Apprentissage non supervisé appliqué au problème de détection de fraude

Jean-Charles Verdier [1] **Wanlin Li** [1] **Cédrick Jonathan Randriamilamina** [1]

1. Introduction

Les paiements par cartes de crédit constituent une part importante des transactions financières dans le monde. Elles offrent une facilité et une efficacité inégalée par d'autres types de paiement, particulièrement pour les transactions effectuées sur internet. Or avec une telle ubiquité vient des opportunités de fraude pour les criminels. Aux États-Unis seulement, on estime les coûts engendrés par la fraude fiscale à 32 milliards pour l'année 2013 [1]. À ce jour, un Américain sur 10 a été victime de fraude (avec un montant médian à 399$) [3]. Le plus récent rapport de la Banque Centrale Européenne comptabilise, pour l'année 2018, le montant total des transactions frauduleuses à plus de 1.8 milliard [15]. Depuis, de nombreuses solutions ont été implémentées afin prévenir cette fraude - comme l'ajout du code de sécurité CVV et l'arrivée des cartes magnétiques. Toutefois, les fraudeurs s'adaptent et développent des nouvelles techniques sophistiquées qui échappent aux systèmes de prévention actuels. Il s'avère donc important de développer des méthodes dynamiques permettant de prévenir ou du moins détecter la présence de transactions frauduleuses.

On distingue deux catégories de fraude : carte présente et carte absente. La première catégorie réfère aux scénarios où la carte est physiquement requise pour compléter un achat comme dans un magasin, par exemple. Elle implique généralement la duplication ou le vol d'une carte de crédit. Ce type de fraude est largement contré par l'arrivée des cartes à puce. En contraste, la catégorie carte absente réfère aux transactions par téléphone, internet ou par poste où la carte physique n'est pas requise. Cette taxonomie est nécessaire car les techniques pour compromettre les cartes de crédit varient en fonction de la présence ou l'absence physique de la carte. Dans notre travail, nous étudions plutôt la deuxième catégorie car elle représente aujourd'hui la majorité des fraudes [10] et parce que nos données proviennent de la compagnie Vesta qui se spécialise dans les transactions en ligne. Détecter manuellement des fraudes s'avère extrêmement onéreux en temps considérant le nombre élevé de transactions quotidiennes. En moyenne, une fraude est découverte après 72 heures d'analyse [1]. Pour cette raison, les techniques d'apprentissage automatique sont de plus en plus considérées car elles permettent une détection beaucoup plus rapide des fraudes. Les méthodes supervisées – qui utilisent les étiquettes lors de leur entraînement – sont délaissées en faveur des méthodes non supervisées car les premières ne permettent pas de détecter des nouveaux types de fraude et pour des considérations pratiques que nous verrons plus loin. Ainsi, dans ce travail, nous étudions quelques techniques de détection de fraude basées sur l'apprentissage non supervisé et comparons leur performance sur le jeu de données IEEE-CIS Fraud Detection (IEEE-FD) offert sur la plateforme Kaggle . Dans la prochaine section, nous définissons plus précisément le problème avant de présenter une revue de la littérature sur le sujet. Ensuite, nous analysons en rapidement les données de IEEE-FD avant de présenter les algorithmes utilisés.

2. Détection de fraude

Le problème de détection de fraude se généralise bien comme un problème de détection d'anomalie dans lequel les transactions frauduleuses représentent les anomalies par contraste aux transactions dites normales ou légitimes. La détection d'anomalie vise l'identification des observations qui se distinguent significativement d'un comportement attendu. Cette définition intuitive cache deux concepts sous-jacents: le concept de normalité et le concept de distance par rapport à celle-ci. Ces concepts prennent des significations différentes en fonction du cadre dans lequel on se trouve. Dans une approche probabiliste, par exemple, on définit l'ensemble des anomalies comme $A = \{x \in \mathcal{X} : \mathbb{P}^+(x) \leq \tau, \tau > 0\}$ où $\mathbb{P}^+$ représente la fonction de densité de la classe normale et $\mathcal{X} \in \mathbb{R}^d$ représente l'espace des données ([11]). Un cadre géométrique utiliserait plutôt une notion de distance numérique pour identifier les anomalies: $A = \{x \in \mathcal{X} : d(x) \leq \tau, \tau > 0\}$ où $d(x)$ représente une fonction de distance quelconque (euclidienne, Manhattan, Mahalanobis, etc.). Dans tous les cas, la tâche de détection de fraude se range dans la catégorie des problèmes de classification binaire où on tente de séparer les transactions en deux catégories : légitimes et frauduleuses. De plus, la détection de fraude présente des défis particuliers :

