UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR
DEDAM KOSLENGAR GERTRUDE

L'APPLICATION DU MACHINE LEARNING POUR LA DÉTECTION DE FRAUDE EN FINANCE

Décembre 2025

Résumé

La fraude financière est une une action intentionnelle causant une perte économique à un individu ou à une institution. Face à la complexité croissante des comportements frauduleux, le *Machine Learning* s'impose comme une solution prometteuse pour concevoir des systèmes de détection adaptatifs et performants. Cependant, le déséquilibre des classes, dû à la rareté des transactions frauduleuses, demeure un obstacle majeur à la performance des modèles. Ce travail vise à améliorer la détection des fraudes financières en développant et en évaluant des modèles d'apprentissage automatique robustes aux changements comportementaux graduels. Une chaîne de traitement complète a été mise en œuvre, intégrant notamment le *CostSensitive Learning*, qui compense le déséquilibre en attribuant des coûts différenciés aux erreurs de classification. Les résultats expérimentaux montrent que le modèle XGBoost offre la meilleure performance avec une AUC de 93%, suivie d'une approche hybride combinant XGBoost et TabNet atteignant 92%. Les modèles de référence comme la régression logistique et TabNet obtiennent des performances plus modestes. Enfin, l'auto-encodeur, utilisé pour la détection non supervisée d'anomalies, présente des résultats inférieurs aux approches supervisées, mais reste utile pour l'identification de comportements atypiques.

Mots clés : Fraude financière ; Machine Learning ; Déséquilibre de classes ; *Cost-Sensitive Learning* ; XGBoost ; TabNet ; Auto-encodeur ; Changements graduels.

DEDAM KOSLENGAR GERTRUDE MAHMED MESFIOUI

Abstract

Financial fraud is an intentional act causing economic loss to an individual or an institution. With the growing complexity of fraudulent behaviors, Machine Learning has emerged as a promising solution for designing adaptive and efficient fraud detection systems. However, the class imbalance, resulting from the rarity of fraudulent transactions, remains a major challenge to model performance. This work aims to improve financial fraud detection by developing and evaluating machine learning models that are robust to gradual behavioral changes. A comprehensive processing pipeline was implemented, notably integrating CostSensitive Learning, which addresses class imbalance by assigning different misclassification costs to fraudulent and non fraudulent samples. Experimental results show that the XGBoost model achieved the best performance with an AUC of 93%, followed by a hybrid approach combining XGBoost and TabNet reaching 92%. Baseline models such as logistic regression and TabNet obtained more modest results. Finally, the autoencoder, used for unsupervised anomaly detection, showed lower performance compared to supervised approaches but remained useful for identifying atypical transaction patterns.

Keywords : Financial Fraud ; Machine Learning ; Class Imbalance ; Cost-Sensitive Learning ; XGBoost ; TabNet ; Auto-encoder ; Gradual changes.

Avant-propos

Je rends grâce à Dieu de m'avoir accordé la grâce, la sagesse et la clarté pour la réalisation de ce projet.

Au terme de ce mémoire, j'exprime ma profonde gratitude à M. Mhamed Mesfioui, mon directeur de mémoire, pour son encadrement bienveillant, sa disponibilité et la confiance qu'il m'a accordée tout au long de ce travail.

Je suis très honorée de compter parmi les membres de mon jury les professeurs Mme. Nadia Ghazzali et M. Boucif Amar Bensaber, que je remercie sincèrement pour avoir accepté d'évaluer mon travail et pour le temps et l'attention qu'ils lui ont consacrés.

J'adresse également ma profonde reconnaissance à mes enseignants qui me font également l'immense plaisir de participer à l'achèvement de ce travail et avec qui les échanges furent riches.

Mes rémerciements vont aussi l'ensemble de mes condisciples du département mathématiques et informatique pour leurs encouragements.

Je tiens à exprimer ma profonde gratitude à mes parents, qui demeurent mon plus grand soutien grâce à leur amour inconditionnel et à leur confiance inébranlable. Mes sincères remerciements s'adressent également à mes amis, pour leur précieux soutien moral tout au long de ce parcours.

Ce parcours a été pour moi une expérience académique fondamentale, à la fois enrichissante par l'acquisition de connaissances approfondies, exigeante, et formatrice pour le développement d'une autonomie scientifique.

À mon père,

Pour ton amour inconditionnel et son sacrifice qui n'a d'égal. Malgré les défis de ton état de santé, tu as été mon pilier, prêt à sacrifier ton propre bien-être pour assurer la réussite et le soutien financier de ta famille, de ta fille. Cette détermination force mon admiration et demeure ma plus grande leçon de vie.

À ma mère,

Pour ton amour inconditionnel, ton soutien inébranlable, ta patience et ta confiance constante, qui ont créé un environnement propice à l'étude et à l'épanouissement. Ton engagement silencieux fut la fondation de ma réussite.

À mes frères et sœur,

Pour votre présence aimante et votre encouragement continu. Merci d'avoir été mon refuge et ma source de joie tout au long de ce parcours universitaire.

Ce travail est le fruit de vos sacrifices et de votre amour.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

1.1 Mise en contexte

L'apprentissage automatique est une branche puissante de l'intelligence artificielle caractérisé par une vaste application dans les secteurs bancaire et financier. Il offre aux institutions financières des avancées tant méthodologiques que conceptuelles, leur permettant de détecter les transactions frauduleuses et d'assister les gestionnaires dans l'évaluation, la classification et la prise de décision relative à l'octroi de crédit. Les chiffres désormais disponibles indiquent une augmentation notable de la fraude financière en raison des avancées technologiques et du passage des transactions financières en espèces aux transactions numériques. Dans ce contexte, les approches traditionnelles de détection de fraude, souvent basées sur des règles statiques, se révèlent rapidement dépassées.

La transformation numérique des systèmes financiers a entraîné une explosion des transactions électroniques notamment via les cartes bancaires, les virements en ligne et les applications mobiles. L'essor du numérique augmente considérablement le risque de cybercriminalité et de fraude sous diverses formes. Ces menaces incluent notamment les courriels frauduleux, l'usurpation d'identité, le spam, l'hameçonnage ou en-

core le piratage des distributeurs automatiques de billets, l'objectif étant d'accéder à des informations financières sensibles.
L'un des principaux objectifs des acteurs du domaine est de trouver des solutions innovantes pour la détection précoce et efficace de la fraude. C'est dans cette optique que *le Machine Learning* émerge comme une technologie prometteuse, grâce à sa capacité à traiter un volume important de données et à développer des modèles prédictifs qui peuvent apprendre et s'améliorer de manière autonome.

1.2 Objectifs et problèmes de recherche

1.2.1 Objectifs

L'objectif principal de ce travail est d'étudier la détection des transactions financières frauduleuses par l'apprentissage automatique, en se concentrant sur le développement de méthodes innovantes et robustes capables de s'adapter au changement de comportement graduel des fraudeurs. Ce changement de comportement graduel aussi appelé concept *drift* permet aux fraudeurs de tester les limites du système et d'augmenter graduellement ses méthodes sans déclencher les alarmes. Dans le cadre de ce travail nous avons exploré plusieurs modèles de détection de fraude dont des modèles supervisés notamment, XGBoost, la régression logistique et TabNet ainsi qu'une approche non supervisée de détection d'anomalies par auto-encodeur. Afin d'améliorer leur performance dans un contexte de données fortement déséquilibrées, nous avons intégré une approche de *cost-sensitive learning*, appliquée spécifiquement aux modèles XGBoost, TabNet et régression logistique. Notre principale contribution réside dans la mise en place d'un modèle hybride qui combine XGBoost–TabNet, permettant d'exploiter les forces respectives de ces deux approches et d'améliorer la détection des fraudes dans le cas de changements comportementaux graduels.

1.2.2 Problèmes de recherche

Dans le cadre de ce travail de recherche consacré à la détection de la fraude financière par l'apprentissage automatique, plusieurs défis méthodologiques et techniques ont été rencontrés. Ces défis sont présentés comme suit :

1. **Données privées.**
   Un système de détection de fraude devient complexe lorsque les ensembles de données disponibles sont limités et que la période de détection est limitée.
2. **Déséquilibre des classes.**
   Dans le système financier, les transactions frauduleuses sont extrêmement rares contrairement aux transactions légitimes qui priment entraînant un déséquilibre sévère entre les classes dans les jeux des données. Le déséquilibre intrinsèque des données peut toujours influencer la performance du modèle.
3. **Variabilité graduelle des comportements frauduleux.**
   Les fraudeurs ne conservent pas un schéma d'action fixe. Ils adaptent progressivement leurs stratégies afin de contourner les systèmes de détection existants rendant la détection plus complexe.
4. **Stratégie complexe.**
   L'utilisation de multiples modèles et techniques nécessite une expertise rigoureuse.

Dans ce contexte, nous avons analysé ces défis et proposé une approche plus robuste pour y faire face. La méthodologie adoptée à cet effet est présentée en détail dans le chapitre 3.

1.3 Approche

La détection de changements comportementaux graduels est une approche qui prend en compte l'évolution progressive du comportement des utilisateurs au fil du

temps. C'est une approche prometteuse et relativement peu explorée qui nous permet de détecter si le comportement dérive progressivement vers quelque chose de suspect. Les changements graduels se produisent de manière continue. Pour capturer ces évolutions, la détection graduelle repose sur une division temporelle des données. A cela s'ajoute le déséquilibre de classe rendant complexe l'exploitation des modèles. Nous avons de ce fait intégrer une chaîne de traitement performante capable de gérer l'important déséquilibre de classes et de capter les motifs complexes associés aux transactions frauduleuses.

Premièrement, pour faire face au déséquilibre des données, nous proposons l'approche du *cost-Sensitive learning*, un modèle spécifiquement conçu pour attribuer un coût différent aux erreurs de classification.

Deuxièmement, nous explorons une combinaison du *Cost-sensitive learning* avec XGBoost afin d'établir une classification supervisée robuste basée sur les arbres de décision.

Nous appliquons également une méthode d'apprentissage non supervisé pour la détection d'anomalies à l'aide d'un auto-encodeur qui apprend à reconstruire les transactions considérées comme normales.

Par ailleurs, nous avons testé d'autres modèles de référence, dont la régression logistique et l'utilisation de TabNet, un réseau neuronal profond particulièrement adapté aux données tabulaires.

Enfin, une méthode hybride innovante combinant XGBoost et TabNet est implémentée pour renforcer leurs performances respectives. En intégrant ces diverses approches, notre objectif est de construire un système de détection performant, autonome et robuste face aux données déséquilibrées, parfaitement adapté aux enjeux cruciaux du domaine financier.

1.3.1 XGBoost

XGBoost est un algorithme d'apprentissage ensembliste basé sur le gradient boosting. En partant d'une prédiction de base comme la moyenne, XGBoost réduit progressivement l'erreur en construisant de nouveaux arbres pour prédire les résidus de l'itération précédente, les intégrant au modèle via un taux d'apprentissage et une fonction de perte qui pénalise la complexité par régularisation des paramètres.

Hypothèses

- Les relations entre variables peuvent être facilement capturées par des décisions binaires successives ;
- Les features importantes peuvent être identifiées de manière itérative ;
- La combinaison additive d'arbres faibles améliore progressivement la performance.

XGBoost est particulièrement adapté à la détection de fraude pour plusieurs raisons concrètes. Il se distingue par sa haute performance sur les données tabulaires déséquilibrées [3] et sa capacité à modéliser des évolutions comportementales progressives, tout en intégrant la pondération des classes, la régularisation et la gestion efficace des valeurs manquantes. De plus, il fournit des scores d'importance des variables, permettant d'identifier les indicateurs de fraude les plus pertinents, et sa rapidité d'exécution le rend viable pour des applications en temps réel. Il s'agit d'un modèle aux applications multiples dans divers domaines, notamment dans le secteur de la santé, de la géomatique et de la cybersécurité. Par exemple, dans le domaine médical, Ogunleye et al. en 2019 [4] ont utilisé ce modèle pour le diagnostic de plusieurs maladies rénales. Dans un autre contexte, ce modèle a été combiné à la méthode SHAP afin d'analyser l'hétérogénéité géospatiale de la susceptibilité aux glissements de terrain, donnant naissance au modèle SHAP-XGBoost par Zhang et al. [5].

XGBoost vise à sélectionner itérativement les règles les plus pertinentes à partir

des données d'apprentissage, en se basant sur des critères comme le lift et le support. À chaque étape, les règles peu informatives sont éliminées, tandis que les plus efficaces sont conservées afin de construire un ensemble optimal de caractéristiques [6].

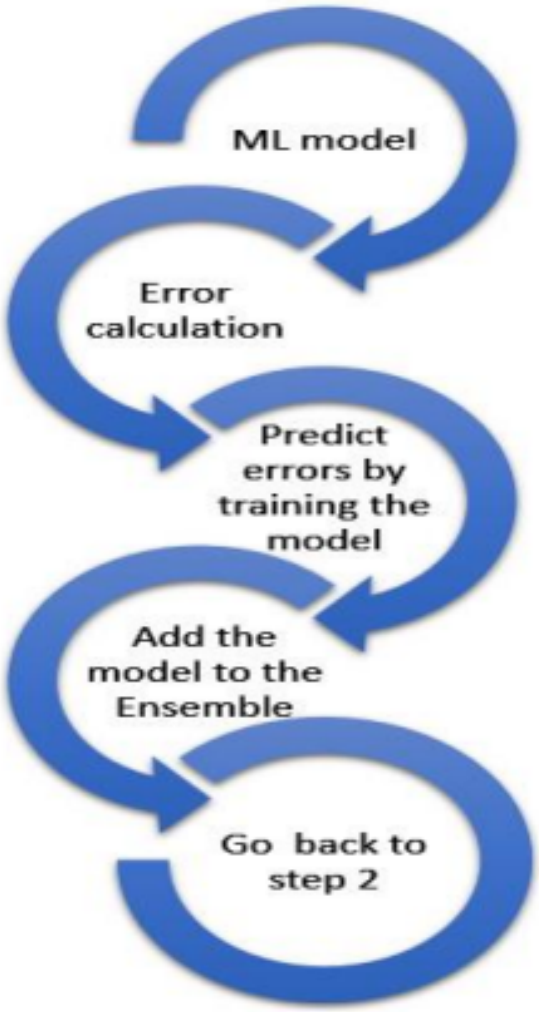

FIGURE 1.1 – Flux d'apprentissage de XGBoost

1.3.2 Tabnet

TabNet est une architecture de deep learning spécialement conçue pour les données tabulaires. Son innovation principale réside dans le mécanisme d'attention séquentielle qui sélectionne les features les plus pertinentes à chaque étape de décision. À chaque

étape, un masque d'attention sparse identifie quelles features sont importantes, créant ainsi une sélection de features adaptative et interprétable.

