jeu de test). (Shung, 2018)

$$Précision = \frac{\text{Nombre total de prévisions correctes}}{\text{Nombre total de prévisions positives}}$$

$$Précision = \frac{TP}{TP+FP}$$

- **Rappel.** Rappel est la proportion de positifs (attaques) qui ont été correctement identifiés (Shung, 2018)

$$Rappel = \frac{TP}{TP+FN}$$

- **F1-Score :** F1 Score est similaire à la précision, mais est une meilleure mesure car elle cherche à créer un équilibre entre précision et rappel, surtout quand il y a une classe un pair. Le score F1 est donné par formule (Shung, 2018) :

$$Score\ F1 = 2 \times Rappel \times Précision + Rappel$$

3.1 Analyse des résultats pour l'algorithme des K plus proches voisins.

Commençons par tester les valeurs triviales de K =1, K=5, et K=10 sur l'ensemble de données, avec partage en 20% pour les données de test et 80% pour les données d'apprentissage.

Tableau 3.1 K triviaux (1,5 et 10) pour 20% de données de test.

K		1	5	10
Exactitude		0.986	0.978	0.978

Ensuite, nous avons tester la classification pour la plage de valeur de K de 1 à 14 pour 40% pour les données de test (60% pour les données d'apprentissage).

Tableau 3.2 Exactitude du modèle KNN (K = 1 à 14) pour 40% de données de test.

Valeur de k	Exactitude (/1)
1	0.986
2	0.978
3	0.978
4	0.981
5	0.978
6	0.980
7	0.981
8	0.977
9	0.977
10	0.970
11	0.970
12	0.969
13	0.971
14	0.967

Selon les résultats obtenus, nous pouvons affirmer que la valeur de K=1 est la plus appropriée pour les données du NSL-KDD, avec 10000 lignes de données.

La figure suivante montre la variation de l'Exactitude en fonction de la valeur de :

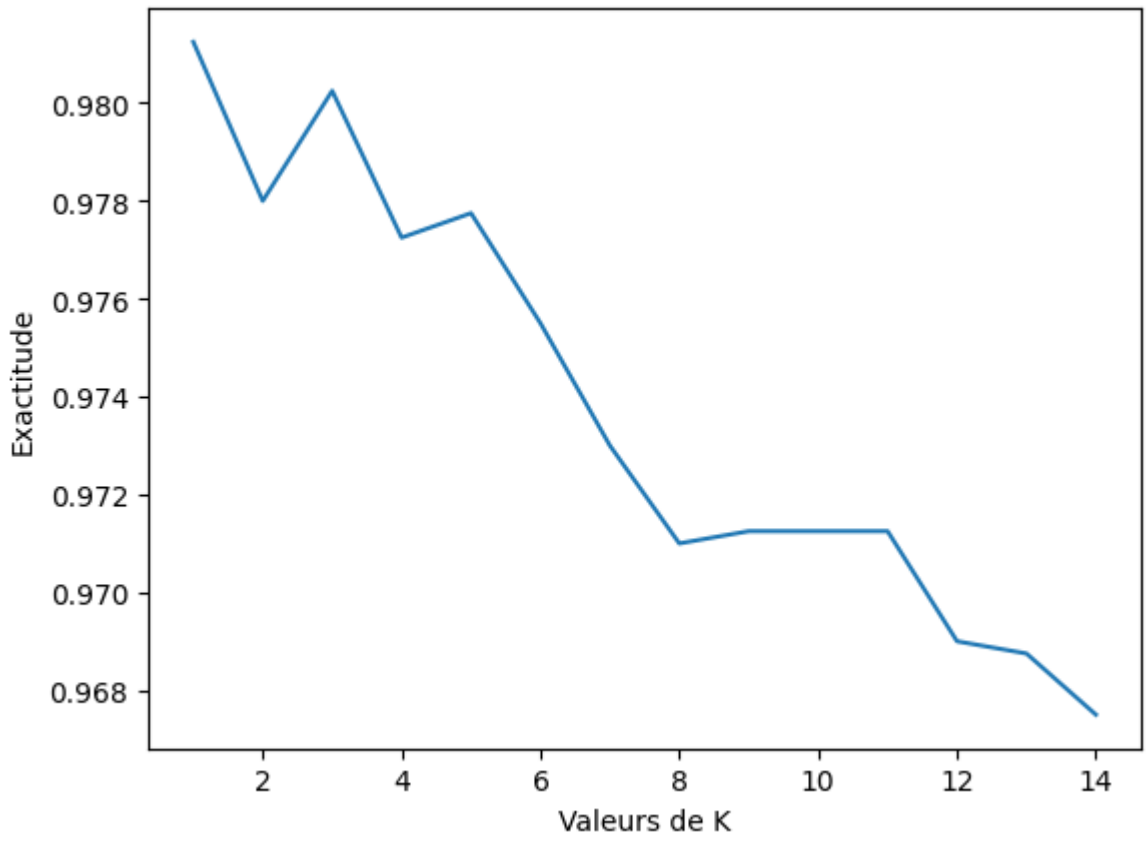


Figure 3.2 Variation de l'exactitude en fonction du paramètre K pour 40% des données de test

La deuxième méthode qui préconise l'utilisation de la racine carrée du nombre d'éléments dans l'ensemble de données (test + apprentissage) estimer la valeur de K optimale. La valeur de K étant 100 = racine carré de 10000 utilisant un fractionnement de 80%-20% pour l'entrainement-test.

