Titre: Title:	Système de détection d'intrusion non supervisé fédéré
Auteur: Author:	Maxime Gourceyraud
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Gourceyraud, M. (2024). Système de détection d'intrusion non supervisé fédéré [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59170/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/59170/
Directeurs de recherche: Advisors:	Nora Boulahia Cuppens, & Frédéric Cuppens
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Système de détection d'intrusion non supervisé fédéré

MAXIME GOURCEYRAUD
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Système de détection d'intrusion non supervisé fédéré

présenté par **Maxime GOURCEYRAUD**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Omar ABDUL WAHAB, président
Nora BOULAHIA-CUPPENS, membre et directrice de recherche
Frédéric CUPPENS, membre et codirecteur de recherche
Hanane DAGDOUGUI, membre

RÉSUMÉ

Les organisations à travers le monde subissent une menace croissante pour leur sécurité informatique. En effet, le marché de la cybercriminalité connaît un développement intense. Pour se protéger, les entreprises constituent des équipes de cyberdéfense. Parmi la panoplie d'outils de défense, on trouve les Systèmes de Détection d'Intrusion (IDS). Ces logiciels alertent en cas de comportement suspect dans le système informatique. Un tel comportement pourrait donc relever d'une intrusion d'un acteur malveillant. Ces IDS assistent les analystes en sécurité informatique et leur indiquent quelle communication sur le réseau inspecter. Ces outils facilitent ainsi le travail des équipes de sécurité.

Les IDS fonctionnent typiquement par signature. Pour créer ce genre d'IDS, il faut d'abord générer une base de données d'attaques déjà identifiées. Ensuite, l'IDS renvoie une alerte lorsqu'il reconnaît un pattern (signature) présent dans sa base d'attaques. Pour tenter de mieux prédire les cyberattaques, la recherche se tourne vers les IDS à base d'apprentissage automatique. Or, la plupart des outils développés dans ce cadre sont supervisés. Cela signifie que les données étudiées doivent être étiquetées pour entraîner le modèle. Toutefois, les entreprises génèrent de grandes quantités de données. Il semble ainsi irréaliste de demander à des analystes d'étiqueter chaque *logs* ou chaque communications sur le réseau comme étant suspecte ou non. C'est pourquoi, au cours de ce projet, nous utilisons de l'apprentissage non supervisé. En effet, les données n'ont pas besoin d'être étiquetées pour pouvoir entraîner un modèle de ce type. Le modèle détermine les ressemblances entre les données par lui-même.

Les données observées par les IDS sont en général critiques puisqu'elles décrivent les actions et les ressources d'un système informatique. Ainsi, les entreprises sont réticentes à dévoiler ces informations pour collaborer les unes avec les autres. Or, les attaques subies par les unes pourraient être menées contre les autres. Il semble donc important de collaborer malgré la sensibilité des données. C'est pourquoi l'IDS proposé dans ce mémoire utilise l'apprentissage fédéré. De cette manière, les données restent chez les participants du protocole et ne sont pas dévoilées tout en coopérant.

Toutefois, l'apprentissage fédéré est vulnérable à des attaques issues de participants malveillants. On peut par exemple penser à de l'empoisonnement de modèles. Nous supposons que les participants sont honnêtes mais curieux (*honest-but-curious*). Ainsi, ils suivent scrupuleusement le protocole et cherchent toutefois à en tirer le plus d'informations.

L'IDS proposé dans ce mémoire utilise du clustering fédéré pour regrouper les données des clients dans des partitions. Nous comparons trois algorithmes de clustering fédéré. Parmi ces

trois algorithmes, nous en proposons deux. Ensuite, chaque client vote selon ses observations pour le type de communications contenues dans chaque partition. Les clusters sont classés selon la classe majoritaire dans leurs données. Par exemple, les clients indiquent qu'un cluster est bénin/normal à 90%. Dans ce cas, le cluster est classé comme normal. Une fois que les partitions sont associées à un type (attaque, normal, quelle attaque...), l'IDS prédit pour chaque donnée le type de son cluster. Les performances de l'IDS ont ensuite été évaluées sur deux jeux de données classiques du domaine. Les performances sont comparables à un clustering réalisé sur des données centralisées. La sélection des hyperparamètres des modèles se fait sur la base de la silhouette.

Certains phénomènes notables sont observés au cours de l'étude. L'IDS présente un compromis entre les efforts à fournir pour analyser les clusters et les performances de l'IDS. En effet, il a de meilleures performances lorsqu'il a de plus en plus de clusters. Ceci coûte néanmoins en travail d'analyse de chaque cluster. Il faut donc trouver un équilibre entre les performances attendues et le travail qu'on souhaite consacrer à l'entraînement de cet IDS. Nous avons aussi observé une faible séparabilité des différents types d'attaques entre eux lors de l'analyse des résultats. Ceci pose problème, car le modèle ne semble pas trouver des différences fondamentales entre tous les types d'attaques.

L'étude de l'IDS proposé présente des limites. En effet, les modèles de clustering utilisés nécessitent la disponibilité de tous les clients. De plus, il n'y a pas de simulation du temps de réponse pour chaque communication. Ceci peut être important puisqu'un IDS doit réagir dans les plus brefs délais. De plus, nous ne supposons pas que les clients puissent être malveillants.

Des travaux futurs pourraient être réalisés sur trois composantes. Tout d'abord, on pourrait améliorer la modélisation des données avec de la corrélation temporelle. Ensuite, la confidentialité pourrait être augmentée avec de la confidentialité différentielle ou bien un chiffrement homomorphe. Finalement, nous supposons que les participants au protocole sont honnêtes mais curieux. Ainsi, ils suivent le protocole mais cherchent à tirer des informations à travers celui-ci. On peut alors travailler à l'avenir sur la présence d'un participant malveillant.

ABSTRACT

With the growing digitization of the world, the cyberthreat is becoming more and more concerning. To protect from cyberattacks companies create teams to defend themselves. These teams use several tools such as Intrusion Detection Systems (IDS). IDS detect if a behaviour or a specific communication is suspicious and raises an alert to analysts. Indeed such tools are designed to detect a threat. This makes defending an organization easier by telling analysts what to inspect.

The typical IDS is signature-based. The first step to create such an IDS is to populate a database with previously identified attacks. Then the IDS raises an alert when it recognizes an attack (signature) stored in its database. Research is now heading towards machine learning-based IDS to detect cyberattacks in a more efficient way. Usually this kind of IDS is based on supervised algorithms. It means that each communication/behaviour needs to be labelled to train the model. But companies generate a lot of data to monitor. Thus it seems unfeasible for analysts to label each log. That is why during this project we used unsupervised machine learning. Indeed there is no need to label each data because the model finds by itself patterns in data.

Machine learning models require data to be trained on but data to train an IDS are sensitive ones. Indeed such data can be exploited to learn confidential information about the company that owns them. Nonetheless companies could benefit from collaboration. Indeed if a company was never attacked but some other was, the first could benefit from the second. To enable collaboration, we used federated learning. This way, the data owned by the companies are kept in the companies. This enhances privacy.

However, federated learning can be attacked by a malicious client. For example, a client could poison the learning process of the model and thus reduce the performances. We do not tackle this issue because we assume that the participants are honest-but-curious. It means that clients dont try to undermine the performances of the model but seek to gain information as they can through the protocol.

To solve the problems stated above, we propose a federated clustering-based IDS. We study three federated clustering algorithms. Among these three algorithms, two are new ones proposed during the project. The clustering algorithm creates a partition of the datasets of the clients. Then each client votes for the composition of each of the partition/cluster. For example, a cluster could be composed of 90% normal data. Then this cluster is classified as a normal cluster and so are the data inside of it. We evaluated the performances of the studied

algorithms on classic datasets in the field of network intrusion detection. We found that the performances of the federated versions are comparable to the centralized ones. Model selection was performed using the silhouette criterion.

During our experiments, we noted some phenomena. First, the proposed IDS shows a compromise between the effort to analyse the clusters and the performances. Indeed the performances increase with the number of clusters of the method. But increasing the number of clusters increases the number of clusters to analyze. We also found that there is little separability between some types of attacks on the studied datasets.

The study presented here faces some limitations. First, we did not take into account the availability of the participating organization. Second, we did not measure the response time of the IDS which can be limiting if its reaction time is not near real time. Finally, the participating organizations (clients) are supposed to be honest-but-curious. So we did not suppose that there can be malicious clients.

Future works can be done on three axes. First the machine learning algorithm could be enhanced by adding time correlation to the modeling. Another axis is privacy. It could be enhanced by adding differential privacy or homomorphic encryption. These techniques could prevent anybody from extracting useful information from the communications from the clients to the server. Finally, future work could focus on preventing a malicious client to poison the learning process of the IDS.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ACP	Analyse en Composantes Principales
DDoS	Distributed Denial of Service
HIDS	Host-based Intrusion Detection System
IA	Intelligence Artificielle
IDS	Intrusion Detection System
i.i.d.	Indépendant et identiquement distribué
IoT	Internet of Things
IPS	Intrusion Prevention System
NIDS	Network Intrusion Detection System
SIEM	Security Information and Event Manager
SOC	Security Operation Center
SVM	Support Vector Machine
WCSS	Within-Cluster Sum of Squares

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Dans un monde où l'informatique et l'interconnexion entre les appareils sont grandissantes, la menace cyber devient de plus en plus pesante économiquement. Statista, cité par l'OMC [1], estime qu'en 2023 le coût de la cybercriminalité était de 8 150 milliards de USD et projette un montant de 13 820 milliards de USD pour 2028. IBM [2] estime que les entreprises découvrent une violation de leur système en moyenne en 204 jours après qu'elle ait eu lieu. Ce chiffre n'a pratiquement pas évolué depuis 2019 où cette durée était de 206 jours en moyenne. Ces attaques coûtent en moyenne 4.45 millions de USD. Ce montant est de 2.3% supérieur aux coûts de 2022. Ainsi, les menaces informatiques, lorsqu'elles se concrétisent, peuvent être difficiles à détecter, très coûteuses pour l'entreprise qui les subit et les montants croissent.

Depuis quelques années, la surface d'attaque informatique augmente. En effet, depuis la pandémie de COVID-19, les entreprises proposent du télétravail à leurs employés. Selon un sondage réalisé par Amazon Canada [3], 55% des employés pourraient refuser un emploi si la présence sur le lieu de travail est obligatoire. Cela indique que le télétravail s'installe dans les mentalités. Toutefois, un salarié qui travaille depuis chez lui agrandit la surface d'attaque de son entreprise. En effet, son réseau informatique domestique peut maintenant être une porte d'entrée vers le réseau de son entreprise pour un attaquant.

Les avancées des dernières décennies en conception de puces électroniques ont rendu possible la création d'appareils compacts pouvant collecter des données et se connecter à Internet. Ainsi, l'Internet of Things (IoT) offre de nombreux services grâce aux données collectées comme l'automatisation de processus industriels, de la maintenance prédictive, de la domotique, etc... Selon Ericsson [4], le nombre de connexions IoT avait atteint 15.7 milliards en 2023 et devrait croître fortement pour atteindre 38.9 milliards en 2029. Tous ces appareils connectés à Internet peuvent être mal pris en compte dans la politique de sécurité d'une entreprise alors qu'ils peuvent collecter des informations importantes, voire personnelles et gérer des processus industriels. Entre le télétravail et l'IoT, la surface d'attaque ne cesse d'augmenter, ce qui offre plus d'opportunités aux attaquants.

Les attaquants peuvent avoir des profils très différents [5]. En effet, des organisations criminelles ou des individus peuvent rechercher un gain financier. D'un autre côté, certains criminels informatiques veulent plutôt promouvoir leurs idées/idéologies (*hacktivists*). Ces

activités par des individus ou des organisations criminelles sont facilitées par l'accès facile à des outils de piratage en ligne. Les États profitent aussi du cyberespace comme vecteur de renseignement et de déstabilisation. Pour cela, ils parrainent en général des auteurs de menaces. Le Centre canadien pour la cybersécurité [6] pointe par exemple de la surveillance de leur diaspora par des États comme la Chine ou l'Arabie Saoudite. Les individus ne sont pas les seuls à être ciblés puisque les entreprises sont aussi visées pour leur propriété intellectuelle ou pour du renseignement commercial.

Ainsi, les menaces en cybersécurité sont grandissantes. Il est important de développer des outils et des méthodes de gouvernance efficaces pour améliorer la cybersécurité. En particulier, la détection d'intrusion est un moyen efficace de réduire le risque dans un système informatique.

1.1 Définitions et concepts de base

1.1.1 Organisation de la défense

Pour organiser sa défense informatique, une entreprise peut avoir recours à un Security Operation Center (SOC). Un SOC est une équipe dédiée à la cybersécurité. Celle-ci peut-être interne à l'entreprise ou être fournie en tant que service par un prestataire. Un SOC a généralement trois missions [7].

La première mission d'un SOC est de préparer son organisation à la menace cyber. Pour cela, il peut mettre en œuvre une politique de réponse aux incidents, faire l'inventaire du parc informatique, se tenir à jour des menaces et des technologies... Le SOC doit aussi s'assurer de la conformité des installations et des processus par rapport aux lois et standards applicables. Cette phase sert à minimiser le nombre d'incidents et de se donner les moyens de réagir en cas d'attaque.

La deuxième mission est la surveillance et la réponse aux incidents. Un SOC peut alors observer l'activité du parc informatique de son entreprise grâce à des journaux (*logs*) de l'activité du réseau voire de différents appareils (serveurs, ordinateurs...). Pour faciliter la surveillance, le SOC utilise en général un Security Information and Event Manager (SIEM). Un SIEM est un outil informatique qui centralise les données importantes à surveiller comme des *logs*, des évènements sur le réseau ou sur un poste de travail et toute autre information utile à la sécurité informatique de l'entreprise. Usuellement, un SIEM organise des *logs*, corrèle les variables et évènements entre eux et alerte le SOC en cas de suspicion [8]. Pour aider un SIEM, on peut mettre en place un IDS. Les IDS seront présentés dans la section 1.1.2. Ainsi,

un SIEM facilite le travail du SOC en organisant et analysant automatiquement le grand volume de données produit par le parc informatique d'une entreprise. Une fois qu'une alerte est remontée par le SIEM, le SOC est chargé d'enquêter pour établir la nature de l'alerte (fausse ou vraie alerte), ses causes et de réagir en conséquence. Certaines actions peuvent être prises pour contenir une attaque : l'isolation de la menace, changer des mots de passe, etc...

La troisième mission d'un SOC est la réaction *a posteriori* en cas d'incident. Cette phase se produit une fois que la menace est contenue. Dans ce contexte, l'équipe doit régler l'incident, restaurer l'intégrité des systèmes d'information. Un SOC s'assure, en accord avec la direction, de la communication conforme aux lois et standards vis-à-vis des autorités, des clients et toute partie prenante pertinente.

Une organisation fréquente de SOC est le SOC par niveaux. On dénombre généralement quatre niveaux [9]. Même si le fonctionnement présenté est répandu, le fonctionnement de chaque SOC peut différer selon l'organisation qui le met en place pour répondre à ses propres besoins.

Le premier niveau est dévolu à la réception des incidents et à leur priorisation. Les analystes du premier niveau peuvent voir défiler un grand nombre d'incidents. Dans ce cas, ils peuvent se désensibiliser à la menace, tarder à répondre, voire l'ignorer. Ce phénomène s'appelle l'*alert fatigue* et doit être pris en compte. Ces analystes sont généralement moins expérimentés que dans le deuxième niveau auquel ils peuvent faire suivre des alertes plus importantes. Les analystes du deuxième niveau peuvent enquêter plus en profondeur sur les incidents pour en comprendre le fonctionnement et réagir. Le troisième niveau est généralement le dernier niveau d'analystes dans un SOC. Il est composé de personnes expérimentées qui recherchent activement la menace dans l'organisation et opèrent une veille. Le quatrième niveau est celui de la direction et gestion du SOC.

Les rôles dans le SOC peuvent se répartir suivant deux grandes équipes, voire plus selon les variantes. La première équipe, l'équipe bleue (*blue team*) s'assure de la protection de l'infrastructure de son organisation. L'autre grande famille d'équipes est l'équipe rouge (*red team*). Cette dernière participe à la défense de l'entreprise en cherchant des failles dans son système informatique. Certaines opérations d'équipe rouge peuvent ressembler à une véritable attaque. Le tout étant supervisé et autorisé par l'entreprise évidemment.

Le projet présenté dans ce mémoire porte sur la détection d'intrusion. Ainsi, le projet vient en aide à l'équipe bleue pour la surveillance. Cela concernerait principalement les deux premiers niveaux d'un SOC puisque ce sont eux qui répondent aux alertes.

1.1.2 IDS

Un Système de Détection d'Intrusion (IDS) est un logiciel qui détecte la présence d'une menace pour le système qu'il protège. Lorsqu'un IDS peut intervenir directement sur tout ou partie des incidents qu'il détecte, on parle d'Intrusion Prevention System (IPS). Les IDS peuvent détecter des évènements sur le réseau qu'ils surveillent. On parle alors de Newtork Intrusion Detection System (NIDS). C'est le cas qui est étudié dans ce mémoire. L'autre grande famille d'IDS est celle des Host-based Intrusion Detection System (HIDS). Ce sont des IDS qui protègent un appareil (*host*) de menaces provenant de son environnement ou de lui-même.

Les IDS sont donc séparés en deux catégories selon ce qu'ils protègent. On peut aussi les définir suivant la méthode de détection. En effet, les IDS peuvent détecter des attaques qui ressemblent à des occurrences observées par le passé. Il faut donc tenir une base de données à jour pour que les nouvelles menaces soient détectées. Par analogie avec les antivirus qui fonctionnent sur le même principe, ces IDS sont dits "par signature". Une autre catégorie est formée par les IDS par détection d'anomalie, qui n'est pas tout à fait la détection d'anomalie de la section 2.2.1. Ces IDS utilisent de l'apprentissage automatique pour différencier ce qui est du trafic normal, aussi appelé bénin, ou de l'attaque. Ce mémoire traite d'IDS utilisant de l'apprentissage automatique. Même si ces IDS peuvent potentiellement détecter des attaques *zero-day*, il peut être difficile de bien modéliser le comportement normal. Ainsi, il faut faire attention aux faux positifs *i.e.* les communications normales classées comme des attaques. En effet, ces faux positifs peuvent nuire à l'efficacité d'un IDS en termes de performances mais aussi en causant de l'*alert fatigue* comme expliqué précédemment.

Dans le cadre d'un IDS par détection d'anomalie, il faut être attentif à la rapidité de la détection. En effet, des modèles d'apprentissage profond peuvent être utilisés pour détecter des attaques. Or, selon la profondeur et le type d'architecture, il se peut que ce modèle soit demandant en puissance de calcul voire que la détection soit trop lente par rapport au flux de données. Ce type de système peut donc détecter une attaque avec du retard voire risque de ralentir le réseau. Ainsi, la détection en temps réel est aussi un enjeu des IDS utilisant de l'apprentissage automatique.

1.1.3 Collaboration

La construction d'un IDS à base d'apprentissage automatique par une seule entreprise peut être une tâche difficile. En effet, celle-ci n'a pas subi toutes les formes d'attaques et donc son IDS risque de ne pas en reconnaître certaines. Pour pallier ce problème et ajouter de la diversité aux données, plusieurs organisations peuvent collaborer. Or, le cadre d'usage classique en apprentissage automatique centralise les données dans une seule organisation ou un seul serveur qui se charge d'apprendre un modèle. Ce type d'apprentissage pose un problème de confidentialité qui est accentué par la criticité des données réseau. En effet, de nombreuses informations sensibles transitent sur le réseau informatique d'une organisation. De fait, les entreprises sont généralement réticentes à divulguer ce type d'information. C'est pourquoi il faut adopter un protocole d'apprentissage automatique qui favorise la confidentialité tout en permettant la collaboration. De plus, le serveur central, ayant collecté les données des participants, représente un *single point of failure*. C'est-à-dire qu'il suffit pour un attaquant de compromettre un seul membre du protocole pour s'emparer de toutes les données, en l'occurrence le serveur central. Ceci représente donc un risque important de fuite de données. Par la suite, l'adjectif "centralisé" fait référence à l'entraînement d'un modèle sur un jeu de données unique par un seul appareil.

1.2 Formulation de la problématique

1.2.1 Apprentissage fédéré

Nous avons abordé l'aspect collaboratif dans la section 1.1.3. Toutefois, l'entraînement centralisé de modèles pose le problème de la confidentialité et de la fuite potentielle de données. Ainsi, il vaudrait mieux que les données restent chez les clients. De fait, si les participants gardent leurs données sur leurs appareils, il faudrait qu'un attaquant compromette tous les clients pour obtenir le même résultat que dans le cas centralisé. Ceci est la raison d'être de l'apprentissage fédéré. Avec un tel protocole d'apprentissage, les clients conservent bien leurs données. Les informations qui sont transmises au serveur sont aussi limitées que possible. Plus d'informations sont fournies dans la section 2.1.4.

Au cours de l'étude, nous supposons que tous les participants du protocole, serveur et clients, sont **honnêtes mais curieux** (*honest-but-curious*). Cela signifie que tous les participants suivent le protocole scrupuleusement sans l'altérer. Ainsi, nous écartons un potentiel adversaire qui empêcherait le protocole de se dérouler correctement. Néanmoins, les participants essaient de tirer le plus d'informations possible sur les autres au cours des interactions issues du protocole. Donc, il faut veiller à ce que les informations transmises préservent au mieux

la confidentialité.

Les clients sont supposés être des organisations. Ainsi, ils sont capables de communiquer avec le serveur, peuvent expertiser les données qu'ils ont (cf. section 3.2) et sont capables d'utiliser un IDS sur leurs ressources. Pour ce qui est du serveur, on considère que c'est une organisation tiers dont les clients supposent qu'elle est au moins honnête mais curieuse. Le serveur pourrait par exemple être une entreprise de cybersécurité mettant en relation des organisations pour réaliser l'apprentissage de l'IDS. Si les clients sont considérés comme vitaux pour un pays, alors le gouvernement pourrait endosser le rôle de serveur.

1.2.2 Apprentissage non supervisé

Les organisations génèrent de grandes quantités de données réseau. Il semble donc difficile d'étiqueter tout ou partie des communications de façon fiable comme étant de l'attaque ou du trafic normal. En effet, cela reviendrait soit à demander à des analystes de classer chaque communication, ce qui est irréaliste et laborieux, soit de faire appel à un IDS par signature. Dans ce dernier cas, on s'expose à la marge d'erreur de cet IDS. De plus, on ne pourrait pas apprendre une meilleure classification que celle de l'IDS par signature puisqu'il serait l'objectif de l'algorithme d'apprentissage automatique.

Ainsi, les méthodes d'apprentissage automatique à utiliser présentent un avantage en ne tenant pas compte d'un étiquetage des données pour trouver des similarités entre elles. Ce type d'apprentissage machine s'appelle l'apprentissage non supervisé, par opposition à l'apprentissage supervisé qui nécessite l'étiquetage des données.

1.2.3 Problématique

Ainsi, de par les points abordés au cours de cette section, nous comprenons l'intérêt de développer un IDS non supervisé. De plus, il est important que l'IDS soit collaboratif et préserve la confidentialité des données des participants. On peut donc formuler la question suivante :

Comment développer un système de détection d'intrusion non supervisé fédéré préservant la confidentialité des données des participants ?

1.3 Objectifs de recherche

Les objectifs suivis lors de ce projet de recherche sont les suivants :

1. Proposer une architecture de détection d'intrusion collaboratif se basant sur un modèle d'apprentissage automatique non supervisé.
2. Développer un protocole d'apprentissage fédéré préservant la confidentialité des données des participants.
3. Évaluer les performances de l'IDS.

Au-delà de ces objectifs de recherche principaux, on peut ajouter quelques contraintes au projet. En effet, l'utilisation de l'apprentissage non supervisé a pour objectif de minimiser le travail humain lors de l'entraînement du modèle. Ainsi, on peut chercher à réduire cette charge de travail au cours des choix des paramètres et des méthodes.

L'apprentissage fédéré implique des communications entre le serveur et les clients. Une des contraintes qu'on peut ajouter est de minimiser ce nombre de communications pour limiter les échanges entre les parties prenantes. En effet, moins il y a d'étapes, moins un participant pourrait tirer d'informations. De plus, limiter les communications implique une bande passante plus faible et donc réduire les coûts de fonctionnement du protocole.

1.4 Plan du mémoire

Le présent mémoire se déroule comme suit. Le prochain chapitre est dédié à la revue de littérature. Dans un premier temps sont présentées en détail les notions essentielles à la compréhension du sujet. Ensuite, un inventaire de plusieurs types de méthodes non supervisées et fédérées sont abordées pour offrir au lecteur une vue plus globale des notions de ce type. Le chapitre se termine finalement sur un tour d'horizon des IDS fédérés et centralisés étant non supervisés.

Après la revue de littérature, nous aborderons la méthodologie de cette étude. Ainsi, nous présenterons tout d'abord l'architecture proposée de l'IDS (objectif 1). Ensuite, nous traiterons des algorithmes fédérés utilisés pour le clustering ou l'évaluation de celui-ci. Les algorithmes proposés visent à améliorer la confidentialité des données des clients (objectif 2) par rapport à l'état de l'art. Après, nous définirons la méthode de classification des clusters (objectif 1). Finalement, le chapitre se termine sur la méthodologie d'entraînement et d'évaluation.

Une fois la méthodologie complète, nous pouvons observer les résultats expérimentaux de cet IDS (objectif 3). Tout d'abord, nous présenterons les jeux de données utilisés et leurs prétraitements. Pour jeter les bases d'une comparaison, nous analyserons ensuite le comportement

et les performances de l'IDS lorsque le clustering est assuré par l'algorithme k-means. Ceci nous permettra de définir ce qu'est le comportement attendu de l'IDS dans un cas idéal. Dans le chapitre suivant, nous présenterons tout d'abord comment nous avons généré des jeux de données fédérés. Ensuite, nous analyserons l'évolution de la silhouette des modèles fédérés selon les différents hyperparamètres. Cette section sera l'occasion de sélectionner les hyperparamètres pertinents pour l'évaluation des performances des modèles, en section suivante.

Les résultats étant présentés et analysés, nous conclurons ce mémoire par une synthèse des travaux suivie d'une exposition des limites du projet. Finalement, nous déterminerons quelles suites peuvent être données à ce projet de recherche.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre recense les travaux connexes, pour situer la production réalisée, et utiles à la compréhension de la suite du mémoire. La section 2.1 traite des travaux essentiels pour comprendre les aspects théoriques et méthodologiques. Les principales notions utiles à ce projet sont réparties en deux catégories. Tout d'abord, le projet mobilise de l'apprentissage non supervisé, présenté dans la section 2.2. Dans celle-ci, le lecteur trouvera à la fois des concepts de détection d'anomalie et de clustering. La deuxième composante majeure est la collaboration. Pour ce faire, nous avons eu recours à de l'apprentissage fédéré présenté en section 2.1.4. Le clustering fédéré est introduit en section 2.3. Finalement, différents travaux de détection d'intrusion non supervisés sont présentés dans la section 2.4.

2.1 Notions essentielles

Au cours de cette section, les notions centrales du mémoire sont présentées avec plus de détails que pour le reste du chapitre 2. Ce sont des points importants à connaître pour comprendre la méthodologie.

2.1.1 K-means

Le clustering est une catégorie d'apprentissage automatique non supervisé dont l'objectif est de regrouper les données en ensembles similaires - des clusters. L'algorithme le plus connu est k-means avec l'algorithme de Lloyd [10]. K-means regroupe les données en k clusters où k est un hyperparamètre. Les clusters sont construits autour de centroïdes. Chaque point est associé au cluster dont le centroïde est le plus proche de lui. Ces centroïdes sont la moyenne des données du cluster, d'où le nom de la méthode. L'algorithme débute avec k centroïdes et les modifie jusqu'à convergence. Un critère classique de convergence est de vérifier si la partition a changé entre deux étapes. Si elle n'a pas changé, l'algorithme termine. L'algorithme de Lloyd est détaillé dans l'algorithme 1.

Posons pour l'algorithme X un jeu de données et $k \in \mathbb{N}^*$ clusters $C = (C_i)_{i\in\{1,\dots,k\}}$ avec leurs centroïdes respectifs $c = (c_i)_{i\in\{1,\dots,k\}}$.

Algorithm 1 K-means - Lloyd

Require: X, $c = (c_i)_{i \in \{1,...,k\}}$

1: # Création des clusters, vides pour le moment.
2: $\forall i \in \{1, ..., k\}, C_i \leftarrow \{\}$
3: **while** non convergence **do**
4: # Chaque point est associé au cluster dont le centroïde est le plus proche.
5: $\forall x \in X, x \in C_m$ où $m \leftarrow \arg\min_{i \in \{1,...,k\}} (||x - c_i||_2^2)$
6: # Les nouveaux centroïdes sont la moyenne des points de leur cluster.
7: $\forall i \in \{1, ..., k\}, c_i \leftarrow \frac{1}{|C_i|} \sum_{x \in C_i} x$
return c

L'algorithme de Lloyd a pour objectif de minimiser la distance entre les points dans chaque partition et leur centroïde (*Within-Cluster Sum of Squares (WCSS)*). La WCSS s'exprime comme suit :

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - c_i||_2^2 \tag{2.1}$$

Toutefois, selon l'initialisation des clusters et le jeu de données, l'algorithme peut trouver des minima locaux et non globaux. Il peut être pertinent de réitérer l'initialisation et l'algorithme puis de garder les centroïdes qui minimisent la WCSS. C'est l'approche implémentée par exemple par Scikit-Learn [11] selon le type d'initialisation choisi.

L'algorithme k-means a l'avantage d'être plutôt léger. En effet, pour prédire le cluster d'un point, il suffit de trouver le centroïde le plus proche. Ceci contraste avec les autres algorithmes de clustering présentés dans la section 2.2.2.

2.1.2 K-means++

K-means pose deux problèmes : l'initialisation des centroïdes et le choix de k, le nombre de clusters. L'initialisation peut être réalisée en tirant uniformément des centroïdes dans le jeu de données mais une approche plus efficace est l'initialisation k-means++ [12]. Posons $c = (c_i)_{i \in \{1,...,k\}}$ l'ensemble des centroïdes. L'ensemble c est construit itérativement :

Algorithm 2 K-means++_init

Require: X, k

1: # Tirage uniforme sur X du premier centroïde.
2: Tirer c_1 avec $c_1 \sim \mathcal{U}(X)$
3: $c \leftarrow \{c_1\}$
4: **for** $i \in \{2, ..., k\}$ **do**
5: # Calcul de la distance au centroïde le plus proche pour chaque point.
6: $\forall x \in X, d(x, c) = \min_{i' \in \{1,...,|c|\}}(||x - c_{i'}||_2^2)$
7: # Tirage d'un point proportionnellement à la distance calculée.
8: Tirer c_i avec $P(c_i) = \frac{d(c_i,c)}{\sum_{y \in X} d(y,c)}$
9: $c \leftarrow c \cup c_i$
return c

La loi de probabilités utilisée dans la ligne 8 pour k-means++ favorise grandement les points qui sont éloignés de tous les éléments de c. Ceci est souhaitable pour obtenir une diversité de clusters. On peut cependant craindre que k-means++ privilégie les données aberrantes qui sont éloignées de tout cluster. Cependant, un cluster est composé de plusieurs points et donc même si individuellement tirer chacun des points est moins probable que de tirer un point isolé, sélectionner un point du cluster peut être plus probable.