[*]Equal contribution [1]Département d'informatique, Université de Sherbrooke, Sherbrooke, Canada. Correspondence to: Jean-Charles Verdier <verj2009@usherbrooke.ca>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

- **Dérive conceptuelle**. Les transactions normales et frauduleuses changent avec le temps. D'un côté les comportements des consommateurs varient en fonction des saisons et journées de la semaine. De l'autre, les criminels innovent et développent de nouvelles techniques pour frauder leurs victimes. Les algorithmes d'apprentissage assument souvent que les données sont i.i.d (indépendantes et identiquement distribuées). Cette hypothèse permet de généraliser le critère de décision de l'ensemble d'entraînement à l'ensemble de test. Or, elle ne tient pas dans la détection de la fraude car la distribution des données change avec le temps.

- **Débalancement de classe**. Les anomalies sont beaucoup moins fréquentes que les transactions normales. Le pourcentage de fraude dans les données est de l'ordre de 1% seulement. On doit donc développer des méthodologies particulières pour entraîner un classifieur sans quoi on court le risque de sur-apprentissage sur la classe majoritaire.

- **Disponibilité des données**. Pour des raisons de confidentialité évidentes, les données de transactions financières ne peuvent pas être partagées publiquement. Lorsqu'elles le sont, les variables doivent être lourdement anonymisées de telle sorte qu'on ne sait plus ce qu'elles représentent.

- **Métriques de performance**. On doit choisir des métriques de performance qui sont robustes au débalancement de classe. Les mesures classiques comme Accuracy et AUROC doivent être abandonnées en faveur de mesures plus sensibles aux capacités prédictives sur la classe minoritaire (les fraudes). En effet, on peut mal classifier toutes les instances de la classe minoritaire et obtenir une excellente Accuracy (de l'ordre de .90 et plus). Quant à l'AUROC, il donne autant de poids aux prédictions sur la classe minoritaire que la classe majoritaire. Un excellent AUROC peut donc cacher de moins bonnes performances sur la classe minoritaire [6].

- **Apprentissage non supervisé**. Étant donné la dérive conceptuelle et des contraintes pratiques, un apprentissage non supervisé est favorisé par rapport à un apprentissage supervisé. En effet, puisque les anomalies ne sont pas fixes et changent avec le temps, un modèle supervisé aura de la difficulté à détecter de nouvelles fraudes. Étant donné le faible ratio de fraudes dans les transactions quotidiennes, il est très coûteux pour une entreprise de mettre en place une équipe dont la tâche est d'étiquetée chaque transaction.

3. Revue de la littérature

Dans cette section nous révisons quelques algorithmes de détection de fraude basées sur l'apprentissage automatique ainsi que quelques techniques d'ingénierie des données utilisées dans le problème. Les transactions peuvent être conçues comme une séquence discrète d'événements pouvant être modélisée par un automate de Markov à états cachés (MMC). Khan et al. [8] ont proposé ce modèle afin de produire une séquence de transactions. Un algorithme de clustering est appliqué à ces séquences afin de diviser les transactions en trois groupes définis par rapport à leur montant (petit, moyen et grand). Les nouvelles transactions sont comparées aux dix plus récentes et sont autorisées si l'intersection entre les deux ensembles est non nul. Les auteurs ne donnent toutefois aucune mesure de performance pour évaluer l'efficacité de leur approche [1]. Des séquences d'événements peuvent aussi être modélisés par des réseaux de neurones spécialisés comme les LSTM (Long Short Term Memory). Bontemps et al. [2] utilisent cette architecture pour produire des profils transactionnels pour chaque utilisateur. Chaque nouvelle séquence est comparée au profil attendu à l'aide d'une métrique de distance qui sert ultimement de mesure d'anomalie.

Des arbres de décision et des séparateurs à vaste marge (SVM) ont aussi été étudiés pour la détection de fraude. Sahin et Duman [13] ont comparé les performances des deux approches sur des données avec des taux de contamination différents. Les données sont divisées en trois groupes avec un ratio entre les anomalies et les transactions normales différent (1:1, 1:4 et 1:9). Les auteurs ont expérimenté avec sept SVM différents et quatre fonctions noyaux et ont constaté la supériorité des arbres de décision avec une exactitude variant entre 83.02 et 94.76. Toutefois, la mesure d'exactitude n'est pas appropriée dans les situations de débalancement de classe car les performances sur la classe minoritaire ont peu d'impact sur le résultat obtenu.

