UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

APPRENTISSAGE PROFOND DISTRIBUÉ SÉCURISÉ : APPLICATION À LA DÉTECTION DE FRAUDES BANCAIRES SUR INTERNET

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN SCIENCES ET TECHNOLOGIES DE L'INFORMATION

PAR

JÉSUTIN JONAS AVOCE

NOVEMBRE 2021

Ce mémoire a été évalué par un jury composé des personnes suivantes :

Prof. Omer Landry Nguena Timo . . . . . Président du jury

Prof. Mohand Saïd Allili . . . . . Membre du jury

Prof. Kamel Adi . . . . . Directeur de recherche

Mémoire accepté le : 26 novembre 2021

A mon père Félix

A ma mère Denise et mon beau-père Eugène

Remerciements

Ce travail de mémoire est réalisé au Laboratoire de Recherche en Sécurité Informatique (LRSI) de l'université de Québec en Outaouais sous la direction bienveillante du Professeur Kamel Adi. La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

D'abord, mes sincères remerciements vont à l'endroit de Monsieur Kamel Adi, professeur au département d'informatique de l'université de Québec en Outaouais, pour son accueil dans son équipe et pour la direction rigoureuse de ce travail de recherche. Aussi, je lui exprime ma reconnaissance pour sa disponibilité, sa patience et surtout ses judicieux conseils, et remarques aussi pertinentes que constructives.

Je désire aussi remercier Madame Myria Bouhaddi, Doctorante au département d'informatique de l'université de Québec en Outaouais pour m'avoir fait bénéficié de ses compétences et pour avoir lu et corrigé mon rapport.

Je tiens à témoigner toute ma gratitude à Monsieur Omer Landry Nguena, professeur au département d'informatique de l'université de Québec en Outaouais pour avoir accepté de présider mon jury de mémoire et de sacrifier une partie de son temps dans l'évaluation scientifique du présent travail.

J'aimerais exprimer ma gratitude à Monsieur Mohand Saïd Allili, professeur au département d'informatique de l'université de Québec en Outaouais pour son cours de techniques d'apprentissages qu'il m'a enseigné et l'honneur qu'il m'a fait en acceptant d'être évaluateur du présent travail.

Enfin, j'adresse mes remerciements les plus chers à mon amie Faridath et mon frère Aymeric qui m'ont fortement soutenu tout au long de ce mémoire.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations, sigles et acronymes

MVS Machine à vecteurs de support.

MPI Message Passing Interface.

ACP Analyse en Composantes Principales.

SGD Stochastic Gradient Descent.

DP-SGD Differentially Private Stochastic Gradient Descent.

NCCL The NVIDIA Collective Communications Library.

GPU Graphics Processing Unit.

KNN K-Nearest Neighbors.

DP Differencial Privacy.

CDG Confidentialité Différentielle Globale.

CDL Confidentialité Différentielle Locale.

RBM Restricted Boltzmann Machine.

IA Intelligence Artificielle.

KDD Knowledge Discovery in Databases.

RUS Random Under-Sampling.

ANN Artificial Neural Network.

CFLNN Chebyshev Functional Link Neural Network.

MLP MultiLayer Perceptron.

CART Classification And Regression Tree.

CAC Centre Antifraude du Canada.

CFR Caractéristique de Fonctionnement du Récepteur.

ASC Aire Sous la Courbe.

MRE Minimisation du Risque Empirique.

IOT Internet des Objets.

P2P Peer-to-Peer.

MLaaS Machine Learning as a Service.

SMOTE Synthetic Minority Over-sampling Technique.

GAD Graphique Acyclique Dirigé.

API Application Programming Interface.

APDDS Apprentissage Profond Distribué Sécurisé.

MER Moyenne des Erreurs de Reconstruction.

Résumé

Le commerce électronique a pris une grande ampleur ces dernières décennies. Ceci a eu pour conséquence, l'apparition de nouvelles formes de fraudes dans les transactions bancaires. L'impact de ces fraudes, de diverses gravités, sur les banques et le mode opératoire des fraudeurs s'amplifient et se diversifient au fil des ans. Pour pouvoir détecter de manière automatique les fraudes bancaires sur Internet, plusieurs chercheurs ont proposé des approches basées sur l'apprentissage automatique. Malheureusement, pour des raisons de conflit d'intérêts et de confidentialités de l'information véhiculée par les données, les modèles d'apprentissage automatique pour la détection des fraudes bancaires sont généralement entraînés en utilisant uniquement les données d'apprentissage propres à chaque institution financière. En conséquence, les modèles obtenus manquent de précisons, notamment pour l'application de techniques d'apprentissage profond qui nécessitent des ensembles de données volumineux et diversifies. Pour faire face à cette problématique, nous proposons dans ce travail de recherche une approche basée sur l'apprentissage profond distribué garantissant la confidentialité des données d'apprentissage et permettant, ainsi, la participation de plusieurs organismes financiers dans la construction du système de détection de fraudes.

Abstract

Electronic commerce, has grown in popularity in recent decades. This has resulted in the emergence of new forms of fraud in transactions banking. The impact of these frauds, of various severities, on banks and the modus operandi of fraudsters have grown and diversified over the years. In order to be able to detect automatic banking fraud on the Internet, several researchers have proposed based on machine learning. Unfortunately, for reasons of conflict of interest and confidentiality of information conveyed by data, learning models automatic bank fraud detection are generally trained using only learning data specific to each financial institution. Consequently, the models obtained lack precision, in particular for the application of techniques learning programs that require large and diverse data sets. For to face this problem, we propose in this research an approach based on distributed deep learning ensuring data confidentiality learning and thus allowing the participation of several financial organizations in the construction of the fraud detection system.

Chapitre 1

Introduction

1.1 Introduction générale

De nos jours, la fraude sur Internet coûte chère aux organismes financiers, notamment au Canada où 18 533 personnes ont été victimes de fraudes en 2020, ce qui a engendré 67,2 M $ en perte financière selon le bilan du Centre Antifraude [7]. En outre, Radio-Canada a noté une augmentation de 84 % de cas de fraude entre 2018 et 2019 et parmi lesquels, le « carding » occupe une place de choix [7]. Le « carding » consiste à voler et à utiliser de façon illégale les informations sur les cartes bancaires, les comptes bancaires et bien d'autres informations financières. Les fraudeurs qui pratiquent le « carding » utilisent souvent des sites web d'hameçonnage pour inciter les utilisateurs à révéler leurs informations financières. Les victimes peuvent recevoir de faux courriels électroniques ressemblant à ceux envoyés par des entreprises de bonne réputation. L'objectif étant de faire en sorte que la victime révèle son numéro de carte de crédit, la date d'expiration de celle-ci et d'autres renseignements sensibles. Par ailleurs, les « carders » utilisent des forums spécialisés pour échanger les informations subtilisées (numéros des cartes de crédit, de débit et autres) afin de commettre toutes sortes de fraudes bancaires.

Étant donné que les pertes financières causées par les fraudes bancaires ont fortement augmenté ces dernières années avec le développement des nouvelles techniques de fraudes, il devient urgeant de trouver une solution pragmatique et efficace pour lutter contre ce phénomène. En effet, la détection de la fraude implique l'identification précise de la fraude aussi rapidement que possible après que celle-ci ait été lancée sur Internet.

Récemment, plusieurs solutions basées sur l'Intelligence Artificielle et l'apprentissage automatique ont été élaborées pour la détection automatique des fraudes bancaires. Ces modèles

basés sur l'apprentissage automatique, notamment ceux basés sur« l'apprentissage profond», ont besoin d'une quantité importante de données de bonne qualité pour être performants. L'obtention de telles données nécessite, cependant, une collecte impliquant plusieurs institutions financières. Malheureusement, la coordination d'études collaboratives à grande échelle devient difficile à cause des restrictions et des obstacles liés à la confidentialité des données détenues par chaque institution. En effet, les informations sur les transactions financières sont considérées comme sensibles tant dans leurs relations avec la vie privée des clients que dans leur importance en tant que source d'informations exclusives pour les banques. Par ailleurs, pour des raisons de conflit d'intérêts et de concurrence du secteur financier, les sociétés de cartes de crédit et les banques hésitent à partager leurs données exclusives entre elles ou avec un référentiel central. En conséquence, les modèles d'apprentissage automatique pour la détection des fraudes sont généralement entraînés en utilisant uniquement les données d'apprentissage propres à chaque institution financière et ceci engendre, souvent, des systèmes moins précis et donc moins fiables. Aussi, deux types d'attaques de rétro-ingénierie peuvent être réalisées à partir d'un modèle entraîné : l'attaque d'appartenance et l'attaque par reconstruction. L'attaque d'appartenance consiste à deviner efficacement si une donnée particulière a été utilisée ou non durant la phase d'entraînement d'un modèle. L'attaque par reconstruction consiste à reconstruire une donnée d'entraînement ou certains de ses attributs. De telles attaques peuvent être conduites soit en disposant des paramètres complets du modèle (attaque de boîte blanche) soit en disposant seulement d'un accès au modèle par des requêtes d'inférences (attaque de boite noire). La mise en œuvre potentielle de ces deux types d'attaques porte atteinte à la confidentialité des données d'entraînement et ouvre la voie à des détournements incompatibles avec les exigences de protection des données bancaires.

Une approche possible pour répondre à ces enjeux est celle de la confidentialité différentielle combinée au calcul distribué. Ainsi, nous proposons une solution basée sur l'apprentissage distribué permettant de protéger les données bancaires à caractère personnel contre le risque de ré-identification en assurant une confidentialité différentielle des données d'apprentissage tout en maintenant la pertinence des résultats du modèle et permettant ainsi de préserver la vie privée des clients des banques participantes.

1.2 Objectifs et contribution

Dans ce travail de recherche, nous évoquons le problème de détection efficace des fraudes bancaires sur Internet et du maintien de la confidentialité des données d'entraînement utilisées

dans la construction d'un modèle.
L'objectif de notre travail consiste premièrement à présenter notre approche d'apprentissage profond distribué sécurisé. Cela inclus l'architecture et le mécanisme de confidentialité différentielle choisi. Deuxièmement, nous appliquerons notre approche à la détection de fraude bancaire sur Internet. À cet égard, nous présentons le DataSet utilisé et les étapes du prétraitement des données effectuées pour évaluer la performance de l'approche proposée, suivis d'une discussion des résultats obtenus. Cependant, il est utile de comparer la performance en précision et niveau de préservation de confidentialité de notre approche avec une technique d'apprentissage profond supervisé. Cette comparaison va démontrer que notre approche préserve mieux la confidentialité des données tout en offrant une très bonne précision.

Essentiellement, voici comment se structure la suite du document : le second chapitre vise à l'introduction des concepts du domaine de l'apprentissage automatique. Premièrement, il donne une vue générale sur l'apprentissage automatique via ses deux sous-domaines principaux, l'apprentissage supervisé et non supervisé tout en mettant un accent particulier sur la technique de détection d'anomalies avec l'apprentissage automatique. Deuxièmement, il présente les architectures d'apprentissage, quelques techniques d'améliorations et métriques importantes de mesures de performances des modèles d'apprentissage automatique.
Le troisième chapitre explique premièrement les concepts, propriétés de la confidentialité, la notion de sensibilité et puis quelques mécanismes différenciellement privés. Deuxièmement, il met un accent particulier sur l'utilisation de la confidentialité différentielle dans les modèles d'apprentissage automatique. Le quatrième chapitre expose un état de l'art synthétisé sur les sujets en relation avec l'apprentissage automatique et la confidentialité différentielle. De plus, nous avons réalisé des analyses critiques permettant de motiver et de justifier notre approche.
Quant au cinquième chapitre, il présente premièrement notre mécanisme basé sur l'apprentissage profond distribué et la confidentialité différentielle. Plus précisément, il présente l'architecture globale de notre système et le mécanisme de confidentialité différentielle choisi. Deuxièmement, il présente l'application de notre approche à la détection de fraude bancaire sur Internet. À cet égard, il présente le DataSet utilisé, le processus du prétraitement des données effectué et les résultats obtenus du classificateur basé sur notre approche. Nous comparerons également nos résultats à ceux obtenus du classificateur basé sur l'apprentissage profond supervisé. Enfin, nous terminerons par une conclusion tout en relevant certaines limites de cette recherche.

Chapitre 2

Apprentissage automatique

Introduction

Des quantités énormes de transactions bancaires sont produites chaque jour par les banques et les compagnies de cartes de crédit. À cause de leur vélocité et leurs volume important, l'analyse de ces données dépassent largement les capacités humaines et les outils de traitement traditionnels. Pour cette raison, l'apprentissage automatique a rencontré un vif succès dans l'analyse de données, permettant ainsi, la détection de motifs et de règles d'associations. Les résultats produits par ces analyses sont alors utilisés pour la prédiction et la prise de décision. L'apprentissage automatique est la discipline donnant aux ordinateurs la capacité d'apprendre sans qu'ils soient explicitement programmés pour cela. Il consiste à permettre à l'ordinateur d'apprendre à partir d'exemples afin d'acquérir la capacité de pouvoir généraliser et faire, ainsi, des prédictions sur des données nouvellement observées. Il existe deux approches principales pour l'apprentissage automatique : l'apprentissage supervisé, qui entraîne l'algorithme sur des ensembles de données étiquetées au préalable et l'apprentissage non supervisé, qui utilise des données non étiquetées pour permettre l'apprentissage des motifs « patterns » et des relations dans ces données. Dans ce qui suit, nous explorons différentes notions liées à l'apprentissage automatique plus en détails.

D'abord, nous expliquerons le fonctionnement des algorithmes d'apprentissage automatique classique et de l'apprentissage profond qui sont les plus utilisés pour la détection de fraudes bancaires sur Internet ainsi que la technique de détection d'anomalies avec l'apprentissage automatique. Ensuite, nous présenterons le principe de Minimisation du Risque Empirique, les techniques d'apprentissage par la descente de gradient et les architectures d'apprentissage. Enfin, nous présenterons quelques techniques d'améliorations de performances

et quelques métriques importantes de mesures de performances des modèles d'apprentissage automatique.

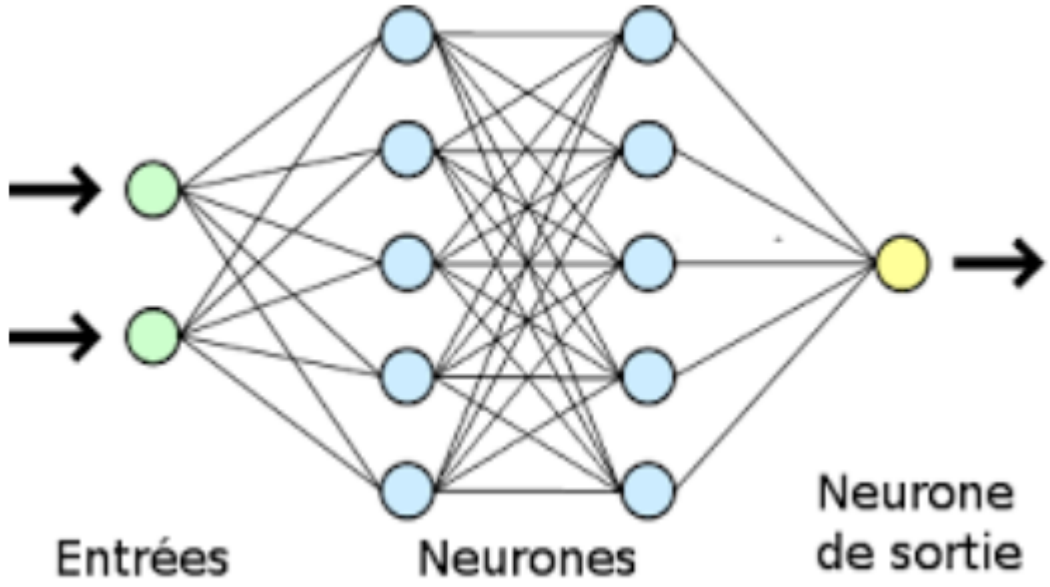
2.1 Apprentissage supervisé

Dans l'apprentissage supervisé, les données d'entrée sont étiquetées par leurs classes respectives. Il est couramment utilisé pour prédire des événements futurs statistiquement probables à partir des données historiques. Le but principal de cette approche est que l'algorithme puisse apprendre en comparant ses sorties réelles avec les sorties enseignées pour trouver des erreurs et modifier ainsi le modèle de prédiction produit par adaptation. La régression et la classification sont les deux méthodes les plus utilisées dans cette approche. On parle d'une régression quand la variable prédite est une valeur réelle tandis qu'il s'agit d'une classification lorsque la variable prédite est discrète. Nous expliquerons plus en détails dans cette partie les algorithmes d'apprentissage qui sont les plus utilisés pour la détection de fraude.

2.1.1 Réseaux de neurones artificiels

Les réseaux de neurones artificiels, organisés en couches, sont inspirés de la structure et du raisonnement du cerveau humain. Ils forment une structure composée d'une succession de neurones et de connexions. Les connexions transmettent l'information entre les neurones qui, pour leur part, procèdent à des calculs à partir des informations reçues comme le montre la FIGURE 2.1 [32]. L'information se transmet dans le sens des connexions, à partir des neurones observés qui correspondent aux données jusqu'à la prédiction. En outre, les réseaux de neurones peuvent être utilisés pour faire de l'apprentissage supervisé, c'est-à-dire que le résultat est déjà connu pour une transaction donnée. De même, ils peuvent être également utilisés pour faire de l'apprentissage non supervisé [50].

FIGURE 2.1 – Structure d'un réseau de neurone



2.1.2 Arbre de décision

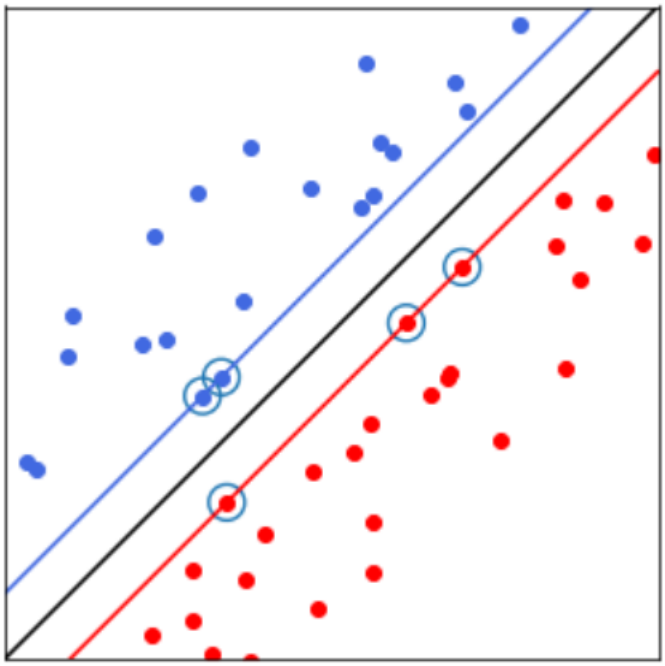
Un arbre de décision est un algorithme utilisé pour la classification et la prédiction. Il offre une représentation arborescente, où chaque noeud interne correspond à un test sur un attribut, chaque branche dénote un résultat de ce test et chaque noeud terminal correspond à une classe. Les arbres de décision permettent de partitionner, récursivement, un ensemble de données en utilisant une approche en profondeur ou en largeur et s'arrêtent lorsque toutes les transactions ont été affectées à une classe particulière [77].

2.1.3 Machine à vecteurs de support (MVS)

La MVS est un algorithme d'apprentissage supervisé qui sépare un ensemble de données en différentes classes à l'aide d'un hyperplan en s'appuyant sur la notion de marge maximale. Les points les plus proches de l'hyperplan dans les différentes classes sont appelés vecteurs de support et ces vecteurs de support sont utilisés pour prédire les classes des nouveaux points. En effet, lorsque qu'un nouveau point est placé sur l'équation de l'hyperplan, il est classé

dans une classe en fonction du côté de l'hyperplan où il est tombé sur l'espace vectoriel [63]. La FIGURE 2.2 [37] montre le principe général de MVS. De cette FIGURE, l'hyperplan est la droite noire, les « vecteurs de support » sont les points entourés (les plus proche de l'hyperplan) et la « marge » est la distance entre l'hyperplan et les droites bleue et rouge[37].

FIGURE 2.2 – Fonctionnement de MVS



2.1.4 Régression logistique

Pour faire face aux anomalies de la régression linéaire qui donne des valeurs supérieures à 1 et inférieures à 0, on fait recours à la régression logistique. Malgré le nom de régression, la régression logistique est utilisée pour les problèmes de classification. Par définition, la régression logistique est un modèle statistique permettant d'étudier les relations entre un ensemble de variables qualitatives X et une variable qualitative Y. Au fond, elle prédit les résultats binomiaux et multinomiaux, dans le but d'estimer les valeurs des coefficients des paramètres à l'aide de la fonction sigmoïde [85].

2.1.5 K plus proches voisins

L'algorithme de k plus proches voisins (KPPV) est un algorithme de classification permettant d'affecter la nouvelle donnée d'entrée x à la cible la plus présente dans les k données les plus proches selon une distance prédéfinie. Cet algorithme est mieux adapté lorsque les frontières sont irrégulières, mais a des limitations liées aux ressources de mémoire et traitement nécessaires. Par ailleurs, l'algorithme de KPPV conserve toutes les données d'entraînement et l'évalue pour chaque donnée de test [20].

2.1.6 Forêt aléatoire

La forêt aléatoire également connue sous le nom de forêt d'arbres décisionnels constitue un ensemble d'arbres de décision construits aléatoirement et en parallèle avec un sous-ensemble de données distinctes. Chaque arbre de décision est entraîné sur un sous-ensemble aléatoire avec remise des données et une projection aléatoire avec remise d'un sous-ensemble d'attributs. La prédiction est basée sur un vote des arbres en cas de classification ou par la moyenne en cas de régression. La forêt aléatoire est robuste au bruit et aux données aberrantes et permet une meilleure généralisation. Elle supporte aussi les données nominales et numériques [77].

2.2 Apprentissage semi-supervisé

Dans de nombreux contextes du monde réel, la quantité de données étiquetées peut être nettement plus petite que celui des données non étiquetées, et il pourrait être trop coûteux d'obtenir des étiquettes de haute qualité. Les algorithmes d'apprentissage semi-supervisé visent à utiliser des données non étiquetées pour apprendre un niveau supérieur de représentations, puis utiliser les exemples étiquetés pour guider la tâche d'apprentissage en aval. Un exemple d'apprentissage semi-supervisé serait d'utiliser une technique d'apprentissage non supervisée en tant que clustering sur des données non étiquetées, puis utiliser un classificateur pour séparer les données d'entraînement représentatives de chaque cluster. D'autres exemples notables sont les modèles génératifs tels que les réseaux adverses génératifs [6].