2.1.3 Silhouette

Le second problème de k-means à traiter est le choix de k. Pour ce faire, la technique du "coude" peut être utilisée. On calcule la WCSS, équation 2.1, pour plusieurs nombres de clusters k. Cette courbe est décroissante mais la pente est plus prononcée pour des valeurs faibles de k. Ainsi, la technique du "coude" consiste à trouver la valeur de k qui représente une cassure sur la courbe *i.e.* une baisse de la décroissance de la WCSS. Cependant, cette méthode semble difficile à automatiser et ne donne pas toujours un résultat clair selon le jeu de données. De plus, elle dépend de l'interprétation de la personne qui observe la courbe de WCSS. De fait, nous avons essayé la méthode du "coude" sur les données présentées dans la section 4.1. Toutefois, le "coude" n'apparaissait pas nettement. Ainsi, la sélection de modèle ne pouvait pas se faire par ce biais. Pour palier ces problèmes, nous utiliserons la silhouette [13]. La silhouette est un score qui se calcule pour chaque point d'un jeu de données. Les données doivent être classées dans k clusters $C = (C_i)_{i \in \{1,...,k\}}$. Soit X le jeu de données et $c(x)$ est l'indice du cluster auquel appartient $x \in X$. On a pour tout x de X :

$$\begin{cases} a(x) = & \frac{1}{|C_{c(x)}|} \sum_{y \in C_{c(x)}} d(x,y) \\ b(x) = & \min\limits_{i \in \{1,\dots,k\} \setminus c(x)} \left(\frac{1}{|C_i|} \sum_{y \in C_i} d(x,y) \right) \\ s(x) = & \frac{b(x)-a(x)}{max(a(x),b(x))} \end{cases}$$

La valeur $a(x)$ correspond à la distance moyenne entre x et les points de son cluster. On mesure ainsi la ressemblance/proximité de x par rapport à son cluster. Pour $b(x)$, on calcule la plus petite distance moyenne aux points d'un autre cluster. Cela permet de mesurer la proximité de x à un autre cluster. Finalement, la silhouette $s(x)$ est une valeur comprise entre -1 et 1. Si $s(x) \simeq 1$ cela signifie que $b(x)$ est "très grand" devant $a(x)$ et donc que x est bien classé dans sa partition. En effet, cela signifie que x est plus proche du cluster auquel il est attribué. Si $s(x) = 0$ cela signifie que x se trouve à la frontière entre deux clusters. Finalement, si $s(x) \simeq -1$ alors $a(x)$ est "très grand" devant $b(x)$ et donc x est devrait plutôt appartenir à un autre cluster que celui auquel il est rattaché. La silhouette indique donc pour chaque point si celui-ci est bien classé, ou non, dans son cluster. L'objectif est de maximiser la silhouette pour chaque point. Pour simplifier, nous chercherons par la suite à maximiser la silhouette moyenne. La silhouette moyenne à l'échelle du jeu de de données est un indicateur global, et non plus au niveau de la donnée individuelle, de la qualité du clustering. Cette manière de faire est tout à fait automatisable puisqu'il suffit de sélectionner la valeur de k maximisant la silhouette moyenne.

Toutefois, le calcul de la silhouette peut s'avérer coûteux sur des jeux de données importants. En effet, pour chaque point, il faut être en mesure de calculer la distance à tous les autres points et donc la complexité est en $\mathcal{O}(|X|^2)$. Pour accélérer les calculs, on peut utiliser une silhouette simplifiée [14] :

$$\begin{cases} a(x) = & d(x, c_{c(x)}) \\ b(x) = & \min\limits_{i \in \{1,\dots,k\} \setminus c(x)} d(x, c_i) \\ s(x) = & \frac{b(x)-a(x)}{max(a(x),b(x))} \end{cases}$$

Cette variante de silhouette a une complexité $\mathcal{O}(|X|k)$ ce qui peut être considérablement plus rapide lorsque $|X|$ est grand. Dans le cadre d'un k-means, un point x est toujours associé au centroïde le plus proche lors de la classification dans un cluster. On a alors toujours $a(x) \geq b(x)$ et donc $s(x) \geq 0$. Tout comme la silhouette, l'objectif est de maximiser cet indicateur.

Au cours de ce mémoire, nous allons observer de nombreuses courbes de silhouette. Ce ne sont pas tant les valeurs prises par celles-ci qui vont nous importer mais plutôt leur forme. De

fait, l'information qui nous importe est le nombre de clusters où la silhouette est maximale, et non la valeur prise par celle-ci. Ainsi, nous commenterons l'allure des courbes et le "$argmax$" plutôt que le "max".

2.1.4 Présentation de l'apprentissage fédéré

L'apprentissage fédéré est un paradigme d'apprentissage automatique décentralisé proposé par des chercheurs de Google, McMahan et al., avec l'algorithme FedAvg [15]. L'objectif de l'apprentissage fédéré est d'apprendre un modèle d'apprentissage profond sur plusieurs jeux de données sans y accéder directement. Le protocole fédéré suppose qu'il existe un ensemble $\mathcal{C}$ de $N \in \mathbb{N}^*$ clients et un serveur. Les clients possèdent les données sur lesquelles le modèle est entraîné et le serveur coordonne le protocole. Ce type d'apprentissage fédéré est dit horizontal. L'autre grande catégorie est l'apprentissage fédéré vertical. Dans ce cadre, les clients ont observé les mêmes entrées mais n'ont pas mesuré les mêmes variables. On peut par exemple penser à un capteur de température et un capteur d'humidité dans une pièce. Ils observent en même temps le même "objet" mais n'enregistrent pas la même variable. Le projet décrit dans ce mémoire se concentre sur de l'apprentissage fédéré horizontal.

L'apprentissage fédéré se positionne ainsi en opposition aux entraînements de modèles classiques qui supposent que les données sont centralisées et/ou accessibles directement par le protocole d'apprentissage. Un des avantages à décentraliser l'entraînement chez les clients est lié à la taille du jeu de données. En effet, une taille de jeux de données trop importante nécessiterait une grande puissance de calcul. Or, en déléguant le calcul sur les jeux de données des clients, la puissance nécessaire s'en trouve réduite pour l'ensemble des participants.

Un cas d'usage de l'apprentissage fédéré serait un modèle de prédiction de mot suivant pour un clavier de téléphone portable. Dans un cadre classique d'apprentissage, le téléphone devrait envoyer à un serveur les textes écrits par son utilisateur. Ainsi, les textes de tous les clients sont centralisés et le prestataire du modèle peut l'entraîner. Ceci pose un problème de confidentialité des données et le serveur est un *single point of failure.* C'est-à-dire que si un attaquant le compromet, il pourrait accéder à l'ensemble des textes des clients. Avec l'apprentissage fédéré, ces données restant sur les appareils clients, ce type d'attaque serait bien plus difficile à mettre en œuvre puisqu'il faudrait compromettre l'ensemble des participants.

L'exemple d'un modèle de prédiction de prochain mot pour téléphone portable permet de saisir plusieurs des enjeux soulignés par McMahan et al. [15]. En effet, les téléphones portables ne peuvent pas être disponibles en permanence. Il faut donc disposer d'un protocole qui ne fonctionne pas avec tous les clients synchronisés. De plus, chaque appareil dispose de données qui sont issues de son usage ou de l'observation de son environnement. Ainsi, ces données

peuvent fortement différer d'un client à l'autre. L'apprentissage fédéré doit donc être robuste face à des clients dont les distributions sont non indépendantes et identiquement distribuées (i.i.d.).

Deux variables aléatoires indépendantes sont des variables aléatoires dont les réalisations de l'une n'influencent pas les réalisations de l'autre. Des variables aléatoires identiquement distribuées suivent la même loi de probabilité. Ainsi, des variables aléatoires i.i.d. suivent la même loi de probabilité et elles ne s'influencent pas entre elles. Les clients dont les distributions de données sont non i.i.d. sont donc des clients dont les données sont générées en ne suivant pas les mêmes lois de probabilité et dont les lois peuvent s'influencer. Le cas de distributions de données non i.i.d. chez les clients représentent donc un cas réaliste. De fait, les clients ont généralement des comportements différents les uns des autres.

L'algorithme FedAvg [15] est un algorithme fédéré proposé par McMahan et al. dans l'article où les auteurs présentent les enjeux de l'apprentissage fédéré. Cet algorithme tient donc compte des points mentionnés précédemment dans cette section. Il se déroule comme suit :

1. Le serveur initialise un modèle d'apprentissage profond.
2. Le serveur envoie les poids de ce modèle à une proportion $p \in]0, 1]$ de clients.
3. Chacun des clients sélectionnés entraîne le modèle sur ses données pendant E *epochs* et une taille de *mini-batch* B.
4. Les clients ayant entraîné un modèle l'envoient au serveur.
5. Le serveur agrège les résultats en calculant le réseau de neurones moyen en pondérant par la taille du jeu de données de chaque client ayant participé à ce *round* de communications.
6. Répéter en repartant du point 2 jusqu'à convergence ou un nombre maximal de *rounds* de communication.

La fonction d'agrégation de FedAvg est relativement simple, mais nous constaterons dans les sections 2.1.5 et 2.3 que les fonctions d'agrégation peuvent être plus sophistiquées.

On observe dans cet algorithme que le nombre de communications entre le serveur et les clients, appelés *rounds* de communications, est un paramètre majeur. L'objectif des auteurs de l'article était de déterminer les conditions dans lesquelles la convergence des modèles est atteinte en un minimum de *rounds*. Durant l'étude des résultats dans le chapitre 4, nous garderons cet objectif en vue. De fait, minimiser le nombre de *rounds* de communications implique un usage plus réduit du réseau et réduit la quantité d'informations transmises.

2.1.5 K-means fédéré

Cette section présente le k-means fédéré de Garst et al. [16], algorithme 3 de ce mémoire. Ce modèle est proche de ce qui est proposé avec le meta k-means (sec. 3.1.2) puisque la fonction d'agrégation est un k-means sur la concaténation des centroïdes des clients. Toutefois, les calculs diffèrent. De plus, nous utilisons cet algorithme comme base lors de la section 3.1.1 où nous proposons l'initialisation k-means++.

Le k-means fédéré commence par une initialisation k-means++ [12] chez chaque client puis le serveur réalise un k-means pondéré par la taille des clusters sur les centroïdes des clients. De cette manière, les clusters plus importants de par leur taille comptent plus dans le modèle. Les clients appliquent ensuite un k-means avec pour initialisation les centroïdes globaux. Toutefois, les clients retirent du calcul les centroïdes dont le cluster est vide sur leur jeu de données. Le serveur reçoit les centroïdes des clients et on retourne à l'agrégation par k-means pondérée par la taille des clusters chez les clients.

Pour cet algorithme 3, nous conservons les notations définies jusqu'à présent. Ainsi, l'ensemble des $N \in \mathbb{N}^*$ clients est $\mathcal{C} = (\mathcal{C}_j)_j$. Chacun des clients $\mathcal{C}_j$ possède un jeu de données X_j. Jusqu'ici, l'ensemble des clusters était désigné par $c = (c_i)_{i\in\{1,\dots,k\}}$. Comme chaque client manipule un ensemble de cluster, on pose c^j l'ensemble des centroïdes à tout moment de l'algorithme pour le client $\mathcal{C}_j$. On pose $s^j = (s_i^j)_{i\in\{1,\dots,k\}}$ l'ensemble de la taille des clusters sur les données du j-ème client.

Tout comme le k-means originel, ce k-means fédéré se déroule à k fixé *i.e.* à nombre de clusters fixe. Pour déterminer une valeur de k pertinente, on pourrait souhaiter utiliser la silhouette comme présentée dans la section 2.1.3. Toutefois, le serveur n'a pas accès aux données des clients pour calculer la silhouette. Pour pallier ce problème, nous proposons une adaptation de la silhouette simplifiée en section 3.1.3 pour un scénario fédéré.

Algorithm 3 The federated kmeans algorithm [16]

Require: k

Init clients :
for $j \in \{1, ..., N\}$ **do**
    $c^j \leftarrow$ kmeans++_init(X_j, k)
    Déterminer s^j
    Envoyer s^j, c^j au serveur
Agrégation des clusters fournis par les clients.
Sur le serveur :
    # On concatène les clusters des clients et les tailles des clusters correspondants.
    # Cela crée un jeu de données pour le k-means du serveur.
    $c \leftarrow [c^1|c^2|...|c^N]$
    $s \leftarrow [s^1|s^2|...|s^N]$
    # Étape d'agrégation avec l'algortihme de Lloyd pondéré par la taille des clusters.
    $c^g \leftarrow$ kmeans$(c, k,$ weights $= s)$
    Envoyer c^g aux clients
for round r **do**
    Pour chaque client $\mathcal{C}_j \in \mathcal{C}$ **:**
        $c^j \leftarrow c^g$
        Déterminer s^j
        # Le client $\mathcal{C}_j$ ignore les centroïdes dont le cluster est vide sur ses données.
        $s^j \leftarrow C^j[s! = 0$ for s in $s^j]$
        $k_j \leftarrow$ taille(s_j)
        # Les clients ne réalisent qu'une étape de l'algorithme de Lloyd (algorithme 1).
        # L'algorithme de Lloyd s'applique sur les centroïdes c^j ici.
        $s^j, c^j \leftarrow$ kmeans$(X_j, k_j,$ init $= c^j)$
        Envoyer s^j, c^j au serveur
    Sur le serveur :
        # On concatène les clusters des clients et les tailles des clusters correspondants.
        # Cela crée un jeu de données pour le k-means du serveur.
        $c \leftarrow [c^1|c^2|...|c^N]$
        $s \leftarrow [s^1|s^2|...|s^N]$
        # Étape d'agrégation avec l'algortihme de Lloyd pondéré par la taille des clusters.
        $c^g \leftarrow$ kmeans$(c, k,$ weights $= s)$
        Envoyer c^g aux clients

2.2 Apprentissage non supervisé

L'apprentissage automatique se compose majoritairement de deux grandes catégories. Soit X un jeu de données et y les étiquettes associées aux données de X. Par exemple, X pourrait être un ensemble de photographies d'animaux et y serait l'espèce associée à chacune de ces images. L'apprentissage supervisé consiste alors à trouver le lien existant entre X et y. Dans le cadre de l'apprentissage non supervisé, l'algorithme d'apprentissage n'a à sa disposition que X. Dans la suite de cette section, nous allons voir plusieurs sous-catégories d'apprentissage non supervisé.

2.2.1 Détection d'anomalie

La détection d'anomalie est une catégorie de méthodes statistiques dont l'objectif est de trouver des données anormales/aberrantes par rapport à un jeu de données d'entraînement. La détection d'anomalie peut être supervisée ou non supervisée. Dans le cadre du projet de maîtrise présenté ici, c'est cette dernière catégorie qui est utile. Les paragraphes suivants présentent quelques méthodes de détection d'anomalie.

One-Class SVM Pour déterminer quel type de données est aberrant ou non, Schölkopf et al [17] ont proposé les *One-Class Support Vector Machine (SVM)*. L'objectif des *One-Class SVM* est de séparer les données en deux ensembles : les données normales et les données anormales. L'optimisation des problèmes des SVM est linéaire et fait intervenir des produits scalaires. Toutefois, pour séparer des données dont les frontières sont non linéaires malgré un problème posé qui est linéaire, on utilise un noyau. Ce noyau k est tel que pour x, y deux données $k(x, y) = (\Phi(x) \cdot \Phi(y))$ avec Φ un plongement inconnu *a priori* des données observées dans un espace de plus grande dimension. Ainsi, le noyau reproduit le comportement du produit scalaire dans un espace de plus grande dimension et l'optimisation des SVM est alors possible. L'utilisation d'un noyau pour mieux séparer des données est appelée l'astuce du noyau (*kernel trick*).

Il existe deux façons de séparer les données avec des *One-Class SVM* :

— Les données normales sont séparées de l'origine par un hyperplan affine. Dans ce cas, pour un vecteur x observé, la fonction de prédiction est $f(x) = sgn((w \cdot \Phi(x)) - \rho))$ où w est un vecteur orthogonal à l'hyperplan de décision et ρ est un paramètre rendant affine la prédiction.

— Les données normales sont contenues dans une hyperboule. La fonction de prédiction

est alors $f(x) = sgn(R^2 - ||\Phi(x) - c||^2)$ où R est le rayon de la boule et c en est le centre.

Dans les deux cas présentés ci-dessus, si $f(x) = 1$ alors x est considéré comme normal. Sinon, si $f(x) = -1$, alors x est considéré comme anormal.

Isolation forests La détection de données anormales est aussi possible avec des arbres de décision. Pour ce faire, définissons les *isolation trees* (iTrees). Un arbre d'isolement est un arbre binaire aléatoire. Le fonctionnement est similaire aux forêts aléatoires [18]. Pour séparer les données, l'arbre d'isolement génère chaque nouveau noeud en choisissant aléatoirement une variable explicative et en choisissant un seuil de décision aléatoire lui aussi. L'arbre cesse de créer de nouveaux nœuds soit lorsqu'une profondeur maximale a été atteinte, soit lorsque toutes les données sont seules dans leur zone définie par le modèle. Une *isolation forest* (forêt d'isolement) est l'agrégation de plusieurs arbres d'isolement [19].

Soit x un point à classer comme normal ou anormal. L'idée principale de Liu et al. [19] est que les points anormaux sont peu nombreux et sont facilement séparables du reste des données. Cela signifie que, en moyenne, un point anormal aurait une distance à la racine d'un arbre d'isolement "faible" comparée aux points normaux. Soit $h(x)$ la profondeur/hauteur dans un arbre d'isolement à laquelle se trouve x. On suppose alors que la profondeur moyenne $\bar{h}(x)$ est "petite" si x est anormal. Cette métrique est calculée via la moyenne de la profondeur fournie par chaque arbre de la forêt d'isolement. Une fois les arbres entraînés, les auteurs définissent un score d'anomalie pour chaque point x dans un ensemble de n points :

$$s(x, n) = 2^{-\frac{\bar{h}(x)}{c(n)}} \tag{2.2}$$

L'équation 2.2 définit le score d'anomalie d'un point. On reconnaît la profondeur moyenne du point $\bar{h}(x)$. Le facteur de normalisation $c(n)$ est la profondeur moyenne pour une recherche qui ne réussit pas dans un arbre de recherche binaire. C'est-à-dire, la longueur moyenne d'une recherche pour une donnée n'étant pas dans l'arbre. Cette valeur $c(n)$ sert de facteur de normalisation pour $\bar{h}(x)$.

Le score s d'anomalie se lit comme suit :

— Si $s(x, n) \simeq 1$, alors x est considéré comme une anomalie. En effet, cela signifie que $\bar{h}(x)$ est petit et donc que x est facilement séparable du reste des données.
— Si $s(x, n) < \frac{1}{2}$, alors x est considéré comme normal. En effet, il faut un nombre "important" de nœuds pour séparer x du reste des données.

— Si tous les scores s pour toutes les données est d'environ $\frac{1}{2}$, alors les données ne semblent pas contenir de données aberrantes.

Il faut donc ensuite définir un seuil de décision sur s pour classer comme normales ou anormales les données. Cette méthode présente l'avantage de ne pas estimer un profil de ce qui est normal ou non et est assez rapide de par la recherche aléatoire de frontières de décision.

Auto-encodeurs Une méthode populaire de détection d'anomalie est la détection par auto-encodeurs. Un auto-encodeur est un modèle d'apprentissage profond composé d'un encodeur E_θ et d'un décodeur $D_{\theta'}$, où θ et θ' sont les paramètres des modèles. Généralement, l'espace latent *i.e.* l'espace en sortie d'encodeur est de dimension inférieure à celle d'entrée. L'objectif de l'auto-encodeur est alors d'apprendre les paramètres θ et θ' de sorte que $D_{\theta'} \circ E_\theta(x) = x$. On cherche ainsi à apprendre une reconstruction des données tout en diminuant l'espace où l'information est contenue. Pour ce faire, on optimise généralement la *mean squared error* sur le jeux de données : $\mathbb{E}_{x \sim p_{data}}[||D_{\theta'} \circ E_\theta(x) - x||_2^2]$. Soit X un jeu de données, la fonction de perte empirique à minimiser est :

$$\mathcal{L} = \frac{1}{|X|} \sum_{x \in X} ||D_{\theta'} \circ E_\theta(x) - x||_2^2 \tag{2.3}$$

Une fois l'auto-encodeur entraîné sur les données considérées comme normales, il faut ensuite déterminer un seuil de détection. Ce seuil $\nu > 0$ est la frontière entre ce qui est considéré normal ou non. Ainsi, les données normales x sont telles que $||D_{\theta'} \circ E_\theta(x) - x||_2^2 \leq \nu$. Ce sont des données "bien reconstruites". Il est important que l'auto-encodeur ait été appris sur des données considérées *a priori* comme normales. En effet, si les données anormales sont exclues de l'apprentissage, l'auto-encodeur devrait les reconstruire alors qu'il n'a jamais observé leur distribution. On s'attend alors à ce que l'erreur de reconstruction soit élevée. Il faut ensuite régler le seuil de décision ν en conséquence. Étant donné que cette méthode nécessite que le jeu de données d'entraînement ne soit composé que de données normales, on peut considérer que cette méthode est semi-supervisée.

Analyse en Composantes Principales (ACP) L'ACP [20] est une méthode pour réduire la dimension d'un ensemble de données. Cette méthode recherche les directions dans l'espace qui maximisent la variance du nuage de point. Cela revient, sur un jeu de données centré et réduit, à calculer les valeurs propres de la matrice de variance-covariance. La réduc-

tion de dimension s'opère lorsqu'on choisit les k valeurs propres les plus grandes. En effet, ce sont elles qui expliquent le plus de variance du jeu de données.

De la même manière que pour les auto-encodeurs, il est possible de détecter des anomalies avec l'ACP. Ainsi, on mesure là aussi l'erreur de reconstruction. C'est par exemple ce que font Nguyen et al. [21]. Dans leur article, ils ont réalisé de la détection d'intrusion fédérée via de la détection d'anomalie à base d'ACP.

Même si la détection d'anomalie peut être un ensemble de méthodes utiles au présent projet, celui-ci s'est orienté vers le clustering pour pouvoir appliquer plus que deux étiquettes : attaque ou normal.

2.2.2 Méthodes de clustering

Au-delà de l'algorithme k-means présenté dans la section 2.1, il existe d'autres méthodes pour réaliser du clustering de données.

GMM Une méthode plus générale que le k-means est la modélisation par mélange de gaussiennes (GMM). Dans cette modélisation, on suppose que les clusters sont issus d'une génération de données par k lois normales. Soient $\mu = (\mu_i)_{i\in\{1,\dots,k\}}$ les moyennes de ces clusters et $\Sigma = (\Sigma_i)_{i\in\{1,\dots,k\}}$ leurs matrices de covariance. Posons $p_i \in [0;1]$ la probabilité qu'une donnée soit générée par la loi $\mathcal{N}(\mu_i, \Sigma_i)$ de telle sorte que $\sum_{i=1}^{k} p_i = 1$. Les paramètres de cette modélisation sont généralement estimés grâce à l'algorithme EM [22]. Pour une donnée x, son cluster $C(x)$ est celui dont la probabilité d'appartenance est la plus grande :

$$C(x) = \arg\max_{i\in\{1,\dots,k\}} (p_i \phi(x, \mu_i, \Sigma_i)) \tag{2.4}$$

avec $\phi(x, \mu_i, \Sigma_i)$ la densité de probabilité de la loi normale de moyenne μ_i et de matrice de covariance Σ_i évaluée en x.

Bien que cette modélisation soit plus fine que le k-means grâce à la covariance appliquée aux clusters, elle est aussi bien plus coûteuse en termes de calculs. C'est pourquoi nous ne nous sommes pas dirigés vers cette méthode dans la suite.

Deep k-means L'algorithme Deep k-means (DKM) [23] est une modélisation de données inspirée par k-means. DKM réalise à la fois l'apprentissage de représentation et l'apprentissage des centroïdes. Dans l'article initial, la représentation est issue d'un autoencodeur AE composé d'un encodeur E et d'un décodeur D. Soit x une entrée du jeu de données, le clustering s'effectue dans l'espace latent. Soit $c = (c_i)_{i\in\{1,2,\dots,k\}}$ l'ensemble des centroïdes où k est le nombre de centroïdes. On calcule la distance aux centroïdes :

$$d(x,c) = \begin{pmatrix} ||E(x) - c_1||_2^2 \\ ||E(x) - c_2||_2^2 \\ \dots \\ ||E(x) - c_k||_2^2 \end{pmatrix}$$

Une fois $d(x,c)$ obtenu, on calcule la probabilité d'appartenance aux clusters avec un softmax sur les distances aux centroïdes : $p(x,c) = softmax(d(x,c))$. La perte sur laquelle la descente de gradient est calculée est composée de deux termes. Le premier $\mathcal{L}_r$ est une perte de reconstruction. Par exemple, on peut utiliser la *mean squared error* pour calculer l'écart entre x et $D \circ E(x)$. La deuxième perte $\mathcal{L}_c$ est la perte de clustering. Celle-ci mesure la qualité des clusters. Pour un seul élément x, $\mathcal{L}_c = \sum_{i=1}^{k} p(x,c_i)d(x,c_i) = p(x,c)^T d(x,c)$. Finalement, pour un jeu de données X, on a la perte $\mathcal{L}$ suivante :

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_c = \frac{1}{|X|}\sum_{x\in X} ||x - D\circ E(x)||_2^2 + \frac{1}{|X|}\sum_{x\in X}\sum_{i=1}^{k} p(x,c_i)d(x,c_i)$$

Cette méthode semble présenter de meilleures performances mais est aussi très coûteuse en temps de calcul.

DBSCAN Les méthodes présentées précédemment génèrent des clusters convexes. DBSCAN [24] est un algorithme de clustering qui agrège les points d'un jeu de données à partir de leur proximité et peut donc former des clusters non convexes. Cet algorithme cherche des zones à forte densité et connecte les points entre eux. Il y a deux hyperparamètres principaux : $\epsilon \in \mathbb{R}_+^*$ et $MinPts \in \mathbb{N}^*$. Le premier est le rayon de la boule autour de chaque point qui définit son voisinage. Le second est le nombre de points nécessaire dans un voisinage pour considérer un point comme central dans un cluster. On pose l'ϵ-voisinage d'un point p : $N_\epsilon(p) = \{q \in X | d(p,q) \leq \epsilon\}$.

Il existe trois catégories de points dans DBSCAN. La première est la catégorie des points

centraux tels que $|N_\epsilon(p)| \geq MinPts$. Ainsi, il peut y avoir plusieurs points centraux par clusters. Ils forment le cœur (*core* dans [24]) du cluster et ne sont pas nécessairement uniques. La seconde est formée des points frontières. Ces points appartiennent à l'ϵ-voisinage d'un point central. Ils appartiennent au cluster constitué par les points centraux mais sont, comme leur nom l'indique, en périphérie de celui-ci. Le dernier type de points est formé par les points aberrants. Ceux-ci ne sont ni des points centraux ni des points frontières. Ils n'appartiennent donc à aucun cluster.

2.3 Clustering fédéré

2.3.1 Clustering de clients

Le clustering fédéré est une partie de l'apprentissage fédéré qui est relativement peu traitée. En effet, la plupart des clustering dans un cadre fédéré est utilisé pour regrouper les clients pour optimiser l'apprentissage d'un modèle supervisé. Par exemple, CADIS [25] utilise l'avant-dernière couche des modèles d'apprentissage profond appris localement pour calculer une matrice de similarité des clients. L'utilisation d'une couche du modèle appris chez le client permet d'observer la répartition des données sans avoir accès aux données elles-mêmes. Au moment de l'agrégation par le serveur des modèles, les modifications apportées localement sont pondérées par la taille du jeu de données du client ainsi que par l'inverse du nombre de clients qui sont similaires à lui. De cette manière, le poids des clients est diminué pour ceux qui appartiennent à un grand cluster et augmenté pour ceux qui appartiennent à un petit cluster.

2.3.2 Clustering fédéré de données

Le clustering fédéré qui est pertinent dans notre cas vise à calculer des clusters sur les données des clients sans que ceux-ci doivent les dévoiler. L'algorithme k-FED [26] nécessite que k clusters soient présents dans les données et que chaque client dispose d'au plus $\sqrt{k}$ clusters dans ses données. Les auteurs utilisent le clustering de Aswathi et al. [27] chez les clients. Ces derniers envoient leurs centroïdes au serveur. Celui-ci agrège les clusters des clients avec un k-means de Lloyd mais dont l'initialisation n'est pas exactement k-means++. Au lieu de tirer des points avec la loi de l'algorithme 2, ligne 8, les auteurs choisissent le point des clients le plus éloigné de l'ensemble des centroïdes déjà choisis par le serveur. Cette méthode a l'avantage de fonctionner en un seul *round* de communications. Toutefois, k-FED suppose que le nombre de clusters est égal chez tous les clients. Cette hypothèse est trop forte dans le cas de notre IDS. En effet, on peut aisément supposer que chaque client subit des attaques

différentes ou même que le comportement normal de leur réseau n'induit pas le même nombre de clusters.

Deux algorithmes parus plus récemment tiennent compte d'un nombre différent de clusters chez les clients. Le premier est le k-means fédéré [16] qui a été abordé plus longuement dans la section 2.1.5. Au cours de cet algorithme, les clients calculent des centroïdes sur leurs données et communiquent la taille des clusters associés au serveur. Ce dernier les agrège avec un k-means pondéré par la taille communiquée. Ensuite, les centroïdes sont renvoyés aux clients jusqu'à convergence. Cette méthode affiche une robustesse face à des données non i.i.d. selon Garst et al. [16].

Le second algorithme est proposé par Holzer et al. [28]. Ces derniers sont plus proches de l'esprit de l'apprentissage fédéré puisque tous les clients ne participent pas à chaque *round* de communication. En effet, à chaque étape, un sous-ensemble des clients est tiré au sort. Ensuite, chaque client reçoit les clusters globaux et les met à jour avec une itération de l'algorithme de Lloyd (algorithme 1). Les nouveaux centroïdes globaux sont calculés avec la moyenne pondérée par la taille des clusters clients. Cette approche ressemble à celle proposée par Tensorflow Federated [29]. Toutefois, Holzer et al. ajoutent un taux d'apprentissage sur les modifications entrantes pour canaliser des mises à jour qui seraient très variables entre deux *rounds*. Pour ajouter en stabilité, les auteurs ont aussi inclus de l'inertie dans le modèle. C'est-à-dire que la mise à jour des centroïdes conserve une mémoire des centroïdes passés. Cela réduit ainsi l'impact de clusters calculés par les clients qui différeraient trop les uns des autres.

2.4 IDS non supervisés

2.4.1 IDS centralisés

La majeure partie de la recherche sur les IDS à base d'apprentissage automatique s'est orientée vers les modèles supervisés. Il y a tout de même de la recherche dans le sens d'IDS non supervisés. Au cours de cette section, nous passerons en revue quelques-uns de ces IDS non supervisés dont les résultats ont été évalués sur CIC-IDS2017 et/ou UNSW-NB15. En effet, ces jeux de données classiques en détection d'intrusion sont ceux utilisés pour évaluer la méthode présentée dans ce mémoire. Ils seront présentés dans la section 4.1.

Prasad et al. [30] ont proposé un algorithme de clustering créant des formes arbitraires. Pour ce faire, leur méthode initialise un grand nombre de clusters initiaux. Ensuite, ces clusters sont rassemblés par similarité. Ils accompagnent leur méthode d'une sélection de variables. Les performances qu'ils obtiennent en termes d'exactitude est de 0.8860 et 0.8828

en terme de score F_1 avec leur méthode sur l'ensemble de CIC-IDS2017 en classification binaire (attaque/normal). Ils comparent ce résultat au k-means classique qui obtient dans leur article une exactitude de 0.7725.

Sirisha et al. [31] ont réalisé une étude comparative des performances de modèles supervisés et non supervisés sur NSL-KDD et sur CIC-IDS2017. Ce sont les résultats sur ce dernier jeu de données qui nous intéressent. Ils ont obtenu respectivement pour k-means et les *isolation forest* 0.79 et 0.79 d'exactitude et 0.37 et 0.88 de score F_1 où l'attaque est la classe positive. On remarque que ce sont des valeurs inférieures à celles présentées dans le paragraphe précédent. Ce sont des performances inférieures à celles des modèles supervisés que Sirisha et al. ont évalués. Par exemple, les forêts aléatoires obtiennent un score d'exactitude de 0.9377 et un score F_1 de 0.8415. Il est récurrent de constater que les modèles supervisés ont de meilleures performances que les modèles non supervisés. Ainsi, les résultats des auteurs sont cohérents.