Hypothèses

— TabNet suppose que toutes les features ne sont pas également importantes pour chaque prédiction et qu'une sélection adaptative améliore la performance ;

— Les décisions peuvent être prises de manière séquentielle, chaque étape affinant la compréhension ;

— Le modèle assume que l'apprentissage non supervisé sur la reconstruction des features peut améliorer les représentations, particulièrement utile quand les données étiquetées sont limitées.

TabNet apporte plusieurs avantages uniques pour la fraude. À partir d'un ensemble de données prétraitées, Tabnet est mis en œuvre pour obtenir une grande précision dans l'identification des transactions frauduleuses.[7]. Pour chaque transaction, on peut voir quelles features ont conduit à la classification, essentiel pour justifier les blocages de transactions auprès des clients grâce a son mécanisme d'attention. De plus TabNet gère bien les features catégorielles nombreuses comme les types de marchands, pays, etc. sans nécessiter d'encodage one-hot massif. Enfin, il peut apprendre des dépendances temporelles subtiles dans les séquences de transactions.

L'architecture de TabNet repose sur un processus séquentiel de traitement de l'information, décomposé en plusieurs étapes clés d'après les recherches de Arik et al. [8] :

(a) *TabNet encoder architecture* : Le modèle traite les données de manière séquentielle à travers plusieurs étapes de décision. À chaque étape, l'information est filtrée, transformée, puis séparée pour alimenter à la fois la prédiction finale et l'étape suivante.

(b) *TabNet decoder architecture* : Ce bloc permet au modèle de reconstruire les

caractéristiques d'origine à partir des représentations encodées, une étape cruciale pour le pré-entraînement.

(c) *Feature Transformer* : Ce composant traite les données via une structure de réseau profond à quatre couches. Il combine des connaissances globales via deux couches partagées par toutes les étapes et des analyses locales via deux couches propres à chaque étape de décision.

(d) *Attentive Transformer* : Ce mécanisme utilise une couche Sparsemax pour générer un masque de sélection. En se basant sur les informations des étapes précédentes, il décide précisément quelles variables doivent être "allumées" ou "éteintes" pour la décision en cours.

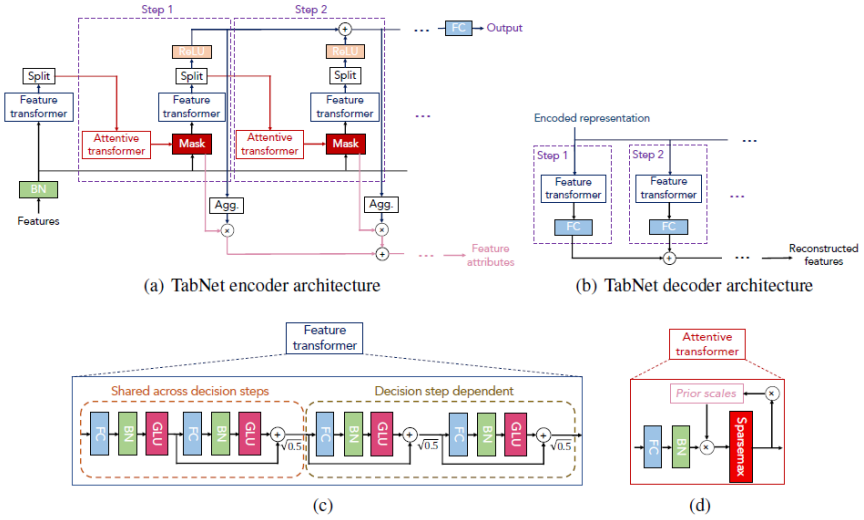

FIGURE 1.2 – Architecture de Tabnet

1.3.3 Régression logistique

La régression logistique est un modèle linéaire de classification qui estime la probabilité d'appartenance à une classe via la fonction sigmoïde. Il est utilisé pour résoudre des problèmes où il existe une relation entre les variables d'entrée et de sortie [9]. Le

modèle calcule une combinaison linéaire des features pondérées par des coefficients, puis transforme ce score via $\sigma(z) = \frac{1}{1+e^{-z}}$ pour obtenir une probabilité entre 0 et 1. L'apprentissage optimise les coefficients en maximisant la vraisemblance ou minimisant la log-loss via des méthodes comme la descente de gradient.

Hypothèses

— Indépendance des observations entre elles ;

— Absence de multicolinéarité forte entre les prédicteurs ;

— Les erreurs suivent une distribution binomiale ;

— Elle suppose que la vraie frontière de décision est approximativement linéaire dans l'espace des features.

Bien que simple, la régression logistique reste pertinente pour plusieurs raisons. Elle sert de baseline robuste pour évaluer la complexité nécessaire. De plus elle est très stable et peu sensible au surapprentissage avec régularisation appropriée. Enfin, elle nécessite peu de données comparé aux modèles complexes et fonctionne bien quand des features capturent déjà les patterns non-linéaires.

1.3.4 Auto-encodeur

Un auto-encodeur est un réseau de neurones non supervisé composé de deux parties dont un encodeur qui compresse les données en une représentation latente de dimension réduite, et un décodeur qui reconstruit les données originales depuis cette représentation. Pour la détection d'anomalies, le principe est d'entraîner l'auto-encodeur uniquement sur des transactions légitimes. Le modèle apprend ainsi à bien reconstruire les patterns normaux. Les transactions frauduleuses, ayant des patterns différents, seront mal reconstruites, produisant une erreur de reconstruction élevée qui sert de score d'anomalie [10].

Hypothèses

— Les données normales peuvent être compressées en une représentation de dimension inférieure sans perte majeure d'information ;
— Les fraudes ne partageant pas cette structure, se manifesteront par des erreurs de reconstruction élevées ;
— La classe majoritaire considérée comme des transactions légitimes est suffisamment homogène pour être modélisée efficacement.

L'auto-encodeur fonctionne en mode semi-supervisé, et ne nécessite que des données normales pour l'entraînement, ce qui résout le problème du déséquilibre extrême des classes. Il peut détecter des fraudes nouvelles, car il identifie tout ce qui dévie du normal plutôt que d'apprendre des patterns de fraude spécifiques.

1.3.5 Hybridation XGBoost + Tabnet

Ce modèle hybride combine les forces complémentaires de XGBoost et TabNet selon une architecture par ensemble ou *stacking*. Cette approche consiste à entraîner XGBoost et TabNet indépendamment, puis leurs prédictions sont combinées via un méta-modèle de régression logistique. Cette approche exploite la diversité des modèles.

Hypothèses

— XGBoost excelle sur les patterns basés sur des seuils et interactions locales ;
— TabNet capture des dépendances globales et des sélections de *features* contextuelles ;
— La combinaison réduit le biais et la variance individuels selon le principe du *bagging* et *boosting* ;
— L'ensemble surpasse les composants individuels grâce à la diversité des approches.

Le modèle hybride offre des avantages particulièrement puissants pour la fraude. Il combine la robustesse de XGBoost sur les features numériques avec la capacité de TabNet à gérer efficacement les features catégorielles et l'attention. XGBoost apporte

rapidité et efficacité pour le *scoring* en production, tandis que TabNet ajoute l'interprétabilité via l'attention et la capacité de pré-entraînement. La diversité des modèles réduit le risque qu'une fraude sophistiquée échappe aux deux systèmes simultanément.

1.4 Organisation du mémoire

Ce document est structuré en plusieurs chapitres, dont chacun aborde un aspect spécifique du travail de recherche. L'organisation du reste du mémoire est présentée comme suit :

- L'état de l'art au chapitre 2 présente les travaux les plus importants en passant par les méthodes traditionnelles aux modernes de la détection de fraude.
- La méthodologie au chapitre 3 présente les modèles de *machine learning* et les métriques utilisées.
- Le chapitre 4 présente les expérimentations globales.
- Le chapitre 5 compare les performances et présente les autres résultats des expérimentations.
- La conclusion au chapitre 6 pour les discussions et la synthèse finale de notre travail de recherche.

Chapitre 2

État de l'art

Dans ce chapitre, nous présentons un résumé de l'état d'art en mettant en perspective les idées, les théories et les concepts, qui étaient nécessaires à comprendre pour la réalisation de notre travail de recherche. Nous abordons la question de la gestion de l'évolution des schémas de la fraude ensuite nous présentons les différentes méthodes utilisées à travers quelques publications.

2.1 Revue de la littérature

Une information financière fiable est une nécessité et non un choix comme l'ont souligné G. Souad et al., [11]. Plusieurs recherches sont réalisées par des chercheurs pour étudier les différentes méthodes de détection de fraude financière. Les nouvelles technologies ont révolutionné le monde de la finance en impactant significativement le travail des entreprises et la fiabilité des tiers.

2.1.1 Concept de la fraude

La fraude est un concept qui se défini par l'intention qu'a une personne de tromper une autre pour l'induire en erreur. Selon J.Le Maux et al., [12] dans leur revue intitulée « De la fraude en gestion à la gestion de la fraude », ce phénomène repose sur trois éléments essentiels : l'intention délibérée, la volonté de dissimulation et le mode opératoire utilisé pour atteindre cet objectif.
D'après le sondage de *Global Economic Crime Survey* [1], le taux de fraude économique était de 34 % en 2011 sur les douze derniers mois étudiés, contre 30 % en 2019. Ces chiffres confirment que la fraude demeure un phénomène récurrent et en constante évolution. De plus, près d'une personne sur dix ayant déclaré un cas de fraude a subi des pertes supérieures à cinq millions de dollars américains. À cet effet, il est nécessaire de surveiller des transactions suspectes, c'est d'ailleurs une méthode de détection de fraude très efficace.

Le tableau 2.1 ci-dessous met en évidence un taux élevé de fraudes signalées dans les pays dont l'économie est développée ou en forte croissance, tandis que le tableau 2.2 présente un taux plus faible de fraudes déclarées. Ce faible taux pourrait s'expliquer, selon le *Global Economic Crime Survey* [1], par une capacité de détection plus limitée ou par des contraintes liées à la confidentialité des informations.
Par ailleurs, entre 2009 et 2011, le taux de fraude signalée a augmenté dans la majorité des pays, à l'exception de l'Afrique du Sud, du Mexique, de la Slovaquie, de la Grèce, de l'Indonésie et du Japon, où une baisse relative a été observée.

Figure 8: Reported fraud by territory

Territories that reported high levels of fraud (40% or more)	% respondents 2011	% respondents 2009
Kenya	66%	57%
South Africa	60%	62%
UK	51%	43%
New Zealand	50%	42%
Spain	47%	35%
Australia	47%	40%
Argentina	46%	39%
France	46%	29%
USA	45%	35%
Malaysia	44%	28%
Mexico	40%	51%

TABLE 2.1 – Territoires ayant signalé un niveau élevé de fraude [1]

Territories that reported low levels of fraud (below 25%)	% respondents 2011	% respondents 2009
Romania	24%	16%
India	24%	18%
Sweden	22%	19%
Slovakia	21%	29%
Turkey	20%	15%
Switzerland	18%	17%
Netherlands	17%	15%
Italy	17%	19%
Greece	17%	23%
Slovenia	17%	(didn't participate in 2009)
Indonesia	16%	18%
Japan	6%	10%

TABLE 2.2 – Territoires ayant signalé un niveau faible de fraude [1]

Donald R. Cressey [13], pour sa thèse en doctorat durant les années 1940 a mené une enquête auprès de 200 criminels condamnés pour Fraude . Cela lui a permis de créer le *Triangle de la fraude* [13] pour comprendre la logique des fraudeurs. Le triangle de Cressey est composé de trois facteurs différents qui sont :

— La pression financière qui amène un individu à commettre un acte frauduleux.

— L'opportunité qu'a un fraudeur d'utiliser les failles du contrôle interne ou encore son sentiment que son acte ne sera pas détecté.
— La rationalisation concerne les arguments qu'utilise le fraudeur pour justifier ses faits, il n'a pas le sentiment de frauder lors de son premier acte.

Au fil du temps, un nouveau modèle basé sur le modèle de Cressey [13] a été développé pour comprendre les actes de ces fraudeurs : *le triangle de l'acte frauduleux* par Kranacher et al., [14] et Dorminey et al., [15]. Ce modèle présente également trois facteurs qui sont les suivants :

— L'acte frauduleux considéré comme la méthodologie mise en place.
— La dissimulation est la technique utilisée pour couvrir les actes de la fraude comme la destruction des documents.
— La conversion qui consiste à transformer les gains de la fraude en actifs utilisables comme le blanchiment d'argent.

Bolton et al., [16] identifient dans leur étude quatre types principaux de stratégies frauduleuses. Premièrement, la fraude par faillite ou *Bankruptcy fraud* consiste à utiliser sa carte de crédit tout en étant insolvable, sans intention de remboursement. Deuxièmement, la fraude par vol et contrefaçon ou *Theft fraud / Counterfeit fraud* se produit lorsqu'une carte de paiement volée est utilisée physiquement ou que les informations de la carte sont exploitées à distance. Troisièmement, la fraude à la demande ou *Application fraud* implique la soumission d'une fausse identité lors de la demande d'une carte de crédit. Enfin, la fraude comportementale ou *Behavioral fraud* se manifeste lorsque les informations d'une carte de crédit légitime sont obtenues frauduleusement et utilisées pour effectuer des achats, principalement par téléphone ou en ligne.

Outre les stratégies mentionnées précédemment [16], d'autres techniques de fraude existent. Le *shoulder surfing* consiste pour le fraudeur à se positionner à proximité de sa victime et à observer par-dessus son épaule afin de dérober des informations confidentielles telles que le code PIN. Le *skimming*, quant à lui, implique le piratage d'une carte de crédit par l'enregistrement des données de la bande magnétique en vue d'une reproduction ultérieure, généralement en installant un faux lecteur de carte sur

un terminal de paiement [2]. La figure 2.1 présente un exemple de lecteur de carte utilisé pour le skimming.

FIGURE 2.1 – Cardblocker PIN reader terminal topper[2]

Aujourd'hui, nous assistons malheureusement à une évolution préoccupante des techniques de fraude. Du *shoulder surfing* traditionnel, les fraudeurs sont passés aux logiciels malveillants (*malware*), aux courriels frauduleux (*phishing*), aux messages

textes frauduleux (*smishing*) et aux appels téléphoniques trompeurs (*vishing*). Plus récemment, des techniques sophistiquées telles que l'imitation de la voix ou du visage par *deepfake* ont évolué, témoignant de la diversification des menaces.