Des réseaux de neurones profonds, quoique dans une proportion beaucoup plus modeste, ont aussi été proposé. Van Vlasselaer et al., [16] ont appliqué une méthode de segmentation du marché (Récence, Fréquence, Montant) pour produire des nouveaux attributs discriminants. Trois classifieurs différents sont évalués sur 78 variables: régression logistique, forêt d'arbres décisionnels et un réseau de neurones [1]. Les auteurs utilisent encore la mesure biaisée d'exactitude afin d'évaluer la meilleure méthode.

La détection de fraude offre des possibilités intéressantes en termes d'ingénierie des données afin de générer de nouvelles variables discriminantes. L'entropie de Shannon peut être utilisée afin de quantifier l'information apportée par une nouvelle transaction pour un consommateur. Une grande entropie peut ainsi être indicatrice d'une fraude [5]. Certaines fraudes sont commises dans des courts laps de temps et

peuvent être facilement détectés en associant chaque transaction à une variable delta qui représente la distance temporelle avec la plus récente transaction. Une petite valeur contribuerait ainsi à l'identification d'une fraude [7]. Une autre approche intéressante est l'utilisation d'un graphe biparti où les nœuds correspondent aux consommateurs et marchands et les arêtes représentent les transactions entre les parties [16]. Celles-ci sont pondérées par le volume de transactions et décroisent de manière exponentielle en fonction du temps. Un algorithme PageRank est utilisé pour mesurer l'exposition des nœuds à de la fraude et ultimement en retirer des attributs pertinents [9].

4. Descriptions des données

Les données proviennent de la base de données IEEE-CIC Fraud Detection et sont offertes publiquement sur le site Kaggle. Elles sont divisées en deux fichiers «identity» et «transaction» qui contiennent respectivement des informations sur les propriétaires de carte de crédit et les transactions. Les deux fichiers sont joints à l'aide de la clé «TransactionID». Il est important de noter que toute transaction n'est pas nécessairement liée avec un utilisateur, ce qui génère beaucoup de valeurs manquantes lors de la fusion des deux fichiers. D'ailleurs, on compte $590\,540$ observations avec 434 variables différentes dont la plupart ont été anonymisées pour des raisons de confidentialité. N'ayant pas accès à la signification de ces variables, notre analyse est fortement contrainte. Le tableau 1 résume les informations principales de notre jeu de données. Dans ce paragraphe, nous essayons de trouver des attributs permettant de bien discriminer les transactions frauduleuses.

4.1. Débalancement de classe

Avant, on note un fort débalancement de classe (voir 4.1). En effet, on compte $569\,877$ transactions légitimes pour seulement $20\,663$ transactions frauduleuses. Ces dernières ne représentent donc que 3.626% des transactions légitimes. Des stratégies de rééchantillonnage devront être considérées pour corriger la situation.

4.2. Valeurs manquantes

Le processus d'apprentissage des algorithmes en analyse de données passent fréquemment par une stratégie de descente de gradient, ce qui nécessite la présence de variables numériques réelles. Or les données de IEEEFraudDetection sont infestées de valeurs manquantes. En tout, on compte 414 colonnes (384 numériques et 30 catégoriques) avec des valeurs manquantes. Ce chiffre représente 95.29% des colonnes. En tout, seulement 19 attributs numériques sont complets et valides: `TransactionID`, `isFraud`, `TransactionDT`, `TransactionAmt`, `card1` et `C1` à `C14`. Toutes les valeurs catégoriques sont incomplètes (à l'exception de l'étiquette `isFraud`). Le tableau 2 montre une fraction des colonnes problématiques. Dans la prochaine section, nous discuterons des possibles stratégies afin de palier à ce défi.

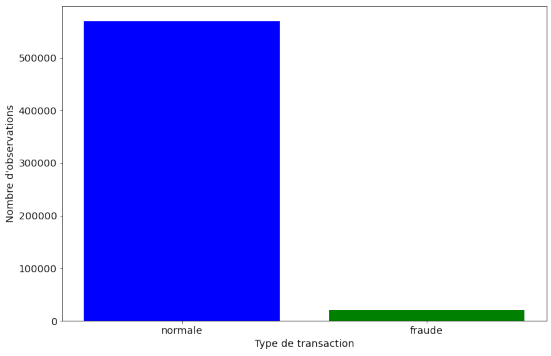

Figure 1. Débalancement de classe

Table 1. Informations de base sur les données IEEE-CIS Fraud Detection

Nombre d'observations N	Nombre de variables D	Ratio d'anomalie ρ
$590\,540$	434	0.035