Comme mentionné en section 2.2.1, les auto-encodeurs peuvent servir à la détection d'anomalie. C'est vers ce type de modèles d'apprentissage profond que ce sont dirigés Yang et al. [32]. Pour leur détection d'anomalie, les auteurs ont donc entraîné un auto-encodeur sur des données normales. En plus de la détection de données anormales avec un seuil, la méthode proposée quantifie la distribution normale (*i.e.* sur des données normales) des couches cachées avec leur moyenne et leur variance. Lors de la phase de détection, la valeur de chacune des couches cachées est comparée à la normalité grâce à la distance de Mahalanobis [33]. Le score final d'anomalie est une combinaison linéaire de l'erreur de reconstruction et de toutes les distances de Mahalanobis des couches cachées. Il faut ensuite déterminer un seuil pour ce score. Avec cette méthode, les auteurs obtiennent sur UNSW-NB15 une exactitude de 0.85 ± 0.01 et un score F_1 de 0.84 ± 0.01.

2.4.2 IDS fédérés

L'apprentissage fédéré étant un paradigme d'apprentissage récent, les articles sont moins nombreux que dans le cas classique du centralisé. De la même manière que dans la partie précédente, il y a peu d'IDS non supervisés qui sont fédérés. Au moment de la rédaction de ce mémoire, nous avons trouvé peu d'IDS fédéré non supervisés utilisant CIC-IDS2017 ou UNSW-NB15 pour valider sa méthode. Nous n'aurons donc pas de point de comparaison direct. De plus, les modèles proposés n'utilisent généralement pas de clustering. De fait, comme constaté dans la section 2.3, le clustering fédéré fait l'objet de relativement peu de recherche et les méthodes sont encore nouvelles.

Un exemple d'IDS fédéré non supervisé est proposé par Shibly et al. [34]. Leur cas d'étude est spécialisé sur les bus CAN des voitures. Les auteurs appliquent l'apprentissage fédéré à

plusieurs modèles supervisés. Ils ont aussi entraîné un auto-encodeur pour de la détection d'anomalie. L'exactitude obtenue est de 0.6625 sur un jeu de données construit par eux-mêmes. Pour ce faire, ils ont mesuré le trafic d'un bus CAN sans attaque et ont ensuite enregistré le trafic lorsqu'ils ont réalisé les attaques. Pour diversifier le jeu de données, ils ont réalisé ces étapes sur trois véhicules de marques différentes.

Dans la lignée du paragraphe précédent, Md Tayeen et al. [35] ont proposé un algorithme d'apprentissage fédéré utilisant FedAvg [15]. Ils entraînent un auto-encodeur de manière fédérée sur des données normales pour ensuite réaliser de la détection d'anomalie. Toutefois, les seuls paramètres transmis aux clients sont les poids de la couche précédant l'espace latent et ceux de la couche succédant à celui-ci. Ainsi, l'entraînement fédéré ne porte à proprement parler que sur l'espace latent en lui-même. Le score F_1 obtenu sur UNSW-NB15 est de 0.91 ± 0.003.

Aouedi et al. [36] ont opté pour une approche supervisée mais en limitant la quantité de données étiquetées disponible. Le modèle proposé est composé de deux parties. La première est un auto-encodeur, non supervisé donc, appris de façon fédérée sur les données des clients. La seconde est un réseau de neurones profond dont l'espace d'entrée est l'espace latent de l'auto-encodeur et la sortie est une classification des différents types de communication. Ce dernier modèle, supervisé, est entraîné sur un jeu de données étiqueté et moins volumineux que la somme des données fédérées. Les auteurs observent notamment l'influence du ratio $R_u = \frac{\text{taille totale des données clients}}{\text{taille des données labélisées}}$ sur les performances. Selon leurs expériences, les performances sont croissantes en fonction de ce ratio. Ainsi, les données non étiquetées guident le modèle vers de meilleures prédictions. Le modèle sélectionné atteint un score d'exactitude de 0.8432 sur UNSW-NB15. Ces performances surpassent celles des modèles appris uniquement sur les données du serveur durant les expériences de Aouedi et al. Même si les résultats de cet article ont été obtenus sur UNSW-NB15, le modèle utilisé *in fine* est supervisé et donc ne peut pas servir de référence pour notre cas.

En cherchant explicitement des IDS ayant du clustering en étant fédéré, on peut trouver des articles qui traitent du clustering sur les clients. Par exemple, Yang et al. [37] proposent de trouver des clients malveillants avec du clustering. Ou bien, on peut trouver des articles qui traitent des clusters (nœuds) de calcul [38, 39]. Ainsi, le clustering fédéré est lui-même peu sujet à des applications pour la détection d'intrusion.

2.4.3 Récapitulatif

Le tableau 2.1 récapitule les performances des modèles présentés durant cette section 2.4. Les cases où se trouvent une barre oblique "/" indiquent que la donnée n'était pas fournie dans l'article correspondant.

Auteur(s)	Jeu de données	Paradigme	Exactitude	F_1
Prasaad et al. [30]	CIC-IDS2017	Centralisé	0.8860	0.8828
Sirisha et al. [31] (k-means)	CIC-IDS2017	Centralisé	0.79	0.88
Sirisha et al. [31] (IForests)	CIC-IDS2017	Centralisé	0.37	0.91
Yang et al. [32]	UNSW-NB15	Centralisé	0.85	0.84
Shibly et al. [34]	Personnalisé	Fédéré	0.6625	/
Md Tayeen et al. [35]	UNSW-NB15	Fédéré	/	0.91
Aouedi et al. [36]	UNSW-NB15	Fédéré	0.8432	/

Tableau 2.1 Récapitulatif des performances des IDS mentionnés

CHAPITRE 3 MÉTHODOLOGIE

L'IDS proposé se décompose en deux parties principales. La première est l'étape de clustering. Les données sont regroupées dans des clusters par un algorithme. Dans un cadre centralisé, on utilise k-means, et dans le cas fédéré, on utilise l'un des algorithmes décrits le long de ce chapitre 3. Une fois les clusters créés, l'objectif est d'en déterminer la nature. C'est-à-dire classer les clusters suivant qu'ils contiennent des communications bénignes ou des attaques. On ne donne qu'une étiquette à chaque cluster. C'est ici qu'intervient l'expertise des clients pour identifier la classe des clusters. Ceci est abordé dans la section 3.2. Une fois que les clusters sont classés selon les communications qu'ils contiennent, on peut réaliser des prédictions sur un jeu de données. Une fois l'entraînement et la validation terminés (section 3.3), on peut utiliser l'IDS dans un cas réel. Cela signifie classer les nouvelles communications en temps réel. Pour une nouvelle donnée, le modèle la classe dans un cluster. Ensuite, le type de communication qui lui est attribué est celui du cluster. Le processus de détection d'intrusion est schématisé par la figure 3.1.

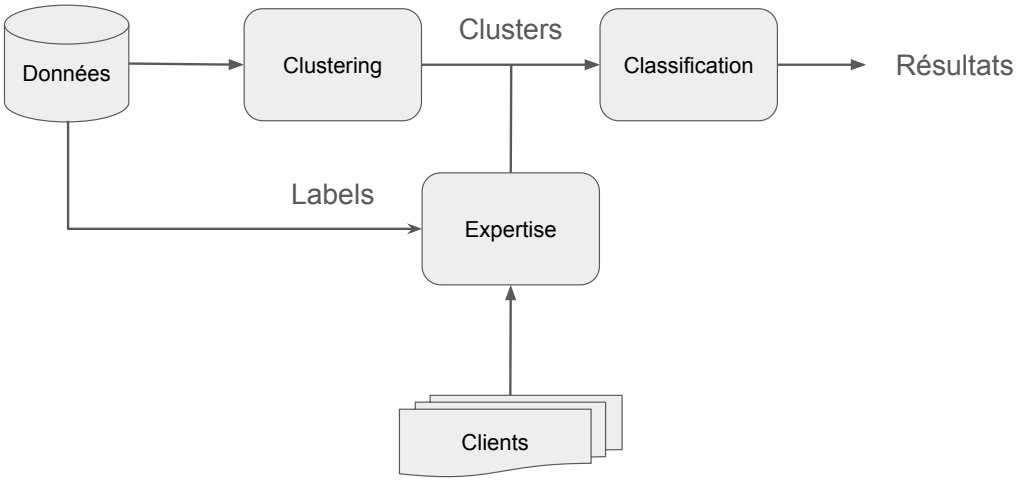


FIGURE 3.1 Schématisation du processus de détection d'intrusion proposé

3.1 Proposition d'algorithmes fédérés

Cette section présente les différents algorithmes fédérés utilisés dans la méthode. Nous avons déjà abordé le k-means fédéré de Garst et al. dans la section 2.1.5. Dans la section 3.1.1, nous améliorons la confidentialité des données des clients par rapport au k-means fédéré de Garst et al.. Nous utilisons la même structure de mise à jour des clusters mais nous modifions l'initialisation pour qu'elle dévoile moins de données des clients. Dans la section 3.1.2, nous proposons le meta k-means. Cet algorithme de clustering ne dévoile pas de données des clients lors de son utilisation et délègue la recherche d'un nombre de clusters optimal aux clients.

Les algorithmes des sections 2.1.5 et 3.1.1 fonctionnent lorsque le nombre de clusters k est fixé. Comme expliqué dans la section 2.1.3, nous utilisons la silhouette pour déterminer un nombre de clusters k pertinent pour le clustering. Dans la section 3.1.3, nous proposons une formulation de la silhouette simplifiée qui est utilisable dans un cadre fédéré.

Pour que les algorithmes mentionnées puissent fonctionner, nous supposons que les clients sont honnêtes mais curieux. C'est-à-dire qu'ils suivent toutes les instructions à la lettre mais cherchent à tirer le plus d'information possible de celles-ci. Ils ne cherchent donc pas à dégrader les performances de l'IDS ou transgresser les instructions.

3.1.1 Initialisation k-means++ fédérée

L'initialisation utilisée dans le k-means fédéré de Garst et al. [16] demande à chaque client de réaliser un k-means++ sur ses données. Or l'algorithme k-means++, algorithme 2, choisit les centroïdes parmi les données sur lesquelles il est appliqué. Ainsi, s'il y a N clients participant au clustering et k clusters à déterminer, les clients divulguent kN communications. Dans cette section, nous proposons une reformulation fédérée de l'initialisation k-means++ qui est équivalente à la version centralisée et qui préserve mieux la confidentialité que l'initialisation de Garst et al. [16].

Reformulation du k-means++

Avant de décrire l'algorithme, on remarque certaines propriétés des données qui seront utiles pour le déroulement du k-means++ fédéré. Posons X le jeu de données obtenu en concaténant les jeux de données des clients $(X_j)_j$. Notons $\mathcal{U}$ la loi uniforme sur X. On suppose qu'il n'y a pas de doublons dans X. Même si c'était le cas, on pourrait distinguer les doublons grâce à un identifiant de base de données. On pose $\mathcal{C} = (\mathcal{C}_j)_{j\in\{1,\dots,N\}}$ l'ensemble des clients et $(X_j)_j$ leurs jeu de données associés. On remarque que la loi uniforme sur X pour $x \in X_j$ s'écrit comme suit :

$$
\begin{aligned}
P_{\mathcal{U}}(x) &= \frac{1}{|X|} \\
&= P_{\mathcal{U}}(C_j) P_{\mathcal{U}}(x|C_j) \\
&= \frac{|X_j|}{|X|} \times \frac{1}{|X_j|}
\end{aligned}
\tag{3.1}
$$

En termes d'interprétation, on peut voir $\frac{|X_j|}{|X|}$ comme la probabilité de sélectionner le client dont le jeu de données contient x. La probabilité de tirer aléatoirement ce client est proportionnelle à la taille de son jeu de données. Ensuite, la probabilité $\frac{1}{|X_j|}$ est la probabilité uniforme sur le jeu de données du client $\mathcal{C}_j$.

Soit c l'ensemble des centroïdes. Posons $\forall x \in X, d(x,c) = \min_{i \in \{1,\dots,|c|\}} (||x - c_i||_2^2)$. Cette notation issue de la définition originelle du k-means++ désigne la proximité d'un point au centroïde le plus proche. Notons $\mathcal{D}^2$ la loi de probabilité telle que pour $x \in X$, $P_{\mathcal{D}^2}(x) = \frac{d(x,c)}{\sum_{y \in X} d(y,c)}$. Définissons $Z = \sum_{y \in X} d(y,c)$ le facteur de normalisation de la loi $\mathcal{D}^2$ sur X. On pose $Z_j = \sum_{y \in X_j} d(y,c)$ où Z_j est donc le facteur de normalisation de la loi $\mathcal{D}^2$ locale du client $\mathcal{C}_j$, sur le jeu de données X_j. On remarque alors que $Z = \sum_{\mathcal{C}_j \in \mathcal{C}} Z_j$. On utilise aussi l'hypothèse selon laquelle tous les points, même égaux, sont distinguables ou uniques. Ainsi, la loi $\mathcal{D}^2$ sur X avec $x \in X_j$ s'écrit :

$$
\begin{aligned}
P_{\mathcal{D}^2}(x) &= \frac{d(x,c)}{Z} \\
&= P(C_j) P_{\mathcal{D}^2}(x|C_j) \\
&= \frac{Z_j}{Z} \times \frac{d(x,c)}{Z_j}
\end{aligned}
\tag{3.2}
$$

On peut voir $\frac{Z_j}{Z}$ comme la probabilité de sélectionner le client contenant x en fonction de la dissimilitude de ses données par rapport aux centroïdes déjà sélectionnés. Ensuite, la probabilité $\frac{d(x,c)}{Z_j}$ est la loi $\mathcal{D}^2$ sur le jeu de données de $\mathcal{C}_j$.

Ainsi, les lois de probabilités du k-means++ peuvent être écrites de sorte que le calcul soit décentralisé sans révéler l'ensemble des données des clients.

L'algorithme 4 exprime le déroulement de la reformulation de k-means++ proposée. On remarque que cette méthode ne dévoile que k communications des clients comparées aux kN de

Algorithm 4 K-means++ fédéré - Federated k-means++

Require: k, $\mathcal{C} = (\mathcal{C}_j)_j$, $(X_j)_j$

Le serveur tire un client $\mathcal{C}_j$ dans $\mathcal{C}$ avec $P(\mathcal{C}_j) = \frac{|X_j|}{|X|}$
Le client $\mathcal{C}_j$ tire un premier centroïde c_1 uniformément sur X_j
$c \leftarrow \{c_1\}$
for $i \in \{2, ..., k\}$ **do**
    Le serveur tire un client $\mathcal{C}_{i'}$ dans $\mathcal{C}$ avec $\forall j, P(C_j) = \frac{Z_j}{Z}$
    Le client $\mathcal{C}_{i'}$ tire c_i où $P(c_i|\mathcal{C}_{i'}) = \frac{d(c_i,c)}{Z_{i'}}$
    $c \leftarrow c \cup \{c_i\}$
return c

l'initialisation du k-means fédéré. Ainsi, l'initialisation k-means++ fédéré proposée préserve mieux la confidentialité des données des clients que celle de Garst et al. [16] (algorithme 3).

Insertion dans k-means fédéré

Pour que le federated k-means respecte mieux la confidentialité, on peut remplacer l'initialisation de cet algorithme par le k-means++ fédéré présenté dans l'algorithme 4. Ainsi, au lieu de réaliser une initialisation k-means++ chez chaque client puis calculer un k-means sur l'ensemble des centroïdes des clients, le serveur coordonne l'algorithme 4. Une fois les centroïdes initiaux calculés, le serveur les partage aux clients. Ceux-ci peuvent alors réaliser les étapes de l'algorithme de Lloyd, algorithme 1. Le serveur agrège ensuite les centroïdes des clients, comme énoncé dans le k-means fédéré. La modification est représentée dans l'algorithme 5. On remarque que pour $r = 0$, l'algorithme 5 revient à simplement utiliser l'initialisation k-means++ fédérée.

Ainsi, avec l'algorithme 5, nous proposons un algorithme de clustering fédéré préservant mieux la confidentialité des données des clients que celui proposé par Garst et al [16]. Ceci correspond à l'objectif 2 de la section 1.3.

3.1.2 Meta k-means

Lors de ce projet, nous proposons un autre algorithme de clustering fédéré. De fait, l'initialisation k-means++ fédérée divulgue tout de même des données des clients. Pour contourner l'initialisation k-means++, le second algorithme que nous proposons impose aux clients de trouver d'abord des clusters dans leur jeu de données avant de les envoyer au serveur.

Chaque client optimise sur son propre jeu de données le nombre de clusters optimal comme décrit dans la fonction opti_clusters de l'algorithme 6. Le critère d'optimisation est la maxi-

Algorithm 5 K-means fédéré avec initialisation k-means++

Require: k

Initialisation k-means++ fédérée.
$c_g \leftarrow$ federated_kmeans++$(k, \mathcal{C}, (X_j)_j,)$
for round r **do**
    Pour chaque client $\mathcal{C}_j \in \mathcal{C}$:
        $c^j \leftarrow c^g$
        Déterminer s^j
        # Le client $\mathcal{C}_j$ ignore les centroïdes dont le cluster est vide sur ses données.
        $s^j \leftarrow C^j[s! = 0 \text{ for } s \text{ in } s^j]$
        $k_j \leftarrow \text{taille}(s_j)$
        # Les clients ne réalisent qu'une étape de l'algorithme de Lloyd.
        # L'algorithme de Lloyd s'applique sur les centroïdes c^j ici.
        $s^j, c^j \leftarrow \text{kmeans}(X_j, k_j, \text{ init } = c^j)$
        Envoyer s^j, c^j au serveur
    Sur le serveur :
        # On concatène les clusters des clients et les tailles des clusters correspondants.
        # Cela crée un jeu de données pour le k-means du serveur.
        $c \leftarrow [c^1|c^2|...|c^N]$
        $s \leftarrow [s^1|s^2|...|s^N]$
        # Étape d'agrégation avec l'algortihme de Lloyd pondéré par la taille des clusters.
        $c^g \leftarrow \text{kmeans}(c, k, \text{ weights } = s)$
        Envoyer c^g aux clients

misation de la silhouette simplifiée moyenne sur chaque jeu de données client. L'itération pour chaque client est présentée dans la fonction opti_client de l'algorithme 7. Ensuite, chaque client envoie ses centroïdes optimaux au serveur ainsi que la taille des clusters associés. Le serveur concatène les centroïdes des clients pour former un jeu de données sur lequel il pourra s'entraîner. Le serveur réalise ensuite une opération similaire. Il optimise le nombre de "meta-clusters" sur les centroïdes envoyés par les clients pondérés par la taille des clusters chez les clients. Il envoie ensuite le résultat à tous les clients.

Pour tenter d'améliorer les performances du modèle, nous nous sommes inspirés du clustering fédéré de Garst et al. [16], abordé dans la section 2.1.5. Les auteurs de cet article itèrent pour obtenir de meilleurs centroïdes. Après la recherche du nombre optimal de centroïdes par clients, ceux-ci envoient au serveur les centroïdes ainsi que la taille des clusters. Ensuite, le serveur réalise sa recherche de centroïdes comme précédemment. A partir de là, le serveur renvoie aux clients ses centroïdes. Les clients éliminent les centroïdes auxquels aucune de leurs données n'est attribuée puis appliquent l'algorithme de Lloyd (algorithme 1) avec pour initialisation les centroïdes globaux, c'est la fonction mise_a_jour_clients. Ils envoient leurs centroïdes accompagnés de la taille des clusters au serveur. Le serveur cherche le nombre optimal de centroïdes. Si la convergence n'est pas atteinte, le serveur diffuse à nouveau les centroïdes globaux dans le réseau pour relancer les calculs, sinon les clients récupèrent simplement les centroïdes. L'algorithme complet est décrit par l'algorithme 7.

Posons pour l'expérience :

- $\mathcal{K} = \{2; ...; K\}$, où $K \in \mathbb{N}$, le panel de clusters à essayer pour chaque client et le serveur.
- $\mathcal{C}$ l'ensemble des clients de l'expérience.
- $(X_j)_j$ l'ensemble des données pour chaque client, X_j étant le jeu de données du client $\mathcal{C}_j$.
- *clust* un algorithme de clustering, en l'occurrence k-means pour notre cas de figure.
- $r \in \mathbb{N}$ est le nombre de *rounds* de communications en supplément de l'initialisation que l'algorithme doit calculer.

Algorithm 6 Optimisation du nombre de clusters

Require: $\mathcal{K}$, X

function OPTI_CLUSTERS($\mathcal{K}$, X)
    $s_{max} \leftarrow -1$
    for $k \in \mathcal{K}$ **do**
        $c^k \leftarrow \text{clust}(k, X)$
        # Moyenne de la silhouette simplifiée. Rappel : $s(x)$ est la silhouette du point x.
        $s^k \leftarrow \frac{1}{|X|} \sum_{x \in X} s(x)$
        if $s^k > s_{max}$ **then**
            $s_{max} \leftarrow s^k$
            # On conserve les centroïdes qui minimisent la silhouette.
            $c_{opti} \leftarrow c^k$
            Déterminer s_{opti}, la taillle de chaque clusters.
    return c_{opti}, s_{opti}

Algorithm 7 Meta k-means

Require: $\mathcal{K}$, r, $\mathcal{C}$, $(X_j)_j$

function OPTI_CLIENT($\mathcal{K}$, $(X_j)_j$)
    # Calcul des centroïdes des clients.
    for $\mathcal{C}_j \in \mathcal{C}$ **do**
        # Chaque client optimise ses clusters.
        c^j, $s^j \leftarrow$OPTI_CLUSTERS($\mathcal{K}$, X_j)
    # Concaténation des centroïdes.
    $c \leftarrow [c^1|...|c^N]$
    # Concaténation des tailles des clusters.
    $s \leftarrow [s^1|...|s^N]$
    return c, s

function META_K_MEANS($\mathcal{K}$, $(X_j)_j$)
    # Récupération de l'ensemble des clusters des clients.
    $c, s \leftarrow$ OPTI_CLIENT($\mathcal{K}$, $(X_j)_j$)
    # Optimisation serveur en pondérant par la taille des clusters des clients.
    $c^g \leftarrow$ OPTI_CLUSTERS($\mathcal{K}$, [c,s])
    for $round \in \{1; ...; r\}$ **do**
        # Les clients mettent à jour les centroïdes serveur avec une étape de
        # l'algorithme de Lloyd sur leurs données.
        $c, s \leftarrow$ mise_a_jour_clients($(X_j)_j$, c^g)
        # Optimisation serveur en pondérant par la taille des clusters des clients.
        $c^g \leftarrow$ OPTI_CLUSTERS($\mathcal{K}$, [c,s])
    Diffuser c^g aux clients.

3.1.3 Silhouette fédérée

Pour sélectionner un nombre de clusters pertinent, par exemple dans le cas du k-means fédéré (algorithme 3), il faut définir un calcul de silhouette à l'échelle des clients. Toutefois, la silhouette originelle [13] nécessite de connaître la distance entre chaque point de chaque client avec ceux des autres. Étant donné que l'objectif 2 de recherche vise à conserver un maximum de confidentialité durant, cette méthode est exclue. En effet, avec suffisamment de points, un client pourrait deviner l'emplacement de ceux de tous les autres clients, ce qui annulerait toutes les autres précautions prises pour limiter les informations à divulguer.

Pour contourner ce problème, la formulation de la silhouette simplifiée [14] est utilisée. En effet, celle-ci ne nécessite pour chaque client que de connaître les centroïdes du clustering. Or les centroïdes calculés par le serveur sont considérés comme une information disponible à tous les clients.

L'algorithme se déroule comme suit. Une fois le clustering fédéré fini, chaque client reçoit les centroïdes de la part du serveur. Une fois les centroïdes obtenus, les clients peuvent calculer la silhouette simplifiée sur leurs propres données. Les clients envoient ensuite au serveur la silhouette simplifiée moyenne ainsi que la taille des données sur lesquelles elle a été déterminée. Le serveur calcule finalement la moyenne pondérée des silhouettes des clients. On obtient alors la silhouette simplifiée moyenne de l'ensemble des données. En effet, en reprenant la notation utilisée précédemment, la silhouette simplifiée moyenne d'un client C_j est donc :

$$\overline{S_{C_j}} = \frac{1}{|X_j|} \sum_{x \in X_j} s(x) \tag{3.3}$$

où $s(x)$ est la silhouette simplifiée de la donnée x (cf. section 2.1.3).

La silhouette simplifiée fédérée calculée par le serveur vaut finalement :

$$\begin{aligned} S &= \frac{1}{\sum_{j=1}^{N} |X_j|} \sum_{j=1}^{N} |X_j| \overline{S_{C_j}} \\ &= \frac{1}{\sum_{j=1}^{N} |X_j|} \sum_{j=1}^{N} \sum_{x \in X_j} s(x) \\ S &= \frac{1}{|X|} \sum_{x \in X} s(x) \end{aligned} \tag{3.4}$$

Donc S est bien la moyenne des silhouettes simplifiées de toutes les données, comme si le calcul avait été effectué par un serveur connaissant toutes les données des clients. Durant ce projet, la silhouette fédérée est utilisée comme critère d'optimisation dans le cadre fédéré. C'est-à-dire qu'on sélectionne le nombre de clusters qui correspond au maximum de la silhouette fédérée moyenne.

3.2 Classification

Une fois que les clusters sont obtenus, il faut déterminer à quelle classe ils appartiennent. Dans cette section seront présentées les différentes méthodes proposées pour classer un cluster dans le cadre de l'IDS de ce mémoire. La dernière sous-section justifie l'utilisation de ce type de classification.

3.2.1 Classification centralisée

L'objectif principal du projet est de développer un IDS fédéré. Pour cela, il faut alors classer les données soit comme des attaques, soit comme du trafic bénin dans le cas de données réseau, ce qui est le cas d'application. On peut aussi penser à une classification plus fine comme abordée dans la section 3.2.3.

Une fois les clusters obtenus, il est nécessaire de faire appel à un certain degré d'expertise pour les analyser, comme dans tout usage du clustering. Dans le cas centralisé *i.e.* quand un k-means est utilisé de façon classique sur un unique jeu de données, chaque cluster est identifié à la classe majoritaire en son sein. C'est-à-dire que pour chaque cluster, on observe par exemple si le trafic bénin est majoritaire. Dans ce cas, le cluster est considéré comme étant un cluster bénin. La logique inverse fonctionne si le cluster est composé en majorité d'attaques. Toutes les communications classées dans un cluster sont alors considérées comme étant du type de celui-ci dans une phase de prédiction.

3.2.2 Vote binaire

Dans le cas présent, on suppose que chaque client est en capacité de mesurer relativement finement la composition de chaque cluster. Ainsi, les participants doivent indiquer pour chaque cluster sa composition entre trafic bénin et trafic d'attaque ainsi que sa taille sur leurs données. Le serveur calcule ensuite la moyenne de la proportion de communications bénignes dans chaque cluster en la pondérant par la taille de ceux-ci. On retrouve ainsi la proportion de communications bénignes du cluster comme dans le cas centralisé. Lors de la phase de

prédictions, chaque donnée est classée selon le type de son cluster comme mentionné dans la section précédente.

Pour chaque cluster, chaque client détermine $p_{i,j}$ la proportion de communications bénignes dans le cluster i pour le client j. On a, comme précédemment, s_i^j la taille du cluster i chez le client j. La moyenne suivante est alors calculée :

$$P_i = \frac{\sum_j p_{i,j} \times s_i^j}{\sum_j s_i^j} \tag{3.5}$$

Le serveur considère le cluster i comme bénin si $P_i > 0.5$ et malveillant sinon. Au cours de ce mémoire, nous appelons ce processus le *vote des clients*.

3.2.3 Vote multi-classes

Une partie de l'étude des résultats porte sur la classification multi-classes. Pour ce faire, on considère que les communications appartiennent à l'une des T catégories de communications connues dans le réseau. Il y a nécessairement dans les T catégories le type de communication normale. Pour faire voter les clients, deux méthodes ont été étudiées.

La classification d'un cluster se fait sur la base de la proportion des types de communications présentes dans celui-ci. Pour ce faire, on calcule une matrice P' de taille $|c| \times T$. Ainsi, la valeur $P'_{i,t}$ est la proportion du t-ème type de communication dans le cluster i.

Classification à un tour Dans le cas d'une classification à un tour, le type du cluster est celui de la communication la plus représentée. Par exemple, un cluster composé de 40% de communications bénignes, 30% de communications *DDoS* et 30% de *DoS Hulk* est considéré comme étant un cluster de communications normales.

Classification à deux tours Dans une classification à deux tours, on choisit d'abord la catégorie bénigne ou attaque selon la catégorie majoritaire. Si le cluster est considéré comme de l'attaque, son type est celui de l'attaque la plus représentée. Dans l'exemple ci-dessus, le cluster serait considéré comme de l'attaque puis l'attaque majoritaire étant le *DDoS*, il serait considéré comme tel. L'objectif de cette méthode est d'éviter un cas où le vote à un tour favoriserait les communications bénignes. En effet, dans l'exemple ci-dessus, le trafic normal est le type le plus représenté mais il n'est pas majoritaire. Il peut donc sembler faux de classer le cluster comme du trafic bénin.

La classification fédérée se base sur les mêmes principes. Toutefois, le serveur n'ayant pas accès aux données des clients, il demande à chaque participant de transmettre la population dans chaque cluster de chacun des types de communications. Ainsi, chaque client envoie au serveur un tableau de taille $|c| \times T$. Le serveur moyenne toutes les tables des clients entre elles pour se ramener au cas décrit dans les paragraphes précédents.

Ainsi, si $n_{i,t,j}$ est le nombre de communications t dans le cluster i du client j, la matrice P' s'obtient comme suit :

$$P'_{i,t} = \frac{\sum_{j=1}^{N} n_{i,t,j}}{\sum_{m=1}^{T} \sum_{j=1}^{N} n_{i,m,j}} \tag{3.6}$$

Il n'y a pas de pondération puisque dans ce cas, on ne passe pas par les proportions de chaque type chez les clients. On utilise directement le nombre de communications. La matrice P' est utilisée par le serveur pour le vote multi-classe comme mentionnée précédemment dans cette section 3.2.3.

3.2.4 Discussion

On pourrait choisir un seuil η différent de 0.5 pour la classification binaire. Cela n'a pas été étudié dans le cadre des travaux présentés dans ce mémoire. Toutefois, un seuil $\eta = 0.5$ semble être une bonne première approche. En effet, en choisissant la majorité du cluster comme étant la classe de celui-ci, l'exactitude sur cette partition est nécessairement supérieure ou égale à 0.5. Cette méthode cause des faux négatifs dans le cas où des attaques se trouvent dans un cluster à plus de 50% bénin. A l'inverse, les faux négatifs apparaissent dans le cas où des communications normales se trouvent dans un cluster d'attaque. Les participants pourraient considérer que les faux négatifs sont importants à limiter. Alors, le seuil η pourrait être réduit à des valeurs inférieures à 0.5. Donc, plus de clusters seraient considérés comme de l'attaque. De cette manière, l'IDS détecterait plus d'attaques mais causerait nécessairement plus de faux positifs. Au contraire, si les faux positifs sont importants à réduire, alors le seuil pourrait être rehaussé au dessus de 0.5. Ainsi, plus de clusters seraient considérés comme bénins et donc cela limiterait les faux positifs au détriment des faux négatifs.

Pour pouvoir appliquer les méthodes de classifications proposées dans cette section, nous supposons que les clients sont au moins capables de déterminer quelles sont les classes dominantes dans les clusters. L'idéal est que les clients puissent déterminer la proportion exacte

de chaque type de communication dans les clusters.

D'un point de vue pratique, on peut se demander comment les clients peuvent déterminer la composition de chaque cluster. Nous observerons dans la section 4.2.2 que la proportion de communications normales dans une partition est en général assez élevée ou au contraire très basse. En tous cas, celle-ci est suffisamment éloignée de $\eta = 0.5$ pour que la décision soit satisfaisante. Nous constaterons dans la section 5.3.2 que c'est aussi le cas dans le cadre fédéré. Ainsi, il suffit pour un analyste d'échantillonner des communications du cluster, de les analyser et d'en déduire la nature de ce dernier. En effet, si une partition est bénigne, la proportion de communications malveillantes est proche de zéro. Ainsi la probabilité de tirer des communications d'attaques est faible et donc les communications bénignes sont majoritaires dans le tirage. Le raisonnement inverse fonctionne pour les clusters malveillants. Une autre approche serait d'utiliser de la détection d'intrusions par signature pour évaluer chaque cluster. Toutefois, cette méthode se restreint aux attaques connues. Dans tous les cas, il n'est pas nécessaire d'analyser l'entièreté de chaque partition pour en déterminer la nature. Ainsi, même si un analyste doit déterminer le type de chaque cluster, cela représente toujours moins de travail que d'analyser chaque communication pour les annoter comme dans le cas de l'apprentissage supervisé. On constate ainsi le gain à utiliser une méthode d'apprentissage non supervisé.