2.1.2 Changement comportementaux graduels

Le concept de changement comportemental graduel décrit une modification lente et persistante du comportement d'une personne ou d'un compte au fil du temps. La détection de comportement graduel est une approche avancée utilisant le ML pour identifier la fraude qui se manifeste par une modification lente et progressive des habitudes normales d'un client ou d'un compte. Ce phénomène est appelé dérive de concept ou *Concept Drift.*

L'objectif principal est de repérer les fraudes à faible intensité utilisées par des fraudeurs sophistiqués pour contourner les systèmes de détection basés sur des règles statiques.

Dans les systèmes financiers, un fraudeur peut dans un premier temps adopter un comportement très proche de celui d'un utilisateur légitime, puis modifier progressivement certains paramètres tels que le montant des transactions, leur fréquence, la localisation géographique ou encore le type de marchands.

Ce changement progressif lui permet souvent d'échapper aux modèles de détection traditionnels, généralement fondés sur des seuils fixes ou sur la détection d'anomalies ponctuelles. Dans ce contexte, les systèmes de détection de dérive comportementale cherchent à établir une ligne de base dynamique propre à chaque utilisateur, construite à partir de ses habitudes transactionnelles. Le modèle analyse ensuite en continu les nouvelles activités afin d'identifier les déviations statistiques par rapport à ce profil de référence, signalant ainsi un possible changement de comportement graduel.

Andrea Dal et al., [17] expliquent comment cette dérive, causée par l'évolution des habitudes des clients ou l'adaptation des fraudeurs, impacte directement les modèles

de détection de fraude financière et rend leur performance non stationnaire.

Dans la même optique, Oluwadare Samuel et al., [18] mettent en lumière le défi du *Concept Drift* en mentionnant que le problème principal est que les modèles formés sur des données historiques ne capturent plus précisément les schémas de fraude en évolution, correspondant exactement au changement graduel.

2.1.3 Panorama des techniques de l'apprentissage automatique

Le *machine learning*, branche de l'intelligence artificielle, offre aux économistes et aux institutions financières de nombreux outils pour l'analyse et la prévention de la fraude. Une panoplie d'algorithmes est ainsi utilisée pour la détection de transactions frauduleuses. Ce chapitre présente les différents algorithmes utilisés, classés selon leur type d'apprentissage, à travers un panorama de techniques allant des approches traditionnelles aux méthodes modernes.

Apprentissage supervisé

L'apprentissage supervisé a pour objectif d'apprendre une fonction de prédiction à partir d'un ensemble de données étiquetées, c'est-à-dire des données dont les résultats sont connus. Cette fonction permet ensuite de prédire les résultats pour de nouvelles données non étiquetées en se basant sur les patterns identifiés dans l'échantillon d'apprentissage.

Bart Baesens et al., [19] ont utilisés la méthode de détection classique qui est la régression logistique pour une classification. Leurs travaux consistent à combiner des techniques de machine learning avec l'analyse de graphes pour détecter des schémas de fraude complexes, comme la fraude en assurance ou la fraude bancaire. Ils expliquent que les méthodes de classification et de régression pour la détection de fraude sont utilisées simultanément.

D'autres algorithmes tels que les machines à vecteurs de support (SVM), les réseaux neuronaux et des méthodes d'ensemble (*bagging, boosting*) font également partie de leurs travaux menés pour la détection de fraude.
Akash Gandhar et al., [20] offrent une revue approfondie des approches en matière de détection de fraude en mettant l'accent sur des algorithmes tels que la régression logistique, les SVM, les forêts aléatoires et les réseaux de neurones. Ils ont utilisé des métriques telles que le rappel, la précision, la courbe *AUC-ROC*, F1-score et *log-loss* pour évaluer les performances des différents modèles. Le *log-loss* évalue la qualité des probabilité assignées aux classes prédictes. Leur étude a montré une amélioration significative de la détection des transactions frauduleuses avec une réduction de faux positifs ainsi qu'une augmentation de la fiabilité des systèmes.

En 2018 Aditya Oza [21] a fait une étude sur l'application du machine learning pour la détection de fraude à partir du SVM linéaire et du SVM avec noyau RBF (Radial Basis Function). Le SVM linéaire est un algorithme de classification qui cherche à trouver l'hyperplan optimal séparant les classes tandis que le SVM avec noyau est l'extension non linéaire de SVM linéaire qui permet de capturer des relations plus complexes entre les caractéristiques. Pour les transactions de type *transfert*, les résultats obtenus ont présenté des AUPRC (*Area Under the Precision-Recall Curve*) très élevés, indiquant une excellente performance de ces modèles, soit un AUPRC de près de 0,98 pour le SVM avec noyau RBF. Des résultats similaires ont été observés pour les transactions de type *cashout*.

Les travaux menés par Y. Zhang et D. Wang [22] montrent que le *Cost-Sensitive Learning* est utilisé pour renforcer un classifieur de base SVM afin qu'il prenne en compte les différents coûts d'erreur. Cette approche garantit que l'algorithme d'ensemble accorde une importance supérieure à la classification correcte de la classe minoritaire. Les résultats expérimentaux démontrent que la méthode d'ensemble intégrant le CSL et le QBC (*Query-by-Committee*) surpasse de nombreuses méthodes existantes, présentant de meilleurs scores pour les métriques cruciales dans le traitement des données

déséquilibrées.

L'article d'Alexandre Alfocéa [23] met en lumière l'importance du *Cost-Sensitive Learning* comme solution au défi majeur du déséquilibre des classes. L'auteur a appliqué un C*ostSensitive Decision Tree* et a défini un ratio de coût spécifique. Les résultats de la recherche soutiennent l'hypothèse selon laquelle les méthodes *cost-sensitive* sont plus efficaces pour détecter la fraude que les méthodes d'échantillonnage.

Nadir Sahllal [24] dans son approche pour les données déséquilibrées aborde les défis posés par les ensembles de données déséquilibrés dans le domaine de la publicité en ligne, en se concentrant sur la prédiction du taux de CTR *(Click-Through Rate)* et la détection de la fraude au clic. L'étude met en œuvre plusieurs techniques d'apprentissage automatique, notamment le *XGBoost* utilisé pour la classification binaire, en particulier pour la détection de la fraude au clic, des méthodes d'ensemble et des techniques de rééchantillonnage telles que *SMOTE* pour traiter le déséquilibre des classes. Les résultats ont montré un AUC atteignant 0,96 dans certaines expériences et un *LogLoss* avec des valeurs aussi basses que 0,15, indiquant une bonne calibration des probabilités prédites.

Beigh et al., [25] ont utilisé la régression logistique pour la détection de fausse monnaie, en s'appuyant principalement sur la précision pour évaluer les performances du modèle. Les résultats de la simulation démontrent que la régression logistique offre une précision élevée pour la détection de fausse monnaie.

Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, l'un des principaux défis de l'apprentissage non supervisé réside dans le traitement de données non étiquetées. Certaines méthodes non supervisées s'appuient sur la classification par regroupement ou *clustering*, considérant les anomalies comme les points qui s'éloignent du centre de gravité des classes. C'est dans cette optique que Bart Baesens et al., [19] ont utilisé le *clustering* par *k*-

means pour identifier des comportements atypiques, en complément de *scores Z* pour la détection d'anomalies.

Alexandre ALFOCEA [23] s'est principalement intéressé au déséquilibre des classes en utilisant la classification one-class SVM. Cette méthode se base uniquement que sur les données de la classe normale en détectant comme anomalie toutes les transactions en dehors de cette classe. Il a obtenu un pourcentage faible de 0,17 pour le F1 score, une précision de 0,22 et un rappel de 0,15.

Plusieurs algorithmes non supervisés de détection des anomalies utilisent des mesures de distance et similarité pour déterminer les observations qui s'éloignent des autres [26]. C'est dans cette optique que Fouad Jabiri [26] applique des méthodes de classification non supervisées plus précisement l'*isolation forest* pour la détection d'anomalie. Cette méthode permet d'isoler des anomalies rares et différentes, en supposant qu'elles sont plus facilement isolables que les données normales. Afin de rendre les séquences exploitables par l'*isolation forest*, il a utilisé le pooling, une technique qui combine les réclamations séquentielles en un seul vecteur. Il a obtenu des résultats satisfaisants en combinant ces deux méthodes avec une parfaite isolation.

Les techniques non supervisées dans le contexte d'audit est crucial. Sutapat et Miklos [27] dans le but d'explorer le *clustering* pour la détection d'anomalie dans les données comptables, regroupent les transactions similaires en huit *clusters* par la méthode des *K-Means Clustering*. Dans ce cas toutes les transactions éloignées du centre de leur *cluster* sont considérées comme des anomalies. Ils ont également utilisé le concept de probabilité d'appartenance qui relève du *clustering* probabiliste comme l'algorithme de maximisation de l'espérance (EM). Les résultats du *clusteing* (K-means et EM) ont permis d'identifier 169 sinistres avec une probabilité d'appartenance de 0,6.

Chergui et al.,[28] ont fait une étude non supervisée par l'auto-encodeur pour réduire la dimension. Cette méthode permet de conserver les performances tout en optimisant

le temps d'entrainements. Ils ont conclu que les auto-encendeurs sont meilleurs que le PCA pour toutes les métriques.

Autres approches d'apprentissage automatique

Plusieurs approches pertinentes d'apprentissage automatique telles que les méthodes hybrides et les méthodes d'ensemble sont mises en evidence pour la détection de fraude en finance. Des auteurs comme Periklis et Theophilos [29] dans leur étude ont précisemment mis la lumière sur ces approches. D'autres méthodes d'ensemble sont utilisées par Richard A et al., [30] notamment un modèle d'empilement appelé *stacking ensemble* combinant plusieurs algorithmes tels que le SVM, *Random Forest*, *Gradient Boosting*. Cette combinaison a donné de meilleurs résultats d'une précision de 92,79 % avec un AUC de 96,98%.

Dans le même contexte O. Ramyateja et al., [31] ont combiné *over-sampling/under-sampling* et des algorithmes d'ensemble comme *Random Forest* et ont obtenu de meilleures performances dans des jeux de données déséquilibrées. L'apprentissage semi-supervisé n'est que très peu exploré à raison de 2% de publications comme l'ont indiqué Juan Bernardo et al., [32] dans leur revue de la littérature.
Les approches de *boosting* sont sujettes au sur-apprentissage, car elles visent à réduire itérativement l'erreur du modèle sur l'ensemble d'apprentissage comme montre J.QUINLAN dans son ouvrage [33].

À l'origine, la détection de fraude reposait sur des méthodes basiques de statistiques définies par des règles manuelles. Cette méthode consiste à fixer une limite de montant qui sera jugée frauduleuse si elle se trouve dépassée. Par faute du temps d'exécution et l'évolution des techniques frauduleuses cette approche ne parvenait plus à s'adapter comme expliquent R. J. Bolton et al., [16]. Puis s'en suit l'age d'or du machine learning traditionnel qui révolutionne le domaine par ses algorithmes de classification tels que la régression logistique, les forêts aléatoires et par sa capacité de gestion du déséquilibre grâce au sur-échantillonnage.

Aujourd'hui la révolution de l'apprentissage profond et de l'intelligence artificielle continue avec des techniques plus performantes pour s'attaquer aux techniques de fraudes de plus en plus complexes. Le machine learning qui était autre fois un outil de classification basique est passé à un système préventif et adaptatif qui évolue continuellement en s'orientant vers des modèles plus intelligents qui non seulement détectent la fraude, mais apprennent et s'adaptent en permanence pour anticiper les nouvelles menaces.

Chapitre 3

Méthodologie

Ce chapitre présente l'approche méthodologique adoptée pour la détection de la fraude par apprentissage automatique. Il décrit de manière détaillée les différentes étapes suivies, depuis la préparation des données jusqu'à l'évaluation des performances des modèles. L'étude s'appuie sur la construction et la comparaison de plusieurs modèles, notamment XGBoost, TabNet, la régression logistique et l'auto-encodeur, auxquels a été intégrée une approche de *cost-sensitive learning* afin de pallier le déséquilibre des classes. Par ailleurs, une hybridation entre XGBoost et TabNet a été proposée dans le but d'exploiter les forces complémentaires de ces deux modèles pour améliorer la détection des fraudes.

3.1 Cost-sensitive learning

Le *cost-sensitive learning* est une approche algorithmique qui vise à corriger le déséquilibre des classes en tenant compte des coûts associés à la mauvaise classification des observations. Elle consiste à adapter les algorithmes d'apprentissage et la théorie de la classification de manière à les rendre sensibles aux coûts d'erreur.

Dans notre jeu de données, les transactions frauduleuses représentent une proportion très faible comparée aux transactions légitimes, ce qui peut conduire les modèles classiques à ignorer la classe minoritaire.
Pour pallier ce déséquilibre, nous avons appliqué la technique du *cost-sensitive learning*, qui introduit un poids supplémentaire à la classe minoritaire afin d'augmenter son influence lors de l'apprentissage. Le choix de cette approche repose sur sa capacité à préserver la distribution initiale des données, contrairement aux méthodes de rééchantillonnage.

Dans le cadre de ce travail, cette stratégie a été intégrée à chacun des modèles testés à savoir XGBoost, TabNet et régression logistique, à l'exception de l'auto-encodeur, pour lequel l'apprentissage non supervisé ne nécessite pas de pondération de classes. En effet, l'auto-encodeur n'a pas pour objectif de prédire une étiquette de classe c'est-à-dire fraude ou non-fraude, mais plutôt de reconstruire les entrées à partir d'un espace latent appris. Son apprentissage repose uniquement sur la minimisation de l'erreur de reconstruction. De plus, il ne manipule pas de variable cible y, et par conséquent, il n'existe aucune pondération de classes à introduire dans sa fonction de perte.

3.2 XGBOOST : Extreme Gradient Boosting

XGBoost est un algorithme de *boosting* particulièrement performant, reposant sur des ensembles d'arbres de décision. Il constitue une implémentation optimisée et évolutive du cadre de *Gradient Boosting* proposé par Friedman [34].
Cet algorithme se distingue par sa capacité à optimiser automatiquement la fonction de coût afin de produire des prédictions précises et robustes. Nous avons retenu XGBoost dans ce travail, d'une part, pour sa capacité à traiter efficacement des données tabulaires de grande dimension, typiques des environnements financiers, et d'autre part, pour son mécanisme intégré de régularisation qui permet de limiter le surap-

prentissage et d'améliorer la généralisation du modèle.