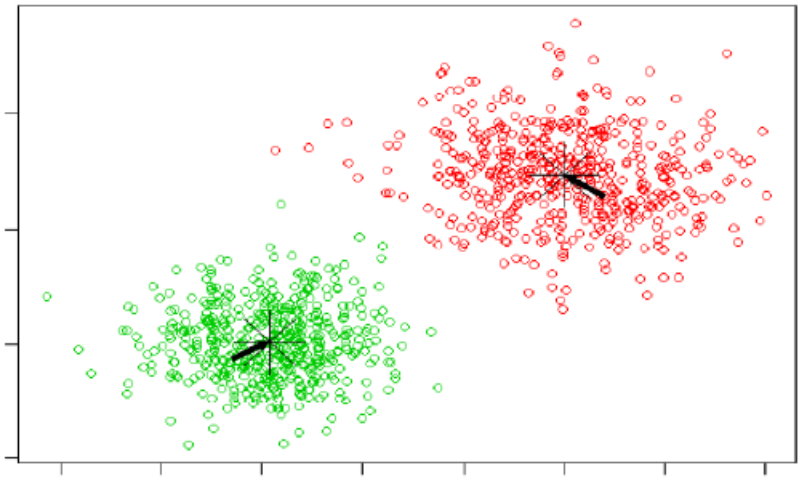
2.3 Apprentissage non supervisé

Dans le cadre de l'apprentissage non supervisé, les données d'apprentissage utilisées sont non labellisées. L'algorithme trouvant tout seul les cas de similarités parmi les données d'en-

trée. Son but est de découvrir des modèles cachés, des corrélations et les différences dans les données qui peuvent être utilisées pour la prise de décision. En outre, la réduction de dimension et le regroupement sont deux exemples classiques d'apprentissage non supervisé. En effet, la réduction de dimension extrait un plus petit nombre de caractéristiques, à partir des caractéristiques d'origine, qui décrivent les données de manière suffisante. Le regroupement quant à lui consiste à partitionner les données en groupes (clusters) de sorte que les objets de chaque groupe partagent certaines caractéristiques communes entre eux. Dans notre travail, nous allons nous focaliser que sur le clustering K-means qui est plus utilisé pour la détection de la fraude.

Le clustering K-means permet de partitionner un ensemble de données en K « clusters » avec une variation minimale au sein de chaque cluster et une variation maximale entre les clusters. L'idée consiste à initialiser le cluster K en sélectionnant au hasard K données comme centres des clusters. L'algorithme affecte alors chaque donnée de l'ensemble des données d'entrées au cluster dont le centre est le plus proche et met à jour les centres de chaque cluster en calculant la moyenne des données qui appartiennent aux clusters. Ce processus répète les étapes de relocalisation et de mise à jour jusqu'à ce que l'algorithme converge. L'algorithme K-means présente plusieurs avantages. En effet, il est facile à implémenter et permet aisément d'identifier des groupes de données inconnus à partir d'ensembles de données complexes. Ainsi, il peut être utilisé pour étiqueter des ensembles de données utilisées pour la classification. L'algorithme K-means s'adapte aux divers changements des données. En cas de souci, l'ajustement du segment de cluster permet d'apporter rapidement des modifications nécessaires à l'algorithme [30]. Cependant, la performance de l'algorithme K-means dépend du choix de la mesure de similarité et le bon nombre de clusters avec leur centre initial respectif. La FIGURE 2.3 tirée de [19] montre le résultat après l'application de K-means avec deux clusters.

FIGURE 2.3 – Résultat après l'application de k-means avec deux Clusters



2.4 Apprentissage et technique de détection d'anomalies

La détection d'anomalies (fraudes) consiste à identifier des comportements inhabituels qui s'écartent des comportements attendus. Par exemple, une valeur aberrante peut être considérée comme une anomalie puisqu'une valeur aberrante est une observation qui s'écarte des observations standards et peut ainsi être utilisée pour la détection de la fraude sur Internet. Par ailleurs, les techniques d'apprentissage supervisé et non supervisés peuvent aussi être utilisées pour la détection d'anomalies. En effet, le résultat d'un algorithme d'apprentissage non supervisé est généralement une nouvelle explication ou représentation des données d'observation. En considérant cette nouvelle représentation des données comme un standard, les observations qui s'écartent alors de ce standard sont classées comme des anomalies. Dans les méthodes supervisées, les modèles sont entraînés pour faire la distinction entre un comportement frauduleux et un comportement non frauduleux afin que les nouvelles observations puissent être classées. Néanmoins, ces méthodes nécessitent une identification précise des transactions frauduleuses et normales dans une base de données historiques.

2.5 Minimisation du risque empirique et apprentissage par la descente de gradient

2.5.1 Minimisation du risque empirique

L'apprentissage d'un modèle signifie déterminer les bonnes valeurs pour toutes les pondérations et le biais à partir d'exemples étiquetés. Plus précisément, l'objectif revient à trouver un modèle de l'espace des hypothèses qui minimise le risque empirique. Dans le cas de l'apprentissage supervisé, un algorithme d'apprentissage automatique crée un modèle en examinant de nombreux exemples, puis en tentant de trouver un modèle qui minimise la perte. Ce processus est appelé la Minimisation du Risque Empirique. En général, ce processus consiste à minimiser la fonction objectif définie par :

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(f_\theta, x_i, y_i), \tag{2.1}$$

où désigne (x_i, y_i) l'instance de donnée, n le nombre d'enregistrements, l la fonction de perte et f_θ une prédiction de y_i.

C'est donc un problème d'optimisation et une solution pour faciliter sa résolution est l'algorithme de la descente de gradient. Par la suite, nous introduirons l'algorithme de la descente de gradient et ses variantes, tout en soulignant les forces et limites de chacune d'elle [33].

2.5.2 Apprentissage par la descente de gradient

La descente de gradient est l'un des algorithmes les plus importants de tout l'apprentissage automatique classique et de tout l'apprentissage profond. Il s'agit d'un algorithme d'optimisation extrêmement puissant qui permet d'entraîner les modèles d'apprentissage automatique. Particulièrement, elle permet de trouver le minimum de n'importe quelle fonction convexe en convergeant progressivement vers celui-ci. En d'autres termes, la descente de gradient permet de trouver le minimum local d'une fonction différentiable.

Considérons une fonction de coût L, qui associe un vecteur de paramètres θ à un scalaire L(θ), que nous voulons minimiser. Plus précisément en apprentissage automatique, la fonction

de coût est généralement la moyenne, ou l'espérance, de la fonction d'erreur définie par :

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(f_\theta, z_i), \tag{2.2}$$

où n représente le nombre d'éléments dans l'ensemble d'apprentissages.

Dans le cadre de l'apprentissage supervisé, $z = (x, y)$, et $f_\theta(x)$ est une prédiction de y indexée par les paramètres θ.

Par ailleurs, le gradient de la fonction L par rapport à un scalaire θ est défini formellement comme suit [14][18] :

$$\frac{\partial L(\theta)}{\partial \theta} = \lim_{\delta\theta \to 0} \frac{L(\theta + \delta\theta) - L(\theta)}{\delta\theta}. \tag{2.3}$$

Cette équation représente la variation ΔL provoquée par un changement $\Delta\theta$, dans la limite où $\Delta\theta$ est très petit.
Lorsque θ est un vecteur, le gradient $\frac{\partial L(\theta)}{\partial \theta}$ est un vecteur contenant un élément $\frac{\partial L(\theta)}{\partial \theta_i}$ pour chaque θ_i.
Mais dans le cas où $\Delta\theta_i$ est petit, $\frac{\Delta L}{\Delta \theta_i}$ devient $\frac{\partial L(\theta)}{\partial \theta_i}$. L'objectif est de trouver une valeur de θ qui minimise $L(\theta)$ en résolvant l'équation :

$$\frac{\partial L(\theta)}{\partial \theta} = 0 \tag{2.4}$$

Pour cela, nous utilisons des méthodes numériques d'optimisation. La plupart de ces méthodes reposent sur l'idée de la descente locale où on modifie itérativement θ pour diminuer $L(\theta)$, jusqu'à ce que ça devienne impossible, c'est-à-dire que nous sommes arrivés à un minimum local. Cependant, la descente de gradient est la technique la plus simple parmi les techniques d'optimisation à base de gradient. Aussi, il existe de nombreuses variantes de l'algorithme de la descente de gradient que nous verrons ci-dessous.

La descente de gradient par lots : Encore appelée la descente de gradient vanille, utilise tous les exemples d'entraînement pour faire une seule mise à jour. Cela signifie que pour que cet algorithme fasse un seul pas dans une certaine direction, il utilise tous les exemples d'apprentissage de l'ensemble de données pour effectuer la mise à jour. Il devient tout à fait évident que si l'ensemble de données est très grand, cette forme de descente de gradient peut

être coûteuse en calcul [22][18]. Voici la formule de la descente de gradient :

$$\theta^{k+1} = \theta^k - \varepsilon_k \frac{\partial L(\theta^k)}{\partial \theta^k} \tag{2.5}$$

où θ^k représente nos paramètres à l'itération k et ε_k est un scalaire appelé pas de gradient, qui peut être fixe, adaptatif, ou déterminé par un échéancier.

La descente de gradient stochastique : Pour la descente de gradient stochastique, nous exploitons le fait que c'est une moyenne sur des exemples généralement (indépendants et identiquement distribués) pour faire des mises à jour de θ de manière beaucoup plus fréquente, à la limite et dans le cas le plus commun après chaque exemple. Voici la formule de la descente de gradient stochastique :

$$\theta^{k+1} = \theta^k - \varepsilon_k \frac{\partial L(\theta^k, z)}{\partial \theta^k} \tag{2.6}$$

où z est le prochain exemple dans l'ensemble d'entraînement, ou le prochain exemple tiré de la distribution d'entraînement dans le cas d'apprentissage en ligne qui signifie que l'on n'a pas accès à un ensemble d'entraînement de taille fixe, mais à un flux d'exemples provenant d'un processus de génération de données. Plus précisément, la descente de gradient stochastique est une procédure itérative où à chaque itération, un lot de données est échantillonné au hasard à partir d'un ensemble de données. L'erreur entre la prédiction du modèle et les étiquettes d'apprentissage est ensuite calculée. Cette erreur, également appelée perte, est ensuite différenciée par rapport aux paramètres du modèle. Ces gradients nous indiquent alors comment mettre à jour chaque paramètre pour rapprocher le modèle de prévision de la bonne étiquette. Par ailleurs, la descente de gradient stochastique peut être beaucoup plus rapide que la descente de gradient par lots, parce que le nombre de mises à jour est beaucoup plus grand. C'est particulièrement vrai pour des jeux de données volumineux, ou pour le cas en ligne. Par contre, la descente de gradient ordinaire est meilleure, lorsque la fonction à minimiser ne peut pas être décomposée comme une moyenne [14][18].

Descente de gradient stochastique par mini-lots : C'est une variante de la descente de gradient stochastique dans laquelle on obtient la direction de la mise à jour en prenant la moyenne de l'erreur sur un petit nombre d'exemples appelé mini-lot. Cela fonctionne comme la descente de gradient par lots, mais évidemment avec des mini-lots. Au lieu de faire des mises à jour après avoir évalué le coût et la dérivée pour l' ensemble de données comme le cas

de la descente de gradient par lots ou en ne considérant qu'un seul exemple d'apprentissage comme la descente de gradient stochastique ; il effectue les mises à jour en considérant un lot plus petit.

C'est en fait le meilleur des deux précédentes approches car il tient compte des dépenses de calcul dans le cas de descente de gradient par lots et de la forte variance dans le cas de descente de gradient stochastique. De plus, la descente de gradient stochastique par mini-lots a l'avantage de travailler avec un estimé du gradient qui est moins bruité (de moins en moins bruité lorsque B augmente). Par contre, lorsque la taille du lot augmente, le nombre de mises à jour diminue par rapport au nombre de calculs faits (à la limite, cela devient très inefficace, autant que la descente de gradient ordinaire). Ainsi, il y a un compromis optimal en terme d'efficacité de calcul, qui peut varier selon la distribution des données et les particularités de la classe de fonctions considérée, ainsi que de la manière dont les calculs sont réalisés, l'architecture matérielle et le parallélisme peuvent faire une différence [1][18].

2.6 Architectures d'apprentissage

Du point de vue de l'architecture du système, nous considérons le processus d'apprentissage comme un processus centralisé ou distribué. Le critère principal derrière cette catégorisation est de savoir si les données et le modèle sont colocalisés ou non.

2.6.1 Apprentissage centralisé

Dans un environnement d'apprentissage centralisé, les données et le modèle sont colocalisés. Il peut y avoir un ou plusieurs producteurs ou propriétaires de données, mais toutes les données sont rassemblées dans un lieu et utilisées pour la formation du modèle. L'emplacement des données peut être dans un seul ou même plusieurs machines dans le même centre de données. L'architecture d'apprentissage centralisé comprend aussi l'apprentissage automatique en tant que service, où le propriétaire des données télécharge ses données vers un service cloud qui est chargé de créer le meilleur modèle possible [5].

2.6.2 Apprentissage distribué

Les exigences qui conduisent au besoin d'architectures d'apprentissage distribuées sont la manipulation et le traitement de grandes quantités de données, le besoin d'informatique

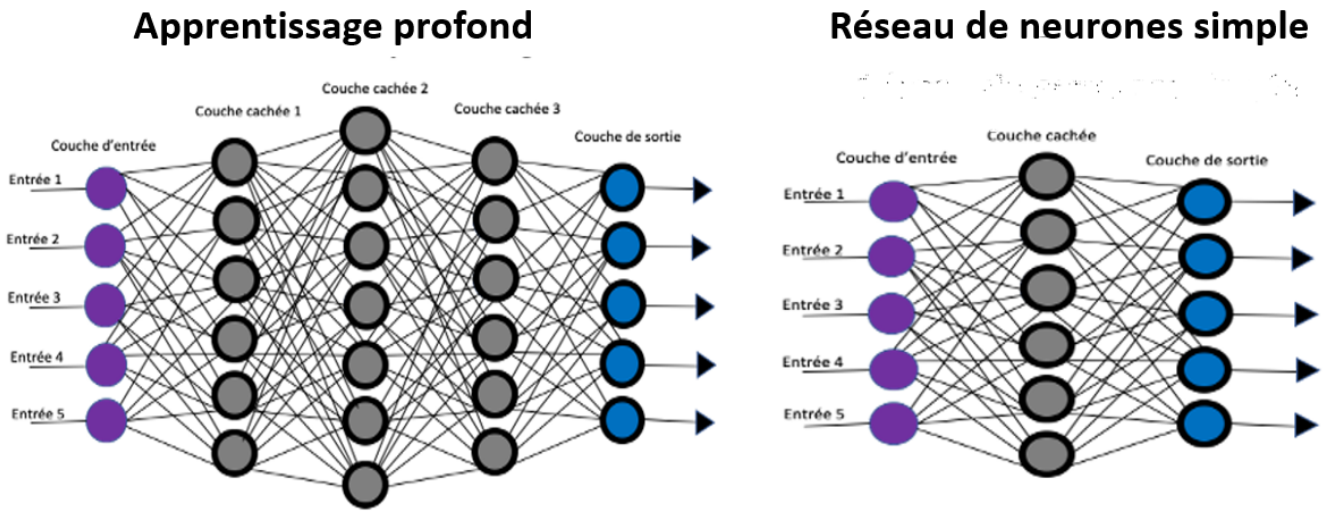
et capacité de mémoire, et des problèmes de confidentialité. À partir des variantes existantes de l'apprentissage distribué, nous présentons celles qui sont pertinentes du point de vue de la confidentialité, à savoir : apprentissage collaboratif ou fédéré, apprentissage entièrement décentralisé ou peer-to-peer (P2P) et apprentissage fractionné. L'apprentissage collaboratif ou fédéré est une forme d'entraînement décentralisé dont le but est d'apprendre un modèle global à partir de données stockées dans plusieurs dispositifs ou emplacements distants. L'idée principale est que les données ne quittent pas les appareils distants. Les données sont traitées localement puis utilisées pour mettre à jour les modèles locaux. Les mises à jour intermédiaires du modèle sont envoyées au serveur central qui les agrège et crée un modèle global. Le serveur central renvoie ensuite le modèle global à tous les participants.

Dans l'apprentissage entièrement décentralisé ou l'apprentissage Peer-to-Peer (P2P), il n'y a pas du serveur central. Au lieu de cela, les appareils communiquent de manière P2P et échangent leurs mises à jour directement avec autres appareils. Cette configuration peut-être intéressante du point de vue de la confidentialité, car elle réduit le besoin de faire confiance à un serveur central. Cependant, les attaques contre les systèmes P2P sont pertinentes dans de tels contextes et doivent être pris en compte. Mais, jusqu'à présent, aucune attaque basée sur la confidentialité n'a été signalée contre de tels systèmes ; bien qu'elles puissent devenir pertinentes à l'avenir. De plus, selon le type d'informations partagées entre les pairs, plusieurs des attaques contre l'apprentissage collaboratif peuvent être applicables. Dans l'apprentissage fractionné, le modèle entraîné est divisé en deux ou plusieurs parties. Les appareils de périphérie gardent les couches initiales du modèle d'apprentissage en profondeur et le serveur centralisé conserve les couches finales [11][4].

2.7 Apprentissage profond

L'apprentissage profond est une technologie de pointe qui a récemment attiré l'attention du milieu informatique. C'est un réseau de neurones possédant de nombreuses couches cachées, contrairement aux réseaux de neurones non profonds qui n'ont qu'une seule couche cachée. À la différence des algorithmes d'apprentissage automatiques classiques, l'apprentissage profond permet à la fois l'extraction, la transformation des caractéristiques et la prédiction. Aussi, il permet d'obtenir des modèles plus performants lorsque nous avons un gros volume de données. Ainsi, l'apprentissage profond offre une solution idéale pour détecter les cas complexes de fraudes dans des grands ensembles de transactions bancaires. La FIGURE 2.4 montre la différence entre les réseaux de neurones non profonds et un réseau de neurones profond.

FIGURE 2.4 – Différence entre un réseau neuronal simple et l'apprentissage profond

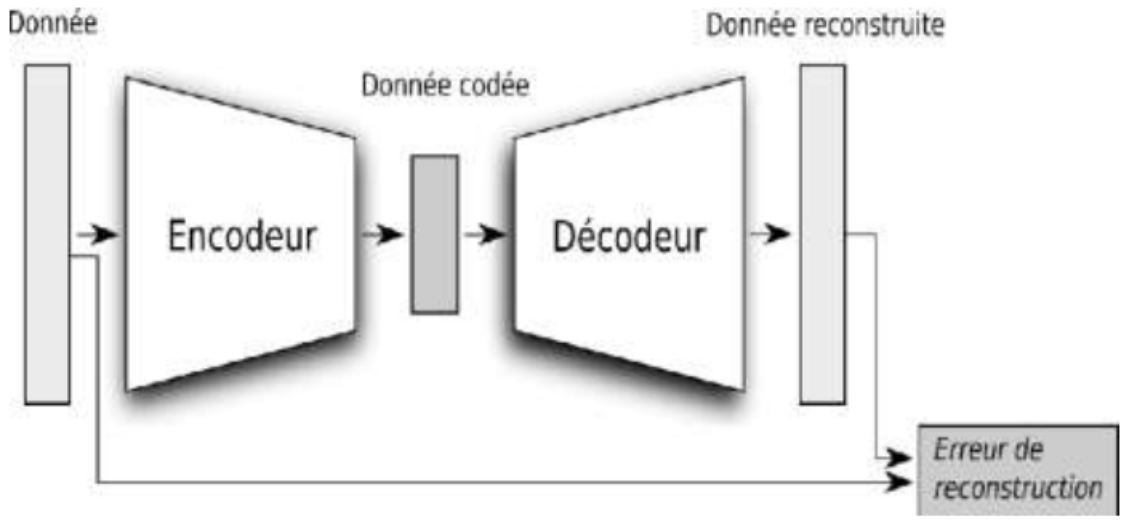


2.7.1 Auto-Encodeurs

Un Auto-encodeur est un type de réseaux de neurones artificiels qui est souvent utilisé dans l'apprentissage de caractéristiques discriminantes d'un ensemble de données. Il est composé d'un encodeur et d'un décodeur. L'encodeur est constitué d'un ensemble de couches de neurones, qui traitent les données afin d'en produire une nouvelle représentation tandis que les couches de neurones du décodeur analysent les données encodées pour essayer de reconstruire les données d'origines. Dans un auto-encodeur, la couche la plus importante est la couche encodée, qui permet d'avoir une nouvelle représentation des données et le nombre de neurones dans les couches cachées doit être inférieur à celui des couches d'entrées pour permettre aux couches cachées d'apprendre plus sur les données tout en ignorant les « bruits ». En effet, si le nombre de neurones dans les couches cachées est supérieur à celui des couches d'entrée, le réseau de neurone aura trop de capacité pour pouvoir apprendre sur les données. Dans un cas extrême, il pourrait simplement copier l'entrée dans les valeurs de sortie, y compris les bruits, sans extraire aucune information essentielle. Généralement, la nouvelle représentation des données a moins de caractéristiques, ce qui réduit la dimensionnalité des données d'origines. Par ailleurs, la différence entre les données d'origines et les données reconstruites par le décodeur permet d'évaluer l'erreur de reconstruction. Au fond, le but de l'entraînement de l'auto-encodeur est de modifier les paramètres afin de minimiser l'erreur de reconstruc-

tion. Aussi, un auto-encodeur ne nécessite pas de données labellisées puisqu'il s'agit d'une technique d'apprentissage non supervisé. Ainsi, il peut être utilisé pour réduire l'effort de labellisation des données. La FIGURE 2.5 extraite de [11] montre le processus d'encodage et de décodage. L'analyse en composantes principales (ACP) [8] est une technique statistique, largement appliquée pour l'analyse de données afin de détecter les valeurs aberrantes. Cependant, il reste que l'ACP induit une transformation linéaire contrairement aux techniques d'auto-encodages qui peuvent supporter des transformations non linéaires avec des fonctions d'activation non linéaire et les couches multiples. Par ailleurs, il est plus efficace de d'utiliser plusieurs couches avec un auto-encodeur, plutôt que d'effectuer une énorme transformation avec l'ACP. Les techniques d'autoencodage montrent donc leurs avantages lorsque les données sont de nature complexes et non linéaires. De plus, l'article [52] a montré qu'un auto-encodeur entraîné produit une erreur plus petite par rapport aux 30 premiers composants principaux d'un ACP et une meilleure séparation des grappes.

FIGURE 2.5 – Processus de codage et de décodage



2.8 Techniques d'améliorations de performances et de validations des modèles d'apprentissage automatique

2.8.1 Bagging

Le concept du bagging trouve son application dans le domaine du data mining prédictif, pour combiner les classifications prévues (prévisions) à partir de plusieurs modèles, ou à partir du même type de modèle pour différentes données d'apprentissage. Il est également utilisé pour résoudre le problème inhérent d'instabilité des résultats lorsque des modèles complexes sont appliqués à des jeux de données relativement petits. En effet, le mot Bagging est une contraction de Bootstrap Aggregation. Il est une technique utilisée pour améliorer la classification notamment celle des arbres de décision, considérés comme des « classifieurs faibles », c'est-à-dire à peine plus efficaces qu'une classification aléatoire. Le principe du bootstrap est de créer de « nouveaux échantillons » par tirage au hasard dans l'ancien échantillon, avec remise. L'algorithme, par exemple l'arbre de décision, est entraîné sur ces sous-ensembles de données. Les estimateurs ainsi obtenus sont moyennés dans le cas d'un arbre de régression où les données sont quantitatives. De même, ils sont utilisés pour un « vote » à la majorité dans le cas d'un arbre de classification où les données sont qualitatives. C'est ainsi, la combinaison de ces multiples estimateurs indépendants qui permet de réduire la variance. Toutefois, chaque estimateur est entrainé avec moins de données. En pratique, la méthode de bagging donne d'excellents résultats notamment sur les arbres de décision utilisés en « forêts aléatoires ».