3.3 Évaluation des performances

Dans cette section 3.3, nous présenterons le protocole d'entraînement que nous avons établis pour l'IDS. Nous aborderons aussi la phase de prédiction puis celle d'évaluation. Finalement, nous introduirons les métriques utilisées pour évaluer les performances de l'IDS.

3.3.1 Entraînement et prédiction

Pour entraîner l'IDS et le faire prédire des classes pour chaque communication, nous avons établis le protocole suivant :

1. L'algorithme de clustering est entraîné selon plusieurs valeurs de nombre de clusters k, et si pertinent le nombre de *rounds* de communications r.
2. Le nombre de clusters optimal est le maximum, ou un maximum local, de la courbe de silhouette moyennée sur les données.
3. Si pertinent, on choisit le nombre r de communications minimal pour que la silhouette présente au moins un maximum local clair. Cela signifie qu'on choisit le modèle le plus simple *i.e.* la valeur de r la plus basse possible qui possède cette caractéristique.

4. Selon le cas :
   (a) Cas central : La proportion de chaque classe présente dans le jeu de données dans chaque cluster est calculée.
   (b) Cas fédéré : Les clients votent pour la composition des clusters et le serveur agrège les proportions obtenues.
5. La classe des clusters est déterminée en fonction des proportions de chaque classe en leur sein selon le type de classification choisi (cf. section 3.2).
6. Chaque communication prédite comme appartenant à un cluster est classée selon la classe de celui-ci.

Les points 1 à 5 relèvent de l'entraînement et l'étape 6 de la phase de prédiction. Les étapes 1 à 3 indiquent la méthode de sélection des hyperparamètres des modèles présentés au cours de ce chapitre.

3.3.2 Méthode d'évaluation

Les jeux de données fédérés sont générés à partir du jeu de données d'entraînement. Les méthodes pour générer les jeux de données i.i.d. et non i.i.d. sont présentées dans les sections 5.1.1 et 5.1.2. Les performances des modèles sont ensuite évaluées sur un jeu de données de test de façon centralisée. Cela revient au cas où le serveur disposerait d'un jeu de données de test et qu'il procède à l'évaluation du modèle dessus. Ainsi, les modèles sont comparés sur des jeux de données de tests qui ont été générés de la même manière et ont les mêmes propriétés que ce soit en configuration centralisée, i.i.d. et non i.i.d..

L'évaluation du modèle aurait pu être décentralisée. En effet, le serveur pourrait demander aux clients leurs matrices de confusion pour calculer les performances. Tensorflow Federated propose par exemple une évaluation fédérée [40]. Ceci est faisable en pratique mais cela n'a pas été implémenté dans le cadre de ce mémoire pour deux raisons. D'une part, à jeux de données égaux entre le cas centralisé et le cas décentralisé et à modèle égal, les métriques seraient les mêmes. Cela signifie que les résultats de l'évaluation seraient les mêmes et ainsi il n'est pas nécessaire de la décentraliser. D'autre part, les jeux de données non i.i.d. construits par la suite n'utilisent pas toutes les données. Donc implémenter une évaluation fédérée n'aurait pas d'intérêt dans notre cas puisqu'elle ne s'effectuerait pas à jeux de données égaux entre toutes les expériences. Ainsi, pour uniformiser les tests des modèles, nous avons utilisé une procédure d'évaluation centralisée comme mentionné dans le paragraphe précédent.

3.3.3 Métriques

Matrice de confusion Lors d'une classification, on utilise généralement une matrice de confusion. On peut ainsi caractériser l'erreur commise par le modèle lors de la prédiction. Les valeurs du tableau sont utiles pour la définition des métriques dans cette section. La matrice de confusion adaptée est présentée par le tableau 3.1. L'attaque est considérée comme la classe positive puisque c'est ce qu'un IDS doit détecter.

	Bénin (prédit)	Attaque (prédit)
Bénin (vrai)	Vrai négatif (TN)	Faux positif (FP)
Attaque (vrai)	Faux négatif (FN)	Vrai positif (TP)

Tableau 3.1 Matrice de confusion

Dans le tableau 3.1, la diagonale vrai négatif - vrai positif correspond aux données qui ont été classées correctement par rapport à leur véritable étiquette. L'autre diagonale correspond donc aux données mal classées. De fait, les faux positifs sont des entrées normales considérées comme des attaques par le modèle. Cette classification pousserait des analystes à enquêter sur des communications bénignes. Ceci risque de causer de l'*alert fatigue* comme mentionné dans les sections 1.1.1 et 1.1.2. A l'inverse, les faux négatifs sont des attaques considérées comme du trafic normal. Cette classification est problématique puisque ce sont des attaques qui ne sont pas détectées par l'IDS et donc le SOC ne peut pas réagir en conséquence.

Pour mesurer la performance des modèles, les métriques suivantes ont été utilisées.

Exactitude L'exactitude (*accuracy*) est le taux de "bonne" classification par le modèle. Dans le cas binaire comme dans la matrice 3.1, cela revient à calculer :

$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.7}$$

Plus généralement, dans la classification multi-classes, le calcul serait :

$$Acc = \frac{\#\text{Bonne classification}}{\text{Taille du jeux de données}} \tag{3.8}$$

Ainsi, un score d'exactitude de 1 signifie que toutes les données ont été prédites dans leur vraie classe. Ce score décrit donc un prédicteur parfait.

Précision La précision (*precision*) est une métrique mesurant la proportion de vrais positifs parmi les prédictions positives. La formule est la suivante :

$$Prec = \frac{TP}{TP + FP} \tag{3.9}$$

Un score de précision de 1 implique que tout ce qui est prédit comme positif est bel et bien positif. Toutefois, cette métrique n'informe en rien sur les faux négatifs. C'est le rôle du rappel.

Rappel Le rappel (*recall*) mesure le taux de vrais positifs rapporté au support de la classe positive. Ainsi, ceci compare le nombre de vrais positifs et les faux négatifs. La formule est la suivante :

$$Rec = \frac{TP}{TP + FN} \tag{3.10}$$

Un score de rappel de 1 signifie donc que toutes les données positives ont bien été classées comme positives. Cela ne renseigne toutefois pas sur le taux de faux positifs car c'est le rôle du score de précision.

Score F_1 Le score F_1 synthétise les résultats de précision et de rappel. Ce score est la moyenne harmonique de ces métriques. Sa formule est :

$$F_1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{3.11}$$

Le score F_1 reflète les performances à la fois en matière de précision et de rappel. Un score F_1 de 1 est atteint par un prédicteur parfait.

Dans le cadre du présent projet, la précision revêt une grande importance. En effet, pour limiter l'*alert fatigue* des analystes, il vaut mieux avoir un faible nombre de faux positifs. Or, plus la précision est proche de 1 plus le taux de faux positifs est faible devant les vrais positifs.

3.4 Synthèse

En introduction de ce chapitre 3, nous avons proposé une architecture d'IDS répondant à l'objectif 1. Cet IDS est composé de deux parties majeures : le clustering et la classification. L'IDS proposé peut être utilisé dans un cadre centralisé lorsque le clustering est réalisé avec

l'algorithme k-means classique (section 2.1.1) et la classification centralisée (sections 3.2.1 et 3.2.3). Dans le cadre fédéré, nous utilisons les algorithmes de clustering des sections 2.1.5, 3.1.1 et 3.1.2. Pour la classification, l'IDS fédéré utilise le système de vote mis en place dans les sections 3.2.2 et 3.2.3. Dans le cas fédéré, la collaboration est assurée grâce aux algorithmes de clustering ainsi que le vote des clients. De fait, les algorithmes de clustering utilisent les données de tous les clients. De même, lors de la phase de votes, les clients mettent en commun leurs expertises pour classer les clusters. L'architecture d'IDS proposée répond donc bien à l'objectif 1.

Pour que l'entraînement de l'IDS se déroule correctement, nous supposons que les clients sont honnêtes mais curieux. Ils suivent donc scrupuleusement les instructions énoncées dans ce chapitre mais cherchent à en tirer le plus d'informations possible. De plus, dans la phase de votes, les clients sont supposés capables de déterminer de façon fiable la composition des clusters. Tout du moins, il faut pouvoir déterminer quelles classes sont les plus importantes en nombre dans les partitions. De cette manière, les différents types de classification décrits en section 3.2 peuvent s'appliquer.

Pour répondre à l'objectif d'amélioration de la confidentialité des données des clients (objectif 2), nous avons proposé les algorithmes de clustering des sections 3.1.1 et 3.1.2. Ces algorithmes dévoilent moins de données des participants que ce que nous avons trouvé dans l'état de l'art (section 2.1.5). Pour assurer l'entraînement de certains des algorithmes de clustering, nous avons proposé la silhouette simplifiée fédérée dans la section 3.1.3. Cette silhouette ne dévoile pas de données des clients et est équivalente à la version centralisée. Ainsi, l'objectif 2 est respecté.

Comme mentionné dans la section 2.1.4, l'apprentissage fédéré permet de décentraliser une partie du calcul. Il n'est donc pas nécessaire d'avoir autant de puissance de calcul au niveau du serveur central que dans le cas d'un apprentissage centralisé. De plus, les protocoles proposés et étudiés s'adaptent à n'importe quel nombre de client $N > 1$. Toutefois, il n'est pas tenu compte de leur disponibilité. De fait, il est nécessaire que tous les clients répondent au serveur central pour que les protocoles étudiés aboutissent. Or, plus il y a de participants, plus il y a de risque de défaut de la part de l'un d'entre eux. Ainsi, l'étude que nous présentons est limitée de ce point de vue.

Dans les chapitres 4 et 5, nous étudierons les performances de l'IDS dans son format centralisé et fédéré respectivement. De cette manière, nous répondrons à l'objectif 3. L'entraînement et l'évaluation de l'IDS suivent les processus et utilisent les métriques décrites dans la section 3.3.

CHAPITRE 4 RÉSULTATS DE L'IDS CENTRALISÉ

Ce chapitre présente les résultats obtenus avec l'architecture de l'IDS présentée dans le chapitre 3 lorsqu'il est utilisé dans un cadre centralisé (objectif 3). Dans un premier temps, le lecteur trouvera une présentation des données utilisées ainsi que les prétraitements appliqués. Une fois les données abordées, nous analyserons les résultats que nous avons obtenu en utilisant le k-means centralisé dans l'IDS proposé. Pour mieux comprendre les phénomènes dans son format fédéré, nous étudions dans un premier temps le cas centralisé. De cette manière, nous fixons un repère pour les attentes des modèles fédérés.

4.1 Données utilisées

4.1.1 Type de données

L'étude menée au cours de ce mémoire porte sur des données réseau. Habituellement, les données réseau sont enregistrées au format pcap. De cette manière, on peut observer toutes les informations des communications. Dans le cas de ce mémoire, le modèle n'a pas accès aux données pcap pour éviter de l'inspection de paquets et pour augmenter la confidentialité. De plus, certaines communications pourraient être chiffrées. On ne peut donc pas avoir accès à toutes les informations d'une communication.

Ainsi, les données utilisées sont sous la forme de statistiques et caractéristiques des communications. Les informations peuvent être les ports de destination, les adresses IP et les statistiques peuvent par exemple être la durée totale de la communication ou le nombre de paquets envoyés.

4.1.2 Prétraitement des données

Avant d'entraîner les modèles de clustering sur les données présentées dans la section 4.1, nous réalisons un prétraitement. Cette étape est la même sur les deux jeux de données. Le prétraitement que nous avons établi se déroule comme suit :

1. Supprimer des entrées ayant une variable vide.
2. Remplacer les valeurs $-\infty$ (resp. $+\infty$) par le minimum (resp. le maximum) de la variable à travers le jeu de données. Il n'est pas précisé pourquoi ces valeurs apparaissent dans le jeu de données.
3. Remplacer les variables catégorielles par des variables binaires pour chaque valeur

possible.

4. Normaliser les variables pour qu'elles soient comprises dans l'intervalle $[0; 1]$.
5. Sélectionner les variables sur la base de la fréquence des valeurs [30]. On élimine par ce biais les variables dont les variations sont trop rares ou trop fréquentes. Réduire le nombre de variables est important pour k-means car cette méthode souffre de la malédiction de la dimension (*curse of dimensionality*).
6. Retirer les doublons.
7. Répartir aléatoirement les entrées dans un jeu de données d'entraînement à 80% et un jeu de données de test à 20%.

Hors mention contraire, la suite du mémoire se déroule en considérant le prétraitement des données comme étant effectué.

4.1.3 CIC-IDS2017

CIC-IDS2017 [41] est un jeu de données créé par l'Université du Nouveau-Brunswick. Il est constitué de données réseau au format pcap. Les auteurs fournissent aussi un format CSV pour l'apprentissage automatique dont les variables sont extraites des fichiers pcap. Nous n'utilisons donc pas les données pcap mais les variables qui en sont extraites pour réaliser de l'apprentissage automatique. Les concepteurs du jeu de données ont construit des réseaux informatiques dans lesquels des profils utilisateurs génèrent un trafic bénin. Ce type trafic a été réalisé avec pour objectif de générer un trafic normal le plus réaliste possible. En plus de cela, les auteurs ont mené des attaques contre leur infrastructure et ont enregistré les communications associées. Ainsi, ils ont pu étiqueter chacune des entrées du jeu de données avec l'un des types de communications produit durant l'expérience. Ce jeu de données utilise une variété de protocoles, d'usages, d'attaques qui le rapproche du réalisme.

Il y a au total 3 119 345 communications avant prétraitement et 2 829 385 après. Cela signifie qu'il y a 9% de doublons ou lignes contenant une variable vide. Le jeu de données a 70 variables explicatives avant prétraitement et 66 après. La répartition des données de CIC-IDS2017 selon le type de communication est présenté dans le tableau 4.1. Celui-ci montre la proportion approximée que représente chaque type de communication dans le jeu de données après prétraitement. Les types sont présentés par ordre alphabétique.

Type	Proportion (%)
Benign	80.32
Bot	0.07
DDoS	4.52
DoS GoldenEye	0.36
DoS Hulk	8.13
DoS Slowhttptest	0.19
DoS slowloris	0.21
FTP-Patator	0.28
Heartbleed	0.0004
Infiltration	0.001
PortScan	5.62
SSH-Patator	0.21
Web Attack - Brute Force	0.05
Web Attack - Sql Injection	0.0007
Web Attack - XSS	0.02

Tableau 4.1 Proportion des communications dans CIC-IDS2017 après notre prétraitement

On remarque dans le tableau 4.1 que les communications bénignes représentent une large majorité des communications du jeu de données ($\simeq 80\%$). Ceci ajoute au réalisme du jeu de données, car les communications normales sont censées être très majoritaires sur le réseau d'une organisation. Les attaques les plus représentées sont les différents types de *DoS* et le balayage de ports. Ceci est aussi en accord avec la réalité car ce sont des attaques qui nécessitent un grand nombre de requêtes. Toutefois, les attaques plus "subtiles" comme les injections SQL ou les *XSS* sont très peu représentées. De part leur mode opératoire, ces attaques engendrent nécessairement un trafic plus faible en volume que celles présentées avant dans ce paragraphe.

4.1.4 UNSW-NB15

Le jeu de données UNSW-NB15 [42–46] a été créé par l'Université de Nouvelle-Galles du Sud (University of New South Wales). Le jeu de données contient 43 variables explicatives avant prétraitement et 54 après. Il compte 257 673 entrées avant prétraitement et 162 735 après. Cela signifie qu'il y avait 36.84% de doublons ou lignes contenant une variable vide. On trouve dans le jeu de données 52.68% de communications normales. Ainsi, il est presque équilibré

contrairement à CIC-IDS2017. Le fait que le jeu de données soit relativement équilibré facilite l'apprentissage automatique mais nuit au réalisme. En effet, on sait que les communications normales sont censées être en large majorité sur un réseau informatique. Le tableau 4.2 présente le pourcentage que chaque type de communications occupe dans le jeu de données. Les types sont triés par ordre alphabétique, excepté pour le type normal qui occupe la première ligne.

Type	Proportion (%)
Normal / Bénin	52.68
Analysis	1.25
Backdoor	1.16
DoS	3.38
Exploits	16.85
Fuzzers	12.88
Generic	4.67
Reconnaissance	6.14
Shellcode	0.89
Worms	0.11

Tableau 4.2 Proportion des communications dans UNSW-NB15 après notre prétraitement

On remarque que les attaques par *Exploits* sont la première catégorie d'attaques du jeu de données. Ceci est étonnant quand on constate que des attaques comme les *DoS* ou la reconnaissance peuvent générer plus de trafic. Toutefois, les *Fuzzers* sont en deuxième place en termes de volume de communications parmi les attaques. Ceci est cohérent avec la façon dont ces attaques sont menées. Les communications normales sont en majorité absolue ce qui est réaliste. Toutefois, pour être encore plus réaliste, il faudrait que les communications normales soient encore plus prépondérantes. Comme pour CIC-IDS2017, les attaques dont la dangerosité se trouve dans les données de communication comme des *Backdoors* ou des *Shellcode* sont très minoritaires.

4.2 Résultats du k-means centralisé

Cette section présente les résultats de l'IDS proposé utilisant un k-means centralisé. Ainsi, nous utilisons les jeux de données CIC-IDS2017 et UNSW-NB15 dans leur entièreté et non pas en format fédéré dans cette section. Les résultats obtenus sont importants puisqu'ils serviront de base de comparaison pour l'IDS avec des modèles fédérés. En effet, comme énoncé dans

la section 2.4.2, il n'y pas de travail clairement identifiable à celui réalisé au cours de ce projet. Ainsi, pour commenter les performances des modèles fédérés, nous utilisons le k-means centralisé comme base.

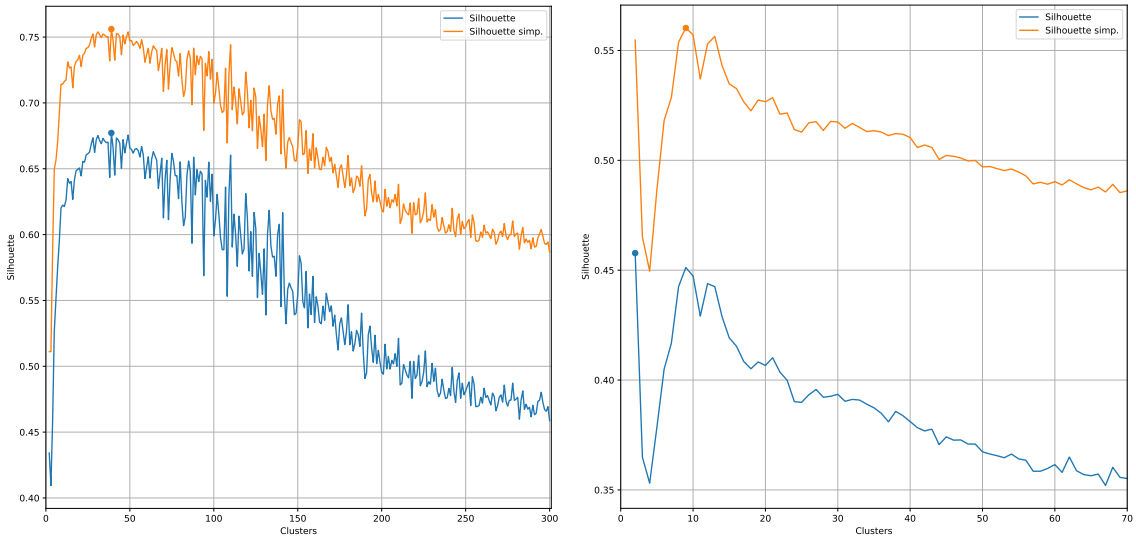
4.2.1 Évolution de la silhouette

La figure 4.1 représente l'évolution de la silhouette (courbe bleue) ainsi que de la silhouette simplifiée (courbe orange) en fonction du nombre de clusters k pour k-means. Les résultats sur CIC-IDS2017 sont la moyenne de 10 courbes tandis que ceux sur UNSW-NB15 viennent de 20 courbes. Les deux figures 4.1a et 4.1b ont des allures correspondant à ce qu'on peut attendre de courbes de silhouettes. En effet, globalement, les courbes sont croissantes jusqu'à un maximum, puis décroissent quand on s'éloigne de celui-ci. De plus, la silhouette et la silhouette simplifiée suivent les mêmes tendances.

Dans la sous-figure 4.1a, les maximums pour les deux métriques sont respectivement de 0.6772 et 0.7561 en $k = 39$. Rappelons toutefois que les valeurs prises par la silhouette ne sont pas les aspects les plus importants de l'étude. Ce qui importe est plutôt le nombre de clusters sélectionnés (k).

Les courbes de la sous-figure 4.1b suivent une tendance légèrement différente par rapport aux résultats sur CIC-IDS2017. Les deux courbes commencent avec un maximum local en $k = 2$ de 0.4578 et 0.5547. C'est d'ailleurs en $k = 2$ que la courbe moyenne de la silhouette atteint son maximum. Après $k = 2$ la silhouette décroît rapidement jusqu'à atteindre un minimum local en $k = 4$ de 0.3531 et 0.4496. Les deux courbes croissent ensuite jusqu'à $k = 9$ où la courbe moyenne de silhouette atteint un maximum local là où la silhouette simplifiée atteint le maximum global. Les valeurs en $k = 9$ sont 0.4512 et 0.5602.

Nous observons donc que CIC-IDS2017 compte 15 classes alors que la méthode indique $k = 39$ clusters. De même, il y a 10 classes dans UNSW-NB15 mais la méthode conseille soit $k = 2$ soit $k = 9$ clusters. Ainsi, on constate que le nombre de clusters optimal n'est pas nécessairement égal au nombre de classes du jeu de données. Ce résultat contredit alors l'intuition qu'on pourrait avoir. Plusieurs classes peuvent se partager un cluster ou bien à l'inverse une classe peut disposer de plusieurs clusters. Nous observerons ceci plus loin dans la section 4.2.3.



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

FIGURE 4.1 Evolution des silhouettes en fonction du nombre de clusters du k-means

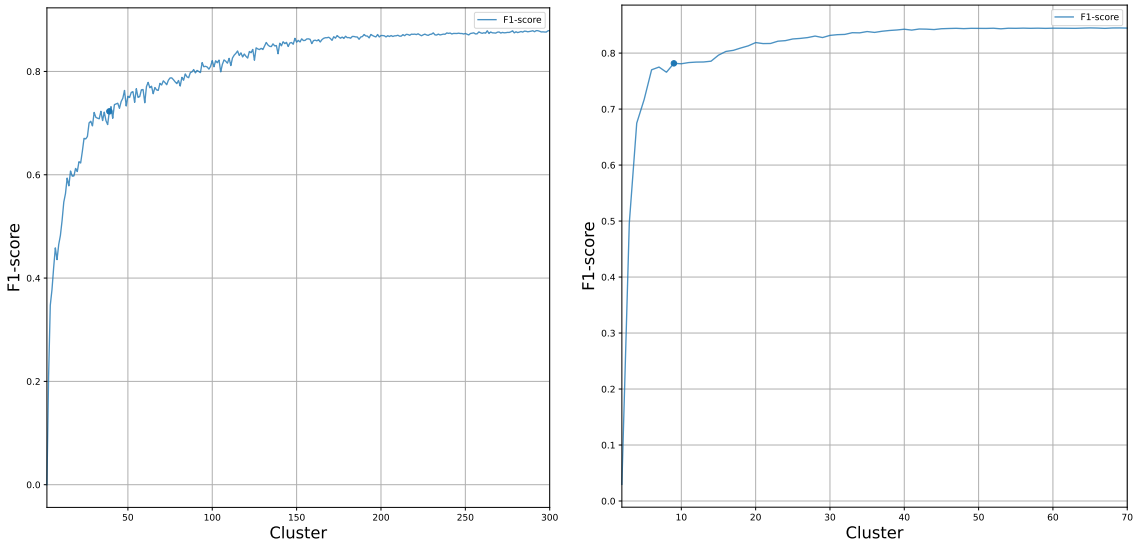
4.2.2 Classification binaire

Performances de l'IDS

L'évolution du score F_1 en fonction du nombre de clusters k est représentée par la figure 4.2. Les sous-figures indiquent les résultats selon le jeu de données utilisé. Les courbes de score F_1 des deux sous-figures suivent des tendances similaires. En effet, plus le nombre de clusters est important, plus le score F_1 est élevé. Cela signifie sans doute qu'avec plus de clusters, le modèle est capable de mieux distinguer les types de données entre eux. Pour expliquer ce comportement, on peut supposer que plus de clusters permet de mieux mailler l'ensemble des données et donc de les classer plus finement.

Dans le cas de CIC-IDS (figure 4.2a), on observe une croissance forte entre $k = 2$ et $k = 40$ environ. La croissance du score F_1 ralentit ensuite. La valeur de score F_1 atteinte lors du maximum de silhouette est de 0.7227. Après $k = 30$, le score F_1 croît en tendance. L'exactitude en $k = 39$ est de 0.8958.

Pour UNSW-NB15 (figure 4.2b), on note que la croissance ralentit vers $k = 6$ pour ensuite atteindre un plateau autour de la valeur 0.8361. La valeur du score F_1 atteinte en $k = 2$ est de 0.0297 et en $k = 9$ de 0.7817. L'exactitude obtenue dans chacun des cas est respectivement de 0.5594 et de 0.7659. On observe donc un écart important d'environ 20 points de pourcentage entre les deux valeurs d'exactitude selon que $k = 2$ est sélectionné ou $k = 9$.



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

FIGURE 4.2 Évolution du score F_1 en fonction du nombre de clusters du k-means

En observant les courbes de la figure 4.2, on comprend ainsi que la silhouette est un indicateur qui permet de fixer un seuil de clusters propre au jeu de données. Ainsi, si on souhaite augmenter les performances du modèle, on peut choisir d'augmenter le nombre de clusters par rapport à la référence qu'est l'optimum de silhouette. Toutefois, augmenter k implique qu'il faut déterminer la nature de plus de clusters. Or ceci implique plus de travail d'analyse des clusters pour déterminer leur composition. Finalement, on fait face à un compromis entre les efforts d'analyse à fournir et les performances obtenues.

Proportion de communications bénignes dans les clusters

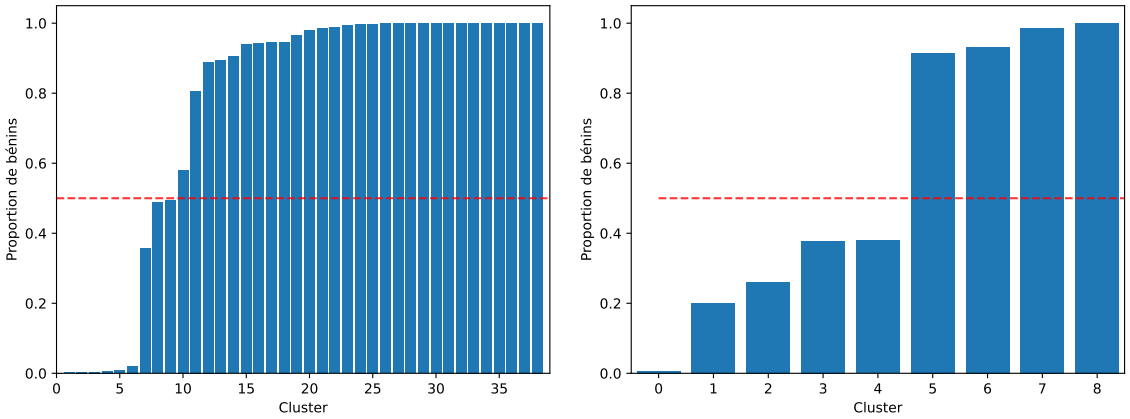
La suite de l'analyse des performances binaires est effectuée sur un k-means entraîné avec $k = 39$ clusters pour CIC-IDS2017 et $k = 9$ pour UNSW-NB15. En effet, ces valeurs étaient les maximums des courbes moyennes de silhouette simplifiée de la figure 4.1.

La figure 4.3 représente la proportion de communications bénignes dans chaque cluster. Les proportions sont triées par ordre croissant. La ligne rouge en tirets est située à la frontière de décision $\eta = 0.5$. Les numéros de clusters n'ont pas de signification particulière et ne correspondent pas par rapport aux numéros de la figure 4.4.

Pour CIC-IDS2017, dans la figure 4.3a, on constate tout d'abord une minorité de clusters d'attaque *i.e.* dont la proportion se situe sous la ligne rouge. Parmi les 9 clusters d'attaque, 7 sont situés à plus de 10 points de pourcentage de la proportion de bénins de 0.5. Ceci signifie

qu'un analyste ou un IDS par signature aurait peu de risques de se tromper en analysant un échantillon de ces clusters. Il en va de même pour les 28 clusters bénins à plus de 60% sur les 29 étant classés comme tels. On note toutefois que 4 clusters sont à moins de 10 points de la lignes des 50% de bénins. Cela signifie que le risque pour un analyste de mal classer ces clusters est plus important que pour les autres. Ils ne représentent toutefois qu'un faible nombre de clusters face aux 39 à analyser.

À l'inverse de ce qui est constaté dans le paragraphe précédent, il y a une minorité de clusters de communications normales pour UNSW-NB15 comme le montre la figure 4.3b. Toutefois, on observe que les clusters bénins et d'attaques sont tous distants d'au moins 10 points de pourcentage de la frontière de décision à 50%. Ainsi, les clusters dans ce jeu de données sont plutôt bien catégorisés. Toutefois, on note que les clusters normaux sont proches de 100% de communications bénignes alors que tous les clusters d'attaques sauf un sont compris entre 20% et 40%. Donc les clusters d'attaques intègrent en proportion plus de communications normales que les clusters bénins n'intègrent d'attaques. Ce qui signifie que cette classification relève plus de fausses alertes en proportion que de faux négatifs.



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

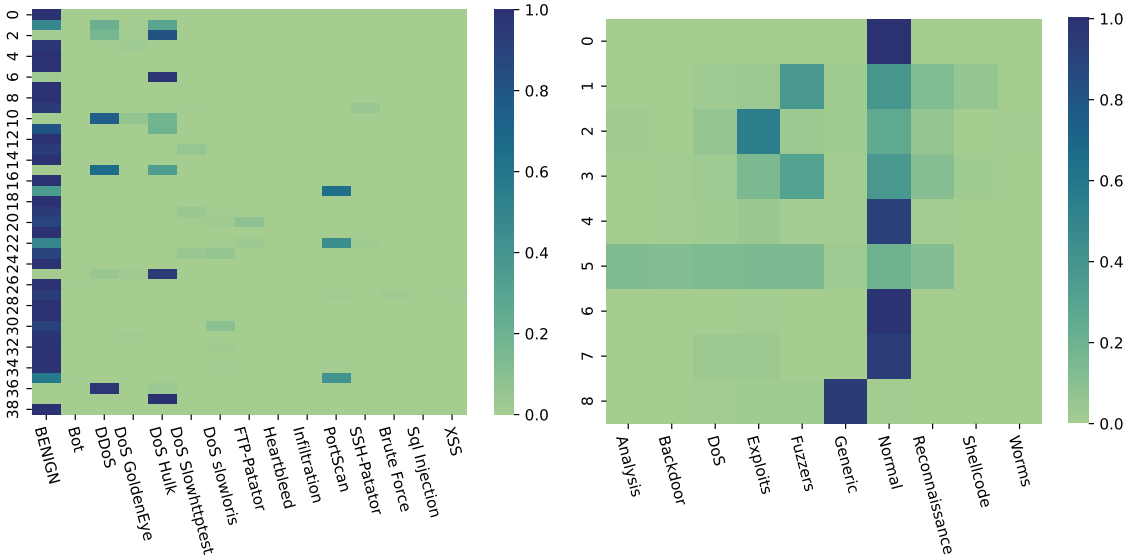
FIGURE 4.3 Proportion de communications bénignes dans chaque cluster

4.2.3 Classification multi-classes

Dans cette section, nous analysons la répartition des communications dans les clusters prédits par k-means avec le modèle maximisant la silhouette moyenne de la figure 4.1 *i.e.* $k = 39$ pour CIC-IDS2017 et $k = 9$ pour UNSW-NB15. Les résultats présentés pour la classification multi-classes se basent sur une réalisation du protocole.