4.3. Jours de la semaine

En premier lieu, vérifions la répartition des fraudes pendant la semaine. Les données originales ne contiennent pas directement cette information. Nous pouvons toutefois l'obtenir à partir du champs «TransactionDT» qui représente l'estampille temporelle de la transaction. Il suffit de diviser la valeur par le nombre de secondes dans une journées $(60 * 60 * 24)$ et calculer son modulo $\Big((60 * 60 * 24)\%7\Big)$. On constate que le jour 0 contient le plus grand nombre de transactions (4.3), mais que les transactions frauduleuses ne sont pas particulièrement présentes lors d'une journée en particulier (4.3).

4.4. Produits

Vérifions maintenant s'il existe une relation entre le type de produit et la fraude. La figure 4 indique que le produit «W» représente plus de 40% de toutes les fraudes et est suivi par «C» avec un peu moins de 40%. Toutefois, «W» représente 74.5% des données contre seulement 11.6% pour «C». Ce dernier est donc clairement surreprésenté dans les fraudes.

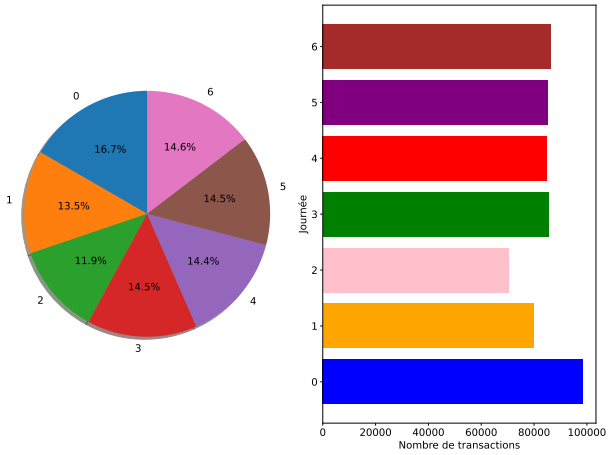

Figure 2. Nombre de transactions et pourcentage par jour

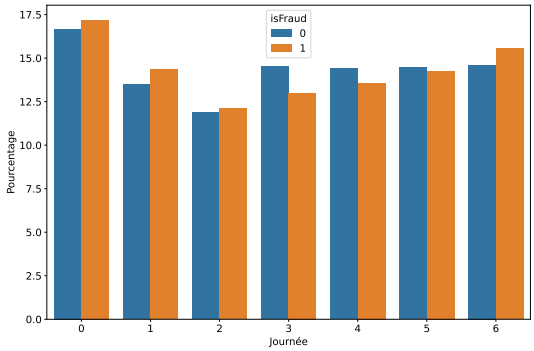

Figure 3. Proportion de fraudes par jour

4.5. Attributs des cartes

Six attributs sont utilisés pour représentées les différentes cartes : «card1», «card2», «card3», «card4», «card5», «card6». Les variables «card4» et «card6» représentent respectivement le distributeur de la carte (Mastercard, Discovery ou Visa) et le type de la carte (crédit ou débit). Les autres champs sont difficiles à interpréter mais contiennent tous des valeurs numériques. Les histogrammes de chaque attribut (figure 5) révèlent que « card3 » contient majoritairement deux valeurs : 150 et 180. Les histogrammes des autres champs montrent une grande variété de valeurs différentes, ce qui implique qu'ils ne sont facilement discriminants. En analysant la distribution de « card3 » en fonction de la classe (figure 7), on constate que la probabilité de rencontrer une transaction frauduleuse après 150 augmente considérablement.

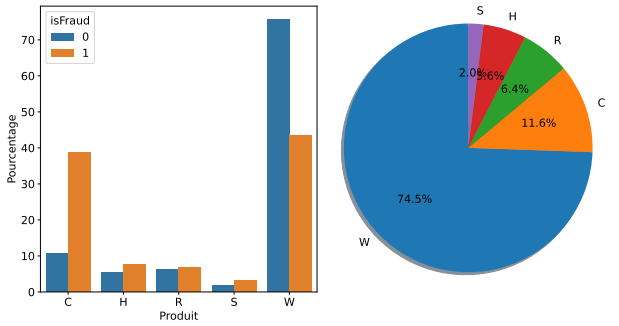

Figure 4. Représentation des produits parmi les fraudes

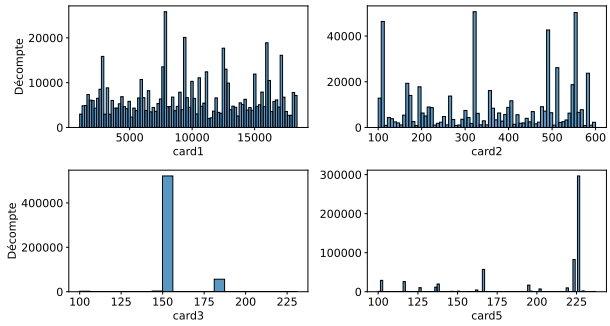

Figure 5. Histogrammes des attributs «card»