L'avantage du bagging est qu'il permet de réduire la variance de l'estimateur, et permet donc de corriger l'instabilité des arbres de décision[28].

2.8.2 Boosting

Le Boosting est une approche en deux étapes, où l'on utilise d'abord des sous-ensembles des données d'origine pour produire une série de modèles aux performances moyennes, puis améliore leurs performances en les combinant à l'aide d'une fonction de coût particulière(vote majoritaire). Plus précisément, le Boosting consiste à optimiser les performances de nombreux algorithmes regroupés qui utilisent des ensembles de classifieurs binaires afin de fournir des décisions très précises. Tout comme le Bagging, le Boosting combine aussi les apprentissages réalisés pour en sortir une règle plus efficace de la classification.

Contrairement au bagging, plutôt que d'utiliser un seul modèle, nous en utilisons plusieurs que nous agrégeons ensuite pour obtenir un seul résultat. Dans la construction des modèles, le Boosting travaille de manière séquentielle. En effet, il commence par construire un premier modèle qu'il va évaluer. A partir de cette mesure, chaque individu va être pondéré en fonction de la performance de la prédiction. L'objectif est de donner un poids plus important aux individus pour lesquels la valeur a été mal prédite pour la construction du modèle suivant. Le fait de corriger les poids au fur et à mesure permet de mieux prédire les valeurs difficiles [29].

2.8.3 Résolution du problème de déséquilibres de classes

Les transactions bancaires étant très déséquilibrées car la quantité des transactions frauduleuses est très petite par rapport à celle des transactions normales. Pour résoudre ce problème afin d'obtenir une meilleure performance de classification, nous utilisons la la technique de sur-échantillonnage ou Synthetic MinorityOver-sampling (SMOTE) qui permet d'équilibrer les données provenant des diverses classes en agrandissant la taille de la classe minoritaire. Plus précisément, elle permet de créer de nouvelles instances à partir de celles de la classe minoritaire en se basant sur l'algorithme des k plus proches voisins où k est défini en fonction du rapport de déséquilibre entre les classes. Les vecteurs résultant de la différence entre les k instances les plus proches en fonction des attributs sont multipliés par un nombre aléatoire compris entre 0 et 1 pour créer la nouvelle instance [67].

Cependant, dans la plupart des cas SMOTE semble plus bénéfique avec des données de faible dimension [64], mais devient moins efficace lorsque les données sont de grande dimension car il n'atténue pas le biais de la classification dans la classe majoritaire pour la plupart des classificateurs. Sauf si le nombre de variables est réduit en effectuant un certain type de sélection de variables dans le cas des classificateurs basés sur l'algorithme des k plus proches voisins[36].

2.8.4 Techniques de validation

Après avoir entraîné un modèle d'apprentissage automatique sur des données étiquetées, celui-ci est supposé fonctionner sur de nouvelles données. Toutefois, il est important de s'assurer de l'exactitude des prédictions du modèle en production. Pour ce faire, il est nécessaire de valider le modèle. Afin de valider les performances d'un modèle d'apprentissage automatique, il est nécessaire de le tester sur de nouvelles données. En fonction des performances des

modèles sur des données inconnues, on peut déterminer s'il est sous-ajusté, sur-ajusté, ou " bien généralisé ".

En effet, il existe plusieurs techniques de validation dont les deux principales sont la validation croisée et la validation non-croisée également connue sous l'expression anglaise holdout method.

Pour ce qui est de la validation non-croisée, elle consiste à diviser l'ensemble de données aléatoirement en deux sous-ensembles disjoints : un ensemble de données d'apprentissage (généralement supérieur à 60) et un ensemble de données de test correspondant à la portion restante. Le modèle est entraîné avec l'ensemble de données d'apprentissage et puis il est validé sur sa performance avec l'ensemble de données de test.

La validation croisée consiste à diviser aléatoirement l'ensemble de données en k plis (fold) égaux de données ayant une représentation similaire des classes. Ensuite, le modèle est entraîné avec k-1 plis et puis validé avec le pli restant. Ce processus est répété jusqu'à ce que tous les plis ont été utilisés comme données de validation. Le classifieur est validé sur la performance moyenne des modèles des diverses itérations [38].

2.9 Sous-Apprentissage et Sur-Apprentissage

Le Sous-Apprentissage et le Sur-Apprentissage sont deux concepts majeurs de l'apprentissage automatique. Ces termes définissent la capacité d'un modèle à prédire les données. Par ailleurs, le Sous-Apprentissage et le Sur-Apprentissage sont les causes principales des mauvaises performances des modèles prédictifs générés par les algorithmes d'apprentissage automatique.

2.9.1 Sous-Apprentissage

Un modèle peut générer des prédictions précises avec des données d'apprentissages et s'adapte mal aux données de tests. Ainsi, le modèle ne se généralisera pas bien sur des nouvelles données, c'est-à-dire les données qu'il n'a pas encore vu parce qu'il est incapable de capturer les modèles complexes dans les données. Par conséquent, le modèle ne produira pas de résultats précis et ne sera d'aucune utilité. Afin d'éviter ce problème, la meilleure stratégie consiste à augmenter la complexité du modèle en augmentant le nombre de paramètres du modèle d'apprentissage. La technique de la validation croisée est aussi une bonne solution pour obtenir un modèle qui possède un bon compromis entre le biais et la variance [34].

2.9.2 Sur-Apprentissage

Le Sur-Apprentissage est simplement l'opposé du Sous-Apprentissage. Cela signifie qu'en plus d'apprendre les données et d'extraire le modèle, le modèle apprend plus que sa capacité. Cette condition indique que les données vont capter du bruit, ce qui conduit au problème de généralisation du modèle pour les nouvelles données. Le bruit est constitué de données non pertinentes qui affectent la sortie de la prédiction lors de la rencontre de nouvelles données. En d'autres termes, ce type de modèle conduit à de mauvaises performances car, trop complexe, il manque de capacité de généralisation. La régularisation est couramment utilisée pour limiter le « Sur-Apprentissage » et permettant ainsi de contrôler l'erreur de type variance pour aboutir à de meilleures performances [34].

2.10 Métriques de mesures de performances pour les algorithmes de classification

2.10.1 Matrice de confusion

Une matrice de confusion est un outil d'analyse prédictive. Plus précisément, il s'agit d'un tableau qui indique le nombre de faux positifs, de faux négatifs, de vrais positifs et de vrais négatifs. En prenant l'exemple de transactions bancaires, alors :

- Le Vrais positifs (VP) représente le nombre de transactions qui étaient frauduleuses et qui ont également été classées comme frauduleuses par le système ;
- Le Vrais négatifs (VN) représente le nombre de transactions qui étaient légitimes et qui ont également été classées comme légitimes ;
- Le Faux positifs (FP) représente le nombre de transactions qui étaient légitimes mais qui ont été incorrectement classées comme transactions frauduleuses ;
- Le Faux négatifs (FN) représente le nombre de transactions qui étaient frauduleuses mais qui ont été incorrectement classées comme transactions légitimes par le système.

Partant de ces définitions, une matrice de confusion permet de savoir non seulement quelles sont les erreurs commises, mais surtout le type d'erreurs commises. Plus précisément, dans le contexte de l'apprentissage automatique, une matrice de confusion est utilisée comme métrique pour analyser les performances d'un classificateur sur un ensemble de données. Par ailleurs, elle permet de générer une visualisation de mesures telles que la précision, l'exac-

titude, la spécificité et le rappel. Ces mesures permettent une analyse plus détaillée que la simple proportion de classifications correctes (Accuracy).

2.10.2 Rappel

Le rappel est donné en pourcentage d'exemples positifs que le modèle a pu classer parmi tous les exemples positifs contenus dans l'ensemble des données. Par ailleurs, cette valeur peut également être appelée « taux de réussite » ou « sensibilité ». Plus formellement :

$$Rappel = \frac{VP}{VP+FN} \tag{2.7}$$

2.10.3 Précision

Comme le rappel, la précision est une valeur qui caractérise les performances d'un modèle en termes de classification des exemples de la classe positive. Contrairement au rappel, la précision concerne le nombre d'exemples que le modèle a étiquetés comme positifs et qui étaient vraiment positifs. Plus formellement :

$$Precision = \frac{VP}{VP+FP} \tag{2.8}$$

Pour faire la distinction entre rappel et précision, la précision vise à déterminer le pourcentage de tous les exemples étiquetés positifs qui étaient vraiment positifs, tandis que le rappel donne le pourcentage de tous les exemples véritablement positifs que le modèle pourrait reconnaître.

2.10.4 Spécificité

Alors que le rappel et la précision sont des valeurs qui suivent les exemples positifs et le taux de vrais positifs, la spécificité quant à elle quantifie le taux de vrais négatifs ou le nombre d'exemples que le modèle a définis comme négatifs qui étaient vraiment négatifs. Plus formellement :

$$Specificite = \frac{VN}{VN+FP} \tag{2.9}$$

2.10.5 F1-Score

Le F1-Score recherche un équilibre entre la précision et le rappel sur la classe positive. Il est plus intéressant que l' « accuracy » car le nombre de vrais négatifs n'est pas pris en consi-

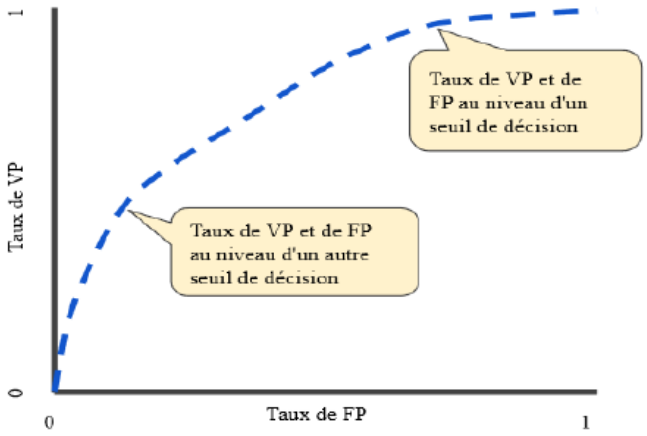
dération. Particulièrement, dans les situations des données de classes déséquilibrées comme le cas des transactions bancaires où, le modèle est plutôt bon pour prédire les vrais négatifs. Pour cette raison, F1-Score est généralement utilisé pour la classification multi-classes ou une classification binaire avec des classes très déséquilibrées. Plus formellement :

$$F1 - Score = 2 \cdot \frac{precision \times rappel}{precision + rappel} \tag{2.10}$$

2.10.6 Courbe CFR

Une courbe CFR est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification. Cette courbe trace le taux de vrais positifs (TVP) en fonction du taux de faux positifs (TFP). Plus précisément, une courbe CFR trace les valeurs TVP et TFP pour différents seuils de classification. En conséquence, lorsqu'on diminue la valeur du seuil de classification, ceci permet de classer plus d'éléments comme positifs, ce qui augmente le nombre de faux positifs et de vrais positifs. La FIGURE 2.6 [9] montre l'évolution d'une courbe CFR.

FIGURE 2.6 – Courbe CFR

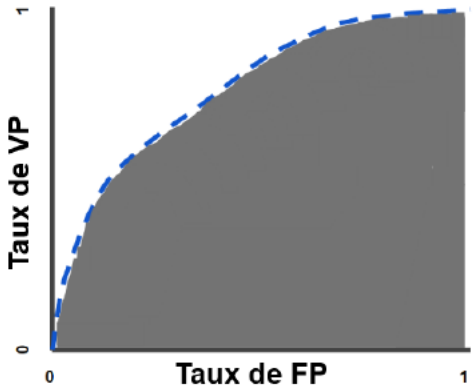


2.10.7 Courbe ROC et ASC

La courbe ROC (Receiver Operating Characteristic) est courbe permettant de mesurer la performance d'un classificateur binaire. Le graphique représente souvent le taux de vrais po-

sitifs (fraction des positifs qui sont effectivement détectés) en fonction du taux de faux positifs (fraction des négatifs qui sont incorrectement détectés) [12]. L'ASC signifie "aire sous la courbe ROC". Elle permet de mesurer l'intégralité de l'aire à deux dimensions situées sous le tracé de la courbe ROC (par calcul d'intégrale) de (0,0) à (1,1). L'ASC fournit une mesure agrégée des performances pour tous les seuils de classification possibles. Ainsi, on peut interpréter l'ASC comme une mesure de la probabilité pour que le modèle classe un exemple positif aléatoire au dessus d'un exemple négatif aléatoire [9]. La FIGURE 2.7 [9] montre une aire sous la courbe ROC.

FIGURE 2.7 – Aire sous la courbe ROC



En résumé, nous avons présenté dans ce chapitre, le fonctionnement des algorithmes d'apprentissage automatique classique et de l'apprentissage profond qui sont les plus utilisés pour la détection de fraudes bancaires sur Internet ainsi que la technique de détection d'anomalies avec l'apprentissage automatique. Aussi, nous avons présenté quelques techniques d'améliorations de performances et quelques métriques importantes de mesures de performances des modèles d'apprentissage automatique. Dans le chapitre suivant, nous présenterons, les principes, les propriétés et formes de confidentialité différentielle, ainsi que quelques techniques d'ajouts de bruits satisfaisantes les propriétés de confidentialité différentielle. Aussi, nous définirons le contexte d'utilisation de la confidentialité différentielle dans les modèles basés sur l'apprentissage automatique.

Chapitre 3

Confidentialité différentielle et apprentissage automatique

Introduction

Ce chapitre présente différentes notions liées à la confidentialité différentielle et son utilisation dans les modèles d'apprentissage automatique. D'abord, nous présentons les principes, les architectures, les propriétés, les différentes formes de la confidentialité différentielle et la notion de la sensibilité. Ensuite, nous expliquerons différents mécanismes différentiellement privés. Enfin, nous définirons le contexte d'utilisation de la confidentialité différentielle dans les modèles d'apprentissage automatique.

3.1 La confidentialité différentielle

Dans cette section, nous décrirons d'abord les principes de base, les propriétés, la notion de sensibilité et puis présenter quelques mécanismes différenciellement privés.

3.1.1 Principe

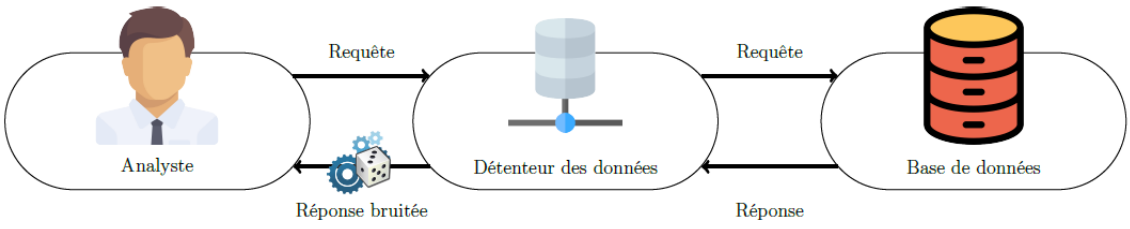
La confidentialité différentielle est un concept qui a été introduit en 2006 par Cynthia Dwork [48], Professeure à l'Université Harvard, afin de protéger les individus fournissant des données à caractère personnel contre le risque de ré-identification. L'objectif est de faire en sorte que peu importe que vos informations personnelles soient récoltées ou non, cela n'a aucune incidence sur ce qu'on peut apprendre sur vous car la personne répondant à une enquête

se sentira plus en sécurité si elle sait que la probabilité que le résultat communiqué soit R est presque la même, indépendamment du fait qu'elle ait répondu ou non. Ainsi, la confidentialité différentielle permet de rendre la confidentialité maximale en minimisant les chances d'identification individuelle des enregistrements [48]. Cette technique consiste à ajouter assez de bruit aléatoire aux données afin de maintenir la confidentialité de ces données en minimisant les risques d'identification des entités qu'elles contiennent. La technique doit aussi autant que possible maximiser la pertinence ou la précision des résultats de requêtes sur les données. Il faut donc trouver un équilibre judicieux entre la quantité du bruit à ajouter (qui agit sur le niveau de confidentialité) et la précision des résultats des requêtes. En effet, une quantité élevée de bruit conduit naturellement à une perte de qualité des données, et donc à une faible précision des résultats des requêtes sur les données.

3.1.2 Architecture

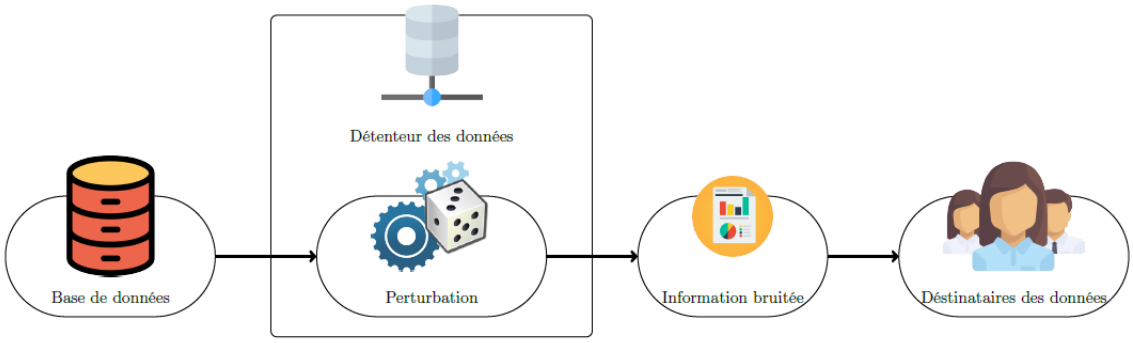
La confidentialité différentielle a été initialement proposée dans une configuration dynamique [46], où un tiers de confiance dispose d'une base de données à laquelle sont soumises les requêtes des utilisateurs. L'hypothèse est qu'il existe un algorithme d'anonymisation entre l'utilisateur qui soumet sa requête et le tiers de confiance qui y répond. Afin de préserver la vie privée des individus, la suppression ou l'ajout d'un enregistrement dans l'ensemble de données, n'affecte pas significativement le résultat de la requête comme montré sur la Figure 3.1 extraite de [79].

FIGURE 3.1 – Architecture dynamique



Quant à la configuration statique [47], le tiers de confiance prépare un agrégat ou une base de données synthétique, destinés à être publiés pour répondre à différents types de requêtes comme montré sur la FIGURE 3.2 extraite de [79].

FIGURE 3.2 – Architecture statique



Définition 3.1.1 (ε-différentiellement confidentiel)**.** *Formellement, un algorithme probabiliste A est dit ε-différentiellement confidentiel, si pour tous jeux de données D1 et D2 qui diffèrent d'un seul élément (information à propos d'une seule personne) et pour toute sortie possible de A dans le sous-ensemble Q,*

$$Pr[A(D1) \in Q] \leq exp(\varepsilon).Pr[A(D2) \in Q] \tag{3.1}$$

où ε est une constante spécifiée par l'utilisateur et la probabilité est fondée sur l'aléa introduit par l'algorithme.

Dans l'équation (3.1)*, A est un algorithme probabiliste qui prend en entrée un ensemble de données. Il est à noter, que dans ce mémoire, nous considérons les transactions bancaires comme l'ensemble de nos données. Puis le paramètre ε représente le budget de confidentialité. En effet, ε est un nombre réel positif qui définit une limite supérieur à la perte de la vie privée de l'algorithme A. Plus ε est proche de 0 et plus la confidentialité est forte [48]. Intuitivement, en connaissant les sorties Q de A, il est difficile pour l'adversaire de déduire les données d'origine D1 ou D2, si le paramètre ε est suffisamment petit. Par ailleurs, ε est une valeur relative car dans [56], il est montré que pour la même valeur de ε, les garanties de protection de la vie privée offertes par la ε-confidentialité différentielle, varient en fonction du domaine en question et de la requête prise en charge. En pratique, la question du choix de ε s'avère difficile et demeure aujourd'hui un défi.*

3.1.3 Propriétés et notion de sensibilité

Cette partie met premièrement en lumière plusieurs propriétés et méthodes fondamentales de la confidentialité différentielle. Deuxièmement, elle présente la notion de sensibilité ainsi qu'une présentation et comparaison de quelques mécanismes différentiellement privés.

3.1.3.1 Propriétés

Propriété 3.1.1 (Protection contre les risques arbitraires)**.** *Cela signifie aller au-delà de la protection contre la ré-identification.*

Propriété 3.1.2 (Quantification de la perte de confidentialité)**.** *La confidentialité différentielle n'est pas un concept binaire, elle a une mesure de perte de confidentialité. Ainsi, elle permet de faire des comparaisons entre différentes techniques utilisées. Par exemple, pour une limite fixée sur la perte de la confidentialité, on peut connaître la technique qui offre une meilleure précision, et aussi qu'en fixant la précision, on peut connaître la technique qui offre une meilleure confidentialité.*

.

Propriété 3.1.3 (Robustesse au post-traitement)**.** *Tout ce qui est dérivé de la sortie d'un algorithme différentiellement privé est elle-même différentiellement privé, et la dérivation n'entraîne aucune autre perte de confidentialité. Ceci dit, peu importe quel post-traitement on applique à une sortie d'un algorithme qui satisfait la confidentialité différentielle, le processus global satisfait la confidentielle différentielle. Ainsi, il n'est pas possible de compromettre davantage la vie privée d'un individu en analysant sa réponse.*
Formellement, soit g une fonction aléatoire, si M est un ε-mécanisme différentiellement privé, alors le mécanisme $M' = M \circ g$ est ε- diférentiellement privé.

Propriété 3.1.4 (Composition parallèle.)**.** *Si on divise un jeu de données en k sous-ensembles disjoints et on applique un mécanisme satisfaisant la confidentialité de niveau ε_i au sous-ensemble i pour $i = 1, ..., k$, on obtient un mécanisme satisfaisant la confidentialité différentielle de niveau $\max_{i=1..k} \varepsilon_i$.*

Propriété 3.1.5 (Composition séquentielle)**.** *Si on applique l'un après l'autre k mécanismes satisfaisant la confidentialité différentielle au même jeu de données, avec paramètres $\varepsilon_i, ..., \varepsilon_k$, on obtient un mécanisme satisfaisant la confidentialité différentielle de niveau $\sum_{i=0}^{k} \varepsilon_i$.*

Propriété 3.1.6 (Confidentialité du groupe)**.** *La confidentialité différentielle permet l'analyse et le contrôle de la perte de confidentialité subie par des groupes de personnes, tels que les familles.*

3.1.3.2 Sensibilité

Une manière d'assurer la confidentialité différentielle consiste à ajouter du bruit au résultat de la requête. Ainsi, la quantité du bruit x à ajouter à la vraie réponse de la requête est calculée selon la notion de sensibilité globale. En effet, pour une requête donnée, celle-ci mesure la différence maximale possible en sortie pour deux bases de données voisines. Plus cette différence est petite, plus le bruit à ajouter pourra être faible.