La figure 4.4a représente sous forme de *heatmap* la proportion de chaque type de commu-

nication dans chacun des clusters pour CIC-IDS2017. Les lignes somment à 1. On constate directement que la majeure partie des clusters est essentiellement composée de communications bénignes. On note aussi une prédominance des attaques *DDoS*, *DoS Hulk* ou *PortScan* dans les clusters. Ceci est compréhensible de par le volume de communications qu'engendre ce genre d'attaques. Les balayages de ports ne sont pas présents dans des clusters où se trouvent des dénis de services ce qui semble montrer que le modèle peut distinguer ces familles d'attaques. On note de plus que les attaques comme *Heartbleed*, les *XSS* ou les injections SQL, sont pratiquement absentes en proportion dans les clusters où elles se trouvent. Ainsi, ces attaques ne sont pas isolées dans leurs propres clusters par le modèle.



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

FIGURE 4.4 Proportion de communications bénignes dans chaque cluster

La *heatmap* 4.4b représente la proportion de chaque type de communication dans chacun des clusters pour UNSW-NB15. On remarque que les communications normales sont majoritaires dans 4 clusters et ce à plus de 90% dans chaque cas. Ceci confirme les résultats de la figure 4.3b. On trouve aussi un cluster, le 8, qui est composé à près de 95% d'attaques *Generic*. Ce type d'attaques est pratiquement absent des autres clusters. Ceci indique une forte capacité du modèle à distinguer ce type de communication par rapport des autres. Toutes les classes ne sont pas aussi bien isolées. Par exemple, les communications normales sont en majorité relative dans le cluster 1. En effet, on trouve 38% de ces communications suivies par 37% de *Fuzzers*. De même, le cluster 5 semble quant à lui très équilibré en termes de communications. En effet, les entrées de ce cluster voient leur proportion osciller entre 12% et 20% environ.

Ainsi, ces clusters ne présentent pas de classe claire et cela montre une certaine incapacité du modèle à séparer les types de communications. Les attaques qui semblent se démarquer en termes de proportions sont les attaques *Generic*, *Fuzzers* et *Exploits*.

La faible capacité du modèle à isoler dans des clusters la plupart des classes autres que les communications bénignes couplée au fait que les classes sont plus ou moins volumineuses pose un problème. En effet, les classes volumineuses rendent invisibles par ces deux phénomènes les classes qui le sont moins. Pourtant, des attaques comme les *XSS* peuvent causer des dommages potentiellement plus importants qu'un DoS.

Classification à un tour

Répartition des clusters La classification à un tour fait référence au fait de classer le cluster comme étant du type de la communication en majorité (cf. section 3.2.3). Le tableau 4.3 indique le nombre de clusters prédit par type de communications sur le jeu de données CIC-IDS2017. Les communications n'ayant pas de clusters à leur nom sont ignorées. Par exemple, les injections SQL n'ont pas de cluster attitré et donc ne sont pas représentées dans ce tableau. On remarque que sur les 39 clusters, 31 sont classés comme bénins. En effet, dans 31 clusters, les communications de ce type sont au moins en majorité relative. On note aussi que les attaques prédites sont des méthodes nécessitant un grand volume de communications. Cela correspond aussi à l'analyse de la *heatmap* (figure 4.4) qui montrait que les communications fortement représentées appartiennent à ces classes.

Total	Bénin	*DDoS*	*DoS Hulk*	*PortScan*
39	31	3	4	1

Tableau 4.3 Nombre de clusters classés à un tour par type de communication (CIC-IDS2017)

Le tableau 4.4 indique le nombre de clusters de chaque type selon la classification multi-classes à un tour pour UNSW-NB15. Sur les 9 clusters, on en trouve 7 de type normal *i.e.* les communications normales sont en majorité relative dans 7 des clusters. On remarque que les seules attaques donnant leur nom à un cluster sont les attaques les plus représentées dans le jeu de données (tableau 4.2) pour le *DoS* ou clairement séparable des autres pour *Generic*. Ce résultat peut correspondre pour les *DoS* à ce qui a été observé dans le cas du jeu de données CIC-IDS2017 comme mentionné dans le paragraphe ci-dessus.

Total	Normal	*Exploits*	*Generic*
9	7	1	1

Tableau 4.4 Nombre de clusters classés à un tour par type de communication (UNSW-NB15)

Discussion des performances Avec la classification multi-classes à un tour sur CIC-IDS2017, on obtient un score d'exactitude de 0.8784 soit une baisse de plus de 0.01 par rapport à la classification binaire. Pour comprendre cette baisse, on peut remarquer que la classification multi-classes à un tour étiquette un cluster selon la classe en majorité relative. Ainsi, en prenant l'exemple du cluster 10 de la *heatmap* 4.4a, on remarque que les *DDoS* sont en majorité relative. Or, les *DDoS* représentent 74.76% d'un cluster composé à 99.25% d'attaques. Ainsi, en classant le cluster comme étant du *DDoS* on exclut les *DoS Hulk* qui représentent 18.30% du cluster et les *PortScan* à 0.0009%.

Dans le cas d'UNSW-NB15, on retrouve le même comportement que sur CIC-IDS2017. En effet, avec la classification multi-classes à un tour, on obtient une exactitude de 0.6199. Cette valeur est distante de près de 0.14 de l'exactitude moyenne dans le cas binaire. Cette baisse de performance s'explique en observant les proportions de bénins dans les clusters où ils sont en majorité relative. Comme mentionné, le cluster 5 de la *heatmap* 4.4b est très équilibré en termes de classes qui le composent. Ainsi, en le classant comme normal parce que ces communications sont en majorité relative avec 19.82%, on classe mal près de 80% du cluster. Un raisonnement similaire s'applique au cluster 3 de la *heatmap* 4.4b où les communications normales sont en majorité relative avec 37.63% du cluster, excluant plus de 60% des entrées.

On comprend ainsi que les clusters où aucun type de communication n'est en majorité absolue sont propices à un mauvais classement de la majorité des communications, réduisant par là même les performances de classification.

Classification à deux tours

Répartition des clusters La classification à deux tours fait référence au fait de classer le cluster comme étant normal si cette classe est en majorité absolue ou selon une des attaques sinon (cf. section 3.2.3). Le résultat de la classification multi-classes à deux tours pour CIC-IDS2017 est présenté dans le tableau 4.5. On remarque que le nombre de clusters bénins est réduit à 29 comparé aux 31 du tableau 4.3. Ceci correspond aux 29 clusters dans lesquels les communications bénignes sont en majorité absolue comme mentionné dans les commentaires de la *heatmap* (figure 4.4). On retrouve dans le tableau 4.5 une augmentation du nombre de

clusters *Portscans* de 1 à 2 et de *DoS Hulk* de 4 à 5. On constate toutefois un maintien du nombre de clusters de *DDoS* à 3. Les classes des clusters sont là encore les communications les plus représentées dans le jeu de données.

Total	Bénin	*DDoS*	*DoS Hulk*	*PortScan*
39	29	3	5	2

TABLEAU 4.5 Nombre de clusters classés à deux tours par type de communication (CIC-IDS2017)

Le nombre de clusters classés par la classification multi-classes dans UNSW-NB15 selon le type de communication est indiqué par le tableau 4.6. Parmi les 9 clusters, 4 sont bénins. Ceci correspond aux 4 clusters où les communications bénignes sont en majorité absolue comme décrit précédemment. Nous observons alors une baisse par rapport aux 7 clusters normaux de la classification à un tour. Les 3 attaques ayant des clusters classés selon leur type sont les *Exploits*, *Fuzzers* et les *Generic*. On retrouve par ailleurs les deux attaques les plus présentes dans le jeu de données (tableau 4.2) comme précédemment. Les attaques *Generic* sont cependant en plus faible effectif que les *Reconnaissance* mais ont tout de même leur propre cluster. Ceci est justifié par le fait que les attaques *Generic* ont été très bien regroupées dans un cluster, le 8 de la *heatmap* 4.4b, là où les attaques *Reconnaissance* sont plus dispersées. On constate que la classification à deux tours remplit son objectif de plus représenter les attaques.

Total	Normal	*Exploits*	*Fuzzers*	*Generic*
9	4	2	2	1

TABLEAU 4.6 Nombre de clusters classés à deux tours par type de communication (UNSW-NB15)

Discussion des performances La classification multi-classes à deux tours prédit les bonnes classes avec une exactitude de 0.8601 sur le jeu de données CIC-IDS2017. Ceci représente une baisse de 0.0183 en comparaison avec la classification multi-classes à un tour. La baisse de performances par rapport à la classification binaire (section 4.2.2) s'explique en partie par le phénomène pointé dans le paragraphe sur la classification multi-classes à un tour en section 4.2.3. De plus, en observant le cluster 1 de la figure 4.4a, on remarque que les communications bénignes représentent 48.79% du cluster. Ainsi, le trafic normal est en minorité par rapport aux trafics d'attaque. Donc, la classification à deux tours classe le

cluster comme étant du type de l'attaque la plus volumineuse du cluster. Dans ce cas, cette attaque est de type DoS Hulk avec ses 29.28% de communications du cluster. Donc, une classe représentant moins d'un tiers du cluster donne son nom au cluster. On classe de façon incorrecte près de 70% des entrées. Ainsi la classification à deux tours cause une réduction de performances.

Une fois de plus, les résultats sur UNSW-NB15 suivent les mêmes tendances que pour CIC-IDS2017. De fait, avec la classification multi-classes à deux tours, le k-means obtient un score d'exactitude de 0.6021. Ce résultat représente une perte de 0.02 par rapport à la classification multi-classes à un tour et près de 0.16 par rapport à la classification binaire. On observe la baisse de performance dans les clusters où la classification a proposé un type n'étant pas en majorité relative. Par exemple, dans le cluster 5 de la *heatmap* 4.4b, la classe en majorité relative est le type de communication normale avec 19.82%. Or, ce cluster est classé comme *Exploits* par la classification multi-classes à deux tours. En effet, ce type est en majorité relative dans le cluster parmi les attaques avec 13.88% du total. Ainsi, en choisissant cette communication, on classe mal 0.06 de plus du cluster qu'en choisissant le type de communication normal.

Dans cette section, nous avons constaté que la classification à deux tours augmente bien le nombre de clusters d'attaque par rapport à la classification à un tour. Toutefois, elle réduit les performances en choisissant, lorsque le cas se présente, une classe minoritaire dans les clusters. De plus, les attaques en large infériotié dans le jeu de données n'ont souvent pas de clusters attitrés. Cela pointe une faible capacité du modèle à discriminer les classes entre elles. Ceci semble montrer certaines limites du clustering sans inspection de paquets (*payload*).

4.3 Discussion sur l'IDS centralisé

Dans le chapitre 4, nous avons étudié l'IDS proposé dans le chapitre 3 dans un cadre centralisé. Cette étude est importante pour jeter des bases de comparaison avec l'IDS dans un format fédéré et répond à l'objectif de recherche 3 (section 1.3).

Au cours de la section 4.2.1, nous avons observé l'évolution de la silhouette en fonction du nombre de clusters. Les silhouettes classiques et simplifiées ont des évolutions similaires. Cela implique que nous pouvons utiliser la silhouette fédérée simplifiée pour l'étude de l'IDS fédéré dans le chapitre 5.

En étudiant les performances en classification binaire de l'IDS dans la section 4.2.2, nous avons observé que plus on utilise de clusters, meilleures sont les performances. La silhouette sert alors de point de repère pour le nombre de clusters et on pourrait choisir une valeur supé-

rieure. En effet, les performances en seraient augmentées. Toutefois, imposer plus de clusters nécessite plus de travail d'analyse de la composition des clusters. Ainsi, nous observons un compromis entre l'énergie à investir dans la classification des clusters et les performances voulues. Dans un cas d'usage réel, il faut donc que les parties prenantes s'accordent sur le nombre de clusters à choisir, si celui proposé par la silhouette ne leur convient pas.

L'analyse de la composition des clusters dans une classification binaire (cf. section 4.2.2) a montré que la discussion de la section 3.2.4 est valide. Ainsi, la méthode de classification choisie est pertinente pour notre IDS. Toutefois, nous avons constaté lors de l'analyse de la classification multi-classes, que certains types de communications étaient en large infériorité dans leurs clusters. Ainsi, on peut penser que si un analyste inspecte un tel cluster, il ne trouverait probablement pas ces attaques en large infériorité. Cela ne change toutefois pas les résultats de la classification en elle-même puisqu'elle se base sur les classes majoritaires.

Nous avons aussi constaté que le nombre de clusters optimal pour la silhouette n'est pas le nombre de classes présentes dans le jeu de données. De fait, nous avons observé dans la section 4.2.3 que le k-means ne réussit pas à isoler les types de communications dans des clusters qui leurs sont propres. On comprend alors que le critère de proximité géométrique utilisé par k-means n'est pas suffisant pour isoler les classes dans des clusters distincts. Ainsi, certaines attaques se retrouvent dans les mêmes clusters et cela les rend donc indistinguables pour l'IDS proposé. Une piste d'amélioration serait d'essayer de nouveaux modèles ou d'intégrer de l'inspection de paquet. Cette dernière proposition viendrait toutefois empiéter sur la confidentialité des données.

Pendant l'étude des résultats multi-classes (section 4.2.3), nous avons constaté que la classification binaire avait de meilleures performances que les classifications multi-classes. Pour obtenir les performances du cadre binaire et les informations disponibles dans la classification multi-classes, on peut combiner les deux façons de faire. La classification se fait d'abord avec les classes normale et attaque. Si une attaque est détectée, l'IDS accompagne l'alerte de la composition du cluster. De cette manière, l'analyste est guidé dans son enquête.

CHAPITRE 5 RÉSULTATS DE L'IDS FÉDÉRÉ

Le chapitre 5 présente les résultats obtenus avec l'IDS introduit dans le chapitre 3 dans son format fédéré (objectif 3). Tout d'abord, nous présenterons comment nous avons généré des jeux de données fédérés et nous les analyserons. Ensuite, nous réaliserons une étude de la silhouette des modèles de clustering fédéré pour sélectionner les hyperparamètres. Une fois les hyperparamètres choisis, nous pouvons étudier les performances de classification. Tout au long du chapitre, nous établirons des comparaisons avec l'IDS centralisé étudié dans le chapitre 4.

5.1 Jeux de données fédérés

Dans la section 5.1, nous montrons comment nous avons créé des jeux de données fédérés et nous les analysons. Les jeux de données fédérés sont générés à partir des données d'entraînement prétraitées (cf. section 4.1).

5.1.1 Construction d'un jeu de données i.i.d.

Le cas le plus simple pour un algorithme d'apprentissage fédéré est celui dans lequel les données des clients sont générés de façon i.i.d.. Pour cela, il faut donc que les données soient réparties entre les clients de sorte que tous aient des données dans tous les clusters. Pour atteindre cet objectif, nous répartissons les données équitablement entre N clients et en les mélangeant uniformément. De cette manière, chaque client est statistiquement une version réduite du jeu de données initial et ressemble aux autres.

Analyse du jeu de données créé avec CIC-IDS2017

Dans le cadre i.i.d. pour CIC-IDS2017, chaque client a un jeu de données de taille égale aux autres. Pour l'expérience, nous utilisons 100 clients. Chaque client dispose de 22 635 entrées. Le tableau 5.1 présente le nombre de clients ayant observé chaque type de communications lors de nos expériences. On constate que tous les clients ont fait face à pratiquement tous les types. Toutefois, les attaques *Heartbleed*, *Infiltration*, et les injections SQL ne sont pas présentes parmi tous les clients. Même si le jeu de données a été mélangé puis découpé en parts égales, les attaques mentionnées sont peu volumineuses. En effet, dans le jeu de données d'entraînement en question, on compte 9 communications *Heartbleed*, 11 injections SQL et 28 *Infiltrations*. Ces valeurs correspondent au nombre de clients les ayant observées.

Ainsi, les résultats obtenus dans le tableau 5.1 attestent que nous avons bel et bien réparti uniformément les données chez les clients.

Type	Clients
Benign	100
Bot	100
DDoS	100
DoS GoldenEye	100
DoS Hulk	100
DoS Slowhttptest	100
DoS slowloris	100
FTP-Patator	100
Heartbleed	9
Infiltration	25
PortScan	100
SSH-Patator	100
Web Attack - Brute Force	100
Web Attack - Sql Injection	11
Web Attack - XSS	99

Tableau 5.1 Nombre de clients ayant observé chaque type de communication dans CIC-IDS2017 (i.i.d.)

Analyse du jeu de données créé avec UNSW-NB15

Les jeux de données des clients dans le cadre i.i.d. avec UNSW-NB15 sont de taille 1 301. Nous utilisons là aussi 100 clients. Le tableau 5.2 présente le nombre de clients ayant observé chacun des types de communication lors de nos expériences. Ainsi, tous les clients ont observé pratiquement tous les types de communication. Toutefois, les attaques de type *Worms* ne sont pas présentes chez tous les participants. Ceci est dû au même phénomène expliqué dans la section 5.1.1. En effet, cette attaque présente un volume inférieur au nombre de clients requis. Ainsi, la répartition des attaques est uniforme chez tous les participants.

Type	Clients
Normal	100
Analysis	100
Backdoor	100
DoS	100
Exploits	100
Fuzzers	100
Generic	100
Reconnaissance	100
Shellcode	100
Worms	79

Tableau 5.2 Proportion des communications dans UNSW-NB15 (i.i.d.)

5.1.2 Construction d'un jeu de données non i.i.d.

Pour construire un jeu de données non i.i.d., nous trions les jeux de données selon une variable qui crée un ensemble cohérent de communications. Par exemple, la variable *Destination IP* de CIC-IDS2017 indique l'adresse IP de destination de la communication. En triant le jeu de données suivant cette variable puis en créant des sous-jeux de données ne possédant qu'une adresse IP de destination, on crée un scénario proche du réel. En effet, en créant des sous-jeux de données n'ayant qu'une IP de destination, on se place du point de vue du receveur qui cherche à savoir si les communications entrantes sont normales ou malveillantes. Toutefois, la cohérence des sous-ensembles de communications créés ne signifie pas une homogénéité statistique des données. De fait, chaque adresse IP dispose de ses propres services, appareils et usages.

La suite de la section 5.1.2 consiste en la présentation et l'analyse des jeux de données que nous avons généré avec la méthode présentée ci-dessus.

Analyse de CIC-IDS2017 trié par adresses IP de destination

Pour CIC-IDS, nous choisissons de trier les clients suivant la variable *Destination IP*. De cette manière, chaque client représente une adresse IP qui reçoit les communications. Cela représente donc un cas réaliste pour un IDS.

Dans CIC-IDS2017, il existe 19 112 adresses IP uniques dans la variable *Destination IP*. Ceci représente un grand nombre de clients et ceux-ci ne sont pas toujours pertinents. Pour le

comprendre, observons la figure 5.1. Celle-ci représente le nombre de clients, en ordonnées, en fonction du nombre de communications chez ce client, en abscisses. Il faut voir cette courbe comme un histogramme. Toutefois, un histogramme n'aurait pas été lisible d'où le format choisi. On voit donc sur la figure 5.1 que le nombre de clients ayant peu de communications est très important. En effet, sur les 19 112 adresses IP distinctes, seules 1 281 ont au moins 100 communications. Les trois clients ayant le plus de communications en ont 685 450, 582 956 et 294 077. Ils représentent à eux seuls environ 55.22% du nombre de communications totales du jeu de données. Ainsi, les tailles de jeux de données sont très déséquilibrées.

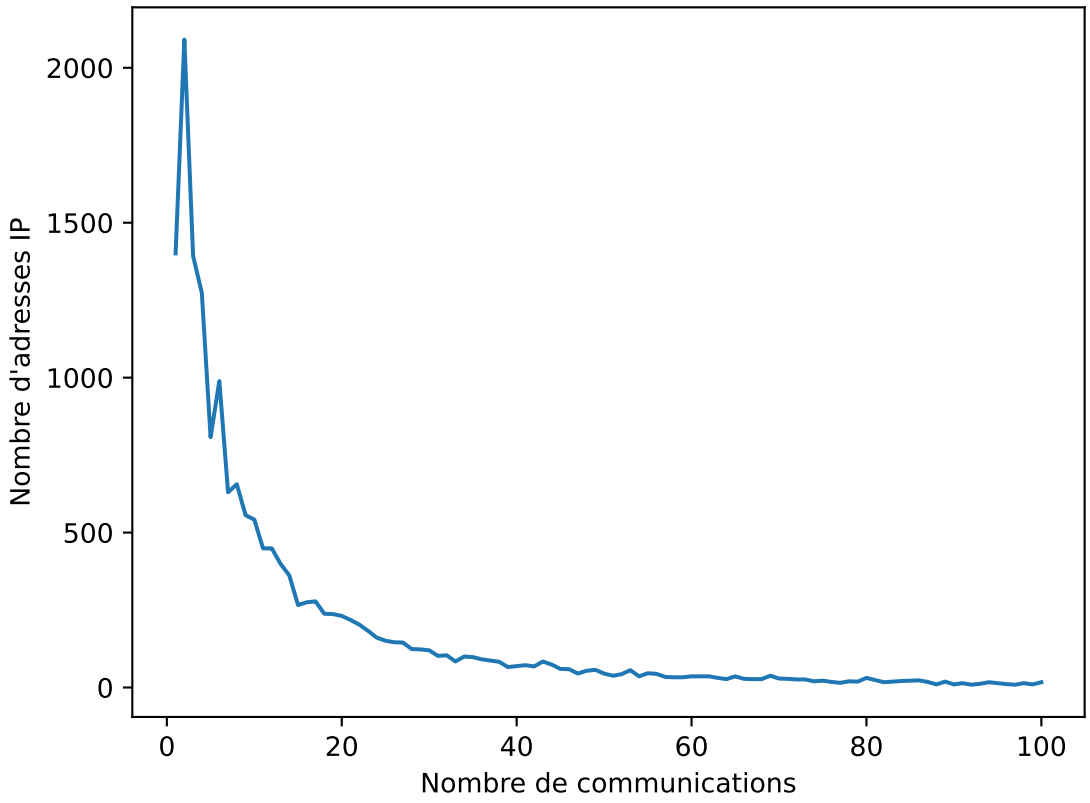


FIGURE 5.1 Nombre de clients en fonction du nombre de communications par adresse IP (CIC-IDS2017)

Comme présenté dans le paragraphe précédent, les jeux de données que nous créons sont très déséquilibrés lorsqu'on les trie par IP de destination. Or, pour faire du clustering, il faut au moins autant de données que le nombre de clusters requis par la méthode. Dans les expériences menées, le nombre maximal étant $k = 300$ et comme nous souhaitons garder une marge, nous conservons les clients ayant plus de 500 communications. Cela représente 212 clients.

Le tableau 5.3 présente le nombre de clients ayant observé chacun des types de communications du jeu de données la séparation par *Destination IP* réalisée. On constate que tous les clients opèrent du trafic bénin. L'attaque la plus vue par des clients différents est l'attaque par *Bot* suivie du *DDoS*. Toutefois, les autres attaques n'ont été observées que par un client chacune. De plus, le client d'adresse IP 192.168.10.50 a subi tous les types de *DoS*, les attaques *Patator*, les *Web Attacks* et le *Port Scan*. C'est donc le client ayant observé le plus d'attaques différentes. Il n'a toutefois pas subi toutes les attaques. Dans l'infrastructure ayant généré le jeu de données, cette adresse IP était reliée à un serveur Ubuntu hébergeant des services et ouvert au public. Ainsi, ce serveur Ubuntu concentre un grand nombre d'attaques mais d'autres en ont aussi subi. Cela montre bien l'hétérogénéité de la répartition des données chez les clients.

Type	Clients
BENIGN	212
Bot	6
DDoS	2
DoS GoldenEye	1
DoS Hulk	1
DoS Slowhttptest	1
DoS slowloris	1
FTP-Patator	1
Heartbleed	1
Infiltration	1
PortScan	1
SSH-Patator	1
Web Attack - Brute Force	1
Web Attack - Sql Injection	1
Web Attack - XSS	1

Tableau 5.3 Nombre de clients ayant observé chaque type de communication dans CIC-IDS2017 (non i.i.d.)

Même si un client rassemble la majeure partie des attaques, certains clients n'en ont jamais observées ou d'autres sont les seuls à les avoir subies. Ainsi, le jeu de données que nous générons est hautement non i.i.d. et nous pourrons constater si le protocole permet la coopération.

Analyse de UNSW-NB15 trié par classes

Pour créer un jeu de données non i.i.d. avec UNSW-NB15, nous avons d'abord essayé de trier les communications suivant la variable *proto* (protocole). Ainsi, chaque client de l'IDS proposé n'utilise qu'un seul protocole. Or dans ce cas de figure, quelques clients observaient pratiquement toutes les catégories de communications. Ceci risque de gommer l'aspect non i.i.d. du jeu de données fédéré. Même en triant suivant plusieurs variables, nous obtenions un résultat similaire. Pour UNSW-NB15, il a donc été décidé que chaque client correspond à un type de communication. Ainsi, la proportion que représente chaque client dans le jeu de données fédéré est présentée en tableau 4.2. Même si cette répartition des données chez les clients n'est pas réaliste comme dans le cas de CIC-IDS2017, cela permet de tester la robustesse des différents modèles face à un autre type de distribution non i.i.d..

Avec cette méthode, tous les clients disposent de suffisamment de données pour réaliser les expérimentations. Nous n'excluons donc pas de clients contrairement au cas de CIC-IDS2017.

5.2 Analyse de la silhouette des modèles fédérés

Au cours de cette section, nous analyserons l'évolution de la silhouette du k-means fédéré avec initialisation k-means++ fédérée (section 3.1.1) en fonction du nombre de clusters k. L'étude des modèles fédérés des sections 2.1.5 et 3.1.2 est réalisée dans l'annexe A. En plus du paramètre k, nous tiendrons compte du nombre de *rounds* r de communication des protocoles. Une fois ces résultats obtenus, nous pourrons ainsi déterminer les paramètres k et r optimaux pour chacun des modèles selon les situations. Ces paramètres seront ensuite utilisés pour comparer les performances des modèles dans la section 5.3. Le choix des hyperparamètres est effectué en tenant compte le moins possible des performances des modèles par souci de réalisme. En effet, dans un cas réel où il n'y a pas d'étiquettes disponibles pour les communications, la sélection de modèles se baserait seulement sur la silhouette.

5.2.1 K-means fédéré avec initialisation k-means++ fédérée

Résultats

La figure 5.2 présente l'évolution de la silhouette en fonction de k sur les deux jeux de données étudiés et dans les configurations de clients i.i.d. et non i.i.d..

CIC-IDS2017 La silhouette calculée sur la répartition i.i.d. des données de CIC-IDS2017 parmi les clients donne les résultats présentés sur la figure 5.2a. On note des ressemblances

avec les courbes pour le k-means fédéré de Garst et al. (figure A.1a). En effet, les courbes de silhouette fédérée pour $r = 5$ et $r = 10$ se ressemblent sur les deux figures. De fait, les courbes croissent, atteignent un maximum et décroissent légèrement en tendance. Toutefois, la courbe pour $r = 0$ est différente et représente le cas où seule l'initialisation k-means++ fédérée proposée est utilisée. On observe que la courbe de silhouette décroît de façon importante après son maximum. Ce comportement est plus conforme à celui observé en figure 4.1a que pour les $r > 0$. Le nombre de clusters optimal est de $k = 51$, $k = 63$ et $k = 63$ pour $r = 0$, $r = 5$ et $r = 10$ respectivement. Ces valeurs sont supérieures aux 39 clusters optimaux du k-means centralisé.

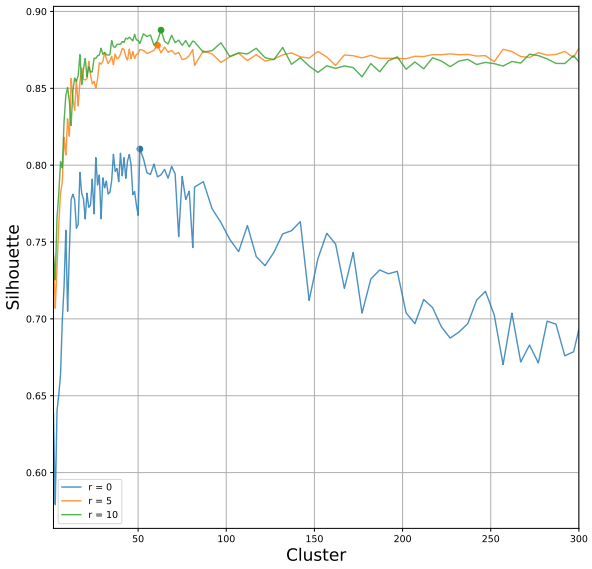
L'allure de la courbe pour $r = 0$ avec les données non i.i.d. de CIC-IDS2017 ressemble fortement à celle pour $r = 0$ aussi sur les données i.i.d.. Cela est cohérent avec le fait que le k-means++ fédéré proposé est équivalent à celui centralisé, qu'importe la répartition i.i.d. ou non. Toutefois, les courbes pour $r > 0$ n'ont pas l'allure escomptée. Il semble que les itérations du protocole sur les données non i.i.d. dégradent l'allure de la courbe. Même si elles sont supérieures à celle pour $r = 0$, les courbes $r > 0$ n'ont pas de maximum clair et lisible. Ce comportement est similaire à celui observé pour $r > 0$ pour le k-means fédéré sur les données de CIC-IDS2017 en répartition non i.i.d. (fig. A.1c).

UNSW-NB15 Pour UNSW-NB15 avec des clients i.i.d. (figure 5.2b), on remarque aussi des similarités avec la figure A.1b. En effet, pour $r = 5$ et $r = 10$, la forme des courbes présente des similarités avec les résultats présentés dans la section A *i.e.* un maximum local en $k = 2$ et un maximum local vers $k = 20$. Toutefois, on note que la courbe de silhouette avec $r = 0$ est "plate" comparée aux courbes avec $r > 0$. Ceci empêche de définir clairement un maximum. Nous aborderons à nouveau ce point dans la sous-section suivante.

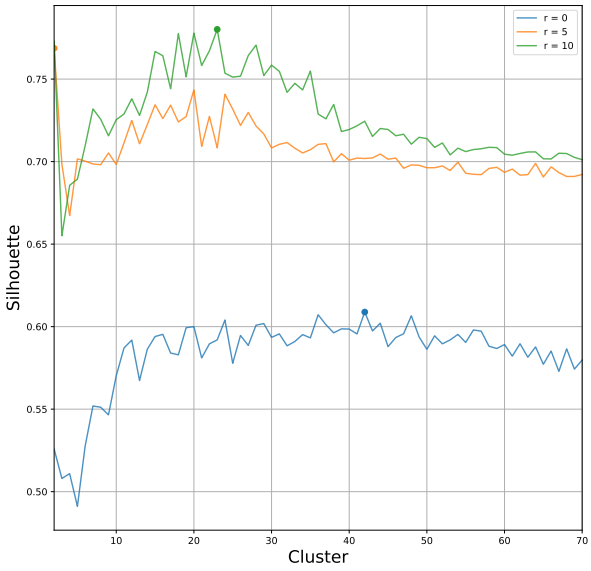
On observe un comportement similaire de la silhouette pour le cas non i.i.d. sur UNSW-NB15 dans la figure 5.2d. De fait, la courbe pour $r = 0$ croît entre $k = 2$ et $k = 10$ semble stagner entre $k = 10$ et $k = 55$ puis semble décroître jusqu'à la borne de $k = 70$ du graphe. *A contrario*, les courbes $r = 5$ et $r = 10$ présentent un maximum local en $k = 2$ puis un maximum local autour de $k = 30$.