La classification devient particulièrement complexe lorsque la distribution des classes au sein de l'ensemble de données est fortement déséquilibrée. Pour remédier à cette difficulté, nous avons proposé dans ce travail une version sensible aux coûts de l'algorithme XGBoost, intégrant le principe du *cost-sensitive learning* afin d'améliorer les performances de détection de la classe minoritaire.
Le principe repose sur l'attribution d'un poids plus important aux observations rares, en proportion de leur sous-représentation dans les données. Cette pondération est réalisée à l'aide du paramètre *scale_pos_weight*, qui ajuste les transactions frauduleuses lors de l'apprentissage. Ce paramètre est défini comme suit :

$$\texttt{scale\_pos\_weight} = \frac{\text{Nombre de transactions non frauduleuses}}{\text{Nombre de transactions frauduleuses}}. \tag{3.1}$$

La classe minoritaire se voit ainsi attribuer un poids plus important qu'initialement. Ce principe repose sur l'idée que plus une classe est rare, plus il est crucial que le modèle y accorde de l'attention afin d'éviter les faux négatifs comme l'explique C. Elkan [35]. Ensuite, cette pondération est intégrée dans la fonction de perte logistique optimisée. Chaque transaction frauduleuse reçoit donc un poids supérieur défini comme suit :

$$\mathcal{L} = -\sum_{i=1}^{N} w_i \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right], \tag{3.2}$$

où

$$w_i = \begin{cases} \text{scale\_pos\_weight} & \text{si } y_i = 1 \quad \text{(transaction frauduleuse)}; \\ 1 & \text{si } y_i = 0 \quad \text{(transaction légitime)}; \end{cases}$$

y_i : ce que le modèle veut prédire ;
p_i : la probabilité de fraude. Ainsi, nous obtenons une fonction de perte logistique pondérée ou le poids w_i ajouté dépend de la classe.

3.3 Auto-encodeur

Le concept d'auto-encodeur a été introduit par LeCun en 1987 [36]. Il s'agit d'un réseau de neurones non supervisé qui apprend à encoder puis à décoder les données. Dans cette étude, l'auto-encodeur a été entraîné uniquement sur les transactions légitimes, représentant le comportement normal. Le choix de ce modèle repose sur sa capacité à détecter les anomalies : en apprenant la structure des transactions normales, il peut identifier les transactions frauduleuses comme des observations présentant une erreur de reconstruction élevée.

La figure 3.1 représente une structure de base d'un auto-encodeur constitué de deux grandes parties a savoir l'encodeur pour la réduction de la dimension des données et le décodeur pour la reconstruction de l'entrée originale à partir de la représentation réduite comme l'explique S. Caron [37]. Une entrée x est construite en passant par les deux fonctions afin d'obtenir une sortie $\hat{x}$. Cette équation est donnée par :

$$\hat{x} = p_\phi\{q_\theta(x)\}, \tag{3.3}$$

avec p_ϕ le décodeur et q_θ l'encodeur.

Cette méthode relève de l'apprentissage non supervisé du fait qu'elle ne nécessite aucune étiquette pour son entraînement. Le modèle reproduit ainsi la valeur de x elle-même. Le schéma répresente un réseau pleinement connecté où les blocs pourraient représenter les neurones et les liens seraient les poids, ou les paramètres du réseau.

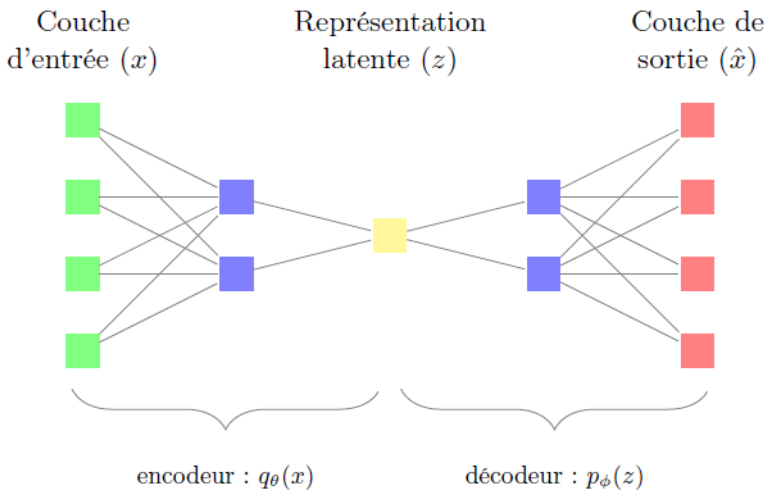

FIGURE 3.1 – Structure de base d'un auto-encodeur

Cette approche repose sur une modélisation non supervisée de la classe majoritaire et donc les transactions normales. Le modèle est constitué d'un réseau de neurones symétrique comprenant plusieurs couches entièrement connectée à savoir :

- une couche d'entrée de dimension égale au nombre de variables qui est de 437 ;
- un encodeur composé de couches de 32-16-8 neurones qui réduit progressivement la représentation des données ;
- un décodeur composé de couches de 16-32-437 neurones qui reconstruit les transactions à partir de cette représentation compressée.

Lors de l'entraînement pour la reconstruction des transactions normales, le modèle minimise l'erreur quadratique moyenne (MSE) entre les données d'entrée et leur reconstruction. Après l'entraînement, le modèle est appliqué sur l'ensemble de test qui contient à la fois des transactions normales et frauduleuses ainsi l'erreur de reconstruction est calculée pour chaque transaction :

$$\texttt{MSE}_i = \frac{1}{d}\sum_{j=1}^{d}(x_{ij} - \hat{x}_{ij})^2, \tag{3.4}$$

où x_{ij} est la valeur réelle et $\hat{x}_{ij}$ est la valeur reconstruite pour la variable j de l'obser-

vation i.

Un seuil T de classification est fixé au 95e percentile des erreurs de reconstruction observées sur l'ensemble de test. La règle de décision est définie comme suit :

$$\hat{y} = \begin{cases} 1 & \text{si } \text{MSE}_i > T \quad \text{(Anomalie)}; \\ 0 & \text{si } \text{MSE}_i \leq T \quad \text{(Transaction normale)}; \end{cases}$$

où

$$T = P_{95}(\text{MSE}_i \text{ pour les transactions normales}).$$

3.4 Tabnet

TabNet est un algorithme d'apprentissage profond performant et interprétable, spécialement conçu pour les données tabulaires. Les travaux de S. Ö. Arik et al., [8] expliquent que cet algorithme utilise un mécanisme d'attention séquentielle pour sélectionner les caractéristiques pertinentes à chaque étape de décision, lui permettant d'offrir à la fois une meilleure interprétabilité et un apprentissage plus efficace . Ce modèle a été choisi pour sa capacité à capturer les interactions complexes entre variables et à gérer efficacement les jeux de données hétérogènes.

Dans notre implémentation, TabNet est employé comme classificateur supervisé pour la prédiction des transactions frauduleuses à partir de variables tabulaires. Afin de mieux gérer le déséquilibre des classes, le modèle intègre le paramètre *class_ weight*, qui calcule automatiquement des poids équilibrés pour chaque classe. Ainsi, les transactions frauduleuses, étant minoritaires, se voient attribuer un poids plus élevé à l'instar de l'approche utilisée avec XGBoost. Dans le cadre d'une classification binaire, TabNet minimise généralement la fonction de perte logarithmique *log-loss*, définie comme suit :

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \Big[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \Big], \tag{3.5}$$

où y_i représente la classe réelle de l'observation i et $\hat{y}_i$ la probabilité prédite par le modèle.

Ensuite pour pallier le déséquilibre entre les classes, le poids est intégré dans la fonction de perte afin d'augmenter l'importance de la classe minoritaire et améliorant ainsi la capacité du modèle à détecter les cas rares de fraude. La fonction de perte pondérée se définit alors comme suit :

$$\mathcal{L} = -\sum_{i=1}^{N} w_i \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right], \tag{3.6}$$

où w_i est le poids associé à i, défini par :

$$w_i = \begin{cases} \alpha & \text{si } y_i = 1 \quad \text{(transaction frauduleuse);} \\ 1 & \text{si } y_i = 0 \quad \text{(transaction légitime);} \end{cases}$$

avec $\alpha > 1$ représentant un facteur de pondération choisi pour compenser la rareté des transactions frauduleuses.

Les variables les plus pertinentes sont sélectionnées à chaque étape de décision. Un seuil de 0,5 est appliqué pour convertir les probabilités en classes binaires.

3.5 Régression logistique

La régression logistique est une méthode statistique utilisée pour modéliser la relation entre une variable dépendante binaire et un ensemble de variables explicatives continues ou catégorielles comme l'indique J. Desjardins [38]. Elle permet d'estimer la probabilité d'occurrence d'un événement en fonction des facteurs susceptibles de l'influencer, ce qui en fait un outil particulièrement adapté aux problèmes de classification, notamment en détection de fraude.

Comme l'ont souligné P. Odermatt et al., [39], l'un de ses principaux atouts réside dans sa capacité à mesurer la force et la direction de l'association entre chaque variable indépendante et la variable cible, tout en contrôlant l'effet des autres variables

incluses dans le modèle .

Notre approche consiste a utilisé le paramètre *class_ weight='balanced'* pour la pondération du modèle. Pour cela, Un poids est attribué à chaque classe. Ce poids est calculé comme suit :

$$w_c = \frac{N}{k \cdot n_c};$$

où N est le nombre total d'observations ; k le nombre de classes et n_c le nombre d'observations de la classe c.

Ensuite ce poids est intégré dans le modèle et nous obtenons une fonction logistique pondérée qui peut améliorer la capacité de la régression logistique en identifiant les fraudes rares définit par :

$$\mathcal{L} = -\sum_{i=1}^{N} w_i \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right]; \tag{3.7}$$

avec

$$w_i = \begin{cases} \alpha & \text{si } y_i = 1 \quad \text{(transaction frauduleuse);} \\ 1 & \text{si } y_i = 0 \quad \text{(transaction légitime).} \end{cases}$$

3.6 Hybridation

Cette méthode repose sur la combinaison de deux modèles supervisés performants, à savoir XGBoost et TabNet, dans le cadre de la détection de fraude.

Comme présenté dans les sections 3.2 et 3.4, chaque modèle a d'abord été pondéré afin de corriger le déséquilibre des classes c'est-à-dire XGBoost à l'aide du paramètre *scale_ pos_ weight*, et TabNet via le paramètre *class_ weight*.

Sur la base de ces versions pondérées, l'approche hybride proposée combine les deux modèles à travers une stratégie d'ensemble de type stacking. L'objectif est de bénéficier de la complémentarité entre les deux algorithmes en exploitant la robustesse de XGBoost dans la modélisation d'interactions complexes, tout en tirant parti de la capacité de TabNet à effectuer une sélection automatique et interprétable des variables

pertinentes.

La méthodologie adoptée consiste tout d'abord à entraîner séparément chacun des modèles de base, à savoir XGBoost et TabNet. À l'issue de chaque entraînement, ces modèles produisent une probabilité de prédiction indiquant qu'une transaction soit frauduleuse.

Ces deux probabilités issues des modèles de base constituent alors les variables d'entrée du modèle hybride. Pour les combiner efficacement, une régression logistique est utilisée comme méta-classifieur dans une architecture de type *stacking*. Ce méta-modèle apprend à pondérer et à fusionner les prédictions issues de XGBoost et TabNet afin d'obtenir une estimation finale plus fiable.

Le modèle hybride ainsi obtenu produit une probabilité finale de fraude, notée P_h, à laquelle est appliqué un seuil de décision défini comme suit :

$$\hat{y} = \begin{cases} 1 & \text{si } P_h \geq 0.5, \\ 0 & \text{sinon,} \end{cases}$$

où $\hat{y} = 1$ indique une transaction frauduleuse prédite, et $\hat{y}= 0$ correspond à une transaction légitime.

Un seuil trop élevé risque de réduire le nombre de fraudes détectées, tandis qu'un seuil trop bas entraîne une augmentation des fausses alertes. Ainsi, le réglage optimal du seuil doit trouver un équilibre entre la précision et le rappel, en fonction des objectifs du système de détection.

3.7 Métriques d'évaluation

La performance des modèles prédictifs repose sur une combinaison de facteurs déterminants, notamment la qualité et la pertinence des données utilisées pour l'entraînement, ainsi que l'optimisation des hyperparamètres. Afin d'atteindre notre objectif de détection de fraude, nous avons exploré et mis en œuvre plusieurs modèles d'ap-

prentissage automatique.

Ce chapitre présente les différentes métriques d'évaluation que nous avons utilisées pour tester et évaluer la performance prédictive de nos modèles.

Matrice de confusion

La matrice de confusion est un outil essentiel pour comprendre et évaluer les modèles de classification supervisée. Elle permet d'analyser globalement les erreurs du modèle en révélant les correspondances et discordances entre les classes réelles et les classes prédites. Il s'agit d'un tableau de contingence de taille nn, où n représente le nombre de classes de notre ensemble de données. Cette matrice fonctionne en croisant les classes cibles réelles avec les classes prédites par le modèle. Ainsi, elle permet d'obtenir un classement des instances correctement classées et mal classées. Elle se présente comme suit :

		Classes actuelles	
		Positif	**Négatif**
Classes prédites	**Positif**	VP	FP
	Négatif	FN	VN
		Rappel	**Spécificité**

TABLE 3.1 – Exemple de matrice de confusion

VP : Vrais Positifs, c'est le nombre d'instances positives correctement classifiées ;

FP : Faux Positifs, c'est le nombre d'instances non positives prédites comme positives ;

FN : Faux Négatifs, c'est le nombre d'instances non négatives classifiées comme négatives ;

VN : Vrais Négatifs, c'est le nombre d'instances négatives correctement classifiées.

Le rappel

Aussi appelé sensitivité, il permet de mesurer le pourcentage des instances positives

correctement identifiées.

$$Rappel = \frac{VP}{(VP + FN)}. \tag{3.8}$$

La précision

Elle mesure la fiabilité des prédictions positives en indiquant la part des vrais positifs.

$$Precision = \frac{VP}{(VP + FP)}. \tag{3.9}$$

La courbe ROC

La courbe ROC est largement utilisée en apprentissage automatique et en traitement du signal. Elle permet de distinguer le signal du bruit en représentant graphiquement le taux de vrais positifs en fonction du taux de faux positifs pour tous les seuils de classification possibles.

L'AUC est une mesure qui permet de quantifier numériquement la performance des classifieurs.

— AUC = 1, un modèle parfait qui fait une séparation parfaite des instances positives ainsi que des instances négatives.

— AUC = 0.5, le modèle ne fait aucune distinctions entre les classes. À ce niveau ce modèle peut être considéré comme erroné. Chaque instance a une probabilité de $1/n$ d'être bien classée, où n est le nombre de classes.

— AUC < 0.5, le modèle fait pire qu'une classification aléatoire.