Définition 3.1.2 (Sensibilité)**.** *Soit $f : D \longrightarrow R$ une fonction d'une requête, la sensibilité Δf est définit comme la distance maximale de la norme L_1 ou L_2 entre les réponses de la requête q ($q(D_1 et q(D_2)$ sur n'importe quelle bases de données voisines. Formellement :*

$$\Delta(f) = max_{D_1,D_2} \big\| f(D_1) - f(D_2) \big\| \tag{3.2}$$

Ainsi définie, la sensibilité permet de paramétrer les mécanismes utilisés pour fournir un bruit adéquat. Intuitivement, elle est la contribution de l'individu le plus influent sur la requête f. Ainsi, cela signifie que plus la requête est sensible, et plus la garantie souhaitée est forte, plus il faut du bruit pour obtenir cette garantie.

Exemple 3.1.1 (Sensibilité)**.** *Soient deux requêtes $f1$ et $f2$, telles que $f1$ = 'Le nombre d'individus atteints d'hypertension' et $f2$ = 'La somme des âges des individus de l'ensemble de données'. Supposons que age $\in$ [0 ; 130]. Nous avons alors, $\Delta(f1)$ = 1 et $\Delta(f2)$ = 130.*

Aussi, voici d'autres concepts importants qui permettent de comprendre la notion de la sensibilité.

Définition 3.1.3 (Sensibilité locale)**.** *La sensibilité locale consiste à calculer la sensibilité d'un ensemble de données local, où les modifications possibles sont liées à l'ensemble de données local et non à l'univers de tous les ensembles de données. Étant donné une fonction de requête f qui fonctionne sur un ensemble de données D_1, la sensibilité locale est alors les différences maximales qu'un changement dans D_1 peut produire :*

$$\Delta(f)_{SL} = max_{D_1,D_2} \big\| f(D_1) - f(D_2) \big\|_1 \tag{3.3}$$

D_1 est l'ensemble de données connu et D_2 est un autre ensemble de données avec au plus un élément différent par rapport à l'ensemble de données D_1.

Définition 3.1.4 (Sensibilité globale)**.** *La sensibilité globale fait référence à la différence maximale de la sortie qu'une fonction de requête peut entraîner lorsqu'une modification est apportée à un ensemble de données. En d'autres termes, la sensibilité globale détermine l'ampleur du bruit nécessaire pour répondre aux exigences de ε-confidentialité différentielle. Intuitivement, la sensibilité globale correspond aux différences maximales de sortie en tenant compte de tous les ensembles de données possibles et dépend donc uniquement de la requête et non de l'ensemble de données. Étant donné une fonction de requête f qui fonctionne sur un ensemble de données D et produisant la différence de résultat maximale pour tous les ensembles de données (D1, D2) avec au plus une entrée différente, la sensibilité globale est définie formellement par :*

$$\Delta(f)_{SG} = max_{D_1,D_2} \left\| f(D_1) - f(D_2) \right\|_1 \tag{3.4}$$

où $\|.\|_1$ est la distance de la norme L_1 entre les ensembles de données différant d'au plus un élément, max est le résultat maximum de $f(D_1) - f(D_2)$ pour tous les ensembles de données D_1, D_2. En outre, d'autres notions importantes de la sensibilité sont la norme L_1, la norme L_2, la sensibilité L_1 et la sensibilité L_2

Définition 3.1.5 (La norme L_1)**.** *La norme L_1 d'un vecteur V de longueur k est définie comme la somme des éléments du vecteur. Formellement :*

$$\|V\|_1 = \sum_{i=0}^{k} |V_i| \tag{3.5}$$

Définition 3.1.6 (La norme L_2)**.** *La norme L_2 d'un vecteur V de longueur k est définie comme la racine carrée de la somme des carrés des éléments du vecteur. Formellement :*

$$\|V\|_1 = \sqrt{\sum_{i=0}^{k} V_i^2} \tag{3.6}$$

Définition 3.1.7 (La sensibilité L_1)**.** *La sensibilité L_1 d'une fonction à valeur vectorielle est égale à la somme des sensibilités élément par élément. Par exemple, si nous définissons une fonction à valeur vectorielle f qui renvoie un vecteur de longueur k de résultats sensibles à 1, alors la sensibilité L_1 de f est k*

Définition 3.1.8 (La sensibilité L_2)**.** *La sensibilité L_2 d'une fonction à valeur vectorielle est une fonction à valeur vectorielle f renvoyant un vecteur de longueur k de résultats sensibles à 1, alors la sensibilité L_2 de f est $\sqrt{k}$. Par ailleurs, pour les vecteurs longs, la sensibilité L_2 sera évidemment bien inférieure à la sensibilité L_1. Notamment, pour les algorithmes d'apprentissage automatique qui renvoient parfois des vecteurs avec des milliers d'éléments, la sensibilité L_2 est nettement inférieure à la sensibilité L_1[49].*

3.1.4 Mécanismes pour assurer la confidentialité différentielle

Nous avons clarifié les mécanismes en deux groupes. D'une part, les mécanismes d'ajouts du bruit au résultat de la requête et d'autres part, les mécanismes qui permettent de choisir de façon confidentielle la meilleure réponse à une requête. Ainsi, nous présentons premièrement les mécanismes de LaPlace, Gaussien et Géométrique et deuxièmement le mécanisme Exponentiel. Le premier groupe permet d'ajouter un bruit aléatoire à la vraie réponse de la requête. D'abord, la vraie valeur de $f(D)$ est calculée, où f est la fonction de requête et D l'ensemble de données, ensuite un bruit aléatoire est ajouté à $f(D)$ et la réponse $A(D) = f(D) + bruit$ est finalement retournée. L'amplitude du bruit est choisie en fonction du plus grand changement que peut provoquer un enregistrement sur la sortie de la fonction requête. Ce paramètre défini dans [49] est appelée sensibilité de la fonction.

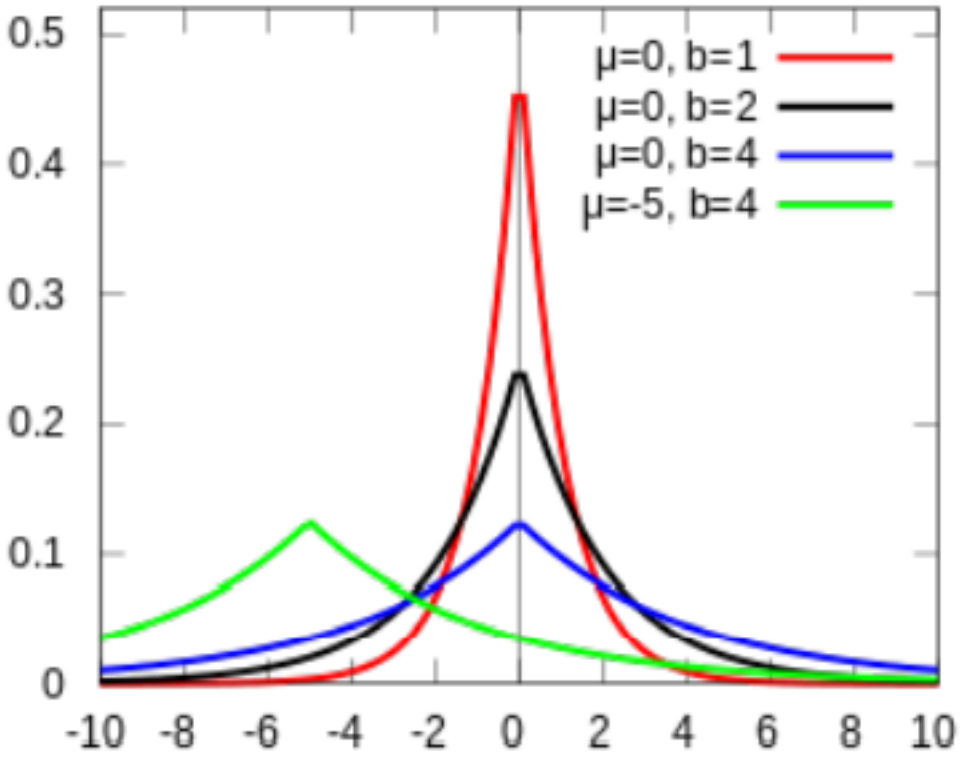
Mecanisme de Laplace Le mécanisme le plus répandu pour assurer la confidentialité différentielle est le mécanisme Laplacien, ce dernier fonctionne en ajoutant un bruit numérique aléatoire à la réponse d'une requête. Il faut noter qu'il est utilisé seulement quand le résultat de la requête est une valeur numérique [79]. Aussi, il faut noter que le mécanisme Laplacien utilise la distribution de LaPlace.

Définition 3.1.9 (Distribution de Laplace)**.** *La distribution de Laplace est une densité de probabilité continue. Elle est aussi appelée la loi double exponentielle, car sa densité peut être vue comme l'association des densités de deux lois exponentielles, accolées dos à dos. De même, la loi de Laplace s'obtient aussi comme le résultat de la différence de deux variables exponentielles indépendantes [25]. Formellement, la distribution de Laplace (centré sur 0) avec échelle b est la distribution de la fonction de densité :*

$$Lap(x|\mu,b) = \frac{1}{2b} exp(\frac{-|x-\mu|}{b}) \tag{3.7}$$

La variance de cette distribution est $\sigma = 2b^2$. Nous écrirons parfois Lap (b) pour désigner la distribution de Laplace avec l'échelle b, et abusera parfois de la notation et écrira Lap (b) simplement pour désigner une variable aléatoire $X \sim Lap(b)$. Par ailleurs, la FIGURE 3.3 extraite du [24] représente la densité de probabilité de Laplace.

FIGURE 3.3 – Densité de probabilité



Selon [49] une quantité du bruit= Lap($\frac{\Delta(f)}{\varepsilon}$) tirée d'une distribution de Laplace, centrée sur 0 et d'échelle $\frac{\Delta(f)}{\varepsilon}$ garantit la ε-confidentialité différentielle où $\Delta(f)$ est la sensibilité de f.

Mécanisme Gaussien Le mécanisme gaussien permet d'ajouter également un bruit aléatoire numérique à un résultat de la requête comme le fait le mécanisme Laplacien mais en utilisant une loi Gaussienne. Par ailleurs, la norme L_2 est utilisée dans ce cas afin de permettre d'avoir une petite valeur de la sensibilité pour réduire la quantité du bruit à ajouter [79].

Définition 3.1.10 (Loi Gaussienne)**.** *Les lois gaussiennes, également appelées une loi normale est une loi de probabilité absolument continue qui dépend de deux paramètres : son espérance,*

un nombre réel noté μ, et son écart type, un nombre réel positif noté σ. Au fond, Parmi les lois de probabilité, les lois normales prennent une place particulière grâce au théorème central limite. En effet, elles correspondent au comportement, sous certaines conditions, d'une suite d'expériences aléatoires similaires et indépendantes lorsque le nombre d'expériences est très élevé. Grâce à cette propriété, une loi normale permet d'approcher d'autres lois et ainsi de modéliser de nombreuses études scientifiques comme des mesures d'erreurs ou des tests statistiques, en utilisant par exemple les tables de la loi normale centrée réduite [27]. La densité de probabilité de la loi normale d'espérance μ, et d'écart type σ est donnée formellement par :

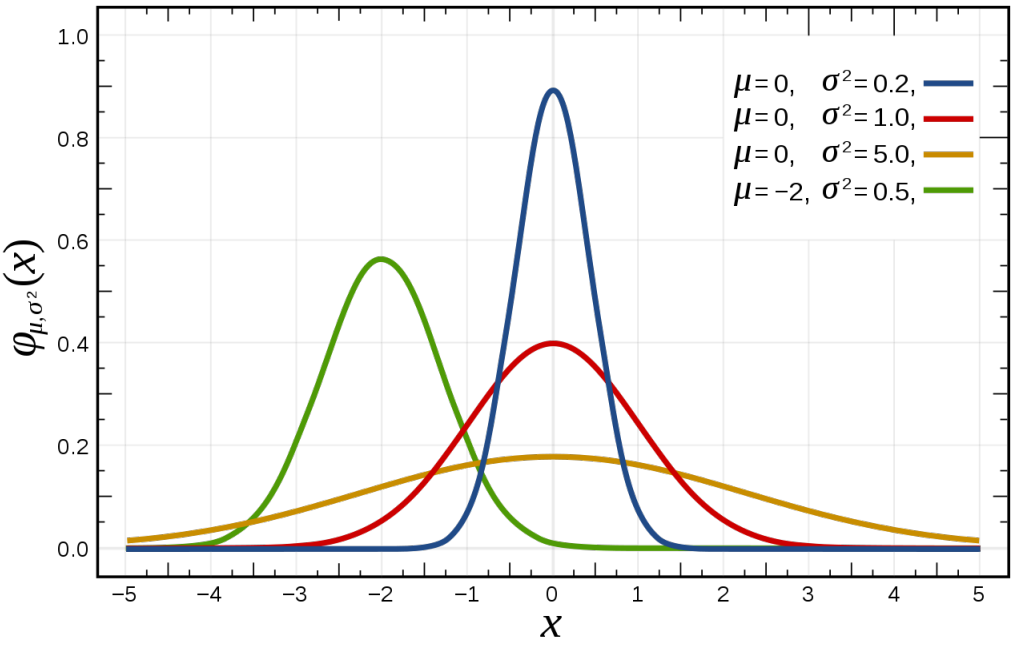
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.8}$$

Puis, la variable aléatoire X suit une loi normale de moyenne μ and d'écart type σ :

$$X \sim \mathcal{N}(\mu, \sigma^2) \tag{3.9}$$

Par ailleurs, la FIGURE 3.4 extraite du [23] représente la densité de probabilité de la loi normale.

FIGURE 3.4 – Loi normale



Selon [49] une quantité de bruit ajoutée avec la loi Normale N(0, $\Delta^2(f)\frac{ln(1.25/\delta)}{\varepsilon^2}$) est (ε, δ)-différentiel.

Mécanisme géométrique Le mécanisme géométrique a été proposé dans le cas des requêtes dont les résultats sont des entiers. Au lieu d'ajouter un bruit réel comme le fait les mécanismes Laplacien et Gaussien, le mécanisme géométrique ajoute un bruit $X \in \mathbb{Z}$ au résultat de la requête $f(D)$ tel que $A(D) = f(D) + X$ où X est tiré à partir de la distribution d'une loi géométrique [79].

Définition 3.1.11 (La loi géométrique)**.** *La loi géométrique est une loi de probabilité discrète qui modélise l'observation du nombre d'épreuves de Bernoulli identiques et indépendantes devant se succéder pour espérer un premier succès [26]. Plus précisément, On répète continuellement et de façon indépendante une épreuve de Bernoulli dont la probabilité de succès est p. Soit X le nombre d'épreuves nécessaires pour obtenir un premier succès, X suit une loi géométrique de paramètre α, dénoté $X \sim G(\alpha)$. La fonction de masse d'une variable aléatoire $X \sim G(\alpha)$ est notée par :*

$$Pr[X = x] = \frac{1+\alpha}{1+\alpha} \times \alpha^x \tag{3.10}$$

L'utilisation d'un bruit issu d'une distribution géométrique, où $\alpha = e^{\frac{-\varepsilon}{\Delta(f)}}$ garantit la ε-confidentialité différentielle.

Mécanisme Exponentiel Les mécanismes Laplacien, Gaussien et Géométrique ont été mis au point pour les requêtes dont les résultats de requêtes sont numériques. Tandis que le mécanisme exponentiel est utilisé pour les requêtes dont les résultats ne sont pas numériques. Celui-ci est basé sur une fonction d'utilité qui évalue l'utilité de chaque résultat possible à une requête, puis sélectionne une sortie $t \in T$ qui est proche de l'optimum (au sens de la fonction d'utilité), tout en respectant les propriétés de la confidentialité différentielle [79]. Ainsi définit, le mécanisme exponentiel est meilleur pour des problèmes qui consistent à faire le meilleur choix sans ajout du bruit. Par exemple, supposons que nous voulions choisir une date pour une grande réunion, qui utilise le calendrier personnel de chaque participant pour maximiser le nombre de participants sans conflit, tout en offrant une confidentialité différentielle pour les calendriers. Dans ce cas, l'ajout du bruit à une date n'a pas beaucoup de sens car cela peut transformer un vendredi en samedi et augmenter considérablement le nombre de conflits. Aussi, il faut noter que le mécanisme exponentiel utilise la loi Exponentiel.

Définition 3.1.12 (La loi Exponentiel)**.** *Une loi Exponentielle modélise la durée de vie d'un phénomène sans mémoire, ou sans vieillissement, c'est-à-dire la probabilité que le phénomène dure au moins s + t heures sachant qu'il a déjà duré t heures sera la même que la probabilité de durer s heures à partir de sa mise en fonction initiale. En d'autres termes, le fait que le phénomène ait duré pendant t heures ne change rien à son espérance de vie à partir du temps t.*

Formellement, une variable aléatoire continue X dénotée $X \sim Exp(\lambda)$ *suit une loi exponentielle de paramètre* Λ *signifie que sa densité est la fonction définie par :*

$$f_X(x) = \begin{cases} \lambda e^{\lambda x} & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases} \tag{3.11}$$

Définition 3.1.13 (Mécanisme Exponentiel)**.** *Pour une base de données D contenant n enregistrements, T un ensemble de résultats possibles, un budget de confidentialité* ε*, et une fonction d'utilité définie par :* $u : D^n \times T \longrightarrow R$.

Intuitivement, cette fonction assigne une valeur réelle (score) à chaque paire (d,t)*, où* $d \in D^n$ *et* $t \in T$*. Il faut noter que le score représente l'importance de la paire. Ainsi, plus le score est elevé, plus la paire est importante. Par ailleurs,*

$$\Delta(u) = max_{(D_1,D_2)\in D^n} \big\| u(D_1,t) - u(D_2,t) \big\| \tag{3.12}$$

désigne la sensibilité de la fonction utilité, elle représente la variation maximum de u à travers deux ensembles de données voisins D_1 *et* D_2*, quel que soit* $t \in T$.

Étant donné l'entrée $d \in D^n$*, l'objectif du mécanisme est de renvoyer un* $t \in T$ *tel que la fonction* $u(d,t)$ *soit approximativement maximisée. En effet, Il tire une sortie t, en construisant une distribution de probabilité à travers l'ensemble de sortie T. La probabilité associée à chaque sortie t est proportionnelle à* $\exp(\frac{\varepsilon u(d,t)}{2\Delta(u)})$*, ainsi, la sortie avec un plus grand score a plus de probabilité d'être retournée. Ainsi donc un algorithme probabiliste qui choisit une sortie* $t \in T$ *avec une probabilité* $\exp(\frac{\varepsilon u(d,t)}{2\Delta(u)})$ *garantit la* ε*-confidentialité différentielle.*

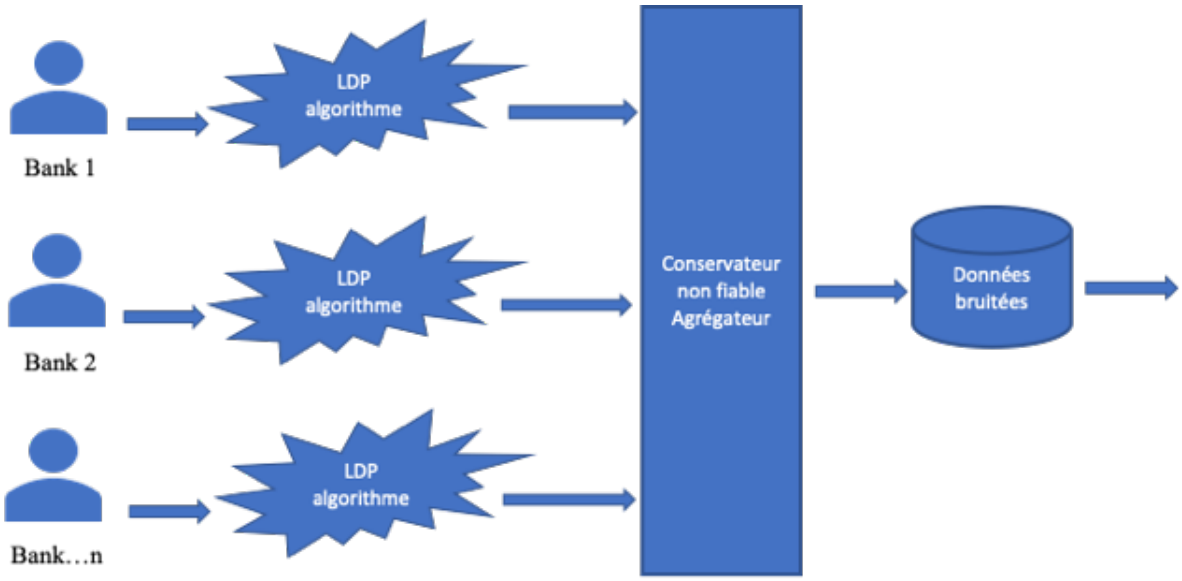
3.1.5 Formes de confidentialités différentielles

Il y a deux formes de confidentialités différentielles : la confidentialité différentielle locale et la confidentialité différentielle globale.

3.1.5.1 Confidentialité différentielle locale

Dans le cadre de la confidentialité différentielle locale, le bruit est ajouté à chaque point de données individuel dans l'ensemble de données. Particulièrement, dans le contexte de données bancaires, le bruit est ajouté à chaque transaction bancaire. Il est ajouté par un des collaborateurs des banques. Ensuite, elles n'envoient leurs données à l'agrégateur qu'une fois qu'elles sont déjà anonymisées. Ainsi, l'agrégateur n'a pas accès aux données réelles. L'agrégateur est un organisme ou un centre de recherche qui collecte et traite les données. Par ailleurs, un algorithme randomisé A satisfait la confidentialité différentielle ε-locale si pour toutes les paires de valeurs d'un client v1 et v2 et pour tout Q $\subseteq$ plage(A) et $\varepsilon \geq 0$, l'équation (1) est vraie. La plage (A) est l'ensemble de toutes les sorties possibles de l'algorithme aléatoire A. Nous rappelons qu'un algorithme randomisé est un algorithme qui fait appel à des données aléatoires. La FIGURE 3.5 montre le fonctionnement de la confidentialité locale.