4.6. Fournisseurs des cartes

La plupart des transactions sont effectuées via les cartes Visa. Or la carte Discover est largement surreprésentée dans les transactions frauduleuses avec 7% de toutes les fraudes pour seulement quelques milliers de achats. Par contraste, seulement un peu plus de 3% des transactions par carte Visa sont frauduleuses alors que celles-ci compte entre $350\,000$ et $400\,000$ de toutes les transactions.

4.7. Domaine de email

Certaines des transactions sont effectuées sur internet et nous avons accès au nom de domaine associé à l'utilisateur de ces transactions. La figure 8 révèle que le domaine «protonmail.com» est relié à 40% des fraudes commises pour les achats en ligne.

Bref, on voit comment les fraudes varient en fonction des produits, des fournisseurs de carte et des noms de domaine des adresses emails. Celles-ci sont surreprésentées dans les produits «c» et pour la carte Discovery. De plus, l'attribut «card6» semble un bon indicateur de la présence ou l'absence de fraude avec un seuil clair à 160.

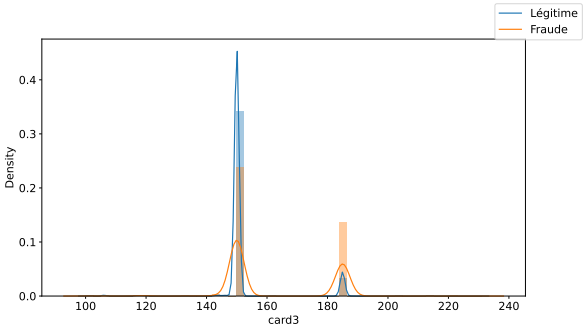

Figure 6. Distribution de «card3» en fonction de chaque classe

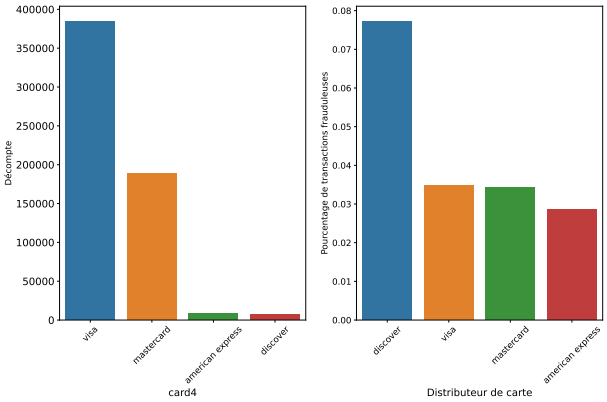

Figure 7. Représentation des fournisseurs de cartes parmi les fraudes

5. Nettoyage et pré-traitement

Comme mentionné dans la quatrième section, les données brutes présentent un grand nombre d'attributs (434) dont la plupart ont été anonymisés, ce qui présente un défi majeur. D'une part, plus le nombre d'attributs est élevé, plus l'interprétation des données et leur traitement par des algorithmes de classification est difficile. D'autre part, l'anonymisation implique qu'il est impossible d'effectuer une réduction de dimension par heuristique. En effet, sans connaître la nature d'une variable, il est difficile de la retirer sans craindre de supprimer potentiellement une variable discriminante. De manière générale, nos données présentent trois défis majeurs:

A. valeurs manquantes;

B. débalancement des classes;

C. hétérogéité des attributs.

Plusieurs stratégies différentes et viables peuvent être envisagées pour répondre à ceux-ci. Pour cette raison, nous présentons trois stratégies de prétraitement distinctes faisant appel à différentes configurations. La figure 9 résume les différentes possibilités pour chacun des problèmes mentionnés. Nous les résumons dans les sections suivantes:

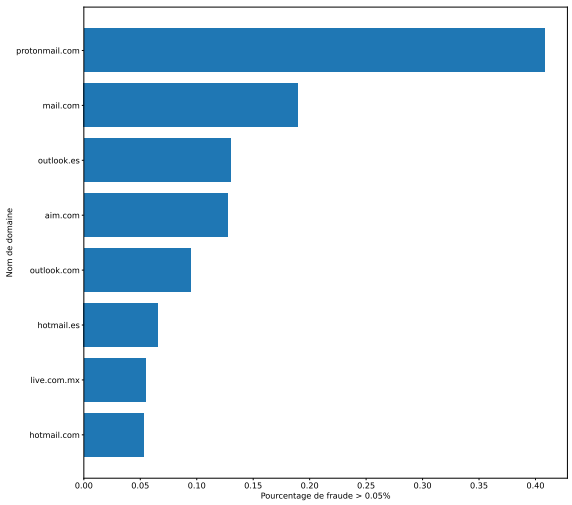

Figure 8. Nom de domaines et les transactions frauduleuses

5.1. Valeurs manquantes

On identifie deux stratégies générales pour gérer les données manquantes:

i. Supprimer les observations ou attributs associés à des valeurs nulles. Pour ce faire, on supprime les attributs dont le nombre de valeurs manquantes dépassent 50% ou 85% des données.

ii. Remplacer les valeurs manquantes par:

   (a) la médiane (pour les valeurs numériques seulement);

   (b) la moyenne (pour les valeurs numériques seulement);

   (c) une valeur déduite par l'algorithme des k-plus-proches-voisins;

   (d) le mode (pour les valeurs catégoriques seulement);

Le remplacement des valeurs manquantes est une option intéressante lorsqu'elles sont marginales. Dans le cas inverse, on modifie significativement la distribution statistiques de

Attribut	Nombre	Pourcentage
id_24	585 793	0.99
id_25	585 408	0.99
id_07	585 385	0.99
id_08	585 385	0.99
id_21	585 381	0.99
id_26	585 377	0.99
id_27	585 371	0.99
id_23	585 371	0.99
id_22	585 371	0.99
dist2	552 913	0.93
D7	551 623	0.93
id_18	545 427	0.92
D13	528 588	0.90
D14	528 353	0.89
D12	525 823	0.89
id_03	524 216	0.89
id_04	524 216	0.89
D6	517 353	0.88
id_33	517 251	0.87
id_10	515 614	0.87
id_09	515 614	0.87

Table 2. Les 20 attributs avec le plus de valeurs manquantes

nos observations de sorte qu'elle risque de ne pas se coller à la réalité empirique et notre modèle risque fortement d'être désuet dès son déploiement. Ainsi, pour palier à ce problème on peut amputer les attributs avec des valeurs manquantes dépassant une certaine proportion.

Les stratégies d'amputation ne sont pas sans problèmes non plus. En effet, on courre toujours le risque de supprimer des variables pertinentes pour la tâche de classification. De plus, sans heuristique, celles-ci sont souvent arbitraires, c'est-à-dire qu'il n'existe presque pas de critères objectifs permettant de déterminer quelles colonnes supprimer. Un expert du domaine pourrait potentiellement permettre d'effectuer une pré-sélection des variables à partir de connaissances déjà acquises, mais nous n'avons pas accès à une telle ressource pour ce travail. Les colonnes à valeurs uniques peuvent être supprimer, mais elles ne sont pas nombreuses dans notre cas.

5.2. Débalancement des classes

Nos modèles de classification assument que la grande majorité des données sont normales et sont ainsi en quelque sorte robuste au problème de débalancement. Toutefois, ce dernier revient lors de l'évaluation des modèles. En effet, il a été démontré qu'on peut manipuler artificiellement le f1-score en jouant avec le ratio de la classe minoritaire dans l'ensemble de tests [4]. Ainsi, il est pertinent de comparer les performances de nos modèles avec un rebalancement des classes. Nous évacuons toutefois les techniques fondées sur du sur-échantillonnage car la différence entre les classes majoritaire et minoritaire est beaucoup trop grande. On créerait beaucoup plus d'anomalies qu'il en existe et on manipulerait directement la distribution sous-jacente des données. Ainsi, on évalue seulement les deux stratégies suivantes:

1. Sous-échantillonnage aléatoire: on retire aléatoirement des instances de la classe majoritaire qui correspodent dans notre cas aux transactions légitimes.
2. Ne rien faire: on laisse le ratio des classes tel quel.