3.8 Environnement technique

Pour mener à bien ce travail de recherche, plusieurs outils ont été mobilisés. L'implémentation des modèles de détection de fraude a été réalisée en utilisant Python, un langage de programmation largement adopté dans divers domaines informatiques tels que l'intelligence artificielle et l'analyse de données. Ce langage a été retenu pour

ce projet en raison de la richesse de ses bibliothèques et de sa facilité d'utilisation et de compréhension.
L'environnement de développement choisi est Google Colab, une plateforme collaborative basée sur le cloud, développée par Google. Cette plateforme gratuite, particulièrement adaptée à l'apprentissage automatique, offre un accès à des ressources de calcul performantes telles que des GPU et des TPU.

Pour la préparation des données, l'exploration et la modélisation, nous avons utilisé les principales bibliothèques suivantes :

- **NumPy** permet un stockage et une manipulation efficaces des tableaux denses typés en Python. Il est adapté au traitement de données numériques homogènes.
- **Pandas** permet à python de travailler avec des données de type tableur pour un chargement, une manipulation, un alignement, une fusion, etc rapides en fournissant de nombreuses fonctions et méthodes qui accélèrent les étapes d'analyse et de prétraitement des données.
- **Scikit-learn** est un module python intégrant une large gamme d'algorithmes d'apprentissage automatique de pointe pour les problèmes supervisés et non supervisés à moyenne échelle [40]. Il est utilisé en ajustant les paramètres *scale_pos_weight* et *class_weight* dans les modèles XGBoost, Tabnet et la régression logistique. De plus il est crucial pour traiter le déséquilibre des classes.
- **TensorFlow** ou **PyTorch** pour l'apprentissage profond avec un mécanisme d'attention adapté aux données tabulaires. Il apprend à reconstruire le comportement normal des transactions.
- **pytorch-tabnet** est un réseau neuronal de pointe conçu spécifiquement pour les données tabulaires combinant la puissance des réseaux profonds avec l'interprétabilité des arbres de décision.
- **Keras** pour l'implémentation des auto-encodeurs, est une bibliothèque qui est intégrée à TensorFlow en tant qu'API dans la version 2.0 de TensorFlow.
- **Matplotlib** ou **Seaborn** pour la visualisation des données notamment les

matrices de confusion et les courbes ROC.

— **XGBoost** ou gradient boosting optimisé est une bibliothèque python performante. C'est un modèle de référence pour sa rapidité et sa précision très efficace en apprentissage automatique.

Chapitre 4

Expérimentation

4.1 Description du jeu des données

Le jeu de données utilisé pour notre travail de recherche provient des transactions e-commerce réelles de Vesta, un leader mondial des services de paiement. Il présente plusieurs caractéristiques, allant du type d'appareil aux caractéristiques des produits. Il a été rendu publique dans le cadre d'un concours sur la plateforme *Kaggle* en 2019 par *IEEE-Computational Intelligence Society*, un institut qui travaille dans divers domaines de l'intelligence artificielle et de l'apprentissage automatique. Cette base de données est l'une des sources de données les plus connues pour l'apprentissage automatique dans le domaine de la détection de la fraude.

En effet, ce jeu de données est important pour notre travail de recherche car elle nous permet de simuler un cas d'usage réel à grande échelle, ce qui est indispensable pour valider la robustesse de nos modèles. De plus il valide notre méthodologie en démontrant notre capacité à résoudre des problèmes concrets et complexes liés au déséquilibre, à la temporalité et au volume.

Fortement déséquilibré, il est divisé en deux fichiers *identity* et *transaction*, reliés par *TransactionID* puis composé de 437 variables détaillées dans le tableau ci-dessous :

Variables	**Description**
`TransactionDT`	Timedelta à partir d'une date/heure de référence donnée.
`TransactionAMT`	Montant du paiement de la transaction en USD.
`ProductCD`	Code produit, le produit pour chaque transaction.
`card1 - card6`	Informations sur la carte de paiement, telles que le type de carte, la catégorie de carte, la banque émettrice, le pays, etc.
`addr`	Adresse.
`dist`	Distance.
`P_ et (R__) emaildomain`	Domaine de messagerie de l'acheteur et du destinataire.
`C1-C14`	Comptage, comme le nombre d'adresses associées à la carte de paiement, etc. La signification réelle est masquée.
`D1-D15`	Timedelta, comme les jours entre la transaction précédente, etc.
`M1-M9`	Correspondance, comme les noms sur la carte et l'adresse, etc.

TABLE 4.1 – Description des Variables

L'analyse de la matrice de corrélation voir figure 4.1) révèle que peu de variables présentent une corrélation significative entre elles. La majorité des relations observées sont faiblement linéaires, traduisant une faible dépendance entre les attributs.
La diagonale rouge représente, la corrélation parfaite de chaque variable avec elle-même. Par ailleurs, on note une corrélation relativement faible entre la plupart des variables de transaction et la variable indiquant la fraude. En revanche, une corrélation

plus marquée est observée entre les distances *dist1* et *dist2*, tandis que les variables liées aux cartes présentent des corrélations modérées, traduisant certaines similarités dans leurs comportements transactionnels.

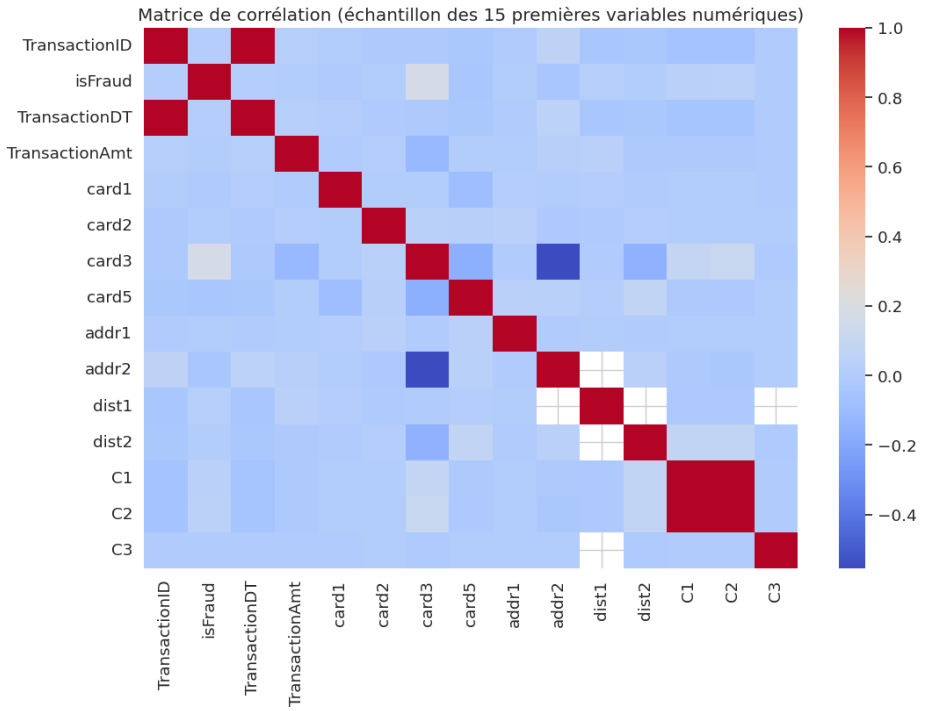

FIGURE 4.1 – Matrice de corrélation

Une étape de statistiques descriptives est réalisée pour nous permettre de mieux comprendre, nettoyer et préparer les données avant toute modélisation.

À partir du tableau 4.2 des variables numériques, nous pouvons voir que le jeu de données est fortement déséquilibré au niveau de la variable cible *isFraud* avec une moyenne indiquant un taux de fraude d'environ 3.68%. Par ailleurs, l'écart-type de *TransactionAmt* est supérieur à la moyenne, suggérant une forte variabilité des montants. De même, l'écart-type de *TransactionDT* par rapport à sa moyenne confirme une large répartition temporelle des transactions au sein de l'ensemble de données.

Enfin, cette analyse a permis de détecter la présence de valeurs manquantes, nécessitant un traitement approprié.

Variable	**count**	**mean**	**std**	**min**
TransactionID	53149.0	$3.282435e+06$	$1.704684e+05$	2987002.000
isFraud	53149.0	$3.678338e-02$	$1.882313e-01$	0.000
TransactionDT	53149.0	$7.376918e+06$	$4.616459e+06$	86469.000
TransactionAmt	53149.0	$1.340188e+02$	$2.308628e+02$	0.292
card1	53149.0	$9.898004e+03$	$4.916096e+03$	1012.000
		...		
id_22	418.0	$1.620096e+01$	$7.268261e+00$	12.000
id_24	382.0	$1.286387e+01$	$2.297392e+00$	11.000
id_25	417.0	$3.343909e+02$	$9.749790e+01$	100.000
id_26	418.0	$1.497823e+02$	$3.263530e+01$	100.000
id_32	7001.0	$2.655506e+01$	$3.747285e+00$	16.000

TABLE 4.2 – Statistiques descriptives des variables numériques

À l'instar des variables numériques, les variables catégorielles présentent également un taux élevé de valeurs manquantes, comme l'indique le tableau 4.3.
L'analyse montre que le code *W* de la variable *ProductCD* est de loin le plus fréquent, ce qui suggère que la majorité des transactions sont associées à cette catégorie de produit ou de service. Par ailleurs, les cartes de type *visa* et la catégorie *debit* constituent les modes de paiement les plus couramment utilisés. En ce qui concerne la variable *P_ emaildomain*, bien qu'elle contienne un nombre important d'observations, elle présente 59 valeurs uniques, avec *gmail.com* comme domaine dominant. Ces valeurs uniques représentent le nombre total de catégories différentes d'une variable pouvant rendre complexe son encodage. De même, la variable *R_ emaildomain* comporte également 59 valeurs distinctes, nécessitant un traitement spécifique pour limiter la complexité et éviter la création d'un trop grand nombre de modalités lors de l'encodage.

Variable	**count**	**unique**	**top**	**freq**
ProductCD	53149	5	W	39541
card4	53019	4	visa	34547
card6	53019	4	debit	39501
P_emaildomain	44682	59	gmail.com	20510
R_emaildomain	12397	59	gmail.com	5151
M1	28722	2	T	28720
M2	28722	2	T	25737
M3	28722	2	T	22649
M4	27754	3	M0	17593
M5	21561	2	F	11928
M6	37896	2	F	20543
M7	21951	2	F	19058
M8	21953	2	F	13906
M9	21953	2	T	18526
id_12	12994	2	NotFound	11119
id_15	12691	3	Found	6016
id_16	11626	2	Found	5900
id_23	418	3	IP_PROXY :TRANSPARENT	291
id_27	418	2	Found	417
id_28	12691	2	Found	6789

TABLE 4.3 – Statistiques descriptives des variables catégorielles

4.2 Préparation des données

Plusieurs manipulations ont été effectuées pour garantir un bon nettoyage et une bonne optimisation de notre jeu de données.

Pour commencer, un sous-échantillonnage aléatoire de 30% de la base initiale a été

réalisé afin de réduire la complexité computationnelle et accélérer l'exécution des modèles, tout en préservant les caractéristiques statistiques du jeu de données complet. Ensuite, les valeurs manquantes ont été identifiées puis imputées à l'aide de constantes, de manière à prévenir les erreurs d'exécution et limiter les biais lors de l'entraînement des modèles. Les variables catégorielles ont été converties en valeurs numériques à l'aide de la méthode de *Label Encoding*, afin de faciliter leur prise en compte par les algorithmes de machine learning.

Enfin, l'ensemble des variables a été normalisé à l'aide de la méthode *StandardScaler*, pour garantir l'homogéneité des données et favoriser la convergence des modèles d'apprentissage.

Le jeu de données pré-traité a été divisé en 80 % pour l'entraînement et 20 % pour le test garantissant un découpage de manière temporel tout en conservant l'ordre chronologique des transactions.

Cette approche reflète une situation opérationnelle réaliste où le modèle est entraîné sur des données passées et évalué sur des transactions futures, ce qui est particulièrement adapté dans le cadre de la détection graduelle de fraude.

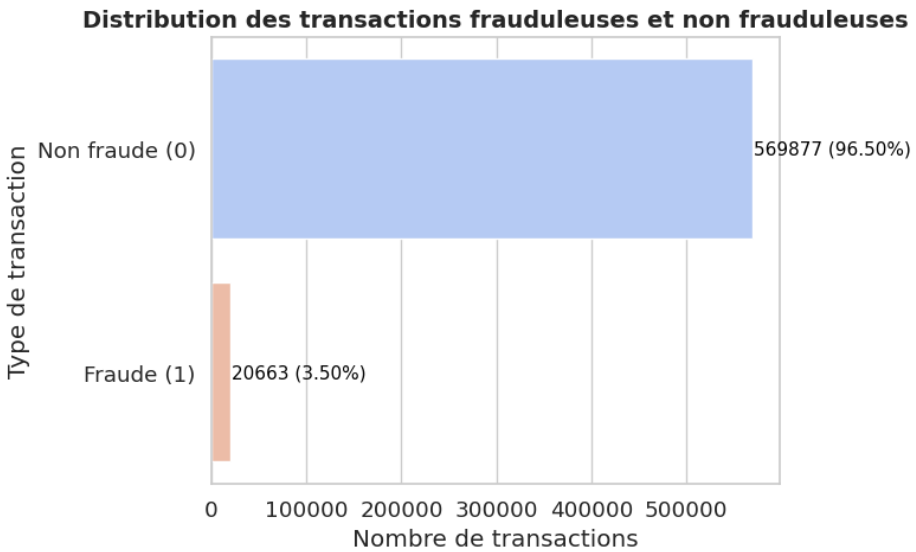

FIGURE 4.2 – Distribution des transactions

La figure 4.2 présente une distribution de l'ensemble des transactions de notre jeu de données. La proportion des transactions frauduleuses est de 3,5% contre celles légitimes de 96,5%.

4.3 Ingénierie des caractéristiques(Feature Engineering)

Étant donné la détection graduelle de fraude, nous avons réalisé une étape d'ingénierie des caractéristiques temporelles afin d'exploiter la dimension chronologique des transactions et de capter les dynamiques comportementales des utilisateurs. Pour cela, plusieurs variables dérivées ont été créées à partir de la variable brute *TransactionDT* comme suit :

- ***hour*** ou l'heure de la transaction est obtenue en divisant le temps écoulé en secondes par 3600, puis en le ramenant à un cycle de 24 heures ;
- ***day*** ou le jour de la semaine est calculé sur un cycle de 7 jours ;
- ***month*** ou le mois est dérivé sur un cycle de 12 mois.

Ces nouvelles variables permettent de mettre en évidence des schémas temporels récurrents par exemple, une fréquence accrue de fraudes à certaines heures ou certains jours.