FIGURE 3.5 – Fonctionnement de la confidentialité locale

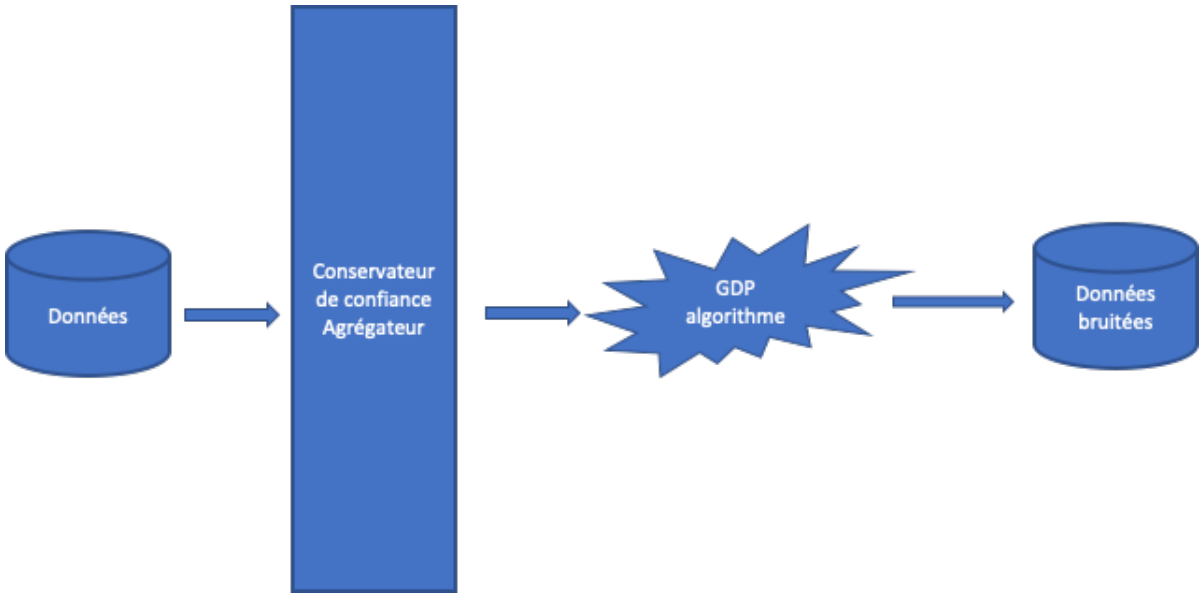


3.1.5.2 Confidentialité différentielle globale

Dans ce modèle, il existe un agrégateur central. Chaque banque ou participant envoie ses données sans bruit à cet agrégateur. L'agrégateur prend les données et les transforme avec un algorithme différentiellement privé tel que décrit précédement. La FIGURE 3.6 montre l'ar-

chitecture permettant d'assurer la confidentialité globale. En règle générale, la confidentialité différentielle globale peut conduire à des résultats plus précis par rapport à la confidentialité différentielle locale, tout en conservant le même niveau de confidentialité. D'autre part, lors de l'utilisation de la confidentialité différentielle globale, les personnes qui font don de leurs données doivent avoir confiance au fait que l'entité destinataire ajoutera le bruit nécessaire pour préserver leur confidentialité.

FIGURE 3.6 – Fonctionnement de la confidentialité globale



3.1.6 Différences entre les confidentialités différentielles globale et locale

La confidentialité différentielle globale (CDG) et la confidentialité différentielle locale (CDL) sont deux approches qui peuvent être utilisées par des algorithmes randomisés pour assurer une confidentialité différentielle. Comme le montre la FIGURE 3.6, CDG emploie un conservateur de confiance qui applique un bruit aléatoire soigneusement calibré aux valeurs réelles renvoyées pour une requête particulière. Le CDG est également appelé modèle du conservateur de confiance [54]. Contrairement au CDG, le CDL n'a pas besoin d'être approuvé par une tierce partie, c'est pourquoi il est appelé modèle de conservateur non approuvé [56]. Avec CDL, les données sont randomisées avant que le conservateur ne puisse y accéder. Cependant, les algorithmes CDL peuvent produire des données trop bruyantes, car le bruit est

souvent intensément appliqué pour assurer la confidentialité des données. Le CDL est considéré comme une notion forte et rigoureuse de la vie privée. Il est donc considéré comme une approche de pointe pour la collecte et la distribution de données privées.

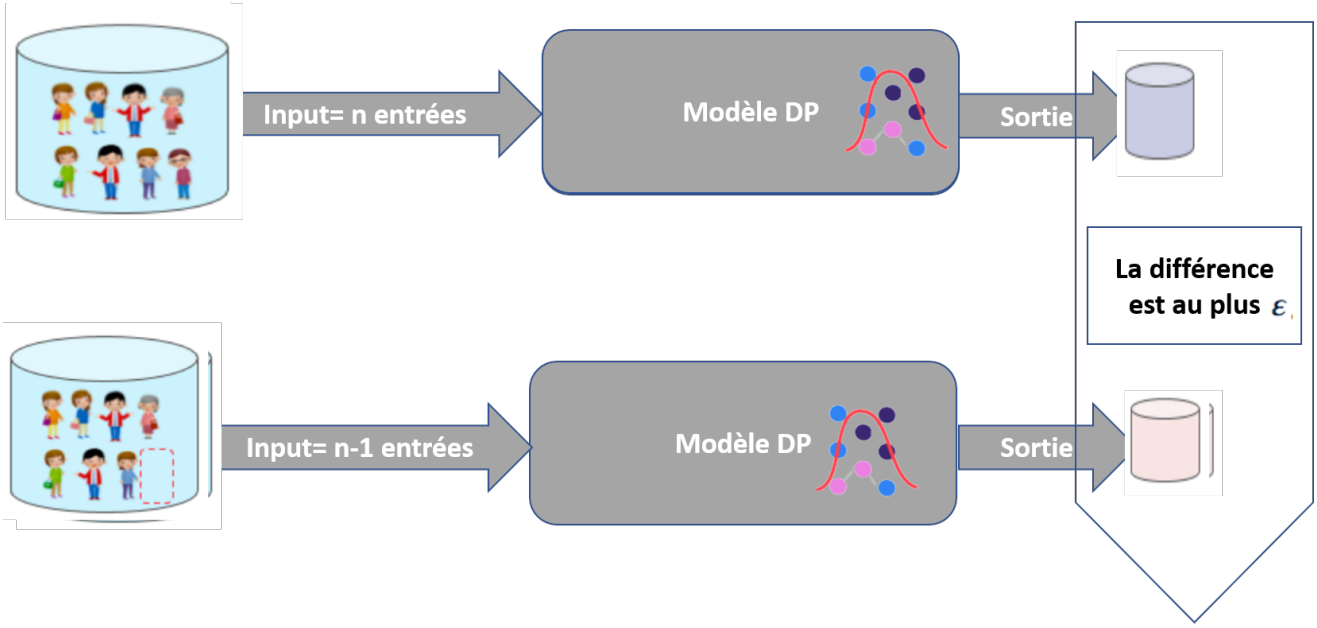
3.2 Contexte d'utilisation avec apprentissage automatique

Les préoccupations liées à la confidentialité et à la sécurité deviennent importantes au fur et à mesure qu'augmente la quantité de données qu'une organisation collecte et utilise à des fins d'analyse. Particulièrement dans le cadre d'apprentissage automatique, plus il y a de données utilisées pour effectuer l'apprentissage des modèles, plus ceux-ci sont précis et efficaces. Cependant, lorsque des informations personnelles sont utilisées pour ces analyses, il est particulièrement important que les données restent confidentielles tout au long de leur utilisation. Or, dans les scénarios traditionnels de l'apprentissage automatique, les analystes utilisent généralement des données brutes qui sont stockées dans des fichiers et des bases de données pour entraîner les modèles. Cette pratique est problématique, car elle peut entraîner une violation de données à caractère personnel. Ainsi, la confidentialité différentielle tente de résoudre ce problème en ajoutant du « bruit » ou une composante aléatoire aux données afin que les utilisateurs ne puissent pas identifier des points de données individuels. Un tel système offre au moins une possibilité de « démenti raisonnable ». Par conséquent, la confidentialité des individus est préservée avec un impact limité sur la précision des données.

Considérons maintenant le contexte d'apprentissage automatique et d'apprentissage profond en particulier, où l'on cherche à modéliser un problème en entraînant un modèle sur un jeu de données de taille importante. Le modèle apprend à partir des données et il ne devrait pas exposer des informations sur le jeu de données d'entraînement. Comme dans le contexte de base de données, avec la confidentialité différentielle, le souhait est de ne pas dévoiler la présence d'un individu dans le jeu de données d'entrée en ayant accès au modèle entraîné. En effet, pour clarifier l'idée derrière la confidentialité différentielle, on propose la FIGURE 3.7. La FIGURE 3.7 montre que si on entraîne un modèle sur une base de données avec n entrées et puis on fournit une donnée à ce modèle, on obtient une sortie O. Ensuite, si on entraîne le même modèle sur une base de données qui est identique à la précédente sauf pour une entrée c'est-à-dire $n-1$ entrées et puis on fournit une donnée à ce dernier, on obtiendra une sortie O' où O et O' sont différentes d'au plus ε comme montré sur la FIGURE 3.7. Par ailleurs, ε représente un paramètre important qui mesure le niveau de confidentialité. De même, l'algorithme A présenté à la section 3.1 serait le processus d'entraînement dont la sortie est le modèle.

En somme, le compromis recherché est donc de construire un modèle utile, qui repose sur des informations du jeu d'entraînement, sans révéler trop d'informations sur chaque exemple particulier.

FIGURE 3.7 – Modélisation avec entraînement pour CD



En conclusion, nous avons présenté dans ce chapitre les principes, propriétés et formes de confidentialité différentielle, ainsi que quelques techniques d'ajouts de bruits satisfaisantes les propriétés de confidentialité différentielle. Aussi, nous avons défini le contexte d'utilisation de la confidentialité différentielle dans les modèles basés sur l'apprentissage automatique. Dans le chapitre suivant, nous présenterons une brève revue de littérature sur la détection de fraude sur Internet.

Chapitre 4

Revue de littérature

Introduction

Ce chapitre présente une brève revue de littérature sur la détection de fraude sur Internet. Comme il existe de nombreuses approches proposées, nous retenons celles ayant reçu le plus d'intérêt de la part de la communauté scientifique et nous présentons en donnant à chaque fois, les motivations sous-jacentes derrière les solutions proposées. Nous commençons par présenter les méthodes d'apprentissage basées sur des modèles simples pour ensuite passer à celles utilisant des modèles hybrides. Nous nous focalisons, en particulier, sur l'étude de performances des modèles proposés. Ensuite, nous porterons notre attention sur les travaux qui proposent des méthodes de détection de fraudes par valeurs aberrantes et des solutions pour le problème de déséquilibre des données. Enfin, nous présenterons les méthodes basées sur les réseaux de neurones artificiels.

4.1 Modèles simples

Plusieurs techniques d'apprentissages supervisées ont été utilisées pour la détection des fraudes sur Internet. Mohammed Emad et autres [60] ont proposé, pour la détection des transactions frauduleuses, différents algorithmes d'apprentissage supervisés comme les arbres de décision, la classification Naïve de Bayes, la régression des moindres carrés, la régression logistique et MVS. Xuan, Shiyang, et autres [81] ont proposés deux méthodes basées sur les forêts aléatoires pour entraîner leurs modèles sur les caractéristiques comportementales des transactions normales et anormales. Il est à noter que les forêts aléatoires sont un cas particu-

lier de « bagging » appliqué aux arbres de décision de type CART. En effet, en plus du principe de bagging, les forêts aléatoires ajoutent de l'aléa au niveau des variables. Pour chaque arbre, on sélectionne un échantillon d'individus appelé Bootstrap et à chaque étape, la construction d'un noeud de l'arbre se fait sur un sous-ensemble de variables tirées aléatoirement. En outre, les forêts aléatoires permettent de faire la moyenne des prévisions de plusieurs modèles indépendants pour réduire la variance et donc l'erreur de prévision. Bien que les forêts aléatoires obtiennent de bons résultats sur de petits ensembles de données, il y a encore des problèmes en cas de données déséquilibrées. L'algorithme des k plus proches voisins (KNN) est également utilisé par N. Malini et autres [66] pour détecter les fraudes par carte de crédit. En effet, KNN est utilisé pour classer chauqe transaction bancaire en calculant son point le plus proche. Si une nouvelle transaction arrive et que ce point est proche d'une transaction frauduleuse, il est alors identifié comme une fraude. Bien que cet algorithme soit simple et facile à mettre en oeuvre, il ralentit, cependant, considérablement au fur à mesure que le nombre de transactions augmente puisqu'il parcourt l'ensemble des transactions pour calculer les distances.

Par ailleurs, les algorithmes d'apprentissage supervisés nécessitent des transactions étiquetées pour fonctionner. Étant donné le volume gigantesque de transactions bancaires et que les techniques des fraudeurs changent régulièrement l'étiquetage des transactions bancaires devient alors une opération très coûteuse.

4.2 Modèles hybrides

Chaque technique d'apprentissage souffre d'un certain nombre de lacunes. De ce fait, une approche hybride composée de plusieurs approches d'apprentissage peut combler certaines limites et permet ainsi d'obtenir des performances améliorées. Pour cette raison, J. Esmaily et R. Moradinezhad [55] ont proposé une approche hybride composée d'un réseaux de neurones artificiel et d'un arbre de décision pour la détection de fraude. La solution proposée procède selon deux phases : dans une première phase, les résultats de classification issus de l'arbres de décision et du perceptron multicouche sont utilisés pour générer un nouvel ensemble de transactions. Dans la deuxième phase, le nouvel ensemble de transaction permet d'alimenter le perceptron multicouche afin de faire la classification. Ce modèle améliore la fiabilité en donnant un taux de fausses alerte très faible. Panigrahi et Behera [75] ont proposé une approche hybride pour la détection de fraude par carte de crédit en utilisant le clustering flou et les réseaux de neurones. Cette approche comporte également deux phases. En effet, dans la première phase, les auteurs ont utilisé un algorithme de clustering k-means pour générer un

score des transactions suspectes. Dans la phase suivante, si une transaction est suspecte, elle est introduite dans le réseau neuronal pour déterminer si elle est vraiment frauduleuse ou non. Aussi, A. Agrawal et autres [76] ont proposé de tester les transactions en utilisant le modèle de Markov caché, basé sur la technique d'analyse du comportement et les algorithmes génétiques. En effet, le modèle de Markov caché généralise le modèle de Markov observable car il produit une séquence en utilisant deux suites de variables aléatoires ; l'une cachée et l'autre observable [31]. À cet égard, le modèle de Markov caché permet de conserver l'enregistrement des transactions précédentes tandis que la technique basée sur l'analyse du comportement permet de faire le regroupement des ensembles de transactions et l'algorithme génétique assure l'optimisation. En effet, l'algorithme génétique permet d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte ou que la solution est inconnue pour le résoudre en un temps raisonnable. Plus précisément, l'algorithme génétique utilise la notion de sélection naturelle et l'applique à une population de solutions potentielles au problème donné [2].

De même, S. Maes et autres [73] ont proposé de détecter les fraudes à la carte de crédit en utilisant les réseaux bayésiens et les réseaux de neurones artificiels. Selon les auteurs, les réseaux bayésiens après une courte durée d'entraînement donnaient de bons résultats et leur vitesse est améliorée par l'utilisation des réseaux de neurones artificiels. E. Duman et M. H. Ozcelik [74] ont proposé une méthode qui améliorerait les systèmes de détection des fraudes par cartes de crédit. Ils ont conçu un système pour attribuer à chaque transaction un certain score et sur la base de ce score, la transaction est jugée. Pour cela, ils ont combiné les réseaux de neurones avec la recherche de dispersion. A. Pouramirarsalani et autres [41] ont proposé également une nouvelle méthode de détection de fraude en utilisant une approche hybride basée sur la sélection de caractéristiques principales et d'algorithme génétique. D'abord, ils ont observé les caractéristiques principales des transactions. Ensuite, ces caractéristiques ont été utilisées pour détecter toute caractéristique inhabituelle et en la signalant comme étant frauduleuse. Par ailleurs, l'algorithme génétique a été utilisé dans ce cas pour des problèmes d'optimisation et de recherche. P. Chougule et autres [78] ont proposé un algorithme combinant un K-means et un génétique pour la détection de fraude. Dans leur article, ils ont utilisé l'algorithme k-means pour regrouper les transactions en fonction des valeurs d'attributs distinctes. Ensuite, l'algorithme génétique est utilisé pour l'optimisation, car l'augmentation de la taille de l'algorithme k-means d'entrée produisait des valeurs aberrantes. De même, S. Fashoto et autres [69] ont utilisé une approche hybride de clustering K-means avec le perceptron multi couches et le modèle de Markov caché. Le clustering K-means a été utilisé afin de re-

grouper les transactions frauduleuses présumées dans un cluster. La sortie de cette étape est utilisée pour entraîner le perceptron multi couches et le modèle de Markov caché, qui classent ensuite les transactions entrantes. En résumé, ils ont trouvé que la précision de détection du perceptron multi couches avec le clustering K-means est plus élevée que celle du modèle de Markov caché avec le clustering K-means. M.R. HaratiNik et autres [62] ont proposé une fusion entre un système expert flou et l'analyse « Foggbehavioural » en le nommant le modèle hybride FUZZGY. En effet, le système expert flou identifie logiquement la réfutation entre les activités actuelles et les activités historiques. Le modèle « Foggbehavioural » décrit en deux dimensions le comportement d'un individu impliqué dans une transaction : sa motivation et sa capacité à commettre une fraude. Le poids de la tendance à la fraude est ensuite calculé pour chaque individu suivi du degré de suspicion pour les transactions entrantes.

P. Krishna et autres [70] ont proposé une approche basée sur la fusion de la théorie de Dempster-Shafer aussi appelée théorie des croyances et l'apprentissage bayésien. Dans leur article, les auteurs décident si une transaction est suspecte ou peu suspecte en utilisant quatre étapes principales. Ces quatres étapes sont basées sur des règles : l'additionneur Dempster-Shafer, l'historique des transactions et l'apprenant bayésien. Dans le premier composant, la mesure dans laquelle la transaction entrante a dévié est déterminée de manière à obtenir le niveau de suspicion. Le deuxième composant combine plusieurs de modèles pour obtenir une croyance initiale et une croyance globale. La transaction est alors classée comme suspecte ou insoupçonnée selon sa croyance. Une fois jugée suspecte, la croyance peut en outre être renforcée ou affaiblie en raison de sa similitude comparative avec un historique transactionnelle frauduleux ou authentique en utilisant l'apprentissage bayésien. La théorie de Dempster-Shafer est une théorie mathématique basée surla notion de preuves utilisant les fonctions de croyance et le raisonnement plausible. Le but de cette théorie est de combiner des preuves distinctes pour calculer la probabilité d'un événement. Tandis que le théorème de Bayes décrit la probabilité d'un événement, basée sur la connaissance préalable des conditions qui pourraient être liées à l'événement.

En conclusion, l'objectif du développement des modèles hybrides est d'élaborer une technique coûteuse qui prend du temps à l'apprentissage mais qui donne des résultats très précis. C'est pour cela que ces approches sont souvent accompagnées d'une technique d'optimisation permettant de réduire le coût du système et faire en sorte que le système s'entraîne rapidement.

4.3 Performance des modèles

Dans le but de déterminer les meilleures techniques de détection de fraudes, plusieurs travaux de recherches ont été consacrés à la comparaison et à l'étude de la performance des diverses solutions proposés. T. Razoogi et autres [80] ont proposé un système de détection de fraude en utilisant la logique floue et les réseaux de neurones. Ils ont découvert que les réseaux de neurones étaient à 33% plus précis que la logique floue. Les données existantes dans le système ont été utilisées pour la prise de décision. En utilisant la logique floue, chaque donnée reçoit un attribut d'appartenance et qui est validé par la suite avec l'utilisation des réseaux de neurones. Y. Sahin et autres [86] ont proposé une solution pour la détection de la fraude avec les cartes de crédit en utilisant une combinaison de machines vectorielles et d'arbres de décision. Les arbres de décision ont surpassé les machines vectorielles lorsque la taille de l'ensemble de données était petite mais avec l'augmentation de la taille de l'ensemble des données, les machines vectorielles ont atteint la précision des arbres de décision. S. Bhattacharya et autres [72] ont fait une étude comparative détaillée sur SVM et les forêts aléatoires avec la régression logistique. Ils ont conclu, au moyen d'expériences, que la technique par forêts aléatoires est plus précise, suivie par la régression logistique et SVM. Y. Jain et autres [82] ont compilé une comparaison des techniques d'apprentissages supervisées pour la détection de fraude par carte de crédit en utilisant des métriques de mesures telles que la précision, le taux de fausses alarmes et l' « Accuracy ». Selon les auteurs, le réseau neuronal et le réseau bayésien naïf donnent les plus grandes précisions. Le K-plus proche voisin, SVM et les arbres de décisions offrent un niveau de précision moyen, tandis que le système à base de logique floue et la régression logistique donnent une précision faible par rapport aux autres. Par ailleurs, un taux de détection élevé est obtenu par utilisation d'un réseau neuronal, les systèmes bayésiens naïfs, flous et l'algorithme des K-plus proches voisins. La régression logistique, SVM et les arbres de décision fournissent, quant à eux un faible taux de détection. Par ailleurs, un faible taux de faux positif est donné par le réseau neuronal uniquement, contrairement à SVM qui donne un taux de faux positif élevé. Il est à souligner que même si les réseaux de neurones artificiels et les réseaux bayésiens naïfs fonctionnent mieux, ils sont toutefois coûteux à entraîner, contrairement, par exemple, à la régression logistique. Pour conclure, une étude comparative permet de choisir une bonne approche pour la détection de fraude.

Cependant, les résultats dépendent des paramètres qui ont été pris en en considération lors de l'étude. Il faut faire au préalable une évaluation des paramètres qui vont permettre une bonne prise de décision en tenant compte de l'environnement et le contexte de l'étude.

4.4 Détection de fraudes par valeurs aberrantes

La détection des valeurs aberrantes est une autre méthode utilisée pour détecter les fraudes avec les techniques d'apprentissage supervisées et non supervisées. En effet, la méthode de détection des valeurs aberrantes supervisée étudie et classe les valeurs aberrantes à l'aide d'un ensemble de données d'apprentissage. De plus, en méthodes supervisées, les modèles sont entraînés pour faire la distinction entre un comportement frauduleux et non frauduleux afin que de nouvelles observations puissent être classées. Les méthodes supervisées nécessitent toutefois une identification précise des transactions frauduleuses dans une base de données historiques.

À l'inverse, la détection des valeurs aberrantes par des méthodes non supervisées utilise la technique de regroupement des données et de la réduction de dimension. Habituellement, le résultat d'un apprentissage non supervisé est une nouvelle explication ou représentation des données d'observation, qui peuvent aider pour la prise de décisions futures. L'apprentissage non supervisé ne nécessite pas d'étiquetage préalables de la base de données, mais détecte plutôt des changements de comportement ou des transactions inhabituelles. Généralement, ces méthodes modélisent une distribution de référence qui représente les comportements normaux, puis détectent les observations qui s'écartent de cette norme. Aussi, N. Malini et autres [65] mentionnent que la méthode de détection des valeurs aberrantes basée sur l'apprentissage non supervisé est préférable, pour détecter la fraude par carte de crédit, à l'apprentissage supervisé par valeurs aberrantes, car l'apprentissage non supervisé par valeurs aberrantes ne nécessite pas d'informations préalables pour étiqueter les données comme frauduleuses et permet ainsi d'agir de façon anticipée. De plus, Wen-Fang et autres [68] ont proposé une méthode de détection de fraude par carte de crédit en utilisant l'extraction des valeurs aberrantes basée sur une distance. Il s'agit de détecter les transactions qui présentent un comportement anormal, c'est-à-dire les transactions qui ne sont pas authentiques. Dans leur travail, ils ont utilisé des attributs qui permettent de connaître le comportement de l'utilisateur et en fonction de la valeur de ces attributs, ils ont calculé la distance entre la valeur observée de cet attribut et sa valeur prédéterminée. Par ailleurs, un avantage d'utiliser des méthodes non supervisées par rapport aux méthodes supervisées c'est de pouvoir agir de façon proactive afin d'anticiper sur les cas de fraudes non connus.