Comparaison avec le k-means++ centralisé

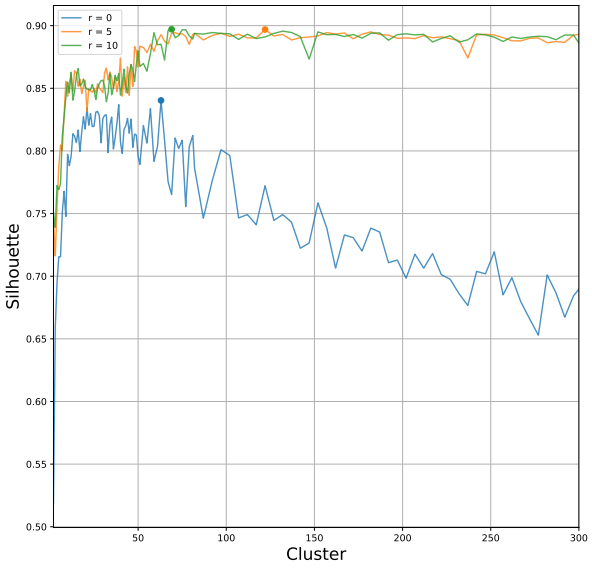
La figure 5.3 compare la silhouette simplifiée (pour le cas centralisé) et la silhouette simplifiée fédérée (pour le cas fédéré) selon le jeu de données. Nous comparons bien des grandeurs équivalentes comme présenté dans la section 3.1.3.



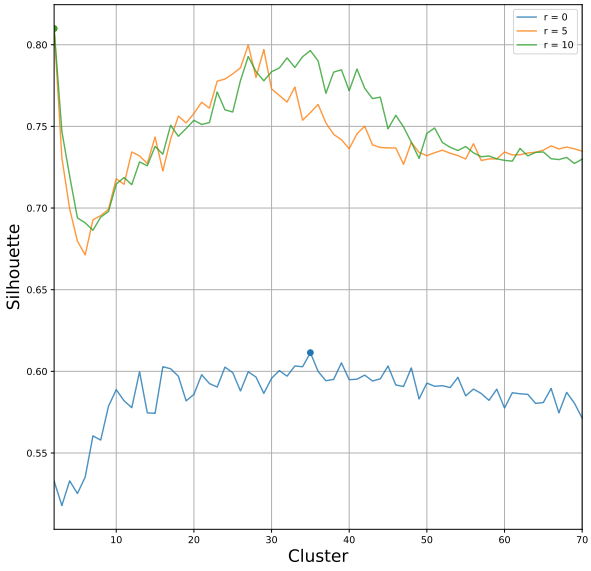
(a) CIC-IDS2017 i.i.d.



(b) UNSW-NB15 i.i.d.

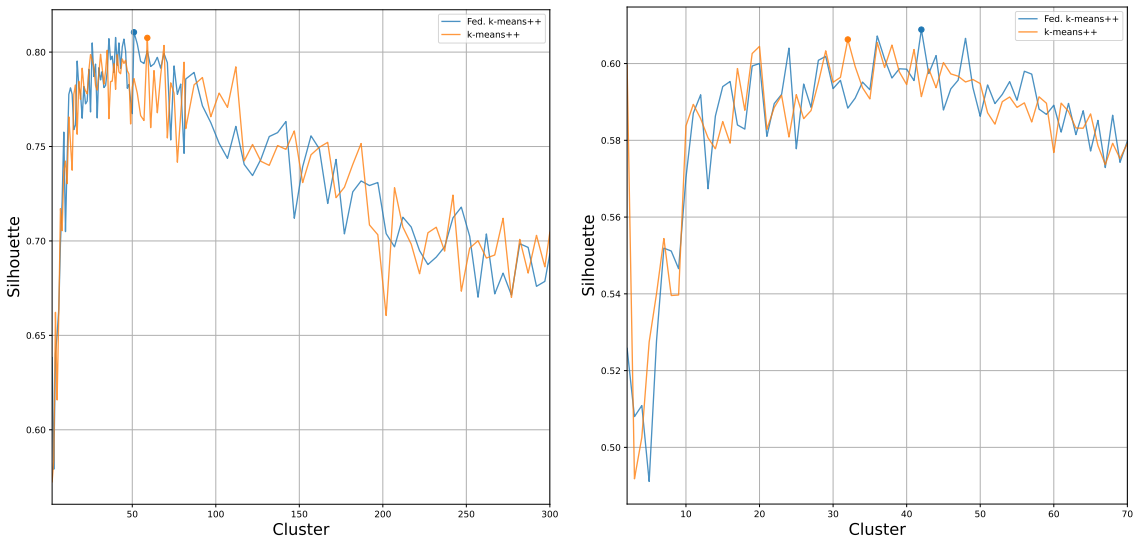


(c) CIC-IDS2017 non i.i.d.



(d) UNSW-NB15 non i.i.d.

FIGURE 5.2 Évolution de la silhouette en fonction du nombre de clusters sur le serveur (k-means fédéré et initialisation k-means++ fédérée)



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

FIGURE 5.3 Comparaison des silhouettes en fonction du nombre de clusters pour k-means++ et k-means++ fédéré

La figure 5.3a présente l'évolution de la silhouette sur CIC-IDS2017. On remarque que les courbes pour le k-means++ centralisé et celui fédéré évoluent dans le voisinage l'une de l'autre. Similairement, la figure 5.3b représente l'évolution de la silhouette des deux modèles sur UNSW-NB15. On a remarqué dans les figures 5.2b et 5.2d que la courbe de silhouette fédérée pour le k-means++ fédéré est "plate". Lorsqu'on compare cette courbe à celle du k-means++ centralisé, on remarque qu'elles sont pratiquement superposées. De plus, on observe mieux avec l'échelle des ordonnées de la figure 5.3b qu'il y a une tendance haussière d'abord puis une décroissance à partir de $k = 40$ environ. Le fait que les deux courbes ne soient pas égales vient de la stochasticité des deux méthodes. Les résultats de la figure 5.3 sont donc conformes à ce qui a été exprimé en section 3.1.1.

Hyperparamètres sélectionnés

Le tableau 5.4 rassemble les hyperparamètres sélectionnés pour le k-means fédéré avec initialisation k-means++ fédérée. Les valeurs sont déclinées selon les jeux de données et la distribution de celles-ci. Pour CIC-IDS2017, on choisit $r = 0$ car la forme de la courbe de silhouette permet plus facilement de choisir un maximum. La valeur de k est déduite de la courbe de silhouette moyenne. Pour UNSW-NB15, $r = 5$ a été choisi pour les mêmes raisons et par souci de minimalité de r. En effet, la courbe de silhouette fédérée pour $r = 10$ a une forme acceptable mais pour $r = 5$ aussi. Ainsi, autant choisir r minimal. Toutefois, k n'est

pas le maximum de la courbe $r = 5$ pour les deux distributions de données. En effet, le maximum est en $k = 2$. Or, on sait que plus k est petit, moins les performances sont bonnes. Ainsi, nous choisissons le second maximum.

Jeu de données	Distribution des données	k	r
CIC-IDS2017	i.i.d.	51	0
CIC-IDS2017	non i.i.d.	63	0
UNSW-NB15	i.i.d.	20	5
UNSW-NB15	non i.i.d.	27	5

Tableau 5.4 Bilan des hyperparamètres (k-means fédéré avec initialisation k-means++ fédérée)

5.2.2 Résultats des autres modèles

Le tableau 5.5 compile les hyperparamètres sélectionnés dans l'annexe A (tableaux A.1 et A.2). On remarque que dans tous les cas de figures, le paramètre $r = 0$ a été choisi et donc que le nombre minimal de *rounds* de communications est suffisant. De plus, pour tous les cas de figures, le nombre de clusters k est supérieur a ce qui a été déterminé pour le cas centralisé. En effet, les valeurs de k pour CIC-IDS2017 sont supérieures à 39 et sont supérieures à 9 pour UNSW-NB15. Nous constatons aussi dans les tableaux 5.4 et 5.5 que le nombre de clusters dans le cas i.i.d. est systématiquement inférieur au cas non i.i.d.. Cela confirme que le cas idéal mathématiquement (i.i.d.) est plus simple pour les algorithmes de clustering que le cas plus réaliste (non i.i.d.).

Modèle	Jeu de données	Distribution des données	k	r
K-means fédéré	CIC-IDS2017	i.i.d.	49	0
K-means fédéré	CIC-IDS2017	non i.i.d.	83	0
K-means fédéré	UNSW-NB15	i.i.d.	13	0
K-means fédéré	UNSW-NB15	non i.i.d.	15	0
Meta k-means	CIC-IDS2017	i.i.d.	73	0
Meta k-means	CIC-IDS2017	non i.i.d.	295	0
Meta k-means	UNSW-NB15	i.i.d.	13	0
Meta k-means	UNSW-NB15	non i.i.d.	24	0

Tableau 5.5 Bilan des hyperparamètres de l'annexe A

5.2.3 Discussion

Le k-means fédéré avec ou sans l'initialisation k-means++ fédérée a tendance à sélectionner un nombre de clusters plus grand que le modèle centralisé. Toutefois, la valeur optimale de k est aléatoire de par la variance de la méthode. Les valeurs optimales obtenues varient donc dans une zone de maximum et peuvent correspondre aux valeurs déterminées en centralisé. Même si les courbes de silhouette sur UNSW-NB15, pour la plupart des valeurs de r, ont des allures satisfaisantes, les courbes sur CIC-IDS2017 sont très plates. Ceci peut poser un problème si la variance est importante. En effet, il y a le risque de s'éloigner de la zone de maximum à cause de la stochasticité du processus. Toutefois, en calculant la courbe moyenne, on réussit à distinguer un maximum sur la courbe. Les courbes de silhouette ont des formes très différentes quand on compare les deux jeux de données. Pour ce qui est d'UNSW-NB15, il semblerait que le jeu de données soit plus facilement partitionnable par clustering. En effet, les formes des courbes sont très satisfaisantes. *A contrario*, CIC-IDS2017 produit des courbes moins claires. On peut supposer que ce jeu de données présente des données moins facile à rassembler par clustering.

Nous avons établi que pour les deux versions du k-means fédéré, il suffit de choisir $r = 0$ dans de nombreux cas. C'est-à-dire le nombre d'allers-retours minimal de la méthode. L'annexe B présente des graphes similaires à ceux de cette section mais pour le score F_1 et non la silhouette. En parcourant cette annexe, on comprend qu'un nombre minimal d'allers-retours est préférable pour profiter des meilleures performances ou des performances équivalentes à k fixé. Ainsi, le k-means++ fédéré semble présenter de meilleures performances sans itérations. Cela signifie peut-être que la méthode choisie pour apprendre et agréger les clusters dans les communications suivantes pose un problème. De plus, nous observons aussi une sous-optimalité du nombre de clusters des méthodes fédérées comparées au cas centralisé. En effet, nous constatons dans les tableaux 5.4 et 5.5 que le nombre de clusters k choisi par les modèles fédérés est supérieur à celui choisi par le k-means centralisé. De plus, le nombre de clusters k est supérieur dans le cas non i.i.d. que dans le cas i.i.d.. Cela signifie que les modèles sont sensibles à l'hétérogénéité de la distribution des données des clients. Ainsi, ce que nous avons énoncé dans ce paragraphe indique que les modèles fédérés n'arrivent pas à atteindre l'optimalité du k-means centralisé. De fait, la fonction d'agrégation proposée par Garst et al. [16], que nous utilisons aussi dans nos modèles, ne présente *a priori* pas d'équivalence avec l'algorithme de Lloyd centralisé. Cela pourrait être la source de l'instabilité des modèles fédérés étudiés.

Un moyen de contourner l'instabilité pour $r > 0$ pourrait être le suivant. On pourrait faire le choix de k avec $r > 0$ si jamais la silhouette n'est pas lisible pour $r = 0$ et garder ce nombre

de clusters pour $r = 0$. Toutefois, cette analyse n'est possible que parce que dans notre cas de figure nous pouvons calculer les performances pour chaque valeur de k. En effet, nous disposons de la vraie classe de chaque communication. Or dans un cas réel, seule la silhouette peut-être observée, sinon on demanderait une expertise à chaque étape du processus pour classer les clusters. Ceci nuirait aux bénéfices d'utiliser de l'apprentissage non supervisé.

5.3 Comparaison des performances des modèles

Au cours de cette section, nous comparons les performances des modèles avec les hyperparamètres sélectionnés au cours de ce chapitre et de l'annexe A. Nous abordons les performances de classification des modèles fédérés. Les modèles évalués ont été entraînés sur les données fédérées réparties de façon non i.i.d. entre les clients. Nous excluons l'étude de la répartition i.i.d. pour éviter d'alourdir le mémoire et parce que nous observons en section 5.2 et dans l'annexe B que ce cas est plus favorable que celui non i.i.d.. De plus, le cas i.i.d. est un cas relativement peu réaliste et idéal mathématiquement. Ainsi, l'évaluation de cette section 5.3 est une comparaison du cas plus réaliste.

5.3.1 Performances binaires

Comparaison des performances binaires

Les performances en classification binaire des modèles sur CIC-IDS2017 et UNSW-NB15 avec une distribution de clients non i.i.d. sont rassemblées dans les tableaux 5.6 et 5.7 respectivement. Ces tableaux présentent les performances des trois modèles de clustering quand ils sont utilisés dans l'IDS proposé. Pour avoir un point de comparaison, les performances du k-means centralisé sont présentes dans le tableau.

Modèle	Exactitude	Précision	Rappel	F_1	k
k-means (centralisé)	0.8958	0.7505	0.6969	0.7227	39
K-means fédéré	0.9121	0.7041	0.9554	0.8107	83
Fed. k-means & fed. k-means++	0.8947	0.6765	0.8613	0.7578	63
Meta k-means	0.9228	0.7335	0.9510	0.8282	295

Tableau 5.6 Performances des modèles sélectionnés (CIC-IDS2017 non i.i.d.)

On remarque dans le tableau 5.6 que les modèles k-means fédéré et meta k-means ont des performances en termes d'exactitude supérieures à celle du k-means centralisé. De plus, tous

les modèles fédérés ont de meilleures performances en score F_1 et rappel que le modèle centralisé. Or, un meilleur rappel implique un taux de faux négatif plus faible *i.e.* moins d'attaques échappent aux modèles. Ces améliorations en rappel se traduisent toutefois par de plus faibles précisions que le modèle centralisé. Cela signifie que ces modèles produisent plus de faux positifs et ainsi risquent de causer de l'*alert fatigue*. L'ajustement entre les faux positifs et faux négatifs peut être réalisé en changeant la valeur du seuil η de classification (cf. premier paragraphe, section 3.2.4). Finalement, le modèle utilisant l'initialisation k-means++ fédérée est le modèle ayant les performances les plus basses des modèles fédérés mais présente tout de même un score F_1 supérieur au k-means centralisé.

Les meilleures performances apparentes des modèles fédérés sont à contraster avec le nombre de clusters optimaux de chaque méthode. Comme constaté dans la section 5.2, tous les modèles fédérés proposent un nombre optimal de clusters plus importants que le k-means centralisé. De ce point de vue, ils ont donc tendance à faire pencher le compromis entre les efforts d'analyse et les performances vers plus d'analyse que le cas centralisé. Cela implique donc plus d'analyse/expertise des clusters puisqu'il y en a plus à étudier. Le nombre de clusters sélectionnés peut donc être un critère de sélection. Ainsi, on pourrait préférer choisir parmi les trois modèles fédérés le k-means fédéré avec initialisation k-means++ fédérée, que nous proposons, puisque c'est celui qui recommande le plus petit nombre de clusters.

Modèle	Exactitude	Précision	Rappel	F_1	k
k-means (centralisé)	0.7659	0.6640	0.9500	0.7817	9
K-means fédéré	0.7706	0.6904	0.9201	0.7889	15
Fed. k-means & fed. k-means++	0.7743	0.7121	0.8456	0.7731	27
Meta k-means	0.7595	0.6881	0.8959	0.7784	24

Tableau 5.7 Performances des modèles sélectionnés (UNSW-NB15 non i.i.d.)

Pour UNSW-NB15, les performances des modèles de k-means fédéré ainsi que de k-means++ fédéré ont une exactitude légèrement supérieure à celle du k-means centralisé. Toutefois, le nombre de clusters optimal trouvé avec la silhouette est plus important que dans le cas centralisé. Le k-means fédéré a des scores de précision et rappels plus proches du k-means centralisé là où le k-means++ fédéré a une précision supérieure et un rappel inférieur. Le meta k-means présente un score d'exactitude en dessous des autres modèles d'au moins 0.01 même si son score F_1 reste proche de moins de 0.005 aux autres. On remarque alors que les performances des modèles ne suivent pas tout à fait ce qui est observé sur CIC-IDS2017.

On comprend alors que la comparaison des modèles sur un jeu de données n'est pas transfé-

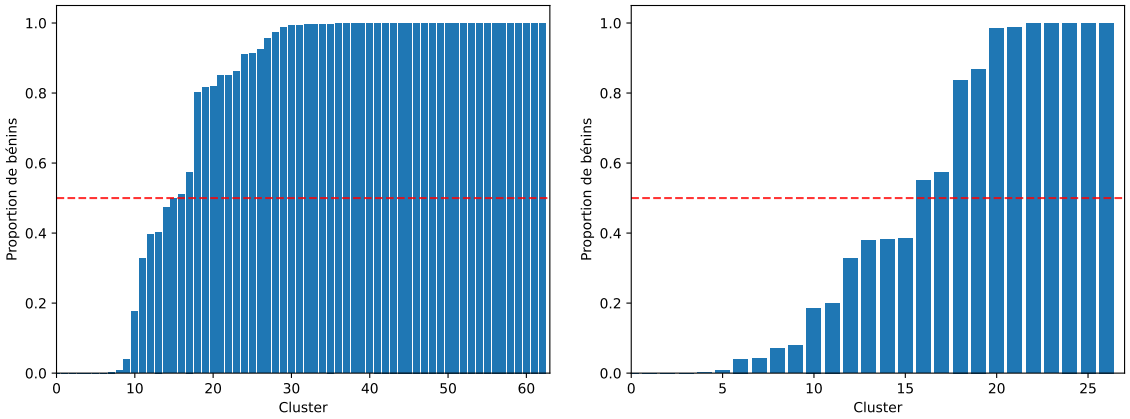
rable à un autre. Ainsi, on ne peut pas dire qu'un modèle est meilleur dans l'absolu puisque les commentaires formulés ont différé entre les tableaux 5.6 et 5.7.

Tous les modèles détectent des attaques lors de la phase de test. Ceci est une bonne nouvelle. En effet, même si certains clients n'ont jamais observé d'attaques, ils bénéficient de ceux qui en ont subi. De fait, ces clients sans attaques disposent après l'entraînement de clusters dont le vote les a déclarés malveillants. Ils connaissent donc un moyen de détecter des attaques. Ceci indique donc qu'il y a bien une forme de **collaboration** à l'œuvre à travers le protocole proposé et les algorithmes de clustering utilisés.

Proportion de communications bénignes

Pour cette partie de la section 5.3.1 et la section 5.3.2, le modèle fédéré étudié est celui utilisant l'initialisation k-means++ fédérée.

La figure 5.4 représente la proportion de communications bénignes dans chacun des clusters dans une configuration non i.i.d. des clients. La ligne rouge en tirets est située en abscisse 0.5. C'est donc la frontière de décision pour déterminer si un cluster est normal ou de l'attaque. Les proportions sont triées par ordre croissant et les numéros de clusters ne correspondent pas à ceux de la figure 5.5.



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

FIGURE 5.4 Proportion de communications bénignes dans chaque cluster (initialisation k-means++ fédérée)

On note que pour CIC-IDS2017 (figure 5.4a), il y a 45 des 47 clusters bénins qui sont composés à plus de 80% de communications normales. Dans le même sens, 11 des 16 clusters d'attaques sont composés à moins de 20% de communications normales. On note de même que 9 des 11

clusters bénins sont normaux à plus de 80% et 12 des 16 clusters d'attaque sont à moins de 20% bénins pour UNSW-NB15 (figure 4.4b). Ainsi, un analyste aurait peu de risques de se tromper en classant la plupart des clusters.

On observe ainsi sur les figures que la plupart des clusters sont éloignés de la frontière 0.5. Pour CIC-IDS2017, on compte 4 clusters à moins de 10 points de pourcentage de la frontière. Pour UNSW-NB15, seulement 2 clusters sont peu éloignés de la limite entre les deux classes. Ainsi, on remarque que pour la plupart des partitions des données, le fait de classer un cluster comme étant de l'attaque ou du trafic normal est assez "sûr". L'analyse de la figure 5.4 rejoint la discussion de la section 3.2.4 et les résultats sur le k-means centralisé en section 4.2.2.

5.3.2 Performances multi-classes

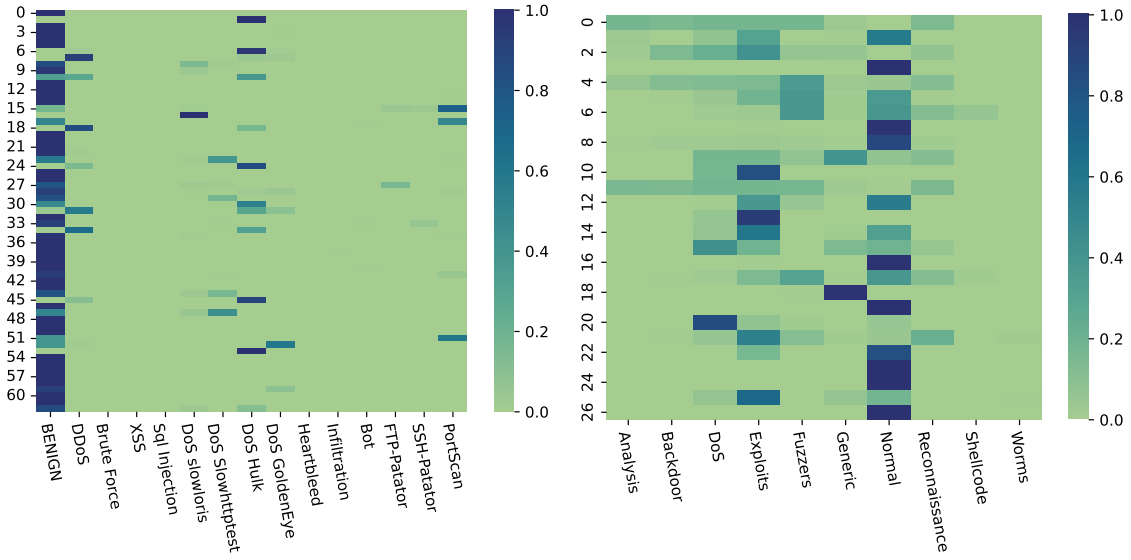
Proportion de communications dans les clusters

La figure 5.5 présente la proportion de chaque type de communication dans chaque cluster pour le modèle utilisant le k-means++ fédéré. Cette figure représente donc le même type de données que la figure 4.4. Les lignes de la matrice décrivent la composition de chaque cluster et somment à 1. Les colonnes représentent la proportion de chaque type de communication parmi les clusters.

Les matrices contenant la proportion de chaque type de communication dans chaque cluster pour les autres modèles fédérés sur des données non i.i.d. sont inclus dans l'annexe C.

La figure 5.5a présente les résultats pour le jeu de données CIC-IDS2017 avec une répartition des données chez les clients qui est non i.i.d.. On remarque que les communications bénignes sont majoritaires dans 47 clusters (cf. fig. 5.4a). On constate que les communications les plus représentées dans les clusters sont les *DDoS*, les *DoS slowloris*, les *DoS Hulk* et les *PortScan*. Cela correspond aux attaques les plus importantes en volume du jeu de données. On remarque que dans plusieurs clusters, comme le 31, plusieurs attaques sont présentes. Ceci révèle donc une proximité de ces types d'attaques.

La proportion de chaque type de communication dans chaque cluster pour la répartition non i.i.d. des données de UNSW-NB15 est représentée dans la figure 5.5b. On remarque que dans les clusters où les communications normales sont majoritaires, les attaques ne sont pratiquement pas représentées. Les attaques de type *Exploits* et *Generic* semblent avoir quelques clusters dans lesquelles elles sont en écrasante majorité. Toutefois, certains clusters comme le 0 ont des répartitions plutôt égales entre les communications d'attaques. Cela indique que le modèle a des difficultés à distinguer certaines attaques entre elles.



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

FIGURE 5.5 Proportion de communications dans chaque cluster (initialisation k-means++ fédérée)

Performances en classification multi-classes

Classification à un tour Le tableau 5.8 présente le nombre de clusters classés selon chaque type de communications avec la méthode de classification multi-classes à un tour sur CIC-IDS2017. On remarque, en cohérence avec la figure 5.4a, que 48 clusters sont classés comme bénins. De plus, les attaques qui ont donné leur nom à des clusters sont globalement des attaques qui nécessitent un volume important de données. On remarque qu'il y a une plus grande diversité dans les types de clusters que dans le cas centralisé (tableau 4.3). Ce commentaire reste toutefois à nuancer puisque les nouvelles classes sont les *DoS GoldenEye* et les *DoS slowloris* et ce sont donc d'autres types de *DoS*.

Total	Bénin	*DDoS*	*DoS GoldenEye*	*DoS Hulk*	*DoS slowloris*	*PortScan*
63	48	4	1	7	1	2

TABLEAU 5.8 Nombre de clusters classés à un tour par type de communication avec le k-means++ fédéré (CIC-IDS2017 non i.i.d.)

Avec cette méthode de vote, le modèle obtient une exactitude de 0.8689 sur CIC-IDS2017. Cette performance est inférieure à la classification binaire. On peut comprendre la différence avec par exemple le cluster 17 de la figure 5.5a. Il est composé à 49.86% de communica-

tions bénignes, de 49.60% de *PortScans* et les communications restantes étant réparties entre *DDoS*, *Inflitration* et *Bot*. On remarque que dans le cas binaire, le cluster serait considéré comme de l'attaque et donc le modèle se tromperait pour moins de la moitié des données. Toutefois, en classification multi-classes à un tour, on considère que le cluster est bénin et donc le modèle se trompe dans un peu plus de la moitié des cas.

Le tableau 5.9 présente le nombre de clusters par type de communications sur UNSW-NB15. On peut formuler les mêmes commentaires sur ce tableau que pour CIC-IDS2017. Les clusters normaux sont en nombre important et les clusters d'attaque sont souvent du type des communications les plus représentées dans le jeu de données. On note toutefois qu'il n'y a pas de cluster *Reconnaissance* alors que c'est une attaque plus commune dans UNSW-NB15 que les *Analysis* ou bien les *DoS*. Cela signifie donc que cette attaque est répartie dans plusieurs clusters sans être majoritaire dans l'un d'entre eux. Cela peut ce confirmer avec l'observation de la *heatmap* (figure 5.5b). On remarque là aussi qu'il y a une plus grande diversité de classes que pour le cas centralisé (tableau 4.4).

Total	*Analysis*	*DoS*	*Exploits*	*Fuzzers*	*Generic*	Normal
27	1	2	7	1	2	14

TABLEAU 5.9 Nombre de clusters classés à un tour par type de communication avec le k-means++ fédéré (UNSW-NB15 non i.i.d.)

Avec la classification à un tour, le k-means++ fédéré obtient une exactitude de 0.6311. Ceci représente une baisse importante de performances par rapport au cas binaire qui présentait une exactitude de 0.7743. Cette baisse de performance peut par exemple s'expliquer par l'étude du cluster 0 de la *heatmap* 5.5b. En effet, la répartition du cluster est assez équilibrée entre les *Analysis*, *Backdoor*, *DoS*, *Exploits*, *Fuzzers* et *Reconnaissance*. De plus, il n'y a pas de communications normales. Or, dans le cas binaire, on rassemblerait ces attaques ensemble là où en classification multi-classes à un tour on classe le cluster comme des *Analysis*. En effet, c'est cette attaque qui est en majorité relative avec 17.05% du cluster.

Classification à deux tours Le tableau 5.10 présente le nombre de clusters par classe avec la classification à deux tours sur CIC-IDS2017. On constate que 47 clusters sont bénins, ce qui est un de moins que dans la classification à un tour. Selon cette méthodologie, ce cluster est considéré comme *PortScan*.

Total	Bénin	*DDoS*	*DoS GoldenEye*	*DoS Hulk*	*DoS slowloris*	*PortScan*
63	47	4	1	7	1	3

TABLEAU 5.10 Nombre de clusters classés à deux tours par type de communication avec le k-means++ fédéré (CIC-IDS2017 non i.i.d.)

Avec la classification multi-classes à deux tours, l'exactitude du modèle est de 0.8622. Cela représente une légère baisse de 0.67 points de pourcentage. La différence de performance s'explique notamment par l'étude du cluster 17 de la *heatmap* 5.5a. En effet, la classe majoritaire parmi les attaques est le *PortScan*. Or, ces communications sont en infériorité numérique comparées aux communications bénignes. Ainsi, le modèle se trompe légèrement plus souvent sur ce cluster que dans le cas de la classification multi-classes à un tour.

Le tableau 5.11 présente le nombre de clusters par type de communications prédits par le k-means++ fédéré sur le jeu de données UNSW-NB15 en répartition non i.i.d.. La différence qu'on observe avec la classification à un tour est la réduction de trois clusters normaux au profit des *Fuzzers*.

Total	*Analysis*	*DoS*	*Exploits*	*Fuzzers*	*Generic*	Normal
27	1	2	7	4	2	11

TABLEAU 5.11 Nombre de clusters classés à deux tours par type de communication avec le k-means++ fédéré (UNSW-NB15 non i.i.d.)

Le modèle couplé à la classification multi-classes à deux tours atteint une exactitude de 0.6059. Là encore, cette baisse par rapport à la classification multi-classes à un tour s'explique parce que dans au moins un cluster, la classe normale est en majorité au plus relative et qu'une classe d'attaque l'emporte alors que sa proportion est plus faible. C'est le cas des clusters 5 et 6 de la figure 5.5b par exemple.

Discussions des résultats Finalement, les analyses de cette section rejoignent celles de la section 4.2.3 dédiée au k-means centralisé. On retrouve les mêmes phénomènes qui font que la classification multi-classes présente des risques de performances réduites. C'est-à-dire que dans le cas de la classification à un tour, la baisse des performances par rapport à la classification binaire s'explique par le fait qu'on choisit parfois la classe normale parce qu'elle est en majorité relative. Ce cas de figure fait baisser les performances dans les clusters où les

attaques sont en majorité et parce qu'on ne les considère plus comme un bloc mais comme plusieurs types différents. Pour la classification à deux tours, les performances baissent encore parce que dans certains clusters on choisit un type de communication qui n'est pas en majorité relative. De fait, lorsqu'on écarte la classe normale, qui est souvent la plus représentée, la seconde classe la plus représentée a nécessairement en proportion inférieure. La prédiction du modèle se trompe donc plus pour ce cluster. Toutefois, le vote à deux tours a été introduit pour éviter de classer tous les clusters comme normaux. En ce sens, il a rempli son rôle puisque dans tous les cas étudiés, il y a plus de clusters d'attaques que via le vote à un tour.

Ces cas de figure apparaissent car certains types de communications sont faiblement séparables. Ainsi, les attaques ne se retrouvent pas dans des clusters distincts les uns des autres. Il n'est donc pas clair de parler de cluster par type de communication mais plutôt par le fait que ce sont des attaques ou du trafic normal. Pour combiner le "meilleur des deux mondes", on peut classer les communications de façon binaire puis, en cas d'attaque, fournir la proportion de chaque type d'attaque dans le cluster pour aiguiller l'enquête d'un analyste.

5.4 Discussion sur l'IDS fédéré

Durant le chapitre 5, nous avons étudié les performances de l'IDS fédéré proposé dans le chapitre 3 pour répondre à l'objectif 3 (section 1.3). Nous avions analysé l'IDS proposé dans son format centralisé dans le chapitre 4 et nous avons donc pu réaliser des comparaisons. Cette section 5.4 est dédiée aux discussions sur l'IDS dans son format fédéré.