Par ailleurs, afin de mieux capter les dynamiques locales des comportements financiers, des variables glissantes basées sur le montant des transactions *TransactionAmt* sont introduites. Ces nouvelles variables ont été agrégées par identifiant de carte *card1* afin de refléter les habitudes propres à chaque utilisateur. Deux indicateurs statistiques ont ensuite été calculés à partir de ces variables glissantes :

- ***TransactionAmt_roll_mean*** ou la moyenne mobile qui permet de détecter des écarts inhabituels par rapport au comportement habituel de l'utilisateur ;
- ***TransactionAmt_roll_std*** ou l'écart-type mobile qui reflète la variabilité locale des montants et permet d'identifier des anomalies soudaines.

Ainsi cette étape d'ingénierie des caractéristiques enrichit notre base initiale en intégrant à la fois des informations temporelles globales et des indicateurs dynamiques locaux en renforçant la capacité du modèle à capter les signaux faibles de fraude dans un contexte évolutif. La figure 4.3 nous montre une activité transactionnelle faible

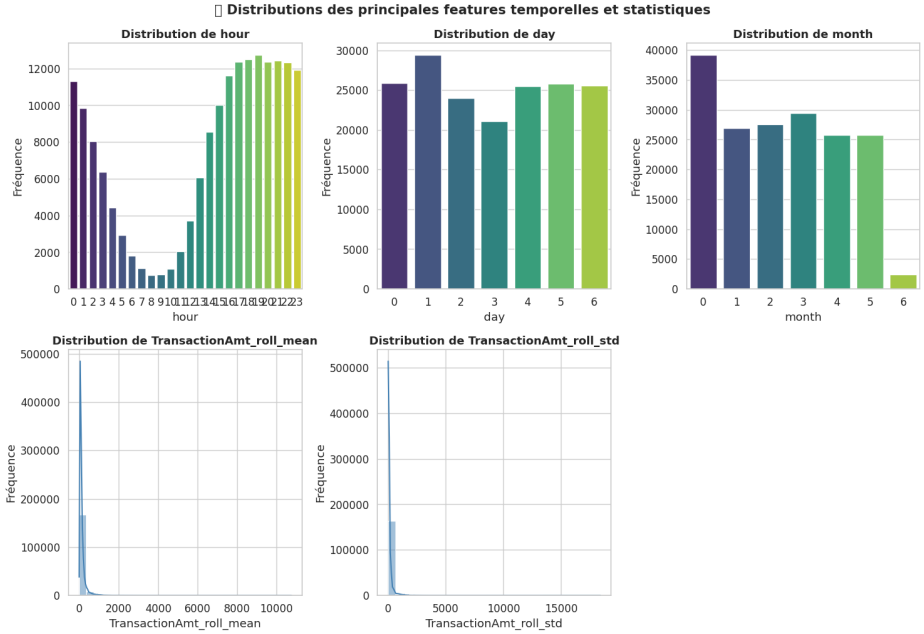

FIGURE 4.3 – Distribution des features temporelles

entre 0h et 7h correspondant aux heures nocturnes. À partir de 13h jusqu'à 23h, le volume des transactions augmente fortement, atteignant un pic en fin de journée.

La distribution du jour est relativement équilibrée sur les 7 jours, mais avec une légère hausse les jours 0 et 1 montrant que les activités financières sont continues, mais un léger effet hebdomadaire peut exister.

La distribution du mois montre que les valeurs sont concentrées sur les premiers mois, avec une diminution sur le dernier mois observé qui peut être du aux données partiellement collectées.

La distribution de *TransactionAmt_roll_mean* est très asymétrique à droite pouvant être du aux transactions avec une moyenne faible. La même tendance est constatée pour la distribution de *TransactionAmt_roll_std* pouvant traduire la stabilité des montants dépensés par les utilisateurs.

4.4 Gestion du déséquilibre des classes

Étant donné le déséquilibre marqué de notre jeu de données, une stratégie d'apprentissage sensible aux coûts de *cost-sensitive learning* a été appliquée à nos modèles d'apprentissage supervisé. Cette approche consiste à attribuer un poids plus élevé aux observations de la classe minoritaire c'est-à-dire la classe fraude, de sorte que les erreurs de classification de ces instances soient davantage pénalisées.

Dans le cas de XGBoost, le paramètre *scale_pos_weight* permet d'ajuster la fonction de perte afin d'améliorer la sensibilité du modèle face aux transactions frauduleuses. Ainsi, le modèle reste attentif aux signaux faibles et aux fraudes rares, même lorsqu'elles se manifestent de façon progressive.
Pour TabNet, la pondération introduite via le paramètre *class_weight* lors de l'entraînement a permis au réseau de mieux considérer la classe minoritaire dans son processus d'optimisation.
De même, pour la régression logistique, le schéma de pondération des classes appliqué via *class_weight="balanced"*, ajuste automatiquement les poids proportionnellement en fonction de l'inverse des fréquences de chaque classe, assurant ainsi un apprentissage plus équilibré.

L'un des principaux avantages de l'approche *cost-sensitive learning* réside dans le fait qu'elle n'altère pas la distribution originale des données, mais agit directement sur la phase d'apprentissage en forçant le modèle à accorder une importance à la classe minoritaire.
En revanche, contrairement aux modèles supervisés classiques, l'auto-encodeur ne traite pas directement le déséquilibre des classes. Son entraînement est réalisé uniquement sur la classe majoritaire c'est-à-dire les transactions normales, ce qui lui permet d'apprendre à reconnaître les schémas normaux et à détecter les anomalies en s'écartant de ce comportement appris.

Chapitre 5

Résultats

Ce chapitre présente et analyse les résultats expérimentaux obtenus à partir de l'application des différentes techniques et modèles de détection de fraude décrits dans le chapitre précédent 3. Les modèles testés incluent le *Cost-Sensitive learning* XGBoost, TabNet, la régression logistique pondérée, l'auto-encodeur pour la détection non supervisée, ainsi qu'une approche hybride combinant XGBoost et TabNet à l'aide d'une stratégie de stacking.
Les performances de ces modèles sont évaluées et comparées à travers divers indicateurs de performance, ainsi que par des graphiques et tableaux.

5.1 Présentation des résultats

5.1.1 Matrice de confusion

Nos résultats présentent un ensemble de 34199 transactions réellement non frauduleuses et de 1234 transactions réellement frauduleuses faisant un total de 35433 transactions dans notre ensemble de test. Les transactions légitimes identifiées comme

étant frauduleuses représentent les coûts opérationnels liés à la vérification manuelle tant dis que les transactions frauduleuses manquées dans l'ensemble de transactions frauduleuses réelles sont des pertes financières réelles due à la fraude non détectée.

XGBoost

	Prédit 0	**Prédit 1**	**Total (Réel)**
Réel 0	33501	698	34199
Réel 1	389	845	1234
Total (Prédit)	33890	1543	35433

TABLE 5.1 – Matrice de Confusion de XGBoost.

Le tableau 5.1 montre un total de 33501 transactions normales identifié correctement par le modèle sur les 34199 et a signalé 698 transactions légitimes comme étant frauduleuses qui représentent les coûts opérationnels liés à la vérification manuelle.
Sur un ensemble de 1234 cas de fraudes réelles, le modèle a correctement identifié 845 transactions frauduleuses et a manqué 389 transactions frauduleuses que nous qualifions de perte financière réelle due à la fraude non détectée.

Régression logistique

	Prédit 0	**Prédit 1**	**Total (Réel)**
Réel 0	27348	6851	34199
Réel 1	327	907	1234
Total (Prédit)	27675	7758	35433

TABLE 5.2 – Matrice de Confusion de la régression logistique.

Le tableau 5.2 montre sur l'horizontal, un total de 34199 transactions légitimes dont le modèle a correctement identifié 27348 transactions normales et 6851 transactions légitimes comme étant frauduleuses.
Ensuite sur un ensemble de 1234 transactions frauduleuses, 907 sont correctement identifiées comme frauduleuses contre 327 représentant des transactions frauduleuses qualifiées de normales.

Auto-encodeur

	Prédit 0	**Prédit 1**	**Total (Réel)**
Réel 0	32586	1613	34199
Réel 1	1075	159	1234
Total (Prédit)	33661	1772	35433

TABLE 5.3 – Matrice de Confusion de l'auto-encodeur.

Le tableau 5.3 nous indique horizontalement que sur les 34199 transactions qui initialement sont réellement normales, 32586 transactions sont correctement classées comme légitimes tandis que 1613 transactions normales sont classées comme frauduleuses.
Nous remarquons verticalement que sur les 33661 transactions prédites comme normales, 1075 sont identifiées comme frauduleuses tandis que sur les 1772 transactions prédites comme frauduleuses, 1613 sont normales et 159 sont effectivement frauduleuses.

Tabnet

	Prédit 0	**Prédit 1**	**Total (Réel)**
Réel 0	27751	6448	34199
Réel 1	344	890	1234
Total (Prédit)	28095	7338	35433

TABLE 5.4 – Matrice de Confusion de Tabnet.

Le tableau 5.4 montre un total de 34199 transactions légitimes dont le modèle a correctement identifié 27751 transactions normales et 6448 transactions légitimes comme étant frauduleuses.
Nous remarquons aussi que sur un ensemble de 1234 transactions frauduleuses, 890 sont correctement identifiées comme frauduleuses contre 344 représentant des transactions frauduleuses qualifiées de normales par le modèle.

Hybridation

Le tableau 5.5 nous indique horizontalement que sur les 34199 transactions qui initialement sont réellement normales, 34029 transactions sont correctement classées comme

	Prédit 0	**Prédit 1**	**Total (Réel)**
Réel 0	34029	170	34199
Réel 1	547	687	1234
Total (Prédit)	34576	857	35433

TABLE 5.5 – Matrice de Confusion hybride.

légitimes tandis que 170 transactions normales sont classées comme frauduleuses.
Sur la verticale, nous constatons que sur les 34576 transactions prédites comme normales, 547 sont identifiées comme frauduleuses tandis que sur les 857 transactions prédites comme frauduleuses, 170 sont normales et 687 sont éffectivement frauduleuses.

5.1.2 Scores-Courbe ROC

Pour rappel, l'AUC-ROC est une métrique fondamentale pour évaluer la performance des modèles de classification qui fournissent une probabilité, en particulier face à un déséquilibre de classes. Sa valeur est comprise entre 0 et 1. Plus l'AUC est proche de 1, plus la performance de discrimination est bonne.
Les résulats des tests des différents modèles ainsi que leurs courbes de performance sont présentés ci-dessous :

Modèles	**AUC**	**Précision**	**Rappel**	**F1-score**
XGBoost	0,934	0,55	0,68	0,61
LogReg	0,839	0,12	0,74	0,20
Autoencodeur	0,68	0,09	0,13	0,11
Tabnet	0,842	0,12	0,72	0,21
Hybride	0,917	0,80	0,56	0,66

TABLE 5.6 – Résultats des Performances des Modèles.

Le score AUC de 0,934 démontre une très bonne capacité de discrimination pour

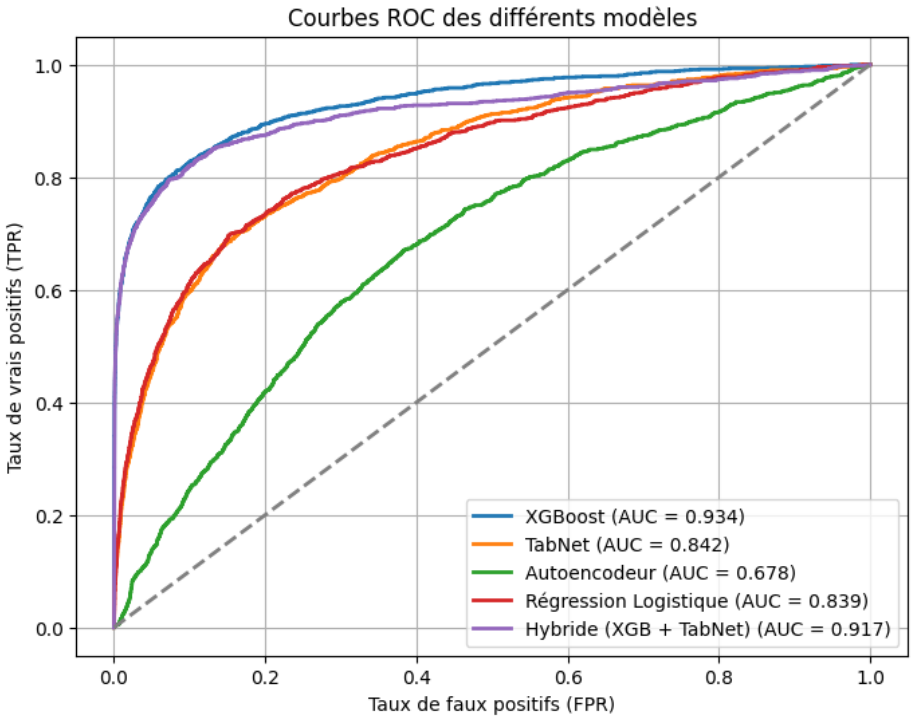

FIGURE 5.1 – Courbe ROC

l'algorithme XGBoost, indiquant qu'il est capable de séparer les classes frauduleuses et non frauduleuses avec une grande fiabilité, quel que soit le seuil de classification. Le rappel de 0,68 indique que le modèle a correctement identifié 68% de toutes les fraudes réelles.

Le modèle de régression logistique affiche un rappel élevé de 74%, indiquant une grande sensibilité à la détection de la fraude. Cependant, ce modèle présente une précision très faible de 12%, suggérant qu'il génère un grand nombre de faux positifs.

L'AUC de 0,678 de l'auto-encodeur est proche de 0,5, le score d'un classifieur aléatoire, ce qui indique que la méthode basée uniquement sur l'erreur de reconstruction a une faible capacité à distinguer les fraudes des transactions légitimes dans notre jeu de données.

Le modèle de Tabnet présente un AUC de 0,842, le plaçant dans la même catégorie de performance de discrimination que le modèle de régression logistique. Semblable à ce dernier, il maintient un bon rappel de 72% au détriment d'une précision faible de 12%.

Le modèle Hybride atteint un score F1 de 0,66 et une Précision très élevée de 0,80. Cette haute précision signifie que 80 % des alertes de fraude émises par ce modèle sont de vraies fraudes, indiquant une grande fiabilité.

5.2 Comparaison des modèles

La synthèse des résultats ci-dessous 5.7 compare les performances de nos quatre modèles, en se concentrant sur les métriques clés et la distribution des erreurs. L'analyse comparative des résultats nous montre des stratégies de classification très différentes entre les modèles, en particulier dans leur gestion de l'équilibre entre la détection des fraudes et la fiabilité des alertes. Nous avons classé les modèles par ordre croissant suivant leur performance.