4.5 Problème de déséquilibre des données

Un problème majeur dans la détection des évènements rares est le déséquilibre des données. En effet, un modèle de classification peut devenir inefficace si les données utilisées pour l'entraîner sont déséquilibrées puisque lorsque les exemples d'une classe dépassent largement le nombre d'exemples des autres classes, les algorithmes traditionnels d'exploration de données ont tendance à favoriser la classification des exemples comme appartenant à la classe majoritairement représentée. Ainsi, un tel modèle serait inefficace pour identifier des exemples de la classe minoritaire, qui est souvent la classe d'intérêt (la classe positive). Au fond, ce sont les exemples de cette classe positive qui entraînent le coût le plus élevé en erreurs de classification. Dans le cadre de la détection des fraudes bancaires, le nombre de transactions frauduleuses est souvent très inférieur à celui des transactions légitimes puisque la fraude par carte de crédit est relativement un événement rare en comparaison au nombre total des transactions effectuées. En conséquence, ceci entraîne un problème de déséquilibre ou de distribution biaisée des données. Pour cette raison, il est nécessaire d'utiliser des techniques d'équilibrage pour que le modèle puisse identifier efficacement ces transactions importantes mais rarement rencontrées.

L'échantillonnage est l'une des techniques les plus utilisées pour atténuer le problème du déséquilibre de données. Ainsi, les trois méthodes fréquemment utilisées pour l'ajustement de la distribution des classes sont le sous-échantillonnage de la classe majoritaire, le sur-échantillonnage de la classe minoritaire, ou une combinaison de ces deux méthodes. En effet, le sur-échantillonnage équilibre la distribution des classes dans les données d'entraînement en ajoutant des exemples à la classe minoritaire tandis que le sous-échantillonnage supprime des exemples de la classe majoritaire. Essentiellement, le sur-échantillonnage aléatoire équilibre un ensemble de données en dupliquant des exemples de la classe minoritaire jusqu'à ce qu'un ratio de classe souhaité soit atteint alors que le sous-échantillonnage aléatoire supprime des exemples au hasard de la classe majoritaire jusqu'à ce que l'équilibre souhaité soit atteint. Par exemple, Anusorn Charleonnan [42] mentionne que le déséquilibre des jeux de données présente de nombreuses caractéristiques qui émergent lors de la classification. Ainsi, il a utilisé le sous-échantillonnage aléatoire pour résoudre le problème du déséquilibre de classe en éditant la distribution de classe des ensembles de données d'entraînement.

4.6 Réseaux de neurones artificiels

L'idée principale des réseaux de neurones artificiels c'est d'imiter l'apprentissage du cerveau humain. La plus petite entité des réseaux de neurones est appelée un perceptron. Plusieurs perceptrons représentant des noeuds sont alors connectés en réseau comme pour le cerveau humain. Chaque noeud a une communication pondérée avec plusieurs autres noeuds de la couche adjacente. Cette pondération est assurée par un poids qui est simplement un nombre à virgule flottante. Ce dernier peut être ajusté lors de l'apprentissage du réseau. Les entrées sont transmises des noeuds d'entrées via les couches masquées aux noeuds de sortie et chaque noeud peut apprendre et s'ajuster pour devenir plus précis et approprié. En outre, on parle d'un perceptron multicouche lorsqu'il s'agit d'un réseau de neurone artificiel composé de 3 couches au minimum.

Plusieurs chercheurs ont proposé des solutions au problème de détection de fraude bancaires en utilisant les réseaux de neurones artificiels. R. Patidar et autres [71] ont affirmé que les réseaux de neurones artificiels produisaient de meilleurs résultats que les forêts aléatoires. M. K. Mishra et autres [61] ont abordé le problème de détection de fraude par carte de crédit en proposant une approche hybride basée sur les réseaux de neurones à liens fonctionnels Chebyshev (CFLNN). En effet, un CFLNN est un réseau de neurones d'ordre supérieur avec une faible complexité de calcul. Non seulement, il n'a pas de couches cachées. Le vecteur d'entrée de CFLNN est étendu fonctionnellement pour obtenir des solutions non linéaires[40]. D'un autre côté, il a été démontré par J. Wang, M. Xu et autres [57] que plus le réseau est profond, plus il obtient de meilleurs résultats que ceux avec un plus petit nombre de couches. Dans ce même article, les auteurs ont aussi comparé les performances de CFLNN, MLP et les arbres de décision. En résumé, les résultats de leur étude suggèrent que MLP surpasse le CFLNN et les arbres de décision en matière de détection de fraude.

De plus, G.Rushin, C.Stancil et autres [53] ont utilisé des algorithmes d'apprentissage profond auto-encodeurs pour la détection des fraudes et ont constaté que les résultats produits étaient meilleurs que les arbres à gradient boosté et la régression logistique. Y. Pandey a également utilisé un réseau neuronal multicouche basé sur H2O pour trouver des modèles de fraude par carte de crédit. En effet, le framework H2O est un « open source » pour l'analyse prédictive de données sur le Big Data. Essentiellement, il propose des outils pour la manipulation et la préparation de données, des algorithmes de modélisation, supervisées, non-supervisées ou de réduction de dimensionnalité[16]. L'auteur a démontré dans son étude[84] que le modèle d'apprentissage profond basé sur le H2O exhibe moins d'erreurs et donc améliore plus la précision

que les algorithmes d'apprentissage automatique classique. C. Wang et autres [51] ont testé un réseau neuronal sur un ensemble de données de transaction par carte de crédit européens. Ils ont utilisé un réseau de neurones à propagation arrière qui est composé de 2 couches d'entrées, 20 couches cachées et 2 couches de sortie, ce réseau de neurones a été optimisé avec l'algorithme Whale. Grâce à l'algorithme d'optimisation, ils ont obtenu des résultats exceptionnels sur 500 échantillons de test : précision de 96,40% et rappel de 97,83%.

De même, A. Pumsirirat et autres [44] ont proposé une approche basée sur un modèle d'auto-encodeur profond et de la machine de Boltzmann qui permet de construire une représentation des transactions normales pour trouver des anomalies à partir de modèles normaux. En effet, la machine de Boltzmann restreinte est un type de réseau de neurones artificiels pour l'apprentissage non supervisé [35]. Elle est couramment utilisée pour avoir une estimation de la distribution probabiliste d'un jeu de données. Aussi, elle permet de faire la réduction de la dimensionnalité et la sélection et l'extraction de caractéristiques. L'avantage de cette approche est qu'elle ne nécessite aucun effort de labellisation des transactions bancaires car il s'agit d'une technique d'apprentissage non supervisée, qui permet donc de détecter les fraudes de façon proactive. Cependant, les approches décritent ci-dessus pour la détection des fraudes sont généralement entrainées en utilisant uniquement des données d'apprentissage propres à chaque institution financière et ceci pour des raisons de conflit d'intérêt et de confidentialités de l'information véhiculées par ces données. En conséquence, les modèles obtenus manquent de précisons, notamment ceux « d'apprentissage profond » qui nécessitent des ensembles de données plus volumineux [44]. Pour faire face à cette réalité, Y. Wang et autres [87] ont développé un algorithme de réseau de neurones profonds distribué préservant la confidentialité, qui permet aux banques et autres entités de partager leurs données sans révéler d'informations sensibles. L'approche de ces auteurs a été expérimentée en utilisant un ensemble de données du monde réel contenant des transactions bancaires. Il est à noter que l'étude [87] produit des performances comparables aux méthodes ne prenant en considération l'aspect confidentialité des données.

Cependant, bien que l'étude [87] propose une approche pour la préservation de la confidentialité des données des banques participantes à la construction du modèle de détection de fraude, la solution ne permet pas d'anticiper sur les cas de fraudes. Étant donné que les techniques et le comportement des fraudeurs sont en perpétuels changements [7], la mise en oeuvre et l'évaluation d'autres topologies d'apprentissage en profondeur dans la détection de la fraude restent donc largement inexplorées. Ainsi, nous proposons dans ce travail une nouvelle approche d'apprentissage profond distribué sécurisé qui tient compte du caractère

changeant du comportement des fraudeurs et garantie la confidentialité des données d'entraînement utilisées. Dans le chapitre suivant, nous présenterons notre approche et les résultats obtenus.

Chapitre 5

Détection de e-fraude bancaire par apprentissage profond distribué sécurisé

Introduction

Dans l'architecture traditionnelle d'apprentissage automatique, les données et le modèle sont colocalisés. Il peut y avoir un ou plusieurs producteurs ou propriétaires de données, mais toutes ces données sont rassemblées dans un centre de données pour être utilisées dans l'entraînement d'un modèle. Cependant, le centre d'entraînement n'est pas nécessairement un organisme de confiance et cela peut être la cause de fuites de données, violant ainsi leurs confidentialité. Cette pratique représente donc un problème de confidentialité. Pour répondre à cette préoccupation, nous proposons un mécanisme d'apprentissage profond sécurisé qui permet d'entraîner efficacement un modèle sans divulguer les informations confidentielles contenues dans les données d'apprentissage. Dans notre cas, la sécurité sous-entend l'utilisation d'un mécanisme de confidentialité différentielle.

Dans la suite de ce chapitre, nous présentons un nouveau mécanisme basé sur l'apprentissage profond distribué utilisant la confidentialité différentielle et son utilisation dans une application de détection de fraude bancaire sur Internet. D'abord, nous présenterons l'architecture globale de notre système d'apprentissage profond distribué avec mécanisme de confidentialité différentielle. Ensuite, nous présenterons l'application de notre approche au cas de détection de fraude bancaire sur Internet. Pour cela, nous présenterons les différents outils utilisés pour son implémentation, le DataSet utilisé, ainsi que les étapes du prétraitement des données effectuées. Finalement, nous comparons les résultats du classificateur obtenus par

notre approche avec un autre classificateur basé sur l'apprentissage profond supervisé afin de montrer ses mérites.

5.1 Mécanisme d'apprentissage profond distribué sécurisé

Dans cette section, nous présentons notre mécanisme d'apprentissage profond distribué et sécurisé. Cela inclus l'approche proposée, l'architecture et le mécanisme de confidentialité différentielle choisi.

5.1.1 Apprentissage profond distribué

L'apprentissage profond distribué permet d'entraîner, en parallèle, le modèle à partir de plusieurs machines ou noeuds. Il est important de souligner ici que cette distribution permet de contribuer à assurer la confidentialité des données d'entraînement. Il existe principalement trois approches pour faire l'apprentissage profond distribué : l'approche basée sur le parallélisme des données, l'approche basée sur le parallélisme des modèles et l'approche hybride.

Approche basée sur le parallélisme des données Le parallélisme des données est une technique qui consiste à partitionner les données d'entraînement et les distribuer sur plusieurs noeuds. Généralement, le nombre de partitions est égal au nombre de noeuds de calcul. Ainsi, chaque noeud possédant une partition indépendante des données fait des calculs locaux sur sa propre partition. Les noeuds synchronisent alors leurs états de paramètres jusqu'à la convergence de l'algorithme d'apprentissage.

Approche basée sur le parallélisme des modèles Contrairement à l'approche basée sur le parallélisme des données, une approche basée sur le parallélisme des modèles permet de partitionner le modèle d'apprentissage automatique en attribuant les partitions du modèle aux noeuds de calcul. Ainsi, les noeuds de calcul exécutent simultanément des partitions différentes du modèle sur le même ensemble de données. En conséquence, le besoin de communication est réduit, car les noeuds n'ont qu'à synchroniser les paramètres partagés. Toutefois, cette approche est plus complexe à mettre en oeuvre que l'approche par parallélisme des données.

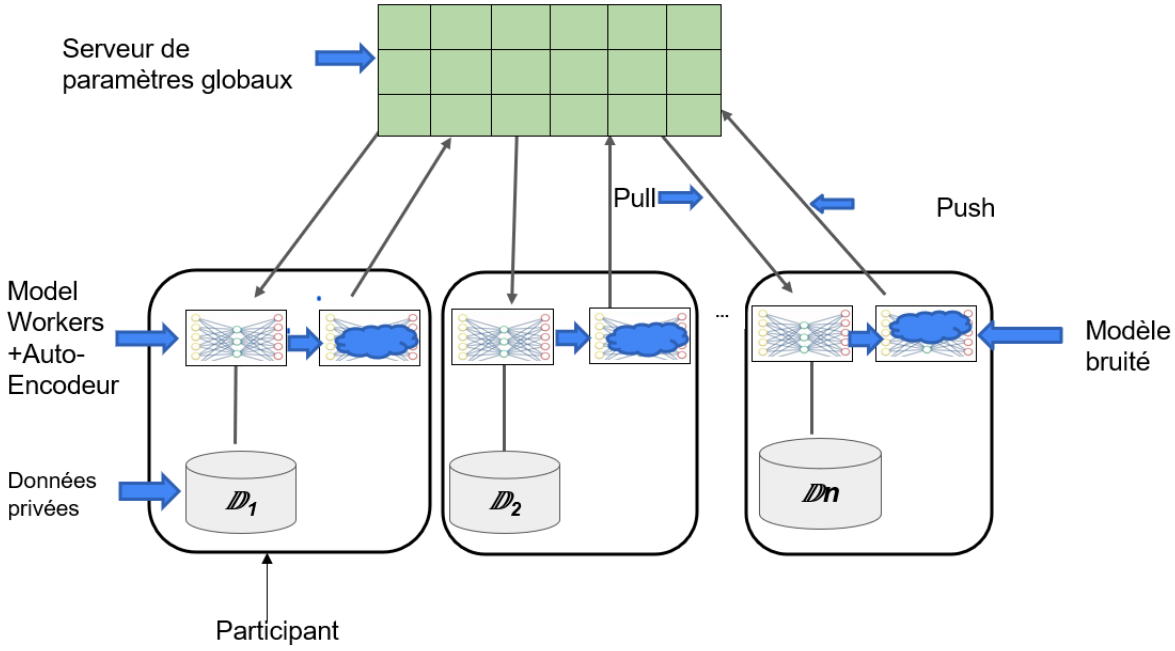
Approche hybride Cette approche consiste à utiliser à la fois le parallélisme des données et celui des modèles. En effet, une partie des noeuds de calcul est utilisée pour le partitionnement des données et le reste pour le partitionnement du modèle.

Étant donné que l'approche basée sur le parallélisme des données permet de traiter localement les données et permettant ainsi d'assurer la confidentialité des données d'entraînement, nous utilisons donc cette approche pour notre mécanisme d'apprentissage profond distribué sécurisé.

5.1.2 Architecture

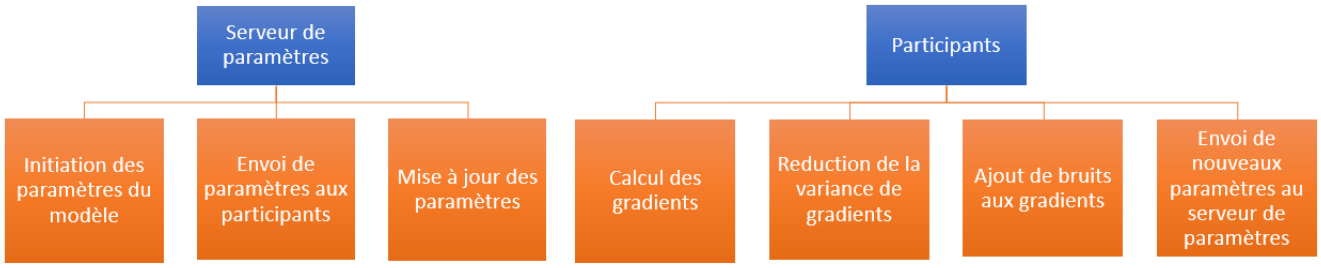
Notre architecture est présentée à la FIGURE 5.1. Dans cette architecture, le serveur de paramètres est responsable de l'initialisation et de l'envoi des paramètres du modèle. Par la suite, chaque participant construit localement un auto-encodeur de manière itérative en sélectionnant au hasard des échantillons d'entraînement. Ceci permet au participant de calculer les gradients locaux sur ses propres données en fonction des paramètres reçus du serveur de paramètres. Durant une itération, chaque exécution est indépendante des exécutions des autres participants. Lorsqu'une itération se termine, chaque participant réduit la variance des gradients et injecte un bruit gaussien, et puis envoi les nouveaux paramètres du modèle obtenus de cette itération au référentiel de façon asynchrone. Le serveur de paramètres aussi appelé référentiel s'occupe de son coté de l'agrégation et de la mise à jour des paramètres du modèle. Enfin, Ces nouveaux paramètres sont diffusés à tous les participants. Ce processus se répète jusqu'à la convergence du calcul du modèle.

FIGURE 5.1 – Apprentissage profond distribué sécurisé



La FIGURE 5.2 présente les tâches au niveau du serveur de paramètres et de chaque participant dans notre modèle d'apprentissage profond distribué et sécurisé.

FIGURE 5.2 – Tâche client-serveur de l'apprentissage profond distribué sécurisé



Il faut noter que dans cette architecture le serveur de paramètres chargé de la mise en place du modèle global, n'a aucune connaissance des données d'entraînement des participants. En effet, le serveur de paramètres s'occupe uniquement de l'initialisation et de la mise à jour

des paramètres du modèle. En outre, les participants sont responsables de la confidentialité de leurs données et de l'entraînement du modèle. Par ailleurs, le bruit gaussien permet d'avoir une grande variance de bruit.

Dans ce qui suit, nous présentons différents mécanismes de confidentialité différentielle et motivons notre choix du mécanisme adopté.

5.1.3 Mécanisme de confidentialité différentielle

Dans cette section, nous présenterons quelques méthodes d'ajouts de bruit lors du processus d'entraînement d'un modèle d'apprentissage automatique en montrant les forces et limites de chacune d'elles afin de choisir la meilleure. Ainsi, nous présentons les méthodes de perturbation de sortie [45], de perturbation de l'objectif [58], de perturbation d'entrées et de perturbation des gradients.

La perturbation de sortie La méthode de perturbation de sortie est couramment utilisée car elle est simple à mettre en œuvre, en ajoutant uniquement du bruit au modèle final. De plus, le niveau de la confidentialité est plus facile à contrôler. Dans la méthode de perturbation de sortie, le bruit est directement ajouté au modèle comme montré par l'équation (5.1). Dans ce contexte, nous désignons le modèle par ses paramètres [83].

$$_{confi} = argmin[L(\theta)] + z \tag{5.1}$$

où z est le bruit garantissant la confidentialité différentielle et L la fonction objective définit dans la section 2.5.1.

La perturbation de la fonction objective Dans la méthode de perturbation de la fonction objective, le bruit est ajouté à la fonction objective elle-même comme le montre l'équation (5.2) [83].

$$L_{confi}(\theta) = L(\theta) + \frac{1}{n} z^T \theta \tag{5.2}$$

où n est le nombre de données contenues dans l'ensemble d'apprentissage et z est le bruit avec une densité de $\frac{1}{\alpha}\exp(-e\|z\|)$. Le paramètre α représente la constante de normalisation. Ensuite, la fonction objective perturbée $L_{confi}(\theta)$ est directement optimisée comme montre l'équation (5.3).

$$\theta_{confi} = argmin[L_{confi}(\theta)] \tag{5.3}$$

Cette méthode est de nos jours rarement utilisée car c'est toujours difficile d'optimiser la fonction objective perturbée et ses performances ne sont pas satisfaisantes.

Les méthodes de perturbation de sortie, de la fonction objective permettent d'ajouter du bruit au modèle final, et à la fonction objective en respectant les standards de la confidentialité différentielle. Cependant, ces méthodes de perturbations ne protègent pas les données d'origines.

Perturbation d'entrées Cette méthode propose d'ajouter du bruit aux instances de données d'entraînement avant de les utiliser pour l'apprentissage. Cette technique est très utile dans le cas où les données locales doivent être centraliser avant l'apprentissage. Dans ce cas, le modèle d'apprentissage automatique est entraîné avec des instances de données perturbées [83]. La FIGURE 5.3 montre l'algorithme d'apprentissage avec perturbation d'entrées.

FIGURE 5.3 – Apprentissage avec perturbation d'entrées

Algorithm 1 Apprentissage avec perturbation d'entrées :

Entrées: Dataset D, itération T, le taux d'apprentissage α

Pour toutes les instances de données (xi, yi) dans D, ajoutez un bruit z :
$(xi, yi) \leftarrow (xi + z, yi)$
La nouvelle donnée $(xi + z, yi)$ est dénotée par 'donnée pertubée'.
Entraîner le modèle avec la donnée perturbée.
La fonction objective devient :
$L(\theta) \leftarrow \frac{1}{n}\sum_{i=1}^{n} L(\theta, x_i + z, y_i)$
Ainsi, la descente de gradient devient :
for t = 0 à T-1 **do**
    $\theta^{t+1} = \theta^t - \alpha\frac{1}{n}\sum_{i=1}^{n}\Delta L(\theta)$
end for

Dans l'algorithme de la FIGURE 5.3, le bruit aléatoire $z \in \mathscr{R}^d$ et chaque élément $z_i \sim \mathscr{N}(0, \sigma^2)$ est échantillonné indépendamment où $\mathscr{N}(0, \sigma^2)$ est la loi normale. Le nombre de données contenues dans l'ensemble d'apprentissage est représenté par n. De plus, on observe que la méthode ajoute du bruit aux données d'origines, entraînant ainsi une perturbation sur le gradient et éventuellement provoquant une perturbation sur les paramètres du modèle. Elle

protège donc simultanément les données d'origines et le modèle final [83]. Malheureusement, l'inconvénient de la méthode de perturbation de l'entrée est qu'il n'est pas trivial de déterminer la quantité du bruit nécessaire pour respecter les propriétés de la confidentialité différentielle. Aussi, elle peut-être trop bruyante provoquant ainsi la précision du modèle construit.

Par ailleurs, la plupart des algorithmes d'apprentissage automatique et en particulier ceux d'apprentissage profonds utilisent l'algorithme de descente de gradient, ainsi la méthode de la perturbation du gradient est plus populaire car mieux adaptée pour cette technique d'apprentissage.

Descente de gradient stochastique différentiellement privée L'idée de base de cette approche, appelée descente de gradient stochastique différentiellement privée, est de perturber les gradients utilisés. Deux modifications sont apportées pour garantir que la descente de gradient stochastique soit un algorithme différentiellement privé. Tout d'abord, la sensibilité de chaque gradient doit être limitée. En d'autres termes, il faut limiter dans quelle mesure chaque point d'entraînement individuel, échantillonné dans un mini-lot, peut influencer le calcul du gradient résultant. Intuitivement, cela permet de déterminer dans quelle mesure chaque point d'apprentissage peut éventuellement avoir un impact sur les paramètres du modèle. Par la suite, nous devons randomiser le comportement de l'algorithme pour qu'il soit statistiquement impossible de savoir si un point particulier a été inclus ou non dans l'ensemble d'apprentissage. Ce test d'inclusion peut être fait en comparant les mises à jour de la descente du gradient stochastique lorsqu'il fonctionne avec ou sans un point particulier de l'ensemble d'entraînement. La randomisation de l'algorithme est réalisée en échantillonnant un bruit aléatoire et en l'ajoutant aux gradients réduites. La FIGURE 5.4 représente l'algorithme de la descente de gradient stochastique différentiellement privée [59].