Lors de la sélection d'hyperparamètres dans la section 5.2, nous avons vu que les courbes de silhouettes en fonction du nombre de *rounds* de communications r n'ont pas de comportement simple. De fait, on aurait pu s'attendre à ce que la silhouette s'améliore lorsque r croît. Cependant, ce n'est pas ce qui est observé. Dans un cas réel, il est donc primordial de bien réaliser cette étape de sélection de variables et d'analyser les courbes de silhouette. Toutefois, grâce aux courbes de l'annexe B, nous constatons que les meilleures performances sont atteintes avec un nombre de *rounds* de communications minimal. Ainsi, la sélection du nombre de clusters k pourrait être réalisée avec un nombre de *rounds* minimal ou plus élevé au besoin et l'appliquer au cas $r = 0$.

Nous avons aussi observé dans la section 5.2 que les modèles de clustering fédérés généraient des courbes de silhouettes moins pertinentes ou décalées vers les valeurs de k plus grandes que le cas centralisé. Cela signifie que le nombre de clusters optimal est plus difficile à lire et/ou plus grand que dans le cas centralisé. Ceci montre une sous-optimalité des modèles de clustering fédéré comparé au k-means centralisé. Ces remarques sont d'autant plus vraies dans

la configuration non i.i.d.. Les modèles sont donc sensibles à l'homogénéité ou l'hétérogénéité des distributions de données entre les clients. Cela peut représenter un inconvénient des méthodes étudiées. Ces commentaires ne s'appliquent pas lorsque l'initialisation k-means++ est utilisée seule puisqu'elle est équivalente à l'initialisation k-means++ centralisée. Ainsi, la cause d'instabilité semble provenir de la fonction d'agrégation utilisée. De fait, il n'y a pas d'équivalence *a priori* entre la fonction d'agrégation des algorithmes 3, 5 et 7 avec l'algorithme de Lloyd (cf. algorithme 1). Ainsi, nous n'avions aucune garantie que les algorithmes fédérés se comportent comme le k-means centralisé.

Dans la section 5.3.1, nous avons comparé les performances de l'IDS selon le modèle de clustering utilisé. Nous avons trouvé que les performances avec les modèles fédérés étaient proches de celles avec k-means centralisé. Toutefois, le nombre de clusters utilisés était plus élevés dans les cas fédérés que centralisés. Ainsi, les modèles fédérés semblent moins efficients mais ont des performances comparables. Ceci vient atténuer la portée du commentaire sur la sous-optimalité de la fonction d'agrégation. De fait, même si elle semble causer de l'instabilité, elle n'empêche pas l'IDS d'atteindre des performances comparables au cas centralisé.

Nous trouvons d'autres similarités entre les IDS fédérés étudiés et celui centralisé. En effet, la proportion de communications bénignes dans les clusters sont généralement éloignées de 0.5. Ceci signifie qu'un analyste aurait peu de risque de mal classer chaque clusters. Ceci valide la méthode choisie pour classer les clusters et rejoint les arguments de la section 3.2.4. Toutefois, pour ce qui est de la classification multi-classes, certaines classes sont en large infériorité dans des clusters et pourraient ne pas être détectées par un analyste dans la phase d'expertise. Comme mentionné dans la section 4.3, cela ne changerait pas la classification prédite. Une autre similarité se trouve dans le compromis entre le travail d'analyse et les performances. Nous avons observé ce phénomène chez les IDS fédérés avec l'annexe B.

Globalement, les performances obtenues dans le cadre fédéré et centralisé ne sont pas suffisantes pour un déploiement de l'IDS proposé. De fait, classer correctement entre 80% et 90% des communications n'est pas suffisant. Cette observation rejoint celle sur la faible capacité des modèles étudiés à distinguer toutes les classes. Pour améliorer ce point, il faudrait aborder une autre modélisation de données.

Ainsi, les remarques et conclusions que nous tirons de ce chapitre 5 se rapprochent fortement de celles exprimées dans la section 4.3. Toutefois, les IDS fédérés présentent l'hyperparamètre r en plus par rapport à celui centralisé. Ceci induit des comportements différents. *In fine*, il est préférable de garder r minimal à la fois d'un point de vue stabilité des modèles, performances, minimisation de la bande passante et minimisation des communications transmises.

CHAPITRE 6 CONCLUSION

Ce chapitre clôt le mémoire par une conclusion en trois parties. Tout d'abord, nous synthétiserons les travaux puis nous en dessinerons les limites. Finalement, nous aborderons les travaux futurs.

6.1 Synthèse des travaux

Au cours de ce projet de maîtrise, nous avons proposé une architecture d'IDS (objectif 1) ainsi que deux protocoles de clustering non supervisés fédérés. Nous avons aussi proposé un outil pour l'étude des modèles de clustering fédérés à travers la silhouette simplifiée fédérée. En adaptant le paradigme de l'apprentissage fédéré au clustering de données, nous avons construit un IDS non supervisé fédéré. La méthode d'initialisation de clustering proposée préserve mieux la confidentialité des données des participants (objectif 2) que le modèle initial de Garst et al. [16]. De plus, l'étude de l'IDS proposé (objectif 3) a montré que les participants collaborent bel et bien et bénéficient donc des données des autres clients.

À travers les expérimentations, nous avons observé qu'il est crucial de bien explorer l'espace des hyperparamètres des algorithmes de clustering. En effet, la présente étude ne permet pas de dégager de modèle manifestement plus performant dans tous les cas de figure. Il est donc important de bien analyser l'évolution de la silhouette pour choisir un modèle adapté à la situation. En effet, dans le cadre non-supervisé, nous ne pouvons pas nous baser sur un étiquetage massif des données pour aiguiller la recherche d'hyperparamètres.

Nous avons aussi observé que les performances des modèles fédérés étaient en général proches de celui centralisé. Ceux qui ont de meilleures performances utilisaient en général bien plus de clusters. De fait, nous avons constaté pour tous les modèles un compromis entre les efforts d'analyse et les performances. De plus, les méthodes fédérées suggèrent un plus grand nombre de clusters que la méthode centralisée. Nous avons remarqué que la silhouette fédérée est équivalente à celle centralisée, et il en va de même pour l'initialisation k-means++. Il semble donc que ce soit la méthode d'agrégation qui est responsable du plus grand nombre de clusters proposé par les algorithmes fédéré. Notre hypothèse est qu'aucune des méthodes d'agrégation n'est équivalente au cas centralisé de l'algorithme de Lloyd. Ainsi, nous n'avons aucune garantie que le nombre de clusters obtenus dans un cadre fédéré soit le même que pour un cadre centralisé dès qu'on inclut l'agrégation utilisée au cours de ce mémoire.

Lors des expérimentations, nous avons pu observer les performances des modèles dans le cas

où la classification est multi-classes. Les résultats semblent pointer que le clustering, en tout cas sur les données étudiées, n'est pas capable de bien séparer toutes les classes. Une piste pour expliquer cela est que le modèle n'a pas accès au contenu de la communication. Ainsi, il se peut que les différents types d'attaques ne soient pas facilement séparables sans le contenu de la communication (*payload*). De plus, les types de communications en volume important dans les jeux de données ont tendance à occuper toute la place dans les clusters. Ceci couplé à la faible séparation entre les attaques rend invisibles les attaques à plus faible volume.

6.2 Limites de la solution proposée

Le modèle proposé ne tient pas compte de la disponibilité des participants. En effet, on suppose que le serveur ne met à jour le modèle que si tous les clients ont répondu. Ceci n'est pas le cas de tous les modèles fédérés.

Dans la même idée, l'étude n'a pas tenu compte de l'organisation d'un véritable réseau informatique. On exclut donc ici des problèmes qui peuvent intervenir sur un réseau et la réactivité du modèle pour détecter des intrusions.

L'IDS proposé suppose que les clients sont capables d'estimer avec fiabilité les proportions de communications. Cette hypothèse, même si elle est viable dans un cas binaire, peut poser problème dans le cas de la classification multi-classes. En effet, les classes en plus faible volume peuvent être difficiles à détecter lors de l'analyse d'un cluster comparé aux classes à plus grand volume.

L'une des hypothèses clefs est que les participants sont honnêtes mais curieux. Ainsi, on ne tient pas compte d'une volonté de la part d'un client ou du serveur d'altérer les résultats du protocole. Ils cherchent simplement à tirer le plus d'informations de celui-ci. Dans un cadre où les entreprises sont sélectionnées pour intégrer le protocole, on peut supposer que les participants suivent ce type de profil. Toutefois, il se peut qu'un client ait subi une attaque. L'attaquant peut ainsi altérer les communications du client vers le serveur et ainsi nuire au protocole. Ainsi, un client malveillant peut devenir un sujet de préoccupation.

6.3 Améliorations futures

6.3.1 Amélioration de la modélisation

Les travaux présentés dans ce mémoire peuvent se poursuivre sur plusieurs plans. Tout d'abord, sur le plan de la modélisation des données, les améliorations ou méthodes à explorer suivantes peuvent être apportées.

Il existe d'autres variantes du k-means++, comme la version gloutonne (*greedy*) utilisée par Scikit-Learn [47]. De plus, il existe d'autres initialisations pour le clustering. Ainsi, une piste d'amélioration pourrait être d'essayer d'autres initialisations pour les modèles de clustering.

Comme exprimé précédemment, les itérations proposées par Garst et al. [16] semblent réduire dans un sens les performances et faire "glisser" les courbes de silhouette vers plus de clusters. On pourrait ainsi rechercher une meilleure manière de calculer les itérations une fois l'initialisation réalisée. Pour ce faire, on pourrait choisir l'initialisation k-means++ fédérée incorporée au k-means de Holzer et al. [28]. De plus, cet algorithme ne nécessite pas que tous les clients participent en même temps, ce qui pourrait être reproché aux méthodes étudiées au cours de ce mémoire.

Le choix du nombre de clusters optimal du meta k-means pourrait être rendu plus stable avec l'utilisation de la silhouette fédérée proposée. Toutefois, ceci augmenterait le nombre de communications entre les clients et le serveur. De fait, cela nécessiterait un appel aux clients à chaque valeur de k parcourue.

Il est récurrent d'utiliser des projections de données avant de réaliser du clustering. On peut alors inclure des méthodes comme l'analyse en composantes principales ou bien les auto-encodeurs. Ces méthodes sont utilisées pour extraire des données les informations les plus importantes et donc faciliter le clustering.

Le clustering pourrait être amélioré en considérant de la corrélation temporelle. Les données seraient ordonnées en séquences. Or, à cause de la haute dimensionnalité d'une séquence de vecteurs, il serait difficile de réaliser du clustering dessus. Il faudrait alors utiliser une modélisation temporelle pour plonger une séquence dans un espace suffisamment "petit" pour que du clustering par k-means soit possible. Ainsi, en utilisant par exemple un auto-encodeur à base de LSTM [48] ou de réseaux de neurones convolutifs [49], on peut réduire suffisamment la dimension de la séquence pour qu'un algorithme de clustering puisse la traiter. Toutefois, de tels modèles augmenteraient nécessairement la puissance de calcul nécessaire ainsi que le temps de réaction de l'IDS.

6.3.2 Amélioration de la confidentialité

La confidentialité des clients dans le protocole proposé peut être améliorée. Par exemple, les clients pourraient bruiter les communications envoyées au serveur. De cette manière, chaque communication individuelle ne serait pas fiable. Toutefois, l'agrégation à l'échelon du serveur le serait. Ainsi, les données des clients ne seraient pas compromises et le protocole pourrait fonctionner. Ceci correspond à de la confidentialité différentielle (*differential privacy*).

En cryptographie, les chiffrements homomorphes permettent de réaliser des opérations sur des messages chiffrés de telle sorte qu'en le déchiffrant, on obtienne le résultat du calcul sur les messages clairs. Posons $\Pi = (Gen, Enc, Dec, Eval)$ un système de chiffrement homomorphe composé respectivement : d'une fonction de génération de clefs, d'une fonction de chiffrement, d'une autre de déchiffrement et d'une fonction d'évaluation. Si Π est homomorphe suivant l'addition par exemple, on aurait alors : $Dec_k(Eval(c_1 + c_2)) = Dec_k(c_1) + Dec_k(c_2)$ pour k une clef donnée et c_1, c_2 deux messages chiffrés. Ce type de chiffrement augmenterait la confidentialité des clients. En effet, si le serveur réalise toutes les étapes d'agrégations sur des données chiffrées, il ne peut pas en tirer d'informations. Le serveur ne servirait alors que d'opérateur pour l'agrégation sans pouvoir être observateur.

6.3.3 Prise en compte de participants malveillants

Comme mentionné dans les limites du projet, une hypothèse formulée plus tôt est que les participants du protocole (clients et serveur) sont honnêtes mais curieux. Ainsi, ils réalisent scrupuleusement les étapes du protocole. Toutefois, ils cherchent à obtenir le plus d'informations possible à travers leurs interactions. Cette hypothèse est valable dans le cas où des entreprises participeraient au protocole et craindraient pour leur sécurité si elles agissaient de sorte à nuire aux performances du modèle. Or, il se peut qu'une menace interne cherche à nuire à l'entreprise ou bien celle-ci peut être compromise. De fait, un client pourrait "empoisonner" le modèle lors de l'entraînement avec des données corrompues et ainsi dégrader le modèle. Ainsi, les travaux futurs pourraient travailler à la détection de client malveillant.

RÉFÉRENCES

[1] Organisation mondiale du commerce. (2024) 2023 was a big year for cybercrime – here's how we can make our systems safer. [En ligne]. Disponible : https://www.weforum.org/agenda/2024/01/cybersecurity-cybercrime-system-safety/

[2] "Cost of a data breach report 2023," IBM, Rapport technique, 2023. [En ligne]. Disponible : https://www.ibm.com/downloads/cas/E3G5JMBP

[3] Amazon Canada. (2022) Ooo until tbd ? majority of canadian office workers want remote work to stay. [En ligne]. Disponible : https://www.newswire.ca/news-releases/ooo-until-tbd-majority-of-canadian-office-workers-want-remote-work-to-stay-897250807.html

[4] Ericsson. Iot connections forecast – mobility report. [En ligne]. Disponible : https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook

[5] ANSSI. (2022) Tendances - les cybermenaces. [En ligne]. Disponible : https://cyber.gouv.fr/tendances-les-cybermenaces

[6] Centre canadien pour la cybersécurité. (2022) Évaluation des cybermenaces nationales 2023-2024. [En ligne]. Disponible : https://www.cyber.gc.ca/fr/orientation/evaluation-des-cybermenaces-nationales-2023-2024#fn3

[7] IBM. (2024) What is a soc ? [En ligne]. Disponible : https://www.ibm.com/topics/security-operations-center

[8] Microsoft. What is a siem ? [En ligne]. Disponible : https://www.microsoft.com/en-ca/security/business/security-101/what-is-siem

[9] Splunk. Qu'est-ce qu'un centre opérationel de sécurité ? [En ligne]. Disponible : https://www.splunk.com/fr_fr/data-insider/what-is-a-security-operations-center.html

[10] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, nº. 2, p. 129–137, mars 1982.

[11] scikit-learn. sklearn.cluster.kmeans. [En ligne]. Disponible : https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[12] D. Arthur et S. Vassilvitskii, "k-means++ : the advantages of careful seeding," dans *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, PA, USA, 2007, p. 1027–1035.

[13] P. J. Rousseew, "Silhouettes : A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, p. 53–65,

nov. 1987. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/0377042787901257

[14] E. Hruschka, L. de Castro et R. Campello, "Evolutionary algorithms for clustering gene-expression data," communication présentée à Fourth IEEE International Conference on Data Mining (ICDM'04), Brighton, UK, 01-04 novembre 2004, p. 403–403.

[15] B. McMahan, A. Moore, D. Ramage, S. Hampson et B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," dans *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54. PMLR, 2017, p. 1273–1282.

[16] S. Garst et M. de Reinders, "Federated k-means clustering," *arXiv*, oct. 2023. [En ligne]. Disponible : http://arxiv.org/abs/2310.01195

[17] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola et R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, n°. 7, p. 1443–1471, 07 2001. [En ligne]. Disponible : https://doi.org/10.1162/089976601750264965

[18] L. Breiman, *Mach. Learn.*, vol. 45, n°. 1, p. 5–32, 2001.

[19] F. T. Liu, K. M. Ting et Z.-H. Zhou, "Isolation forest," dans *2008 Eighth IEEE International Conference on Data Mining*, 2008, p. 413–422.

[20] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, vol. 24, p. 498–520, 1933.

[21] T.-A. Nguyen, J. He, L. T. Le, W. Bao et N. H. Tran, "Federated pca on grassmann manifold for anomaly detection in iot networks," dans *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, p. 1–10.

[22] A. P. Dempster, N. M. Laird et D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society : Series B (Methodological)*, vol. 39, p. 1–22, sept. 1977. [En ligne]. Disponible : https://rss.onlinelibrary.wiley.com/doi/10.1111/j.2517-6161.1977.tb01600.x

[23] M. M. Fard, T. Thonet et E. Gaussier, "Deep k-Means : Jointly clustering with k-Means and learning representations," *arXiv*, déc. 2018. [En ligne]. Disponible : http://arxiv.org/abs/1806.10069

[24] M. Ester, H.-P. Kriegel, J. Sander et X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," dans *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[25] N. H. Nguyen, D. L. Nguyen, T. B. Nguyen, T.-H. Nguyen, H. H. Pham, T. T. Nguyen et P. Le Nguyen, "Cadis : Handling cluster-skewed non-iid data in federated learning with clustered aggregation and knowledge distilled regularization," dans *2023 IEEE/ACM 23RD INTERNATIONAL SYMPOSIUM ON CLUSTER, CLOUD AND INTERNET COMPUTING, CCGRID*, I. Altintas, Y. Simmhan, A. Varbanescu, P. Balaji, A. Prasad et L. Carnevale, édit. IEEE ; Assoc Comp Machinery ; IEEE Comp Soc ; IEEE Tech Comm Scalable Comp ; IEEE Tech Community Cloud Comp ; Assoc Comp Machinery, Special Interest Grp Comp Architecture ; Meta ; HPE ; IBM ; Google ; Microsoft, 2023, p. 249+, 23rd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Bangalore, INDIA, MAY 01-04, 2023.

[26] D. K. Dennis, T. Li et V. Smith, "Heterogeneity for the win : One-shot federated clustering," dans *INTERNATIONAL CONFERENCE ON MACHINE LEARNING, VOL 139*, ser. Proceedings of Machine Learning Research, M. Meila et T. Zhang, édit., vol. 139, 2021, international Conference on Machine Learning (ICML), ELECTR NETWORK, JUL 18-24, 2021.

[27] P. Awasthi et O. Sheffet, "Improved spectral-norm bounds for clustering," dans *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, A. Gupta, K. Jansen, J. Rolim et R. Servedio, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 37–49.

[28] P. Holzer et S. Jacob, "Dynamically weighted federated k-means," *arXiv*, nov. 2023. [En ligne]. Disponible : http://arxiv.org/abs/2310.14858

[29] Tensorflow. (2024) tff.learning.algorithms.build_fed_kmeans. [En ligne]. Disponible : https://www.tensorflow.org/federated/api_docs/python/tff/learning/algorithms/build_fed_kmeans

[30] M. Prasad, S. Tripathi et K. Dahal, "Unsupervised feature selection and cluster center initialization based arbitrary shaped clusters for intrusion detection," *Computers Security*, vol. 99, p. 102062, 2020. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0167404820303357

[31] A. Sirisha, K. Chaitanya, K. Krishna et S. Kanumalli, "Intrusion detection models using supervised and unsupervised algorithms - a comparative estimation," *International Journal of Safety and Security Engineering*, vol. 11, p. 51–58, 02 2021.

[32] D. Yang et M. Hwang, "Unsupervised and ensemble-based anomaly detection method for network security," dans *2022 14th International Conference on Knowledge and Smart Technology (KST)*, 2022, p. 75–79.

[33] P. C. Mahalanobis, "On the generalised distance in statistics," *Proceedings of the National Institute of Sciences of India*, vol. 2, p. 49–55, 1936.

[34] K. H. Shibly, M. D. Hossain, H. Inoue, Y. Taenaka et Y. Kadobayashi, “Personalized federated learning for automotive intrusion detection systems,” dans *2022 IEEE Future Networks World Forum (FNWF)*, 2022, p. 544–549.

[35] A. S. Md Tayeen, S. Misra, H. Cao et J. Harikumar, “Cafnet : Compressed autoencoder-based federated network for anomaly detection,” dans *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, 2023, p. 325–330.

[36] O. Aouedi, K. Piamrat, G. Muller et K. Singh, “Intrusion detection for softwarized networks with semi-supervised federated learning,” dans *ICC 2022 - IEEE International Conference on Communications*, 2022, p. 5244–5249.

[37] R. Yang, H. He, Y. Wang, Y. Qu et W. Zhang, “Dependable federated learning for iot intrusion detection against poisoning attacks,” *Computers Security*, vol. 132, p. 103381, 2023. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0167404823002912

[38] S. k. Kang, D. Lindskog et H. Samuel, “An implementation of hierarchical intrusion detection systems using snort and federated databases,” dans *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, p. 1521–1525.

[39] L. Barreto, J. Fraga et F. Siqueira, “An intrusion tolerant identity provider with user attributes confidentiality,” *Journal of Information Security and Applications*, vol. 63, p. 103045, 2021. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S2214212621002337

[40] Tensorflow Federated. ff.learning.build_federated_evaluation. [En ligne]. Disponible : https://www.tensorflow.org/federated/api_docs/python/tff/learning/build_federated_evaluation

[41] I. Sharafaldin, A. H. Lashkari et A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” communication présentée à 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, 01-04 novembre 2018, p. 108–116. [En ligne]. Disponible : https://www.scitepress.org/papers/2018/66398/66398.pdf

[42] N. Moustafa et J. Slay, “UNSW-NB15 : a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” dans *2015 Military Communications and Information Systems Conference (MilCIS)*. IEEE, nov. 2015.

[43] ——, “The evaluation of network anomaly detection systems : Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset,” *Information Security Journal : A Global Perspective*, p. 1–14, 2016.

[44] N. Moustafa, J. Slay et G. Creech, “Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks,” *IEEE Trans. Big Data*, vol. 5, n°. 4, p. 481–494, déc. 2019.

[45] N. Moustafa, G. Creech et J. Slay, “Big data analytics for intrusion detection system : Statistical decision-making using finite dirichlet mixture models,” dans *Data Analytics and Decision Support for Cybersecurity*. Cham : Springer International Publishing, 2017, p. 127–156.

[46] M. Sarhan, S. Layeghy, N. Moustafa et M. Portmann, “Netflow datasets for machine learning-based network intrusion detection systems,” dans *Big Data Technologies and Applications*, Z. Deze, H. Huang, R. Hou, S. Rho et N. Chilamkurti, édit. Cham : Springer International Publishing, 2021, p. 117–135.

[47] scikit-learn. sklearn.cluster.kmeans_plusplus. [En ligne]. Disponible : https://scikit-learn.org/stable/modules/generated/sklearn.cluster.kmeans_plusplus.html

[48] S. Hochreiter et J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, n°. 8, p. 1735–1780, 11 1997. [En ligne]. Disponible : https://doi.org/10.1162/neco.1997.9.8.1735

[49] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard et L. Jackel, “Handwritten digit recognition with a back-propagation network,” dans *Advances in Neural Information Processing Systems*, D. Touretzky, édit., vol. 2. Morgan-Kaufmann, 1989. [En ligne]. Disponible : https://proceedings.neurips.cc/paper_files/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf

ANNEXE A ANALYSE DE LA SILHOUETTE DES MODÈLES FÉDÉRÉS

L'annexe A présente l'analyse des hyperparamètres pour le k-means fédéré de Garst et al. [16] et pour le meta k-means (section 3.1.2). La discussion correspondant aux résultats présentés dans cette annexe est réalisée dans la section 5.2.3.

A.1 K-means fédéré

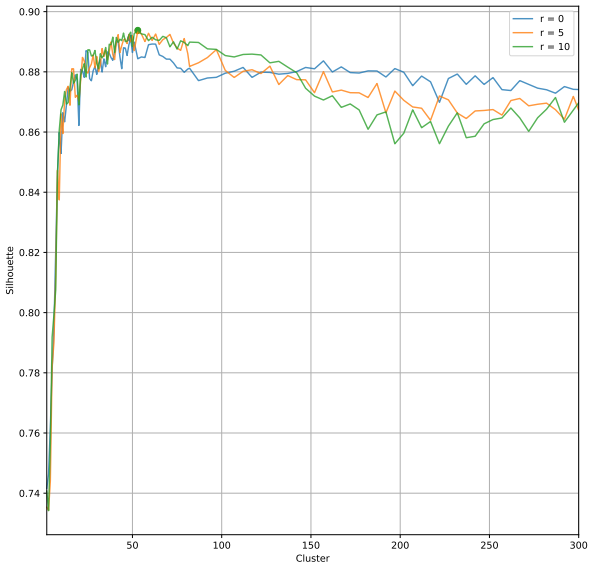
A.1.1 Résultats

La figure A.1 rassemble les silhouettes obtenues sur CIC-IDS2017 et UNSW-NB15 avec des distributions de données chez les clients qui sont i.i.d. et non i.i.d..

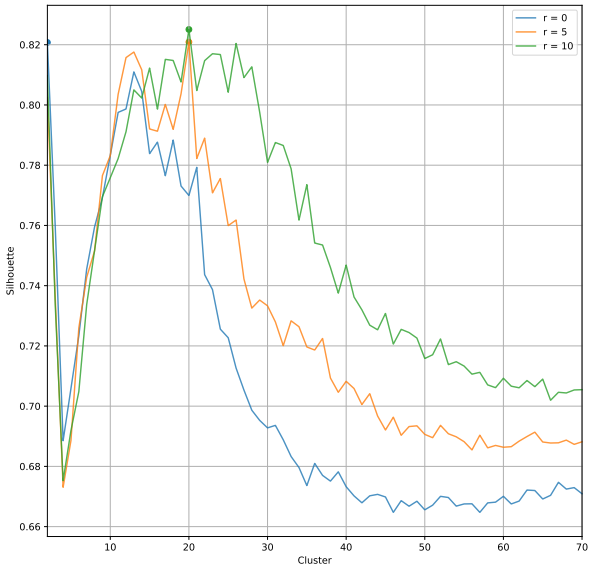
CIC-IDS2017 La sous-figure A.1a présente les résultats sur une répartition i.i.d. parmi les clients du jeu de données CIC-IDS2017. On remarque que pour toutes les valeurs de r, la silhouette est d'abord croissante puis légèrement décroissante. De plus, les courbes sont plutôt proches les unes des autres. Les maximums de silhouette sont situés en $k = 49$, $k = 53$ et $k = 53$ pour $r = 0$, $r = 5$ et $r = 10$ respectivement. Ainsi, les valeurs de k sélectionnées sont plus élevées que pour le k-means centralisé. Pour rappel, le maximum de silhouette de ce modèle était en $k = 39$.

Le comportement des courbes de silhouette pour la répartition non i.i.d. des données de CIC-IDS2017 est présenté dans la sous-figure A.1c. On observe que la silhouette pour $r = 0$ est supérieure aux courbes pour $r = 5$ et $r = 10$. Ces dernières sont par ailleurs très plates une fois la croissance pour $k < 60$ passée. Ces courbes ne semblent donc pas présenter de maximum clair. *A contrario*, même si la courbe de silhouette pour $r = 0$ n'a pas de maximum évident, on note la décroissance de la silhouette à partir de $k = 100$. De plus, l'écart entre les courbes pour $r = 0$ et $r > 0$ est très marqué ce qui peut signifier que le clustering est de moins bonne qualité pour ces dernières valeurs de r.

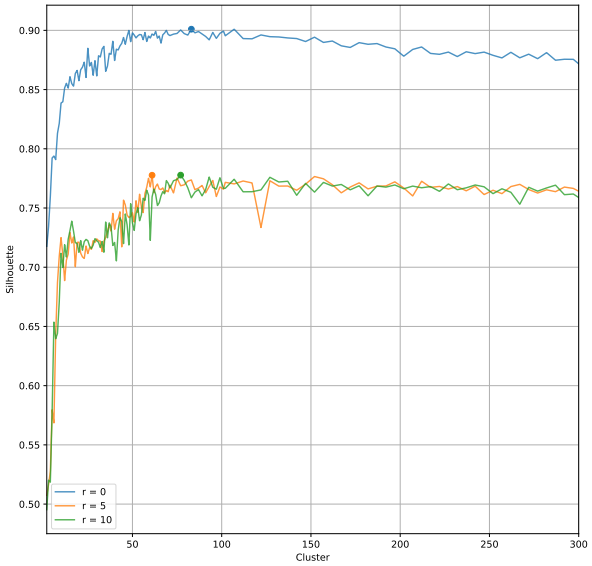
UNSW-NB15 Les résultats pour les données i.i.d. sur UNSW-NB15 sont présentés dans la sous-figure A.1b. On note que les tendances de la silhouette ressemblent à celle du k-means centralisé (fig. 4.1b). En effet, on observe un maximum local en $k = 2$, une baisse puis une croissance de la silhouette. Ensuite, la silhouette atteint une zone de maximum puis baisse en tendance. On note toutefois que plus r est grand, plus la courbe de silhouette connaît des valeurs grandes. On observe aussi que plus r est grand, plus la courbe s'étend vers les



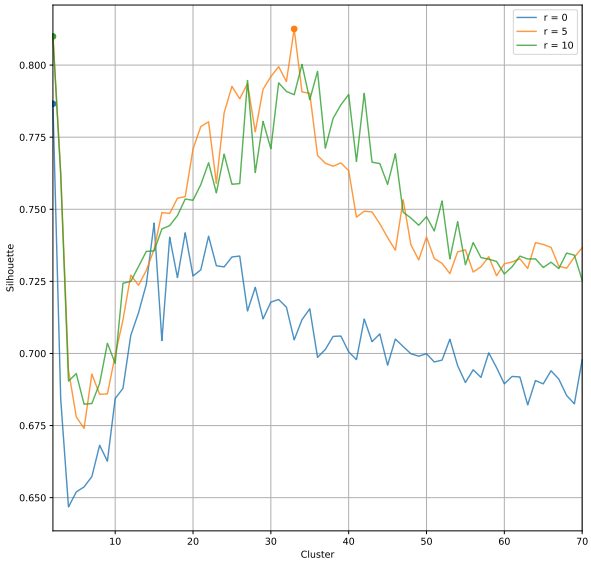
(a) CIC-IDS2017 i.i.d.



(b) UNSW-NB15 i.i.d.



(c) CIC-IDS2017 non i.i.d.



(d) UNSW-NB15 non i.i.d.

FIGURE A.1 Évolution de la silhouette en fonction du nombre de clusters sur le serveur (k-means fédéré)

plus grandes valeurs de k et de silhouette. En effet, la décroissance des courbes de silhouette commence à des valeurs de k plus élevées plus r augmente. Les maximums de silhouette se situent en $k = 2$, $k = 20$ et $k = 20$ pour $r = 0$, $r = 5$ et $r = 10$ respectivement. La valeur $k = 2$ correspond au premier maximum local observé dans le k-means centralisé. Toutefois, $k = 20$ est en dehors du second maximum local de la figure 4.1b.

Pour les données non i.i.d. de UNSW-NB15, la sous-figure A.1d présente des allures de courbes ayant des similarités avec le cas i.i.d.. En effet, on observe comme dans le cas i.i.d. un maximum local en $k = 2$ pour tous les r, une décroissance suivie d'une nouvelle croissance de la silhouette. La silhouette atteint une zone de maximum local puis décroît. Autre point de similarité, les courbes s'étendent vers les plus grandes valeurs de k par rapport au k-means centralisé. Le maximum de la silhouette est situé en $k = 2$, $k = 33$ et $k = 2$ pour $r = 0$, $r = 5$ et $r = 10$ respectivement. Avec les variations de la méthode, le maximum peut se trouver soit en $k = 2$, soit autour du second maximum local. Dans ce cas si on exclut $k = 2$, on obtient $k = 15$, $k = 33$ et $k = 34$ pour $r = 0$, $r = 5$ et $r = 10$ respectivement. Ainsi, seul le cas $r = 0$ se rapproche en termes de nombre de clusters du cas centralisé.