Modèle	**ROC AUC**	**Précision**	**Rappel**	**F1-score**	**VP**	**FN**	**FP**
XGBoost	**93%**	55%	**68%**	61%	845	389	698
Hybride	92%	**80%**	56%	**66%**	687	547	**170**
LogReg	84%	12%	74%	20%	907	327	6851
TabNet	84%	12%	72%	21%	890	344	6448
Autoencodeur	68%	09%	13%	11%	159	1075	1613

TABLE 5.7 – Comparaison des Performances des Modèles

XGBoost

Avec un AUC de 0,9342, XGBoost se positionne comme le meilleur discriminateur parmi les modèles testés, sa courbe ROC étant la plus proche de 1, ce qui demontre une capacité de discrimination optimale entre les deux classes. De plus, il a réussi à identifier des relations profondes et durables entre les caractéristiques. XGBoost offre également le meilleur rappel de 68%, n'ayant manqué que 389 faux négatifs. Cependant, sa précision de 55% indique qu'il signale 698 transactions légitimes comme frauduleuses. Cette caractéristique est particulièrement importante dans le contexte de la détection de changements graduels car la capacité de XGBoost à capturer des relations complexes et non-linéaires lui permet de détecter des évolutions subtiles de comportement sur plusieurs transactions. Les arbres de décision, de par leur nature,

sont relativement robustes aux variations non-linéaires et peuvent identifier des patterns de dérive progressive difficiles à repérer avec des modèles linéaires.

Hybride

Le modèle Hybride se distingue par sa stabilité remarquable face aux changements de comportement. Il obtient de loin la meilleure précision de 80% et le meilleur F1-score de 66%. Ce résultat de 80% signifie que quatre alertes sur cinq correspondent effectivement à de vraies fraudes, réduisant considérablement les fausses alertes. Le modèle gère efficacement les erreurs en limitant les faux positifs à seulement 170, comparativement aux autres modèles. En contrepartie, il affiche un rappel plus faible de 56%, manquant 547 fraudes. Toutefois, c'est le modèle qui démontre la meilleure validité externe immédiate sur les transactions futures.

Cette stabilité du modèle Hybride est particulièrement précieuse dans le contexte de détection de fraudes évolutives. Sa haute précision de 80% indique qu'il est moins sensible aux variations normales du comportement des utilisateurs légitimes, réduisant ainsi les faux positifs même lorsque ces utilisateurs modifient progressivement leurs habitudes transactionnelles. Le modèle Hybride établit un profil comportemental plus robuste qui distingue efficacement les dérives légitimes des escalades frauduleuses.

Tabnet-Régression logistique

Ces deux modèles présentent des performances et des stratégies très similaires. Avec un rappel de 74% pour la régression logistique et de 72% pour TabNet, ils démontrent une sensibilité élevée à la détection de fraude. Cependant, leur précision est extrêmement faible voir 12%, générant un volume considérable de fausses alertes.

Cette faible précision constitue un indicateur révélateur de la présence de changements comportementaux graduels dans les données. Ce phénomène peut être dû à l'évolution des patterns de transactions normales, ou la formulation des hypothèse trop larges faites par ces modèles, classifiant comme frauduleuses toute transaction s'écartant des patterns historiques, sans capacité à distinguer les évolutions légitimes des escalades frauduleuses.

Auto-encodeur

Considéré comme le moins efficace, l'auto-encodeur affiche les performances les plus basses sur l'ensemble des métriques clés, avec un AUC de 68% et un F1-score de 11%. Ce modèle génère le plus grand nombre de faux négatifs voir 1075, indiquant qu'il ne détecte qu'une fraction minime des fraudes réelles.

Face à des fraudes graduelles où les fraudeurs miment progressivement le comportement légitime, les transactions frauduleuses évoluant lentement vers des montants anormaux ne génèrent pas d'erreur de reconstruction suffisamment élevée pour être détectées. Ainsi le modèle apprend progressivement ces nouveaux patterns comme faisant partie de la normalité d'où sa difficulté à détecter ces transactions frauduleuses.

5.3 Menace pour la Validité

Cette section analyse les différentes menaces potentielles à la validité de notre étude sur la détection de fraude par l'analyse des changements comportementaux graduels. Nous examinons les menaces selon trois dimensions : la validité interne, la validité externe et la validité de construction.

5.3.1 Validité interne

Dans ce travail, plusieurs éléments peuvent influencer la validité interne. Bien que des améliorations ont été faites, la qualité et la complexité des données limitent notre compréhension des mécanismes sous-jacents de détection.

Tout d'abord, le sous-échantillonnage aléatoire de 30 % de la base initiale, bien qu'il permette de réduire le temps d'exécution, peut introduire un biais de sélection si certaines fraudes spécifiques sont sous-représentées.

De plus, le choix de la taille des fenêtres temporelles et du nombre minimum de transactions est arbitraire et peut influencer significativement les résultats.

En fin, le découpage temporel de 80 % pour l'entraînement et de 20 % pour le test simule un scénario réaliste de détection graduelle, mais peut exposer le modèle à des variations saisonnières non apprises.

5.3.2 Validité externe

Nos modèles ont été évalués sur le jeu de données de IEEE-CIS provenant d'un environnement commercial unique avec ses propres caractéristiques de transaction et ses méthodes de fraude spécifiques. Par conséquent, leurs performances peuvent varier si les données proviennent d'un autre système de paiement, d'une autre période ou d'un environnement géographique différent.
De plus, la nature évolutive de la fraude c'est-à dire les motifs et méthodes de fraude évoluant constamment pourrait faire perdre la performance d'un modèle qui autre fois avait été efficace pour la détection.

5.3.3 Validité de construction

Dans cette étude, la variable cible étant *isFraud* repose sur une étiquette binaire supposée correcte. Cependant, des erreurs d'étiquetage comme des fraudes non détectées ou mal classées pourraient fausser l'apprentissage.
De plus, les variables temporelles et indicateurs glissants reposent sur des transformations arbitraires, comme la taille de la fenêtre de trois transactions pour le calcul des moyennes mobiles.
Aussi, l'agrégation des transactions au niveau utilisateur suppose que le comportement utilisateur est une entité cohérente, alors qu'un compte peut être partagé ou compromis.
Enfin, les métriques d'évaluation choisies mesurent la performance globale, mais ne traduisent pas toujours parfaitement l'impact réel d'une erreur de détection dans un contexte financier.

Chapitre 6

Conclusion et perspectives

La fraude dans les transactions financières représente un défi majeur pour l'industrie des paiements en ligne, avec des pertes estimées à plusieurs milliards de dollars annuellement. L'objectif principal était de concevoir et d'évaluer des modèles de *Machine Learning* capables de détecter efficacement les transactions frauduleuses malgré la dérive temporelle des comportements des utilisateurs et la forte asymétrie des classes.

Notre approche expérimentale a été développée, intégrant à la fois des méthodes supervisées et non supervisées , ainsi qu'un modèle hybride. La stratégie de détection graduelle a consisté à diviser les données de manière temporelle plutôt qu'aléatoire, simulant ainsi un scénario réaliste où les modèles sont entraînés sur des données passées et testés sur des transactions plus récentes. Cette approche a permis d'évaluer la robustesse temporelle et la capacité d'adaptation des modèles face à l'évolution progressive des comportements.

Sur le plan méthodologique, l'utilisation du *Cost-Sensitive Learning* a permis d'améliorer significativement la précision et la stabilité des modèles face au déséquilibre des classes. Les résultats obtenus démontrent la supériorité du modèle XGBoost, qui a atteint une AUC de 93,42%, confirmant son efficacité dans des contextes de classes

déséquilibrées et dynamiques. Le modèle hybride XGBoost–TabNet a également montré de très bonnes performances, combinant la puissance d'apprentissage en *gradient boosting* et la capacité d'interprétation sélective de TabNet. En revanche, la régression logistique et TabNet ont présenté des performances plus modestes, tandis que l'auto-encodeur, bien qu'intéressant pour la détection non supervisée d'anomalies, s'est révélé moins performant. Ces résultats confirment que les méthodes supervisées pondérées ou hybrides sont les plus adaptées à la détection de fraude évolutive.

Les résultats de ce travail confirment l'efficacité des approches de machine learning pour la détection de fraude basée sur des comportements graduels, tout en mettant en évidence des perspectives d'amélioration. Celles-ci incluent l'intégration explicite de la dimension temporelle à l'aide de modèles séquentiels, tels que les *Transformers* et les réseaux de neurones récurrents, afin de mieux capturer l'évolution progressive des comportements. Par ailleurs, le recours à des mécanismes d'apprentissage continu et adaptatif apparaît essentiel pour maintenir la performance des modèles face à l'évolution constante des stratégies de fraude.

En définitive, l'analyse des changements comportementaux graduels ne remplace pas les approches de détection d'anomalies ponctuelles, mais les enrichit significativement, créant un système de défense multi-couches plus robuste contre les techniques de fraude sophistiquées et évolutives. Dans un contexte où les fraudeurs innovent constamment, cette capacité à détecter les patterns d'escalade représente une avancée vers des systèmes de détection plus intelligents, proactifs et adaptatifs.

Bibliographie

[1] L. PricewaterhouseCoopers, « Cybercrime : Protecting against the growing threat global economic crime survey », 2011.

[2] Y. Lucas, *Credit card fraud detection using machine learning with integration of contextual knowledge*. Thèse doctorat, Université de Lyon ; Universität Passau (Allemagne), 2019.

[3] A. Facci, B. Pinaud, J. Cavarroc et A. Pidash, « Apprentissage machine appliqué à la détection de fraudes bancaires », *in Extraction et Gestion des Connaissances, EGC'2025*, p. pp–335, 2025.

[4] A. Ogunleye et Q.-G. Wang, « Xgboost model for chronic kidney disease diagnosis », *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 17, no. 6, p. 2131–2140, 2019.

[5] J. Zhang, X. Ma, J. Zhang, D. Sun, X. Zhou, C. Mi et H. Wen, « Insights into geospatial heterogeneity of landslide susceptibility based on the shap-xgboost model », *Journal of environmental management*, vol. 332, p. 117357, 2023.

[6] S. S. Azmi et S. Baliga, « An overview of boosting decision tree algorithms utilizing adaboost and xgboost boosting strategies », *Int. Res. J. Eng. Technol*, vol. 7, no. 5, p. 6867–6870, 2020.

[7] E. Fekade, *Detection of Competency Certification Fraud Using Deep Learning*. Thèse doctorat, St. Mary's University, 2025.

[8] S. Ö. Arik et T. Pfister, « Tabnet : Attentive interpretable tabular learning », *in Proceedings of the AAAI conference on artificial intelligence*, vol. 35, p. 6679–

6687, 2021.

[9] C. A. C. Da, *Vers une détection efficace et robuste des fraudes bancaires grâce à l'apprentissage automatique.* Thèse doctorat, Université du Québec à Trois-Rivières, 2024.

[10] Y. M. Thiam, « Système de détection d'anomalies dans les transactions par cartes de débit bancaires à ecobank senegal par apprentissage non supervisé », 2025.

[11] G. Souad et E. G. Leila, « L'impact de la nouvelle technologie sur les mécanismes de gouvernance de l'entreprise face à la qualité de l'information financière : Étude exploratoire dans le contexte marocain », *Int. J. Econ. Stud. Manag*, vol. 3, no. 4, 2023.

[12] J. Le Maux, N. Smaili et W. B. Amar, « De la fraude en gestion à la gestion de la fraude », *Revue française de gestion*, vol. 231, no. 2, p. 73–85, 2013.

[13] D. R. Cressey, « The criminal violation of financial trust », *in American Sociological Review*, p. 738–743, American Sociological Association, 1950.

[14] M.-J. Kranacher et R. Riley, *Forensic accounting and fraud examination.* John Wiley & Sons, 2019.

[15] J. Dorminey, A. S. Fleming, M.-J. Kranacher et R. A. Riley Jr, « The evolution of fraud theory », *Issues in accounting education*, vol. 27, no. 2, p. 555–579, 2012.

[16] R. J. Bolton et D. J. Hand, « Statistical fraud detection : A review », *Statistical science*, vol. 17, no. 3, p. 235–255, 2002.

[17] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi et G. Bontempi, « Credit card fraud detection and concept-drift adaptation with delayed supervised information », *in 2015 international joint conference on Neural networks (IJCNN)*, p. 1–8, IEEE, 2015.

[18] O. S. Adebayo, T. A. Favour-Bethy, O. Otasowie et O. A. Okunola, « Comparative review of credit card fraud detection using machine learning and concept drift techniques », *International Journal of Computer Science and Mobile Computing*, vol. 12, no. 7, p. 24–48, 2023.

[19] B. Baesens, V. Van Vlasselaer et W. Verbeke, *Fraud analytics using descriptive, predictive, and social network techniques : a guide to data science for fraud detection*. John Wiley & Sons, 2015.

[20] A. Gandhar, K. Gupta, A. K. Pandey et D. Raj, « Fraud detection using machine learning and deep learning », *SN Computer Science*, vol. 5, no. 5, p. 453, 2024.

[21] A. Oza, « Fraud detection using machine learning », *Transfer*, vol. 528812, no. 4097, p. 532909, 2018.

[22] Y. Zhang et D. Wang, « A cost-sensitive ensemble method for class-imbalanced datasets », *in Abstract and applied analysis*, vol. 2013, p. 196256, Wiley Online Library, 2013.

[23] A. ALFOCEA, « Comment le machine learning permet de détecter la fraude bancaire ? », *Management & Datascience*, vol. 5, no. 4, 2021.

[24] N. Sahllal, « Machine learning approaches for imbalanced data : Click-through rate prediction and click fraud detection », 2023.

[25] T. Beigh, V. P. Venkatesan *et al.*, « Counterfeit currency detection using machine learning », 2023.

[26] F. Jabiri, « Applications de méthodes de classification non supervisées à la détection d'anomalies », 2020.

[27] S. Thiprungsri et M. A. Vasarhelyi, « Cluster analysis for anomaly detection in accounting data : An audit approach. », *International Journal of Digital Accounting Research*, vol. 11, 2011.

[28] H. Chergui, L. Abrouk, N. Cullot et N. Cabioch, « Détection de fraude financière dans un système de transactions interbancaires. », *in INFORSID*, vol. 22, p. 141–156, 2022.

[29] P. Gogas et T. Papadimitriou, « Machine learning in economics and finance », *Computational Economics*, vol. 57, no. 1, p. 1–4, 2021.

[30] R. A. Bauder et T. M. Khoshgoftaar, « Medicare fraud detection using machine learning methods », *in 2017 16th IEEE international conference on machine*

learning and applications (ICMLA), p. 858–865, IEEE, 2017.