5.3. Valeurs catégoriques

La présence de nombreux attributs catégoriques pose problème. En effet, nos algorithmes de classification fonctionnent seulement avec des nombres réels. Il faut donc trouver une manière de convertir les valeurs catégoriques en valeurs numériques. Une stratégie traditionnelle consiste à les binariser (*one hot encoding*). Toutefois, lorsqu'un attribut a plusieurs modalités distinctes (comme c'est le cas pour `DeviceInfo` et `DeviceType` notamment) on fait exploser le nombre de variables. On doit donc opter pour une stratégie plus sophistiquée. Ainsi, nous considérons trois alternatives différentes:

1. Suppression: nous supprimons les 31 attributs catégoriques.
2. *Mean encoding*: les variables catégoriques sont remplacées par leur probabilité d'être associées à chaque étiquette.
3. Approche mixte: nous supprimons les attributs avec plusieurs modalités. Le reste des colonnes peuvent être encodées soit avec du *mean encoding* ou du *one-hot encoding* sans trop ajouter de nouvelles colonnes.

5.4. Sous-ensembles

Les différentes stratégies énumérées pour les différents défis nous ont mené à trois processus de nettoyages distincts qu'on nomme v1, v2 et v3. On désire ainsi évaluer la différence de performance entre les nettoyages sophistiqués et simples. Ces différentes stratégies sont résumées dans le tableau 4. Les sous-sections suivantes soulignent les autres étapes suivies pour chacun de ces sous-ensembles.

5.4.1. Sous-ensemble v1

Pour ce sous-ensemble, on a d'abord converti l'attribut `TransactionDT`, qui représente l'estampille temporelle (*timestamp*) de la transaction, en journées et en heures. Par la suite, on a éliminé les attributs dont le nombre de valeurs nulles dépasse 85% des données originales. Cette opération a permis d'éliminer 74 colonnes en tout. Les attributs catégoriques "id_31", "DeviceInfo", "DeviceType",

"P_emaildomain" et "R_emaildomain" ont été converti en attribut booléen en fonction de la présence ou non d'une valeur. Ayant ainsi éliminé les attributs avec beaucoup de modalités, les 21 variables catégoriques restantes ont été binarisées, ce qui ajoute 30 nouvelles colonnes. Finalement, un sous-échantillonnage aléatoire est effectué pour équilibrer le ratio d'observations entre les deux classes.

5.4.2. SOUS-ENSEMBLE V2

Pour cette version, nous avons simplement éliminer toutes les variables non numériques. Les valeurs manquantes ont été remplacées par leur médiane respective. Un sous-échantillonnage aléatoire a été effectué sur les observations de la classe majoritaire. Après cette opération, le ratio d'observations pour les deux classes est équivalent.

5.4.3. SOUS-ENSEMBLE V3

Pour ce sous-ensemble, nous avons tenté d'adopter une approche un peu plus chirurgicale. Tout d'abord, en analysant une possible relation entre les valeurs manquantes et les deux classes, on a repéré plusieurs attributs (75) dont les valeurs manquantes sont associées aux mêmes lignes. En les supprimant, on perd seulement 1441 observations ou moins de 0.02% des données totales. Même après cette opération, il reste 75.35% (327) des colonnes avec des valeurs manquantes.

Ensuite, on compte exactement 200 attributs dont les valeurs manquantes dépassent 50% des données. Nous avons réussi à conserver certains attributs avant de supprimer les autres. La colonne `DeviceInfo` décrit les compagnies et les versions des appreils mobiles. Ainsi, une valeur manquante signifie simplement que la transaction a été effectuée sans appareil mobile. Elles sont donc remplacées par la valeur `NoDevice`. Nous avons aussi modifié la même colonne pour séparer les noms d'appareils et numéro de versions, ce qui permet de réduire considérablement le nombre de modalités. Les valeurs catégoriques restantes sont converties par leur probabilité d'être frauduleuses. Avec ce pré-traitement, l'attribut `DeviceType`, qui indique si la transaction est effectuée sur un ordinateur ou un appareil mobile, devient désuette et nous la retirons en conséquence.

Nous appliquons une stratégie similaire pour les variables `P_emaildomain`, `R_emaildomain` et `id_31` qui contiennent respectivement les domaines des adresses électroniques des acheteurs et des marchands ainsi que la version et le type du navigateur web. La colonne `id_31` aurait pu, comme `DeviceInfo`, être simplifiée en regroupant les valeurs similaires sous une seule valeur. Par exemple, les différentes version du navigateur Chrome (Chrome v33 et Chrome v56, etc.) auraient pu être réunies sous l'unique étiquette "Chrome". Toutefois, en analysant leur relation avec les fraudes, on constate que les versions désuettes des différents navigateurs sont plus fréquemment associées à des fraudes que les nouvelles versions. Ainsi, nous les conservons toutes dans des catégories distinctes. Contrairement aux autres sous-ensembles, on conserve le débalancement entre les classes.