FIGURE 5.4 – Descente de gradient stochastique différentiellement privée

Algorithme : confidentialité différentielle SGD

Input: Transactions $\{x_1, , x_N\}$, Fonction de perte $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$.

Parametres: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialiser θ_0 aléatoirement

Pour $t \in [T]$ **faire**

Prendre un échantillon aléatoire L_t avec une probabilité de L/N

Calcul de gradient

Pour chaque $i \in L_t$, calcul $g_t(x_i) \longleftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, x_i)$

Normalisation du gradient

$g'_t(x_i) \longleftarrow g_t(x_i) / \max\left(1, \frac{\|g_t(x_i)\|_2}{C}\right)$

Ajouts des bruits

$g'_t \longleftarrow \frac{1}{L}\sum_i (g'_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathrm{I}))$

Descent

$\theta_{t+1} \longleftarrow \theta_t - \eta_t g'_t$

Output : θ_T ,coût de la confidentialité local (ε, δ)

En conclusion, nous avons choisi la méthode de la descente de gradient stochastique differentiellement privée car la plupart des modèles basés sur l'apprentissage automatique et en particulier ceux basés sur l'apprentissage profond utilisent l'algorithme de la descente de gradient. De plus, cette méthode utilise particulièrement l'algorithme de la descente de gradient stochastique par mini-lots, qui est le meilleur des algorithmes de la descente de gradient puisqu'il tient compte des dépenses de calcul dans le cas de la descente de gradient par lots et de la forte variance dans le cas de la descente de gradient stochastique standard.

5.2 Application à la détection de fraude bancaire sur Internet

Dans le cas particulier de la détection de fraude bancaire sur Internet, nous désignons par serveur de paramètres l'organisme qui coordonne la mise en place du modèle et les banques représentent les participants.

Dans cette section, nous présenterons premièrement notre stratégie de détection de fraude avec les auto-encodeurs, les outils utilisés et les métriques de mesures de performances. Deuxièmement, nous présenterons le DataSet du monde réel utilisé et les étapes du prétraitement des données effectuées. Finalement, nous comparons les résultats du classificateur obtenus avec notre approche avec un autre classificateur basé sur l'apprentissage profond supervisé afin de montrer ses mérites.

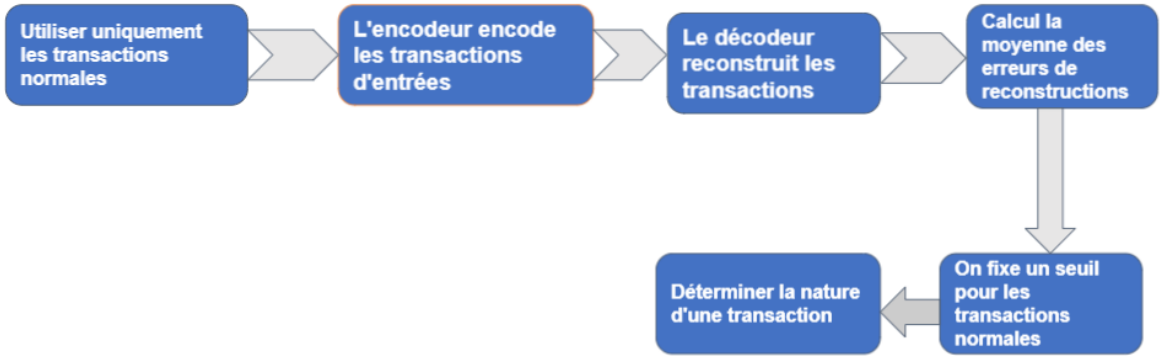
5.2.1 Détection de fraude avec les auto-encodeurs

Pour l'entraînement de la machine auto-encodeur, nous avons utilisé uniquement les transactions normales afin d'apprendre le comportement des transactions légitimes. Premièrement, l'encodeur réduit la dimension des transactions d'entrées sur un ensemble de caractéristiques plus petit. Deuxièmement, le décodeur permet de reconstruire la transaction d'origine. Ensuite, on calcule la moyenne des erreurs de reconstruction des transactions normales comme suit :

$$MER = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y}_i)^2 \tag{5.4}$$

où n représente le nombre de transactions bancaires normales, Y_i la transaction d'origine et $\overline{Y}_i$ la transaction reproduite. Il est important de noter qu'une erreur de reconstruction est la différence en distance entre la transaction d'origine et celle prédite. Enfin, nous fixons un seuil pour les transactions légitimes en utilisant la valeur de la MER pour laquelle nous ajoutons une marge. Il est à noter que la valeur de la marge est relative. Si la valeur d'erreur de reconstruction d'une transaction est supérieure au seuil fixé, alors la transaction est considérée comme frauduleuse, autrement, elle est considérée comme normale. La FIGURE 5.5 résume le processus de détection de fraude avec un auto-encodeur.

FIGURE 5.5 – Processus de détection de fraude



5.2.2 Les métriques de mesures de précision et de confidentialité utilisées

5.2.2.1 Accuracy

L' « accuracy » d'un algorithme de classification encore appelé « Exactitude » dénote la proportion de points correctement prédits. Plus formellement, il est défini comme la somme de vrais positifs et de vrais négatifs divisé par la somme de vrais positifs, de vrais négatifs, de faux positifs et de faux négatifs. En effet, un vrai positif ou un vrai négatif est un point de données que l'algorithme a correctement classé. Un faux positif ou faux négatif, en revanche, est un point de données que l'algorithme a mal classé. Par ailleurs, nous pouvons penser que si nous avons un « accuracy » élevé, notre modèle serait meilleur. Malheureusement, l'« accuracy » représente une excellente mesure uniquement lorsque nous avons des ensembles de données symétriques où les données de la classe positive et celles de la classe négative sont équilibrées. Dans le cas contraire, c'est-à dire si l'ensemble des données est déséquilibré, la valeur de l'« accuracy » serait trompeuse. En effet, lorsque le nombre d'observations dans différentes classes varie considérablement, par exemple, s'il y avait 95 transactions légitimes et seulement 5 transactions frauduleuses dans les données, un mauvais classificateur pourrait classer toutes les transactions comme légitimes en maintenant une précision de 95%, avec un taux de reconnaissance de 100% pour la classe de transactions légitimes et un taux de reconnaissance de 0% pour la classe de transactions frauduleuses. Pour résoudre ce problème, nous avons fait appel à l'équilibrage des classes des transactions par la méthode SMOTE afin de générer des données synthétiques.

5.2.2.2 Mesure de la garantie de confidentialité

La garantie est souvent appelée le budget de confidentialité. Un budget de confidentialité plus faible limite la capacité d'un adversaire à deviner un point d'entraînement. Intuitivement, cela est dû au fait qu'il est plus difficile pour un seul point d'entraînement d'affecter le résultat de l'apprentissage. De plus, les informations contenues dans le point d'entraînement ne peuvent pas être mémorisées par l'algorithme de l'apprentissage automatique. Ainsi, la vie privée de la personne qui a contribué à ce point d'entraînement est conservée.

Par ailleurs, deux métriques sont utilisées pour exprimer la garantie de la confidentialité différentielle d'un algorithme d'apprentissage automatique :

- Delta(δ) : ce paramètre limite la probabilité que le budget de confidentialité ne tienne pas. Une règle d'or consiste à le définir comme inférieure à l'inverse de la taille de l'ensemble de données d'apprentissage.
- Epsilon(ε) : également appelé budget de confidentialité. Il s'agit de la distance maximale entre une requête sur une base de données (x) et la même requête sur une base de données (y). Autrement dit, il s'agit d'une métrique de perte de confidentialité lors d'un changement différentiel des données (c'est-à-dire ajout ou suppression d'une entrée) [15]. Une valeur plus petite pour ε n'est qu'une limite supérieure et une implique une meilleure garantie de confidentialité. Cependant, la valeur ε n'est qu'une limite supérieure et une valeur élevée pourrait encore signifier une bonne confidentialité dans la pratique. Tensorflow Privacy fournit un outil, compute_dp_sgd_privacy.py, que nous avons importé en tant que dépendance, pour calculer la valeur de ε avec une valeur fixe de δ.

5.2.3 Outils utilisés

Cette section présente les différents outils utilisés pour implémenter notre approche de détection de fraude bancaire sur Internet. Ainsi, nous présenterons, les librairies Python Keras, Tensorflow Privacy et Amazon SageMaker.

5.2.3.1 Keras

Keras est une librairie Python Open Source qui encapsule l'accès aux fonctions proposées par plusieurs librairies d'apprentissage automatique, en particulier Tensorflow. De fait, Keras n'implémente pas nativement les méthodes, mais, elle sert d'interface avec Tensorflow.

Plus simplement, elle facilite grandement la vie en proposant des fonctions et procédures relativement simples à mettre en œuvre [10]. Au fond, Keras est l'une des APIs de réseaux de neurones les plus utilisées pour le développement et le testing de réseaux de neurones.

Par ailleurs, Keras propose deux APIs principales : l'API de modèle séquentielle et fonctionnelle. En effet, l'API de modèle séquentielle est une pile de layers linéaires. Les layers peuvent être décrites de façon très simple. Chaque définition de layer requiert une ligne de code [21]. Elle est idéale pour développer des modèles de deep learning dans la plupart des situations, mais présente certaines limites. Par exemple, elle ne permet pas de définir aussi directement des modèles pouvant avoir plusieurs sources d'entrée différentes, produire plusieurs destinations de sortie ou des modèles réutilisant des couches. Or, l'API fonctionnelle de Keras permet de créer des modèles plus flexibles que ceux créés avec l'API séquentielle. En effet, l'API fonctionnelle peut gérer des modèles avec une topologie non linéaire, des modèles avec des couches partagées et des modèles avec plusieurs entrées ou sorties. Elle permet spécifiquement de définir des modèles à plusieurs entrées ou sorties ainsi que des modèles qui partagent des couches. De plus, elle permet de définir des graphiques de réseau acycliques ad hoc. L'idée principale est qu'un modèle de deep learning est généralement un graphique acyclique dirigé (GAD) de couches. L'API fonctionnelle est donc une manière de créer des graphiques de couches. En résumé, elle offre une manière plus flexible de définir des modèles.

5.2.3.2 « Tensorflow Privacy »

Tensorflow est l'un des outils les plus populaires pour créer des applications basées sur l'apprentissage automatique. Il est utilisé comme Framework de référence pour faire l'apprentissage profond [17]. Récemment Google a créé la librairie Tensorflow Privacy basée sur la confidentialité différentielle qui permet aux développeurs en intelligence artificielle de protéger les données de leurs utilisateurs. Grâce à la librairie Tensorflow Privacy, les éléments identifiables sont automatiquement supprimés des ensembles de données sans pour autant changer leur sens. Ainsi, avec quelques lignes de code supplémentaires, les développeurs sont en mesure d'améliorer la confidentialité de leurs modèles. La librairie Tensorflow Privacy utilise la descente de gradient stochastique différentiellement privée. Comme montré à la FIGURE 5.6, la mise en oeuvre doit passer par la spécification de trois hyperparamètres liés à la confidentialité et un hyperparamètre général :

- l2_norm_clip (float) : la norme euclidienne maximale de chaque gradient est appliquée pour mettre à jour les paramètres du modèle. Cet hyperparamètre est utilisé pour limiter la sensibilité de l'optimiseur aux points d'entraînements individuels.

- noise_multiplier (float) : la quantité de bruit échantillonnée est ajoutée aux gradients pendant l'entraînement. Généralement, plus de bruit se traduit par une meilleure confidentialité et une faible précision. Ainsi il y a un compromis entre la confidentialité et la précision en considérant la quantité de bruit ajoutée.
- microbatches (int) : chaque lot de données est divisé en unités plus petites appelées micro-lots. Par défaut, chaque micro-lot doit contenir un seul exemple d'entraînement. Cela permet de découper les dégradés, par exemple plutôt qu'après qu'ils ont été moyennés sur le mini-lot. Ceci, à son tour, diminue l'effet (négatif) de l'écrêtage sur le signal trouvé dans le gradient et maximise généralement l'utilité. Cependant, les frais généraux de calcul peuvent être réduits en augmentant la taille des micro-lots pour inclure plus d'un exemple d'entraînement. Le gradient moyen sur ces multiples exemples d'entraînement est ensuite coupé. Le nombre total d'exemples consommés dans un lot, c'est-à-dire une étape de descente de gradient, reste le même. Le nombre de micro-lots doit diviser également la taille des lots.
- learning_rate (float) : cet hyperparamètre existe déjà en descente de gradient stochastique vanille. Plus le taux d'apprentissage est élevé, plus chaque mise à jour compte. Si les mises à jour sont bruyantes (comme lorsque le bruit additif est important par rapport au seuil d'écrêtage), un faible taux d'apprentissage peut aider la procédure d'apprentissage à converger.

FIGURE 5.6 – Paramètres à régler dans Tensorflow Privacy

```
#traning parameters
batch_size= 250
epochs = 60
learning_rate = 0.15

# DP-SGD parameters
l2_norm_clip = 1.0
noise_multiplier = 1
microbatches= 1
```

5.2.3.3 Amazon SageMaker

D'abord, l'apprentissage automatique a tendance à paraître plus compliqué qu'il ne le devrait pour de nombreux développeurs, car le processus de formation et de création de modèles avant de les héberger est plutôt lent et difficile [39]. Cependant, Amazon SageMaker [3] supprime cette complexité pour aider les développeurs à créer, former et déployer des modèles d'apprentissage automatique rapidement et facilement. Plus précisément, il fournit des modèles d'apprentissage automatique pré-entraînés pour le déploiement, fournit un certain nombre d'algorithmes d'apprentissage automatique intégrés qu'on peut entraîner sur nos propres données. Amazon SageMaker fournit aussi des instances de TensorFlow et Apache MXNet qu'on peut utiliser pour créer nos propres algorithmes d'apprentissage automatique. Par ailleurs, il permet de mettre en place facilement des clusters de machines pour l'exécution des algorithmes d'apprentissage profond distribué.

5.2.4 « Data Set» et Prétraitement des données

5.2.4.1 « Data Set»

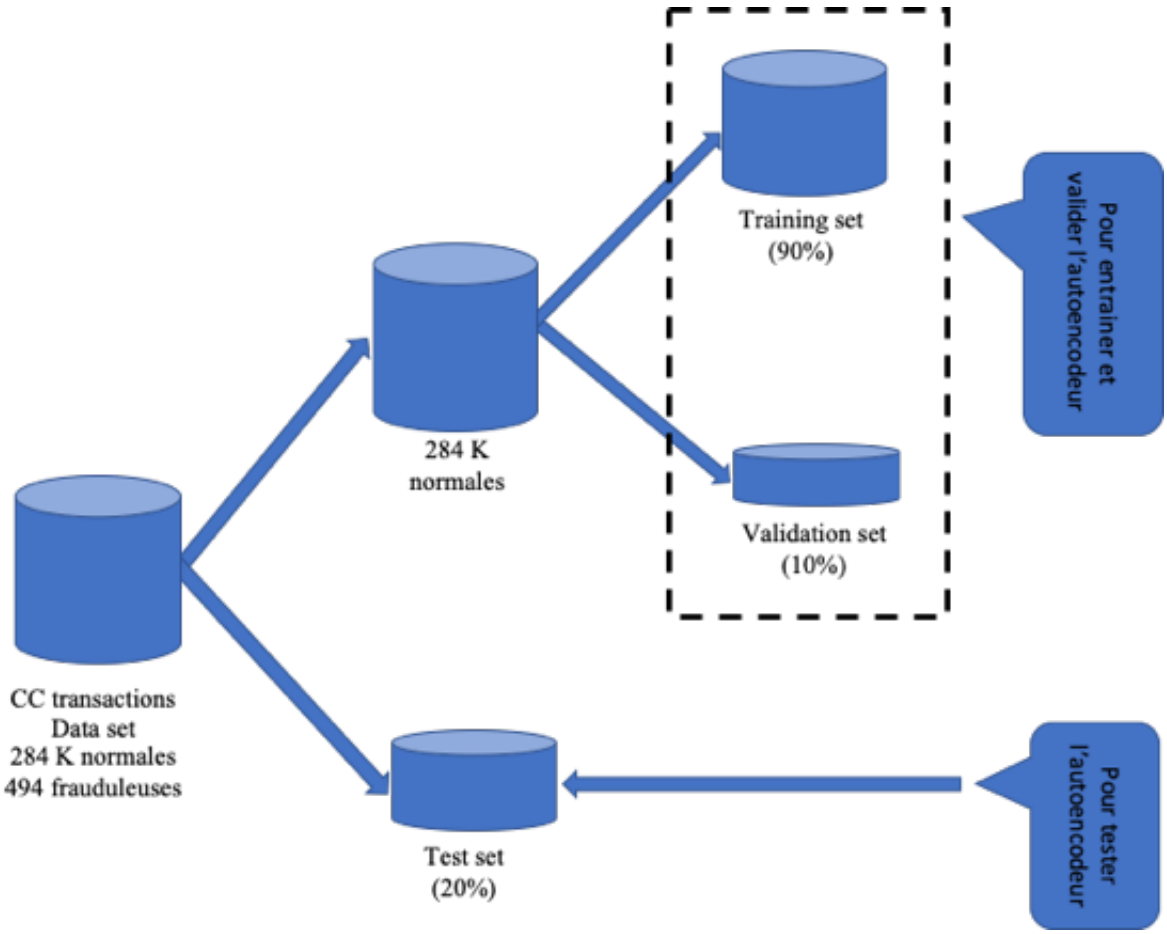
Nous avons utilisé les données de carte de crédit fournies par Kaggle [13]. Le Data Set contient les transactions par carte de crédit effectuées en septembre 2013 par des titulaires de cartes européennes. Le nombre total de transactions est de 284 807. Le Data set est très déséquilibré, car seuls 492 d'entre elles (0,172%) sont frauduleuses, mais par ailleurs, cela est un avantage puisque le but est d'apprendre plus sur la représentation des transactions normales afin de détecter les transactions frauduleuses. Malheureusement, en raison de problèmes de confidentialité, les caractéristiques d'origine et d'autres informations de fond sur les données n'ont pas été fournies. Les caractéristiques V1, V2, ... V28 sont les attributs obtenus avec ACP, les seuls attributs qui n'ont pas été transformés avec ACP sont « Time » et « Amount ». En effet, l'attribut « Time » contient les secondes écoulées entre chaque transaction et la première transaction de l'ensemble de données. L'attribut « Amount » est le montant payé dans chaque transaction. L'attribut « Classe » est la variable de réponse et prend la valeur 1 en cas de fraude et 0 sinon [43].

5.2.4.2 Prétraitement des données

À première vue, la variable « Time » n'a pas d'impact sur la prédiction. Ainsi, nous avons supprimé cette variable du Data Set. En autre, nous avons extrait uniquement les transactions

normales pour l'entraînement de l'auto-encodeur. Parmi ces données d'entraînement, 20% sont utilisées pour fixer le seuil des erreurs de reconstruction et la validation du modèle. Ensuite, nous avons pris 10% pour les données de test de l'auto-encodeur. Les données de test sont composées des transactions normales et frauduleuses. Nous avons enfin normalisé toutes les données d'entrée pour n'avoir que des valeurs appartenant à [0,1] puisque les réseaux de neurones n'acceptent que des vecteurs d'entrée dont les valeurs sont comprises entre 0 et 1. La FIGURE 5.7 montre la procédure utilisée pour le prétraitement des données dans le cadre de l'auto-encodeur. En ce qui concerne l'apprentissage profond supervisé, nous avons utilisé 80% des données pour l'entraînement du modèle et 20% pour le test.

FIGURE 5.7 – Modélisation pour le prétraitement des données



5.2.5 Résultats et analyses

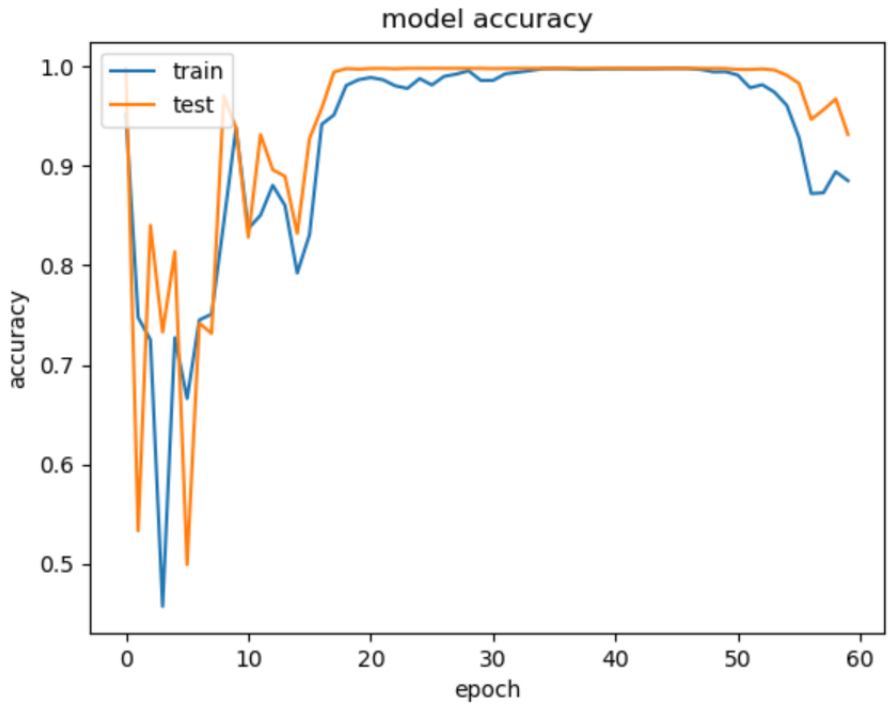
Dans cette section, nous présentons les performances de notre classificateur et celles obtenues avec le classificateur basé sur l'apprentissage profond supervisé afin de montrer les mérites de l'approche proposée.

5.2.5.1 Classificateurs et performances

Nous avons utilisé l'outil Amazon SageMaker présenté dans la section 5.2.3.3 pour l'environnement distribué et la bibliothèque Tensorflow Privacy qui fournit une implémentation de DP SGD. La bibliothèque Tensorflow Privacy a été utilisée pour implémenter les algorithmes d'apprentissage profond qui sont différentiellement privés afin de préserver la confidentialité des transactions utilisées lors de l'entraînement des modèles. En effet, pour tester l'efficacité de l'approche proposée, nous avons implémenté hormis notre classificateur, un autre classificateur basé sur l'apprentissage profond supervisé.

Le classificateur basé sur l'apprentissage profond supervisé Nous avons implémenté un modèle différentiellement privé d'apprentissage profond constitué de 5 couches dont 3 couches cachées et 1 couche d'entrée et 1 sortie. Chaque couche est formée de 20 neurones. Étant donné que les transactions bancaires sont déséquilibrées, nous avons également traité le problème de classification lié au déséquilibre des données (instances positives et négatives) avec la technique de sur-échantillonnage SMOTE. En effet, pour une quantité de bruit e = 1, nous avons le résultat, présenté à la FIGURE 5.8, qui montre la comparaison de la performance en matière de précision d'apprentissage profond supervisé pour les données d'entraînement et de test. De ce résultat, nous constatons qu'il n'y a pas de sur-apprentissage. En effet, le sur-apprentissage se produit lorsqu'on obtient un bon ajustement d'un modèle sur les données d'entraînement, alors qu'il ne se généralise pas bien sur de nouvelles données invisibles.