[31] T. Srinivasarao, N. Leelavathy, M. H. Narayana, T. S. Varsha, S. S. S. Sai et M. B. M. Deepak, « Enhancing medicare fraud detection through ml : Addressing class imbalance with smote-enn », *in Algorithms in Advanced Artificial Intelligence*, p. 316–324, CRC Press, 2025.

[32] J. B. T. Fuentes et E. J. M. Arias, « Modelos de machine learning para la detección de fraudes financieros : Una revisión de la literatura », *UNESUM-Ciencias. Revista Científica Multidisciplinaria*, vol. 9, no. 2, p. 220–234, 2025.

[33] J. Quinlan, « Bagging, boosting, and c4. 5. aaai/iaai », 1996.

[34] J. H. Friedman, « Greedy function approximation : a gradient boosting machine », *Annals of statistics*, p. 1189–1232, 2001.

[35] C. Elkan, « The foundations of cost-sensitive learning », *in International joint conference on artificial intelligence*, vol. 17, p. 973–978, Lawrence Erlbaum Associates Ltd, 2001.

[36] Y. LeCun, « Connexionist learning models », *PhD thesis*, 1987.

[37] S. Caron, « Détection d'anomalies basée sur les représentations latentes d'un autoencodeur variationnel », 2021.

[38] J. Desjardins, « L'analyse de régression logistique », *Tutorial in quantitative methods for psychology*, vol. 1, no. 1, p. 35–41, 2005.

[39] P.-M. Preux, P. Odermatt, A. Perna, B. Marin et A. Vergnenègre, « Qu'est-ce qu'une régression logistique », *Rev Mal Respir*, vol. 22, no. 1, p. 159–62, 2005.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, « Scikit-learn : Machine learning in python », *the Journal of machine Learning research*, vol. 12, p. 2825–2830, 2011.

[41] A. ABRAICH, H. D. NGUYEN et M. Tounsi, « Détection de fraude ieee-cis », *IEE*, 2020.

[42] C. C. Meng, K. M. Lim, C. P. Lee et J. Y. Lim, « Credit card fraud detection using tabnet », *in 2023 11th International Conference on Information and Communication Technology (ICoICT)*, p. 394–399, IEEE, 2023.

[43] G. Metzler, *Learning from imbalanced data : an application to bank fraud detection.* Thèse doctorat, Université de Lyon, 2019.

[44] B. Krawczyk, « Learning from imbalanced data : open challenges and future directions », *Progress in artificial intelligence*, vol. 5, no. 4, p. 221–232, 2016.

[45] J. Huang et C. X. Ling, « Using auc and accuracy in evaluating learning algorithms », *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 3, p. 299–310, 2005.

[46] C. Cortes et M. Mohri, « Auc optimization vs. error rate minimization », *Advances in neural information processing systems*, vol. 16, 2003.

[47] P. N. C. P. de Sá, « Fraud detection with algorithms for tabular data », Mém. D.E.A., Universidade de Coimbra (Portugal), 2023.

[48] Q. Cai et J. He, « Credit payment fraud detection model based on tabnet and xgboot », *in 2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, p. 823–826, IEEE, 2022.

Annexe A

Acronymes

ML : *Machine learning*

CSL : *Cost-sensitivelearning*

FP : Faux Positifs

FN : Faux Négatifs

VP : Vrais Positifs

VN : Vrais Négatifs

IA : Intelligence Artificielle

F1-score : Moyenne harmonique de la précision et du rappel

AUC : *Area Under the Curve*

ROC : *Receiver Operating Characteristic*

XGBoost : *Extreme Gradient Boosting*

SVM : *support-vector machine*

RBF : *Radial Basis Function*

AUPRC : *Area Under the Precision-Recall Curve*

QBC : *Query-by-Committee*

CTR : *Click-Through Rate*

SMOTE : Synthetic Minority Over-sampling Technique

PCA : *Principal Component Analysis*

MSE : *Mean Squared Error*

API : *Application Programming Interface*

Annexe B

Listing 1 – Script Python pour les Modèles Cost-sensitive learning, XGBoost, Tabnet, Régression logistique, Auto-encodeur, Hybridation.

```
#Librairies
# Install the pytorch-tabnet library
!pip install pytorch-tabnet
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import StandardScaler,
LabelEncoder
from sklearn.metrics import (
    roc_auc_score, classification_report,
    confusion_matrix,
    precision_score, recall_score, f1_score
)
from sklearn.linear_model import LogisticRegression
from sklearn.utils.class_weight import compute_class_weight
from xgboost import XGBClassifier
```

```
from pytorch_tabnet.tab_model import TabNetClassifier
import torch
import tensorflow as tf
from tensorflow.keras import layers

#Importation des bases des données
train_transaction = pd.read_csv("/content/train_transaction.csv")
train_identity = pd.read_csv("/content/train_identity.csv")

# Fusion des transactions avec identité
df = train_transaction.merge(train_identity, on="TransactionID",
how="left")
# Sous-échantillonnage de 30% pour une éxecution plus rapide
df = df.sample(frac=0.3, random_state=42)

# Rédéfiniton de X et y sur df.sample
y = df["isFraud"]
X = df.drop(columns=["isFraud", "TransactionID"])

# Statistiques pour les variables numériques
print("\n=== Statistiques descriptives (numériques) ===")
print(df.describe().T)
# Statistiques pour les variables catégorielles
print("\n=== Statistiques descriptives (catégorielles) ===")
print(df.describe(include=['object', 'category']).T)

# Matrice de corrélation (extrait 15 premières variables)
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True).iloc[:15, :15],
cmap="coolwarm", annot=False)
```

```
plt.title("Matrice de corrélation")
plt.show()

# Distribution fraude
sns.set(style="whitegrid", font_scale=1.2)
fig, ax = plt.subplots(figsize=(8, 5))
ax = sns.countplot(
    y="isFraud",
    data=train_transaction,
    palette="coolwarm",
    order=[0, 1]
)
plt.title("Distribution des transactions frauduleuses et non
frauduleuses", fontsize=14, fontweight='bold')
plt.xlabel("Nombre de transactions")
plt.ylabel("Type de transaction")
ax.set_yticklabels(["Non fraude (0)", "Fraude (1)"])

total = len(train_transaction["isFraud"])

for p in ax.patches:
    width = p.get_width()
    percentage = 100 * width / total
    x = width + total * 0.002
    y = p.get_y() + p.get_height() / 2
    ax.annotate(f"{int(width)} ({percentage:.2f}%)", (x, y),
                ha='left', va='center',
                fontsize=11, color='black')

plt.tight_layout()
```

```
plt.show()

# Features temporelles
if "TransactionDT" in X.columns:
    X["TransactionDT"] = pd.to_numeric(X["TransactionDT"],
    errors="coerce")
    X["hour"] = (X["TransactionDT"] // 3600) % 24
    X["day"] = (X["TransactionDT"] // (3600*24)) % 7
    X["month"] = (X["TransactionDT"] // (3600*24*30)) % 12

if "card1" in X.columns and "TransactionAmt" in X.columns:
    X["TransactionAmt_roll_mean"] = (
        X.groupby("card1")["TransactionAmt"]
        .transform(lambda s: s.rolling(3, min_periods=1).mean())
    )
    X["TransactionAmt_roll_std"] = (
        X.groupby("card1")["TransactionAmt"]
        .transform(lambda s: s.rolling(3, min_periods=1).std())
    )

# Visualisation des features
features_to_plot = ["hour", "day", "month",
"TransactionAmt_roll_mean",
"TransactionAmt_roll_std"]

sns.set(style="whitegrid", font_scale=1.1)
plt.figure(figsize=(15, 10))

for i, feature in enumerate(features_to_plot, 1):
    plt.subplot(2, 3, i)
```

```
    if X[feature].nunique() < 30:

        sns.countplot(x=feature, data=X, palette="viridis")
    else:

        sns.histplot(X[feature].dropna(),
        bins=30, kde=True, color="steelblue")

    plt.title(f"Distribution de {feature}",
    fontsize=13, fontweight="bold")
    plt.xlabel(feature)
    plt.ylabel("Frequence")
    plt.tight_layout()

plt.suptitle("Distributions des principales features temporelles
et statistiques", fontsize=15, fontweight="bold", y=1.02)
plt.show()

# 3. Prétraitement
X = X.fillna(-999)
for col in X.select_dtypes(include=["object"]).columns:
    X[col] = LabelEncoder().fit_transform(X[col].astype(str))

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Découpage temporel (graduel)
split_index = int(0.8 * len(X_scaled))
X_train, X_test = X_scaled[:split_index], X_scaled[split_index:]
```

```
y = df["isFraud"]
y_train, y_test = y.values[:split_index], y.values[split_index:]

# XGBoost (Cost-Sensitive)
scale_pos_weight = (len(y_train) - sum(y_train)) / sum(y_train)

xgb = XGBClassifier(
    n_estimators=500, learning_rate=0.05, max_depth=8,
    subsample=0.8, colsample_bytree=0.8,
    scale_pos_weight=scale_pos_weight,
    tree_method="hist", eval_metric="auc", random_state=42
)
# Convertir y_train en entiers
print(f"Shape of X_train: {X_train.shape}")
print(f"Shape of y_train: {y_train.shape}")
xgb.fit(X_train, y_train.astype(int))
y_pred_xgb = xgb.predict(X_test)
y_prob_xgb = xgb.predict_proba(X_test)[:, 1]

# Régression Logistique
logreg = LogisticRegression(class_weight="balanced",
max_iter=500,random_state=42)
logreg.fit(X_train, y_train)
y_pred_lr = logreg.predict(X_test)
y_prob_lr = logreg.predict_proba(X_test)[:, 1]

#Auto-encodeur
X_train_auto = X_train[y_train == 0]
input_dim = X_train_auto.shape[1]
```

```
autoencoder = tf.keras.Sequential([
    layers.Input(shape=(input_dim,)),
    layers.Dense(32, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(8, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(32, activation="relu"),
    layers.Dense(input_dim, activation="linear")
])
autoencoder.compile(optimizer="adam", loss="mse")
autoencoder.fit(
    X_train_auto, X_train_auto,
    epochs=20, batch_size=256,
    validation_split=0.2, verbose=0
)

reconstructions = autoencoder.predict(X_test, verbose=0)
mse = np.mean(np.power(X_test - reconstructions, 2), axis=1)
threshold = np.percentile(mse, 95)
y_pred_ae = (mse > threshold).astype(int)
y_prob_ae = mse / mse.max()

# Calcul des poids de classes à partir des données
d'entrainement
classes = np.unique(y_train)
class_weights = compute_class_weight(class_weight="balanced",
classes=classes, y=y_train)
class_weights = torch.tensor(class_weights, dtype=torch.float)

# Définir la fonction de perte pondérée
```

```
loss_fn = torch.nn.CrossEntropyLoss(weight=class_weights)

#TabNet
tabnet = TabNetClassifier(
    optimizer_fn=torch.optim.Adam,
    optimizer_params=dict(lr=2e-2),
    scheduler_params={"step_size":10, "gamma":0.9},
    scheduler_fn=torch.optim.lr_scheduler.StepLR,
    mask_type="entmax",
    seed=42, verbose=0
)

tabnet.fit(
    X_train, y_train,
    eval_set=[(X_test, y_test)],
    eval_name=["test"],
    eval_metric=["auc"],
    max_epochs=50,
    patience=5,
    batch_size=1024,
    virtual_batch_size=128,
    loss_fn=loss_fn
)

# Prédictions
y_pred_tabnet = tabnet.predict(X_test)
y_prob_tabnet = tabnet.predict_proba(X_test)[:, 1]

#Stacking (métamodele LogReg)
stack_train = np.vstack([
```

```
    xgb.predict_proba(X_train)[:, 1],
    tabnet.predict_proba(X_train)[:, 1]
]).T

stack_test = np.vstack([
    y_prob_xgb,
    y_prob_tabnet
]).T

meta_model = LogisticRegression()
meta_model.fit(stack_train, y_train)

y_pred_hybrid = meta_model.predict(stack_test)
y_prob_hybrid = meta_model.predict_proba(stack_test)[:, 1]

#Hybride XGBoost + TabNet (Stacking)
stack_train = np.vstack([
    xgb.predict_proba(X_train)[:, 1],
    tabnet.predict_proba(X_train)[:, 1]
]).T
stack_test = np.vstack([y_prob_xgb, y_prob_tabnet]).T

meta_model = LogisticRegression()
meta_model.fit(stack_train, y_train)
y_pred_hybrid = meta_model.predict(stack_test)
y_prob_hybrid = meta_model.predict_proba(stack_test)[:, 1]

# 9. Évaluation
models = {
    "XGBoost": (y_pred_xgb, y_prob_xgb),
```

```
    "LogReg": (y_pred_lr, y_prob_lr),
    "Autoencoder": (y_pred_ae, y_prob_ae),
    "TabNet": (y_pred_tabnet, y_prob_tabnet),
    "Hybride (XGB+TabNet)": (y_pred_hybrid, y_prob_hybrid)
}

for name, (y_pred, y_prob) in models.items():
    auc = roc_auc_score(y_test, y_prob)
    precision = precision_score(y_test, y_pred,
    zero_division=0)
    recall = recall_score(y_test, y_pred,
    zero_division=0)
    f1 = f1_score(y_test, y_pred, zero_division=0)
    cm = confusion_matrix(y_test, y_pred)

    print(f"\n=== {name} ===")
    print(f"ROC AUC: {auc:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Rappel: {recall:.4f}")
    print(f"F1-score: {f1:.4f}")
    print("Matrice de confusion :")
    print(pd.DataFrame(cm,
        index=["Reel: Non Fraude (0)", "Reel: Fraude (1)"],
        columns=["Predit: Non Fraude (0)",
        "Predit: Fraude (1)"]))

# Courbes ROC
# ----------------------------

# Dictionnaire des modèles et de leurs probabilités
```

```
models = {
    "XGBoost": y_prob_xgb,
    "TabNet": y_prob_tabnet,
    "Autoencodeur": y_prob_ae,
    "Regression Logistique": y_prob_lr,
    "Hybride (XGB + TabNet)": y_prob_hybrid
}

plt.figure(figsize=(8,6))

# Génération des courbes ROC
for name, y_prob in models.items():
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    roc_auc_value = auc(fpr, tpr)
    plt.plot(fpr, tpr, lw=2, label=f"{name}
    (AUC = {roc_auc_value:.3f})")

# Ajout de la diagonale aléatoire
plt.plot([0,1], [0,1], color="gray", linestyle="--", lw=2)

# Mise en forme du graphique
plt.xlabel("Taux de faux positifs (FPR)")
plt.ylabel("Taux de vrais positifs (TPR)")
plt.title("Courbes ROC des differents modeles")
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```