A.1.2 Hyperparamètres sélectionnés

Le tableau A.1 présente la valeur choisie des paramètres k et r de la méthode. Le nombre de *rounds* des communications r est fixé à 0 pour tous les cas. En effet, la méthode semble assez stable avec un nombre de *rounds* minimal. De plus, minimiser r revient à limiter le nombre de communications entre le serveur et les clients, ce qui est bénéfique dans l'objectif de confidentialité des données. Les valeurs de k sont les maximums de la courbe de silhouette, sauf pour la distribution non i.i.d. de UNSW-NB15. En effet, nous avons choisi le maximum local différent de 2 pour avoir de meilleures performances. Ce choix pourrait très bien être fait dans un cas réel, car le compromis entre les efforts d'analyse et les performances a déjà été formulé dans la section 4.2.2.

Jeu de données	Distribution des données	k	r
CIC-IDS2017	i.i.d.	49	0
CIC-IDS2017	non i.i.d.	83	0
UNSW-NB15	i.i.d.	13	0
UNSW-NB15	non i.i.d.	15	0

Tableau A.1 Bilan des hyperparamètres (k-means fédéré)

A.2 Meta k-means

A.2.1 Résultats

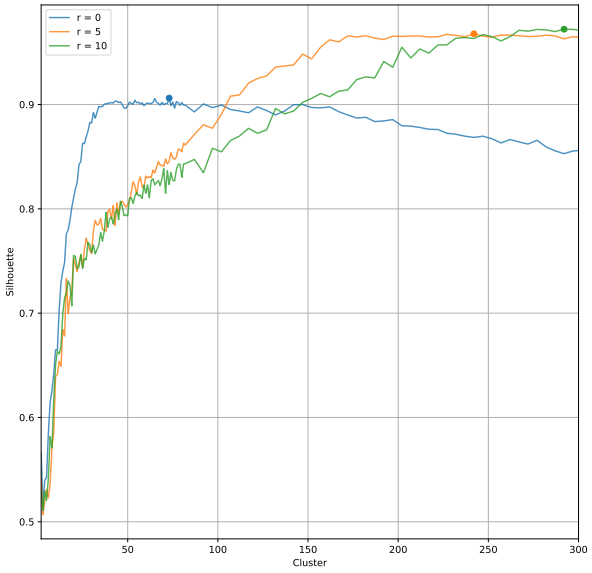
La figure A.2 présente l'évolution de la silhouette en fonction de k sur les deux jeux de données étudiés et dans les configurations de clients i.i.d. et non i.i.d. avec le meta k-means.

CIC-IDS2017 La figure A.2a présente l'évolution de la silhouette sur le jeu de données CIC-IDS2017 avec une répartition de données i.i.d. entre les clients. On remarque que la courbe de silhouette pour $r = 0$ est croissante puis décroissante. Ceci est un comportement assez typique de la silhouette. Le maximum de cette courbe se situe en $k = 73$. C'est un nombre important de clusters en comparaison avec les autres méthodes fédérées et le k-means centralisé. Les courbes pour $r = 5$ et $r = 10$ ne décroissent à aucun moment dans l'intervalle calculé. Au contraire, les courbes atteignent un plateau maximal. On pourrait suggérer de calculer la silhouette sur un plus grand intervalle. Toutefois, on sait de par le k-means fédéré et le k-means que le nombre optimal de clusters n'est pas censé se trouver au delà de 300 pour CIC-IDS2017.

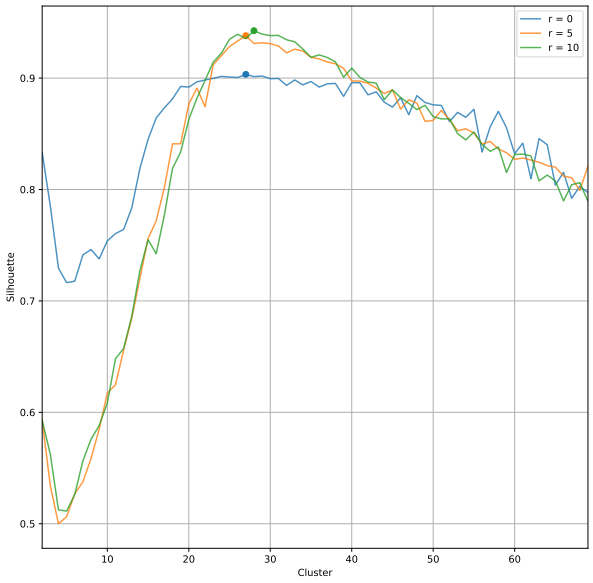
Dans le cadre non i.i.d., le modèle semble ne pas être capable de déterminer correctement le nombre de clusters. En effet, pour CIC-IDS2017 en figure A.2c, on constate que pour toutes les valeurs de r, la courbe ne décroît jamais en tendance et atteint un plateau de silhouette.

UNSW-NB15 Dans le cas du jeu de données UNSW-NB15 avec des clients i.i.d. (figure A.2b), on remarque que l'allure des courbes a des similarités avec celles du k-means centralisé sur les mêmes données (figure 4.1b). En effet, on constate tout d'abord un maximum local de silhouette en $k = 2$ puis une décroissance. Ensuite, la silhouette croît jusqu'à son maximum global compris entre $k = 25$ et $k = 30$. La silhouette décroît ensuite en tendance. Contrairement aux résultats sur CIC-IDS2017, on remarque que les courbes pour $r = 5$ et $r = 10$ ont des tendances plus marquées. En effet, la différence entre le minimum vers $k = 5$ et le maximum entre $k = 25$ et $k = 30$ ainsi que la rapidité de la décroissance ensuite sont plus importants que pour $r = 0$. Toutefois, le modèle avec $r = 0$ trouve tout de même un optimum de silhouette convenable.

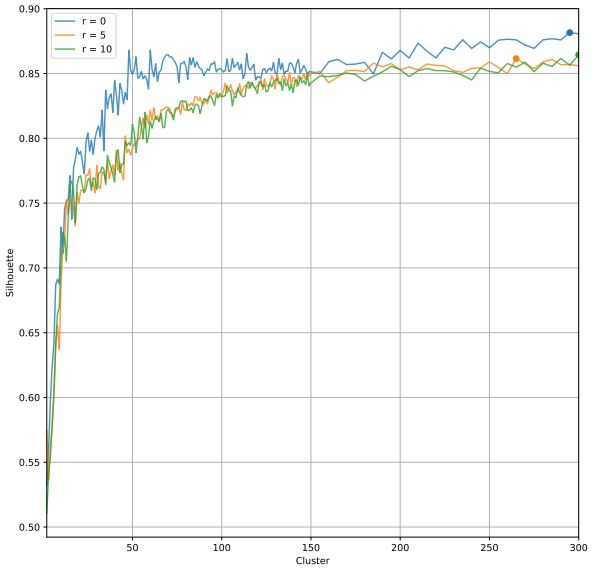
Pour UNSW-NB15 (figure A.2d) avec une configuration non i.i.d., la courbe $r = 0$ s'arrête avant 70 parce que les clients n'ont pas envoyé au serveur assez de points pour réaliser le clustering. Ainsi, le maximum de silhouette se trouve nécessairement à la taille du jeu de données du serveur *i.e.* $k = 24$ dans cet exemple. En soi, $k = 24$ est un bon nombre de cluster quand on compare avec les modèles fédérés précédents. Toutefois, avec $r = 5$ et $r = 10$ on constate que la silhouette n'est que croissante en tendance et donc que le maximum de



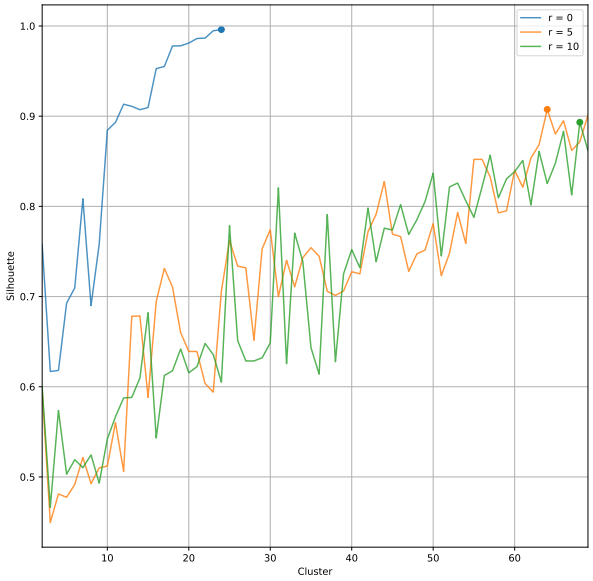
(a) CIC-IDS2017 i.i.d.



(b) UNSW-NB15 i.i.d.



(c) CIC-IDS2017 non i.i.d.



(d) UNSW-NB15 non i.i.d.

FIGURE A.2 Évolution de la silhouette en fonction du nombre de clusters sur le serveur (meta k-means)

silhouette se trouve environ en fin d'intervalle observé.

Le modèle de meta k-means semble donc ne pas bénéficier d'un nombre de *rounds* de communications supérieur à $r = 0$ pour les résultats présentés. Seul le cas des données i.d.d. sur UNSW-NB15 indique une amélioration pour $r > 0$. Or, les résultats pour $r = 0$ sont satisfaisants. De plus, les *rounds* de communications supplémentaires ajoutent en instabilité dans le choix du nombre optimal de clusters. Finalement, le meta k-means en l'état ne semble pas robuste relativement aux données non i.i.d. comme sur CIC-IDS2017.

A.2.2 Hyperparamètres sélectionnés

Le tableau A.2 synthétise les choix d'hyperparamètres pour les différents jeux de données étudiés et les répartitions de données. Pour toutes les configurations, on choisit $r = 0$ car c'est ce qui semble être le plus stable et obtenir les meilleures allures de courbe de silhouette. Le nombre de clusters choisi est celui où la courbe est à son maximum pour chacune des configurations.

Jeu de données	Distribution des données	k	r
CIC-IDS2017	i.i.d.	73	0
CIC-IDS2017	non i.i.d.	295	0
UNSW-NB15	i.i.d.	13	0
UNSW-NB15	non i.i.d.	24	0

Tableau A.2 Bilan des hyperparamètres (meta k-means)

ANNEXE B PERFORMANCES SELON LE NOMBRE DE *ROUNDS* DE COMMUNICATION

Cette annexe B présente l'évolution du score F_1 en fonction du nombre de clusters k au niveau du serveur. Les courbes de score F_1 en fonction du nombre de *rounds* r de communication ont été ajoutées pour comprendre l'influence de ce paramètre. L'analyse qui suit a été insérée dans les annexes pour éviter d'alourdir le corps du mémoire et pour éviter les répétitions avec la section 5.3.

K-means fédéré La figure B.1 présente l'évolution du score F_1 du modèle k-means fédéré pour les jeux de données CIC-IDS2017 et UNSW-NB15 dans les configurations i.i.d. et non i.i.d..

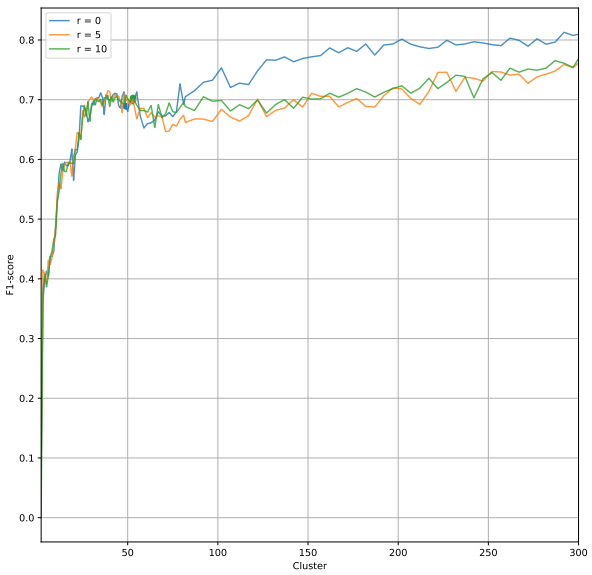
Les courbes des sous-figures pour tout r sont croissantes en tendances. Ceci correspond à l'analyse faite pour le k-means centralisé 4.2.2.

On remarque dans l'ensemble des sous-figures que la courbe bleue ($r = 0$) est soit très proche des courbes pour $r > 0$ soit nettement supérieure. Ainsi, même si la silhouette ne semble pas présenter de maximum clair, les performances semblent se comporter de la même manière.

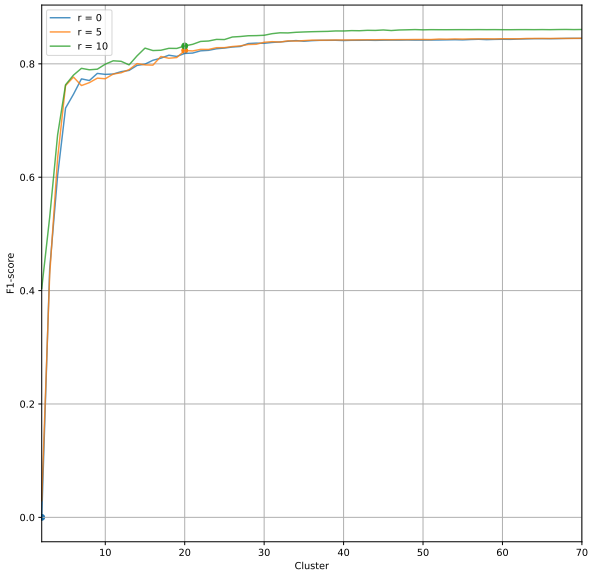
K-means fédéré avec initialisation k-means++ fédérée Les commentaires formulés dans le paragraphe précédent sont là aussi valables. Ils sont même d'autant plus vrais quand on observe les courbes $r = 0$ comparées aux courbes $r > 0$ des sous-figures B.2a, B.2c et B.2d. En effet, la courbe bleue se démarque clairement des autres courbes ou bien en est très proche.

Meta k-means Pour le meta k-means, on remarque les mêmes caractéristiques que pour les autres modèles. Les courbes pour tout r et pour toute distribution de données sont croissantes. La courbe pour $r = 0$ est supérieure aux autres pour les distributions i.i.d. de données. Toutefois dans les expérimentations avec des données non i.i.d., on note que la courbe pour $r = 0$ est proche ou inférieure aux autres. Ainsi, il est moins clair que choisir $r = 0$ est meilleur en termes de performances.

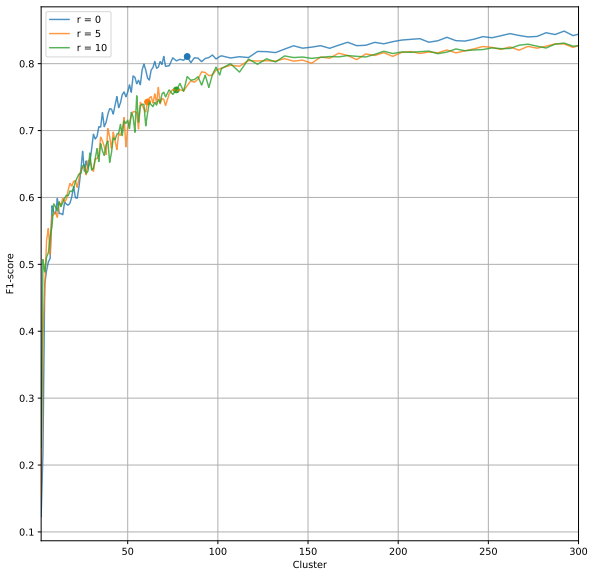
Discussions Ainsi, on constate que choisir le nombre minimal de communications entre les clients et le serveur permis par la méthode est souvent un choix judicieux. De plus, on



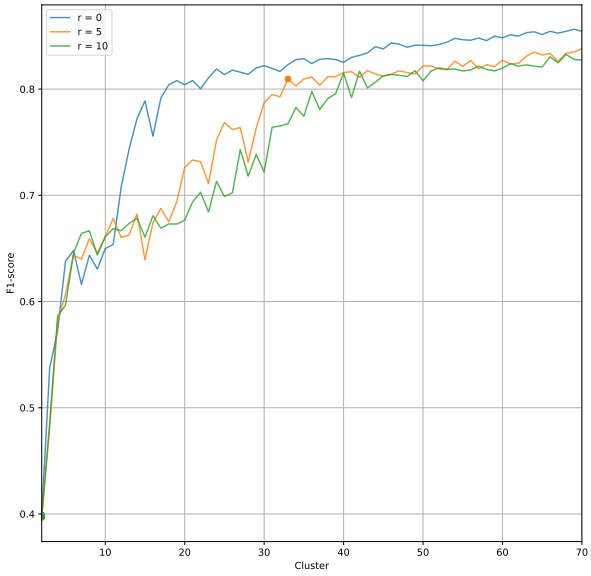
(a) CIC-IDS2017 i.i.d.



(b) UNSW-NB15 i.i.d.

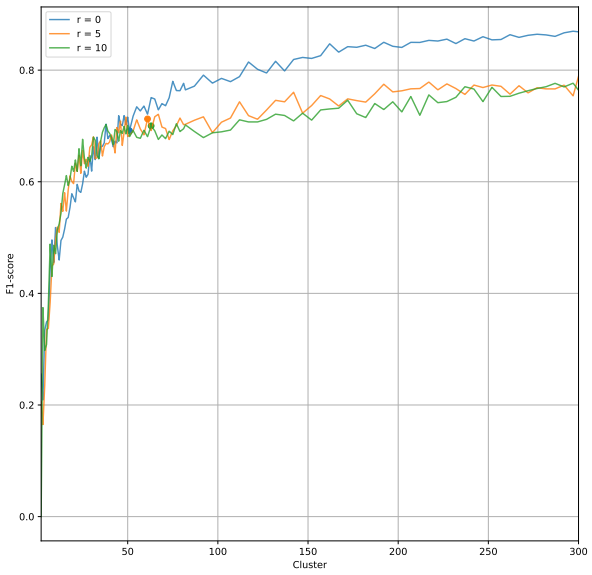


(c) CIC-IDS2017 non i.i.d.

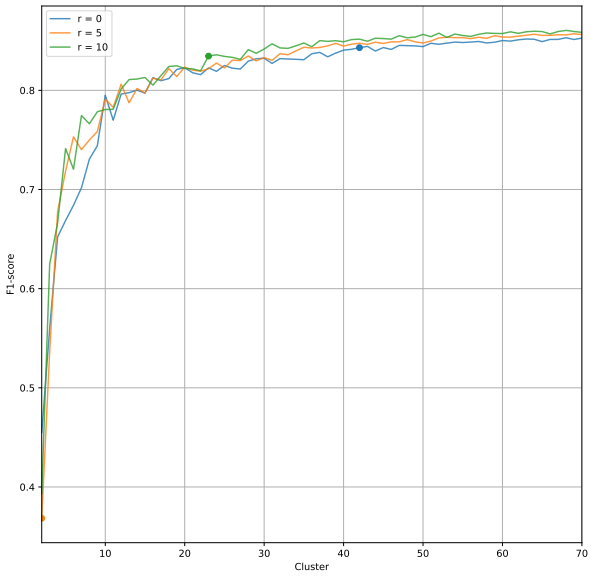


(d) UNSW-NB15 non i.i.d.

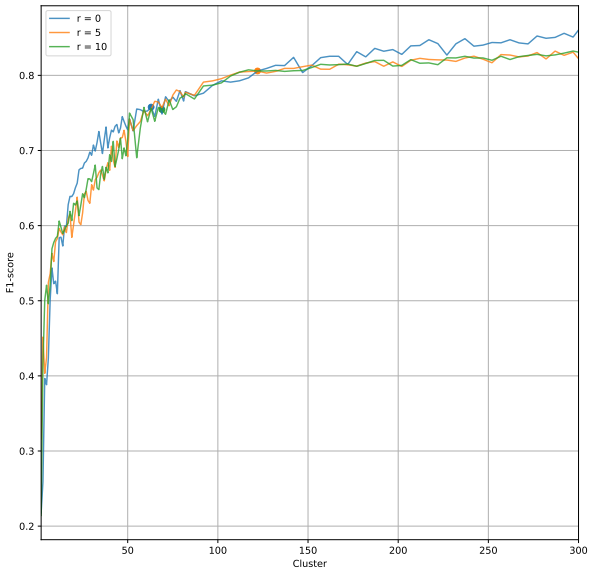
FIGURE B.1 Évolution de la f1 en fonction du nombre de clusters sur le serveur (k-means fédéré)



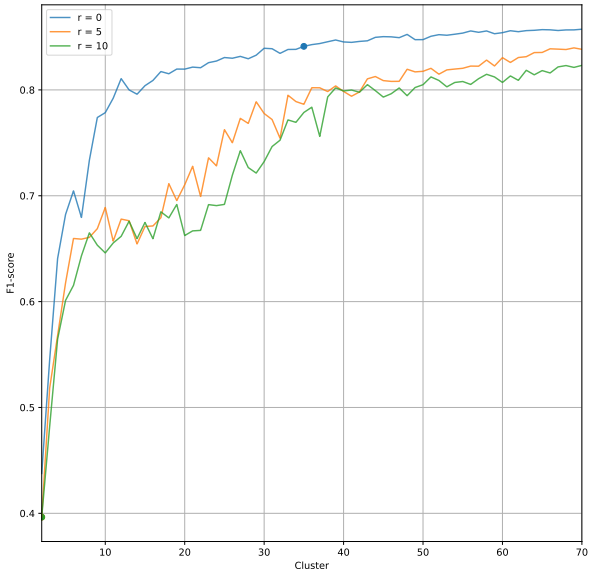
(a) CIC-IDS2017 i.i.d.



(b) UNSW-NB15 i.i.d.

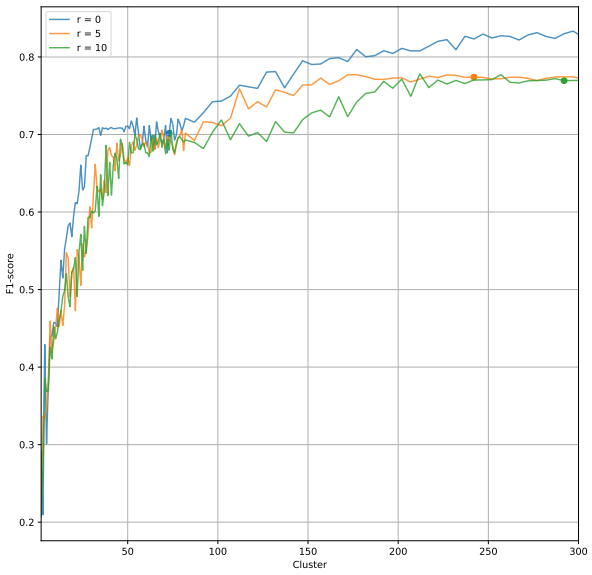


(c) CIC-IDS2017 non i.i.d.

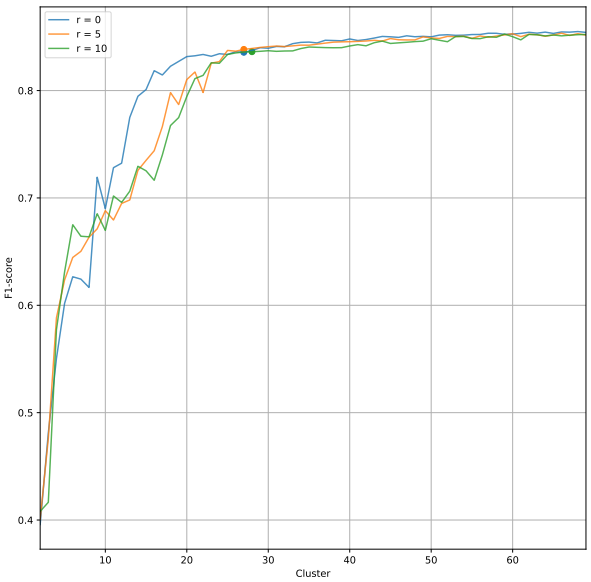


(d) UNSW-NB15 non i.i.d.

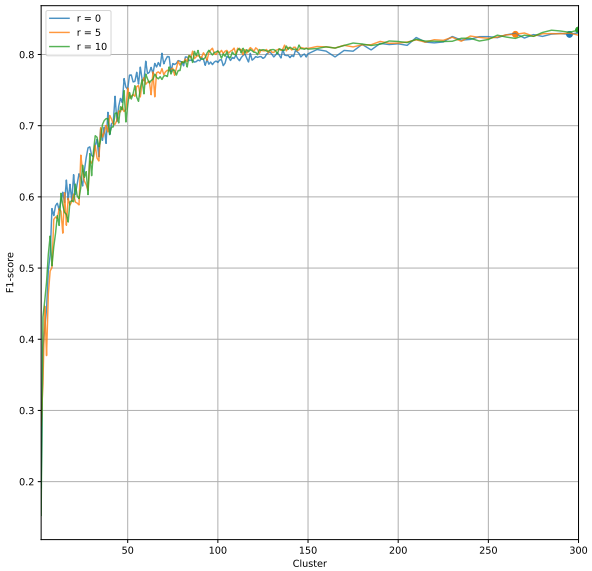
FIGURE B.2 Évolution de la f1 en fonction du nombre de clusters sur le serveur (initialisation k-means++ fédérée)



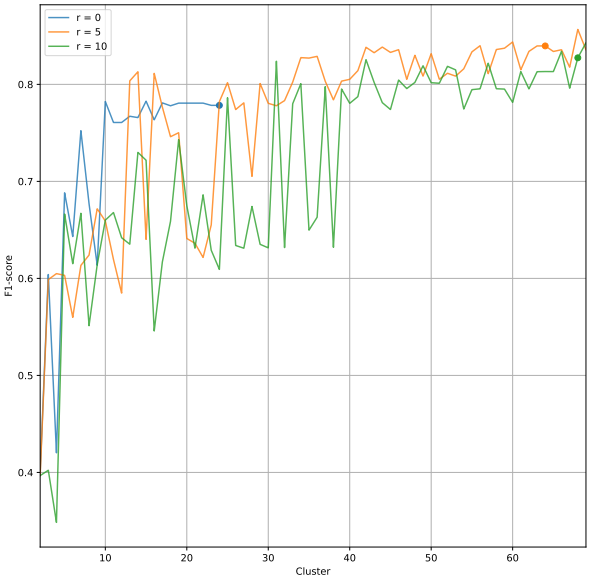
(a) CIC-IDS2017 i.i.d.



(b) UNSW-NB15 i.i.d.



(c) CIC-IDS2017 non i.i.d.



(d) UNSW-NB15 non i.i.d.

FIGURE B.3 Évolution de la f1 en fonction du nombre de clusters sur le serveur (meta kmeans)

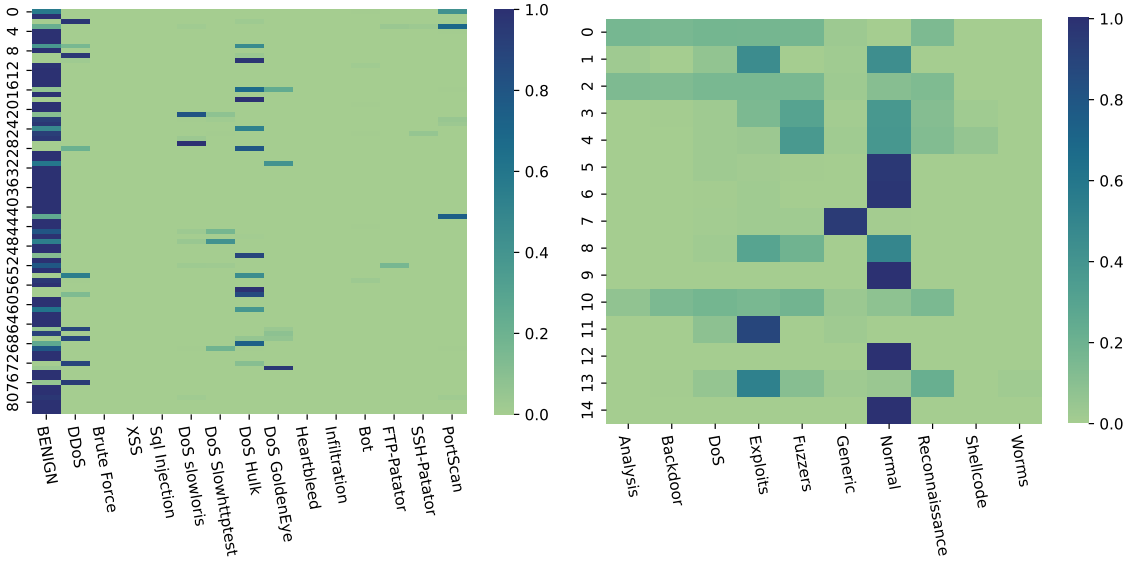
retrouve le compromis entre les efforts d'analyses et les performances souligné dans la partie 4.2.2.

Toutefois, cette analyse n'est possible que parce que dans le cadre de cette étude nous disposons des classes des communications. Or dans le cas d'usage décrit en introduction, les communications ne sont pas étiquetées *a priori* et on classe seulement les clusters. Ainsi, les évolutions des performances présentées dans cette annexe ne pourraient pas être facilement calculées en pratique.

ANNEXE C PROPORTIONS DE COMMUNICATIONS DANS LES CLUSTERS DES MÉTHODES FÉDÉRÉES

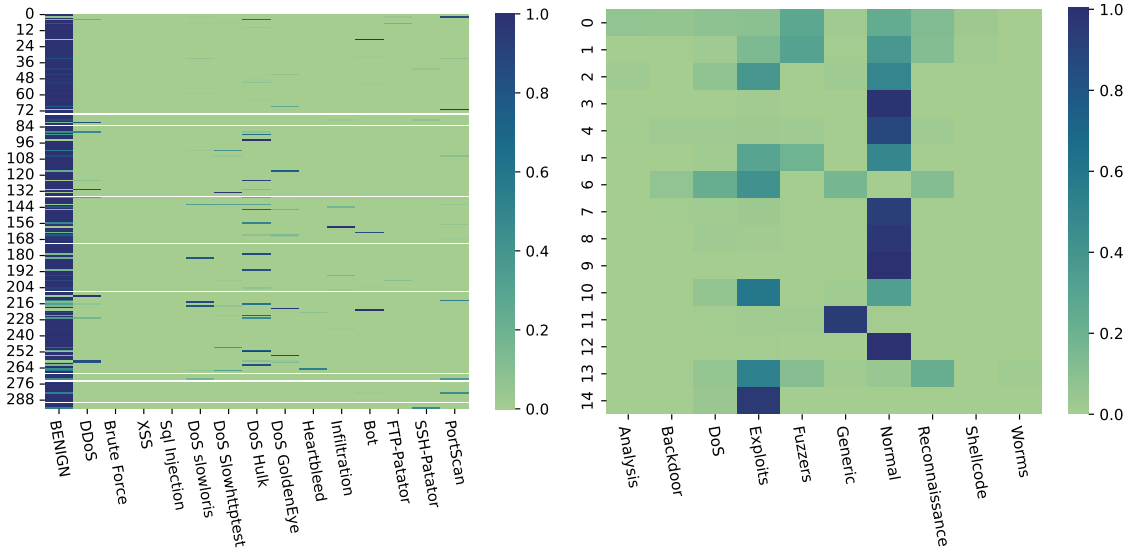
Les matrices présentées dans cette annexe permettent d'arriver aux mêmes conclusions que celles décrites plus en longueur dans la section 5.3.2. Les *heatmaps* présentées dans cette annexe C sont calculées sur des données non i.i.d. avec les hyperparamètres des modèles déterminés dans l'annexe A. Les lignes blanches de la figure C.2a sont celles où aucun client n'a voté. Ainsi, on remarque que les méthodes fédérées peuvent potentiellement générer des clusters vides. De fait, la méthode en question conseille trop de clusters et *a priori*, l'agrégation n'est pas équivalente au cas centralisé. Il n'y a donc pas de raison que tous les clusters aient des données chez les clients.

On note toutefois que pour les résultats du k-means fédéré sur CIC-IDS2017 avec des données non i.i.d. (fig. C.1a), tous les clusters ont une classe en majorité absolue. Ainsi, les commentaires traitant de la perte de performances lors du passage de un à deux tours lors de la classification multi-classes ne s'appliquent pas.



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

FIGURE C.1 Proportion de communications dans chaque cluster (k-means fédéré)



(a) Résultats sur CIC-IDS2017 (b) Résultats sur UNSW-NB15

FIGURE C.2 Proportion de communications dans chaque cluster (k-means fédéré)