FIGURE 5.8 – Comparaison de la performance en matière de précision du classificateur basé sur l'apprentissage profond supervisé pour les données d'entraînement et de test



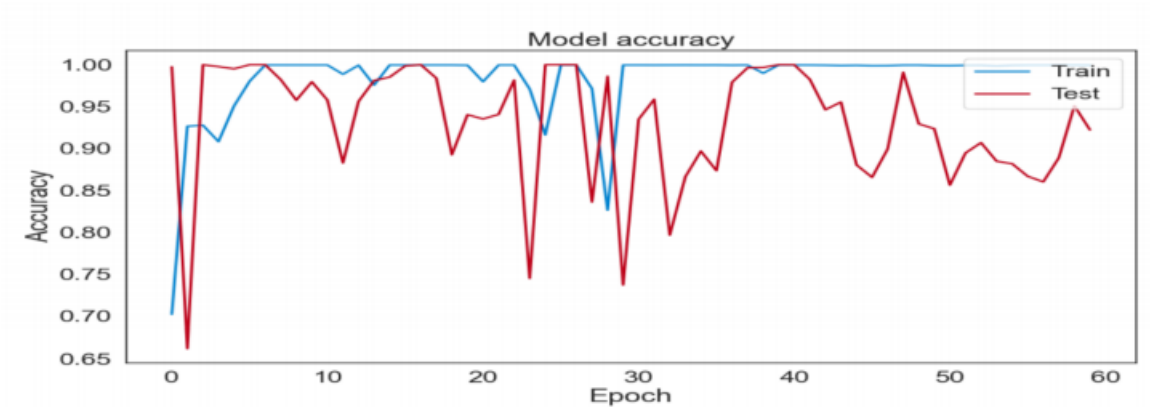
Noise	0	1	2	3	4	5	6
Epsilon(ε)	Niveau Maximal	3.83	1.46	0.92	0.62	0.54	0.45
Accuracy	0.9983	0.97	0.96	0.81	0.88	0.84	0.80

TABLE 5.1 – Rapport entre le niveau de confidentialité (Epsilon), Accuracy et la quantité du bruit pour le classificateur basé sur l'apprentissage profond supervisé

En outre, la TABLE 5.1 décrit le niveau de confidentialité (Epsilon) et Accuracy en fonction de la quantité du bruit ajoutée aux données. De ce tableau, nous constatons que plus la quantité du bruit augmente, plus la valeur de Epsilon diminue c'est à dire un niveau de confidentialité élevé et la précision des résultats diminue. Par ailleurs, Niveau Maximal représente la valeur du budget de confidentialité lorsqu'aucun bruit n'est ajouté.

Notre classificateur Nous avons implémenté un modèle différentiellement privé d'auto-encodeur constitué d'une couche d'entrée, de deux couches cachées et d'une couche de sortie. Chaque couche est formée de 20 neurones. Pour une quantité de bruit e = 1, nous avons le résultat, présenté à la FIGURE 5.9, qui montre la comparaison de la performance en matière de précision de notre classificateur pour les données d'entraînement et de test. De ce résultat, nous constatons que notre modèle ne souffre pas de sur-apprentissage car il a une bonne capacité de généralisation pour les nouvelles données invisibles.

FIGURE 5.9 – Comparaison de la performance en matière de précision de notre classificateur pour les données d'entraînement et de test



Noise	0	1	2	3	4	5	6
Epsilon(ε)	Niveau Maximal	3.49	1.33	0.84	0.62	0.49	0.41
Accuracy	0.9996	0.93	0.83	0.81	0.85	0.80	0.88

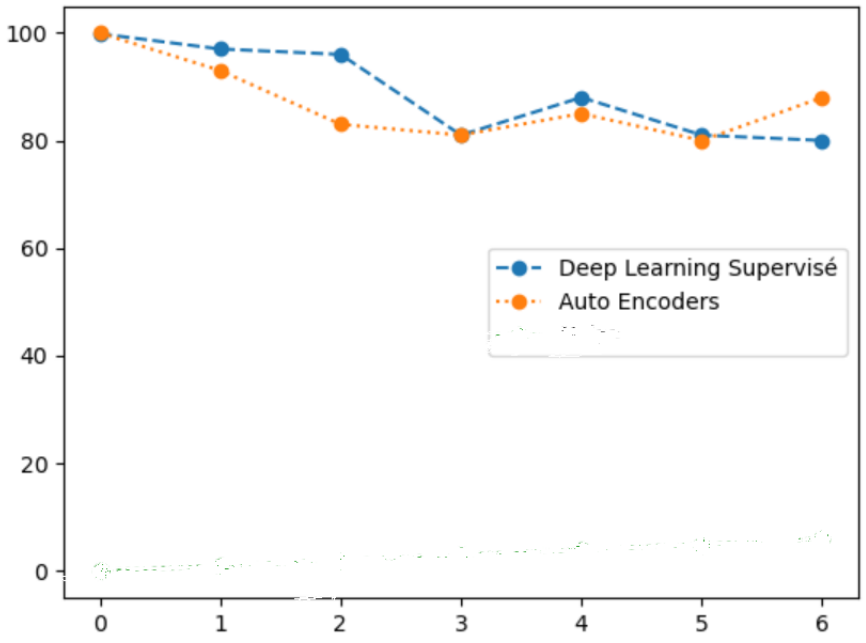
TABLE 5.2 – Rapport entre le niveau de confidentialité(Epsilon), Accuracy et la quantité du bruit pour notre classificateur

En outre, la TABLE 5.2 décrit le niveau de confidentialité (Epsilon) et Accuracy en fonction de la quantité du bruit ajoutée aux données pour le modèle obtenus de notre approche. A partir de ce tableau, nous constatons que plus la quantité du bruit augmente, plus la valeur de Epsilon diminue c'est à dire un niveau de confidentialité plus élevé mais la précision n'a pas

connu beaucoup de variation. Par ailleurs, Niveau Maximal représente la valeur du budget de confidentialité lorsqu'aucun bruit n'est ajouté.

Comparaison de la performance du classificateur basé sur l'apprentissage supervisé et de notre classificateur La FIGURE 5.10 montre que la précision du classificateur basé sur l'apprentissage profond supervisé et de notre classificateur sont presque similaires quel que soit la quantité du bruit.

FIGURE 5.10 – Comparaison de la performance du classificateur basé sur l'apprentissage supervisé et de notre classificateur en matière de précision en fonction de la quantité du bruit



En conclusion, nous avons présenté dans ce chapitre notre approche d'apprentissage profond distribué sécurisé. Afin de valider l'approche proposée, nous l'avons appliqué au cas de détection de fraude bancaire sur Internet et obtenus de meilleurs résultats comparativement aux solutions existantes. Il est à noter que l'auto-encodeur permet d'apprendre efficacement la représentation des transactions normales, qui ont des structures plus complexes. Ce qui augmente ainsi, la précision pour la détection des transactions frauduleuses.

Chapitre 6

CONCLUSION

Plusieurs approches basées sur différentes méthodes d'apprentissage automatique ont été proposées par plusieurs chercheurs pour la détection automatique des fraudes bancaires sur Internet. Malheureusement, ces approches ne permettent pas de détecter les fraudes de façon efficaces tout en préservant la confidentialité des données des banques utilisées lors de la construction du modèle.

Pour résoudre cette problématique, nous proposons dans ce mémoire une solution basée sur l'apprentissage profond distribué permettant d'assurer une confidentialité différentielle des données d'apprentissage et permettant ainsi de préserver la vie privée des clients des banques participantes. Plus précisément, nous implémentons un mécanisme distribué et sécurisé qui permet de détecter les fraudes de façon efficace en respectant les standards de la confidentialité différentielle.

Dans le but d'implémenter notre approche, nous avons d'abord, réalisé le prétraitement des données afin d'avoir un ensemble d'apprentissage de qualité, permettant ainsi une bonne performance de nos modèles basés sur des algorithmes d'apprentissage profond. Aussi, nous avons traité le problème de classification lié au déséquilibre des données (instances positives et négatives) avec la technique de sur-échantillonnage SMOTE puisque les transactions bancaires sont très déséquilibrées. Par la suite, nous avons implémenté avec la librairie Tensorflow Privacy notre approche d'apprentissage profond distribué sécurisé et un autre classificateur basé sur l'apprentissage profond supervisé en respectant les standards de la confidentialité différentielle. Nous avons ainsi obtenu deux modèles d'apprentissage profond qui sont différentiellement privés.

Les résultats montrent une bonne performance de notre approche en matière de préservation de la confidentialité des données ainsi dans la précision du modèle obtenu. En effet, bien

que les performances en précision des deux modèles soit similaires, on constate que notre approche permet d'avoir un niveau de confidentialité plus élevé que celle qui utilise l'apprentissage profond supervisé.

Compte tenu des délais, nous n'avons pu analyser l'ensemble de ce sujet très vaste. Cependant, il nous semblerait intéressant, dans l'avenir, d'explorer une nouvelle méthode basée sur l'« équilibre de Nash » qui permet de fixer automatiquement la quantité du bruit en fonction de la sensibilité des données et de précision souhaitée.

Bibliographie

[1] Algorithme de descente du gradient stochastique. https://fr.wikipedia.org/wiki/Algorithme_du_gradient_stochastique. consulté le 15 Août 2021.

[2] Algorithme génétique. https://fr.wikipedia.org/wiki/Algorithme_g%C3%A9n%C3%A9tique. consulté le 8 Août 2021.

[3] Amazon sagemaker. https://www.amazonaws.cn/en/sagemaker/. consulté le 10 janvier 2021.

[4] Analyse en composantes principales (acp). https://rstudio-pubs-static.s3.amazonaws.com/155513_cc852faa1ed9446cbd50e9a70d9fc41b.html. consulté le 31 decembre 2020.

[5] Apprentissage centralisé. https://fr.wikipedia.org/wiki/Apprentissage_f%C3%A9d%C3%A9r%C3%A9. consulté le 8 Août 2021.

[6] Apprentissage semi-supervisé. https://fr.wikipedia.org/wiki/Apprentissage_semi-supervis%C3%A9. consulté le 15 Août 2021.

[7] Centre antifraude du Canada. https://www.antifraudcentre-centreantifraude.ca/index-fra.html. consulté le 13 octobre 2020.

[8] Centre antifraude du canada. https://www.antifraudcentre-centreantifraude.ca/index-fra.html. consulté le 13 octobre 2020.

[9] Classification : Roc et auc. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=fr. consulté le 20 novembre 2020.

[10] Comprendre keras. http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/fr_Tanagra_Tensorflow_Keras_Python.pdf. consulté le 15 Août 2021.

[11] Contribution au développement de l'apprentissage profond dans les systèmes distribués. https://tel.archives-ouvertes.fr/tel-02284916. consulté le 20 décembre 2020.

[12] Courbe roc. https://fr.wikipedia.org/wiki/Courbe_ROC. consulté le 11 janvier 2021.

[13] Credit card fraud analysis. https://www.kaggle.com/mlg-ulb/creditcardfraud. consulté le 20 Mars 2020.

[14] Descente de gradient. https://exo7math.github.io/deepmath-exo7/descente/descente.pdf. consulté le 15 Août 2021.

[15] Differential privacy definition. https://en.wikipedia.org/wiki/Differential_privacy. consulté le 31 Juillet 2021.

[16] H20. http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/fr_Tanagra_Package_H2O_Python.pdf. consulté le 8 Août 2021.

[17] Introducing tensorflow privacy : Learning with differential privacy for training data. https://opensource.google/projects/tensorflow-privacy. consulté le 29 Août 2021.

[18] Introduction à l'apprentissage par descente de gradient. http://www.iro.umontreal.ca/~pift6266/H12/html/gradient_fr.html. consulté le 16 Décembre 2021.

[19] K-means clustering in machine learning. https://blogs.oracle.com/bigdata/post/k-means-clustering-in-machine-learning-simplified. consulté le 02 Novembre 2020.

[20] K plus proches voisins (knn). https://www.xlstat.com/fr/solutions/fonctionnalites/k-nearest-neighbors-knn. consulté le 15 Août 2021.

[21] Keras : tout savoir sur l'api de deep learning. https://datascientest.com/keras. consulté le 15 Août 2021.

[22] La descente de gradient. https://machinelearnia.com/descente-de-gradient/. consulté le 31 Juillet 2021.

[23] La distribution de probabilité de la loi normale. https://fr.wikipedia.org/wiki/Loi_normale#/media/Fichier:Normal_Distribution_PDF.svg. consulté le 17 Août 2021.

[24] La distribution de probabilité de laplace. https://fr.wikipedia.org/wiki/Loi_de_Laplace_(probabilit%C3%A9s)#/media/Fichier:Laplace_distribution_pdf.png. consulté le 17 Août 2021.

[25] La loi de laplace. https://www.bibmath.net/dico/index.php?action=affiche&quoi=./l/loilaplace.html). consulté le 31 Juillet 2021.

[26] La loi géométrique. https://www.techno-science.net/glossaire-definition/Loi-geometrique.html. consulté le 31 Juillet 2021.

[27] La loi normale. https://fr.wikipedia.org/wiki/Loi_normale. consulté le 31 Juillet 2021.

[28] Le bagging. https://dataanalyticspost.com/Lexique/bagging/. consulté le 31 Juin 2021.

[29] Le boosting. https://datascientest.com/algorithmes-de-boosting-adaboost-gradient-b consulté le 31 Juin 2021.

[30] Le clustering k-means. https://analyticsinsights.io/k-means/. consulté le 30 décembre 2020.

[31] Les modèles de markov cachés. http://www2.agroparistech.fr/ufr-info/membres/cornuejols/Teaching/Master-ISI/ISI-10/livre2-v3(ac)-ch-12.pdf. consulté le 8 Août 2021.

[32] Les neurones formels. http://tpe-ia.lescigales.org/maths.php. consulté le 02 décembre 2020.

[33] Minimisation du risque empirique. http://www.crest.fr/ckfinder/userfiles/files/pageperso/Atsybakov/Projet5.pdf. consulté le 15 Août 2021.

[34] Overfitting et underfitting. https://mrmint.fr/overfitting-et-underfitting-quand-vos-algorithmes-de-machine-learning-derapent. consulté le 16 Décembre 2021.

[35] Rbm. https://fr.wikipedia.org/wiki/Machine_de_Boltzmann_restreinte. consulté le 31 Juillet 2021.

[36] Smote. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-106. consulté le 31 Juillet 2021.

[37] Svm. https://dataanalyticspost.com/Lexique/svm/. consulté le 02 Décembre 2021.

[38] Techniques de validation. https://scikit-learn.org/stable/modules/cross_validation.html. consulté le 31 Juillet 2021.

[39] What is amazon sagemaker ? https://www.amazonaws.cn/en/sagemaker/. consulté le 20 novembre 2020.

[40] What is functional link artificial neural network. https://www.igi-global.com/dictionary/functional-link-artificial-neural-network-flann/61451. consulté le 31 Juillet 2021.

[41] A.CHARLEONNAN. Credit card fraud detection using rus and mrn algorithm. *The 2016 Management and Innovation Technology International Conference (MITiCON-2016)* (2016).

[42] A.CHARLEONNAN. Credit card fraud detection using rus and mrn algorithm. *The 2016 Management and Innovation Technology International Conference (MITiCON-2016)* (2016).

[43] A.D.POZZOLO, O.CAELENA, R.A.JOHNSON, AND G.BONTEMPI. Calibrating probability with undersampling for unbalanced classification. *Symposium on Computational Intelligence and Data Mining (CIDM)* (2015).

[44] A.PUMSIRIRAT, AND L.YAN. Credit card fraud detection using deep learning based on auto- encoder and restricted boltzmann machine. *International Journal of Advanced Computer Science and Applications 9*, 1.

[45] C.CHEN, J.LEE, AND D.KIFER. Renyi differentially private erm for smooth objectives. *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), 2037–2046.

[46] C.DWORK. Differential privacy. *Automata, Languages and Programming* (2006), 1–12.

[47] C.DWORK. Differential privacy in new settings. *InProceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algo-rithms 4052* (2010), 174–183.

[48] C.DWORK, AND A.ROTH. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science 9*, 3-4 (August 2014), 221–407.

[49] C.DWORK, F.MCSHERRY, K.NISSIM, AND A.SMITH. Calibrating noise to sensitivity in private data analysis. *In Theory of Cryptography Conference (TCC)* (2006).

[50] C.MISHRA, D.L.GUPTA, AND R.SINGH. Credit card fraud detection using neural networks. *International Journal of Computer Science 4* (Juillet 2017).

[51] C.WANG, Y.WANG, Z.YE, AND L.YAN. Credit card fraud detection based on whale algorithm optimized bp neural network. *3th International Conference on Computer Science and Education (ICCSE)*, 1–4.

[52] G.E.HINTON, AND R.R.SALAKHUTDINOV. Reducing the dimensionality of data with neural networks. *Science 312*, 1777 (2006).

[53] G.RUSHIN, C.STANCIL, M.SUN, AND S.ADAMS. Horse race analysis in credit card fraud— deep learning, logistic regression, and gradient boosted tree. *Systems and Information Engineering Design Symposium (SIEDS)* (2017), 117–121.

[54] H.CHAN, LI, M., E.SHI, AND W.XU. Differentially private continual monitoring of heavy hitters from distributed streams. *International Symposium on Privacy Enhancing Technologies Symposium 4052* (2012), 140–159.

[55] J.ESMAILY, R.MORADINEZHAD, AND J.GHASEMI. Intrusion detection system based on multilayer perceptron neural networks and decision tree. *International conference on Information and Knowledge Technology* (2015).

[56] J.LEE, AND C.CLIFTON. How much is enough ? choosing for differential privacy. *In International Conference on Information Security* (2011), 25–340.

[57] J.WANG, M.XU, H.WANG, AND J.ZHANG. Classification of imbalanced data by using the smote algorithm and locally linear embedding. *Signal Processing, 2006 8th International Conference 3* (2006).

[58] K.CHAUDHURI, C.MONTELEONI, AND SARWATE, D. Differentially private empirical risk minimization. *Journal of Machine Learning Research 12* (March 2011), 1069–1109.

[59] M.ABADI, AND I.GOODFELLOW. Deep learning with differential privacy. *ACM SIGSAC Conference on Computer and Communications Security* (Octobre 2016), 308–318.

[60] M.EMAD, AND B.FAR. Supervised machine learning algorithms for credit card fraudulent transaction detection : A comparative study. *IEEE Annals of the History of Computing* (Juillet 2018).

[61] M.MISHRA, AND R.DASH. A comparative study of chebyshev functional link artificial neural network, multi-layer perceptron and decision tree for credit card fraud detection. *2014 13th International Conference on Information Technology* (2014).

[62] M.R.HARATINIK, M.AKRAMI, S.KHADIVIE, AND M.SHAJARI. Fuzzgy : A hybrid model for credit card fraud detection.

[63] N.DEMLA, AND A.AGGARWAL. Credit card fraud detection using svm and reduction of false alarms. *International Journal of Innovations in Engineering and Technology 7*, 2 (2016).

[64] NITESH V. CHAWLA, KEVIN W. BOWYER, L. O. H., AND KEGELMEYER, W. P. Smote : Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* (Octobre 2002), 321–357.

[65] N.MALINI, AND M.PUSHPA. Analysis on credit card fraud identification techniques based on knn and outlier detection. *3rd International Conference on Advances in Electrical*.

[66] N.MALINI, AND M.PUSHPA. Analysis on credit card fraud identification techniques based on knn and outlier detection. *3rd International Conference on Advances in Electrical, Electronics* (2017).

[67] N.V.CHAWLA, AND K.W.BOWYER. Smote : Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16 (2002), 321–357.

[68] N.W.WEN, AND F.YU. Research on credit card fraud detection model based on distance sum. international joint conference on artificial intelligence. *Hainan Island* (2009).

[69] O.OWOLABI, O.ADELEYE, J.WANDERA, AND S.FASHOTO. Hybrid methods for credit card fraud detection.

[70] P.KRISHNA, AND K.TRIPATHI. Survey on credit card fraud detection methods. *M.E Compute*.

[71] R.PATIDAR, AND L.SHARMA. Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)* (2011), 32–38.

[72] S.BHATTACHARYA, S.JHA, K.THARAKUNNEL, AND J.C.WESTLAND. Data mining for credit card fraud : A comparative study. *Elsevire 50*, 3 (2011), 602–613.

[73] S.KUMAR, A.MISHRA, AND A.AGARWAL. Credit card fraud detection : a hybrid approach using fuzzy clustering and neural network. *IEEE* (2015).

[74] S.MAES, K.TUYLS, B.VANSCHOENWINKEL, AND B.MANDERICK. Credit cards fraud detection using bayesian and neural networks. *IEEE* (August 2002), 7.

[75] S.PANIGRAHI, AND T.K.BEHERA. Credit card fraud detection : a hybrid approach using fuzzy clustering and neural network. *International Conference on Advances in Computing and Communication Engineering* (2015).

[76] S.PANIGRAHI, AND T.K.BEHERA. Credit card fraud detection : a hybrid approach using fuzzy clustering and neural network. *International Conference on Advances in Computing and Communication Engineering* (2015).

[77] S.PATIL, H.SOMAVANSHI, AND GAIKWAD, J. Credit card fraud detection using decision tree induction algorithm. *International Journal of Computer Science and mobile computing 4*, 92–95.

[78] S.U.HEGDE, P.M.TODD, AND G.MILLER. Designing neural networks using genetic algorithms.

[79] T.BENKHELIF. *Publication de données individuelles respectueuse de la vie privée : une démarchefondée sur le co-clustering*. PhD thesis, Université de Nantes, 2018.

[80] T.RAZOOGI, P.KHURANA, K.RAAHEMIFAR, AND A.ABHARI. Credit card fraud detection using fuzzy logic and neural networks. *Society for modelling and simulation International (SCS)* (2016).

[81] X.SHIYANG. Random forest for credit card fraud detection. *IEEE 15th International Conference on Networking* (2018).

[82] Y.JAIN, N.TIWARI, S.DUBEY, AND S.JAIN. Comparative analysis of various credit card fraud detection techniques. *International Journal of Recent Technology and Engineering (IJRTE) ISSN : 2277-3878 7* (Janvier 2019).

[83] Y.KANG, AND Y.LIU. Input perturbation : A new paradigm between central and local differential privacy.

[84] Y.PANDEY. Credit card fraud detection using deep learning. *Int. J. Adv. Res. Comput. Sci. 8* (May–Jun 2017).

[85] Y.SAHIN, AND E.DUMAN. Detecting credit card fraud by ann and logistic regression. *Computer Science 2011 International Symposium on Innovations in Intelligent Systems and Applications* (Juin 2011).

[86] Y.SAHIN, AND E.DUMAN. Detecting credit card fraud by ann and logistic regression. *Computer Science 2011* (June 2011).

[87] Y.WANG, S.ADAMS, AND P.BELING. Privacy preserving distributed deep learning and its application in credit card fraud detection. *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), 1070–1078.