3.2 Foret Aléatoire

Le tableau suivant montre les résultats d'exécution du modèle des forets aléatoires, en variant le nombre d'estimateurs dans l'ensemble {10,25,50,100}.

Tableau 3.3 Classifieur de forêt aléatoire

Nombre d'estimateurs (nombre d'arbres)	Exactitude (/1)
10	0.995
25	0.998
50	0.998
100	0.996

A partir du tableau 3.3, on peut conclure qu'une valeur optimale du nombre d'estimateurs pour l'ensemble de données est situé entre 25 et 50 estimateurs.

3.3 Discussion des résultats

En considérant l'ensemble de données utilisé, composé de 10000 lignes de la base NSL-KDD, le classifieur KNN assure des résultats satisfaisants, avec des valeurs basses de l'hyper-paramètre K, où 1 était suffisante pour obtenir les meilleurs résultats. Nous n'avons pas aussi remarqué de différence significative entre une partition de 20 et 80% d'une part et de 40 et 60% d'autre part des données d'apprentissage.

Pour le modèle des forets aléatoires, il a donné des résultats meilleurs que celui des KNN, mais certainement avec plus de temps de calcul. Pour le nombre d'estimateurs nécessaire à ce modèle, un nombre entre 25 et 50 est à utiliser avec l'ensemble considéré de données d'apprentissage.

4 Extrait du code Python

4.1 Modules utilisés

```
#Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.kernel_approximation import RBFSampler
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
import sys
from sklearn.metrics import confusion_matrix
```

4.2 Préparation des données d'apprentissage (X et Y)

```
# Dataset (KDDTrain3 avec 10000 entrées)
data_bin =
pd.read_csv("/content/drive/MyDrive/KDD/KDDTrainB3.csv",delimiter=",")

#import seaborn
import seaborn as sns
# Affichage graphique dans le notebook de Google colab
%matplotlib inline
# Préparation des données
data_bin = data_bin.values
X = data_bin[:,:-1]
y_bin = data_bin[:,-1:]
```

4.3 Partition des données d'apprentissage (20/80) et variation de K

```
X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size=
0.2, random_state= 4)

# Pour K = 1
knn = KNeighborsClassifier(n_neighbors= 1)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("pour =1 : ",metrics.accuracy_score(y_test, y_pred))

# Pour K = 5
knn = KNeighborsClassifier(n_neighbors= 5)
knn.fit(X_train, y_train)
```

```
y_pred = knn.predict(X_test)
print("pour =5 : ",metrics.accuracy_score(y_test, y_pred))

# Pour K = 10
knn = KNeighborsClassifier(n_neighbors= 5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("pour =10 : ",metrics.accuracy_score(y_test, y_pred))
```

4.4 Partition des données d'apprentissage (40/60) et variation de K et graphique

```
# 40% données de test et variation de k entre 1 et 14
X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size=
0.4, random_state= 4)
k_range = range(1, 15)
scores = []
for k in k_range:
  knn = KNeighborsClassifier(n_neighbors= k)
  knn.fit(X_train, y_train)
  y_pred = knn.predict(X_test)
  scores.append(metrics.accuracy_score(y_test, y_pred))
  print("k=",k,":",metrics.accuracy_score(y_test, y_pred))

import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(k_range, scores)
plt.xlabel('Valeurs de K')
plt.ylabel('Exactitude')
```

4.5 Test de la valeur K=100 = racine carré de 10000

```
# Pour K = 100 = racine carré de 1000
X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size=
0.2, random_state= 4)
knn = KNeighborsClassifier(n_neighbors= 100)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
knn.score(X_test,y_test)
print("pour K=100 : ",metrics.accuracy_score(y_test, y_pred))
```

4.6 Test des modèles “Foret aléatoire” pour différents nombres d’estimateurs

```
# Foret aléatoire avec n_estimators = 10
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators= 10)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Estimateurs=",10,":",metrics.accuracy_score(y_test, y_pred))

# Foret aléatoire avec n_estimators = 25
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators= 25)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Estimateurs=",25,":",metrics.accuracy_score(y_test, y_pred))

# Foret aléatoire avec n_estimators = 50
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators= 50)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Estimateurs=",50,":",metrics.accuracy_score(y_test, y_pred))

# Foret aléatoire avec n_estimators = 100
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators= 100)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Estimateurs=",100,":",metrics.accuracy_score(y_test, y_pred))
```

5 Conclusion

Dans ce chapitre nous avons présenté la mise en œuvre de nos tests des deux méthodes étudiées, à savoir la méthode KNN, et le modèle des forets aléatoires. Avant cela, nous avons présenté le langage de programmation utilisée (Python), et l’environnement de développement utilisé (Google Colab). A travers nos résultats, nous avons pu recommander l’ensemble des paramètres optimaux (K pour KNN et le nombre d’estimateurs pour le modèle de foret aléatoire), avec le sous-ensemble de données utilisé.

Conclusion générale

Au cours de ce projet de master, une méthode basée sur l'apprentissage automatique pour la détection d'intrusion a été développé en utilisant l'algorithme du K plus proches voisins et l'algorithme de forêts aléatoires. Le modèle a été exécuté plusieurs fois avec différents paramètres afin d'étudier comment chaque paramètre affecte le modèle. Dans certains cas, on a observé que même bien qu'il n'y ait pas eu de changement visible dans la précision de détection, il y avait un retard dans le temps qu'il a fallu pour faire un classement surtout avec le classifieur des k plus proches voisins. Une sélection préalable de de caractéristiques fait toute la différence pour les modèles basé sur l'algorithme du k plus proches voisins.

Dans la forêt aléatoire, ce sont les effets des paramètres tels que le nombre d'estimateur qui ont été également souligné. Un inconvénient récurrent était que le modèle commençait à mémoriser l'ensemble de données et donnant la même précision de classification à mesure que le nombre d'estimateurs augmentait. C'est un signe que le modèle a mémorisé les données. Ce problème peut être résolu de plusieurs manières : l'ensemble de tests pourrait être divisé afin que des données de test différentes soient utilisées à chaque fois que le modèle est utilisé. Dans un vrai scénario, cela ne serait pas un problème car les données changent tout le temps.

L'objectif principal de ce projet était d'évaluer les conditions dans lesquelles la précision de la classification des modèles sur l'ensemble de données NSL-KDD a été atteinte.

Dans les perspectives de ce travail, nous comptons utiliser d'autres classifieurs performants tels que SVM, et d'autres algorithmes ensemblistes tels que Adaboost.

LES REFERENCES

A. 1 A. Buczak, E. G. (n.d.). survey of data mining and machine learning methods for cybersecurity intrusion detection. IEEE Communications Surveys & Tutorials,1153 1176.

B. A. Elike Hodo, X. J. (2017). Système de détection d'intrusion des réseaux superficiels et profonds : A taxonomy and survey,. Récupéré sur le site du CoRR : https://arxiv.org/abs/1708.07174

C. A. P. Singh. (2016). nalysis of host-based and network-based intrusion detection system.

D. Abliz, M. (2011). Internet denial of service attacks and defense mechanisms (Attaques par déni de service sur Internet et mécanismes de défense). Université de Pittsburgh of Pittsburgh,Department of Computer Science.

E. Aliyev, V. (2010). Using honeypots to study skill level of attackers based on. Chalmers University of Technology.

F. Chumachenko, K. (2017). Machine learning methods for malware detection and classification.

G. google developers. (2019). Machine learning crash course. Retrieved from google developers: https://developers.google.com/machine-learning/crashcourse/classification/true-false-positive-negative

H. Ho, T. K. (1995). trees, Random Decision. Bell Labs.

I. J. G. Noraini, M. R. (2011). Genetic algorithm performance with different selection strategiesin solving tsp. World Congress on Engineering. World Congress on Engineering.

J. José, I. (2018). KNN (K-Nearest Neighbors) #1. Retrieved from Towards Data Science : https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d

K. M. Gupta. (2015). Hybrid intrusion detection system: Technology and development,. InternationalJournal of Computer Applications.

L. Oscar Jimenez-del-Toro, S. O. (2017). Analysis of Histopathology Images: From Traditional Machine Learning to Deep Learning. In O. S.-K. Adrien Depeursinge, Biomedical Texture Analysis (pp. 281-314). Academic Press.

M. Özgür A, E. H. (n.d.). . A review of KDD99 dataset usage in intrusion detection and

machine learning between 2010 and 2015. PeerJ Preprints. PeerJ Preprints.

N. S. Paliwal, R. G. (2012). Denial-of-service, probing & remote to user (r2l) attackdetection using genetic algorithm. International Journal of Computer Applications,, 57-62.

O. Saxena, R. (2016). Knn sklearn, k-nearest neighbor implementation with scikit learn. Retrieved from Dataaspirant: http://dataaspirant.com/2016/12/30/k-nearestneighbor-implementation-scikit-learn/

P. Shung, K. P. (2018). Accuracy, Precision, Recall or F1? Récupéré de Towards Data Science: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Q. Livre blanc de T. M. I. (2012). Addressing big data security challenges: The righttools for smart protection. Retrieved from Trend Micro: https://www.trendmicro.com/en_us/business.html

R. V. Shmatikov, M.-H. W. (2007). ecurity against probe-response attacks in collaborativeintrusion detection. 2007 Workshop on Large Scale Attack Defense,ser. LSAD ’07 (pp. 129-136). NY: ACM

S. Structure généralisée pour les forêts aléatoires par Hafsa Binte Kibria Rajshahi Université d’ingénierie et de technologie (researchgatem 2022)

T. Real-Life Example par Amir Ali The Art of Data Scicne (medium, 2018)