Table 3. Informations de base sur les différents sous-ensembles

Sous-ensemble	Nombre d'observations N	Nombre de variables D	Ratio d'anomalie ρ
v1	41 326	389	0.500
v2	41 326	160	0.500
v3	589 099	224	0.035

Finalement, notons qu'une normalisation de type *MinMax* a été appliquée a tous les attributs numériques pour tous les sous-ensembles énumérés. Le tableau 3 résume les informations de base sur les trois sous-ensembles.

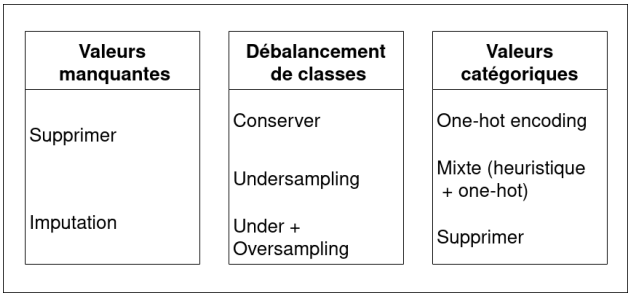

Figure 9. Résumé des stratégies de pré-traitement

6. Expériences

Dans cette section, nous présentons brièvement les algorithmes de détection de fraude non supervisés utilisés ainsi que notre procédure d'entraînement sur les données de IEEE-CIS Fraud Detection.

6.1. Algorithmes

Nous sélectionnons les algorithmes en fonction de leur impact historique (en termes de citations) et de leur originalité. Nous tentons aussi de sélectionner une variété de modèles suivants chacun des approches différentes afin de permettre une interprétation des résultats plus intéressante.

DAGMM: Deep Autoencoding Gaussian Mixture Model [17]. Dans cette architecture, un auto-encodeur est entraîné pour apprendre un sous-espace dimensionnel permettant de représenter les données originales tout en minimisant l'erreur de reconstruction entre celles-ci et leur reconstruc-

Nom	Valeur manquante	Ré-échantillonnage	Données catégoriques
v1	Imputation par algorithme K-NN (avec $k = 10$).	Sous-échantillonnage aléatoire	Suppression et *one-hot encoding*
v2	Imputation par valeur moyenne et suppression	Aucun	Suppression et *mean encoding*
v3	Imputation par valeur médiane	Sous-échantillonnage aléatoire	Suppression complète

Table 4. Résumé des différentes stratégies de nettoyage des données

tion à partir du sous-espace. L'erreur de reconstruction est concaténée avec la représentation latente et est fournie à autre réseau de neurones qui estime les paramètres d'une mixture gaussienne (GMM). Une fonction objectif conjointe permet d'entraîner les deux réseaux simultanément. Finalement, une fonction d'énergie (le logarithme de la vraisemblance) est utilisée comme mesure d'anomalie pour une observation quelconque. L'intuition derrière cette approche est que les anomalies sont incompressibles et vont générer des erreurs de reconstruction importantes comparativement aux instances normales.

DeepSVDD: Deep One-Class Classification [12]. Cette approche, beaucoup plus géométrique, nécessite moins d'hyperparamètres que DAGMM et n'admet aucune présupposition sur les données. Elle prend ses racines dans les méthodes SVDD qui minimisent le volume d'une sphère contenant les données, mais utilise en même temps un réseau de neurones pour apprendre une représentation des données. Autrement dit, un réseau de neurones est entraîné afin d'apprendre une représentation des données se trouvant dans une hypersphère de volume minimal. La distance euclidienne par rapport au centre de cette hypersphère est utilisée pour identifier les anomalies.

One-Class SVM [14]. Cette variante des SVM cherche une représentation des données et un hyperplan de marge maximal permettant de bien séparer les données projetées dans le nouvel espace dimensionnel. Une observation est jugée normale ou anormale en fonction de sa position par rapport à cet hyperplan. Ce problème peut être traduit par une formulation quadratique et ainsi être résolue à l'aide des multiplicateurs de Lagrange. La procédure d'optimisation est donc plus simple que celle d'un réseau de neurones, mais nécessite le maintien en mémoire de la matrice des noyaux qui croît en fonction du nombre de données fournies en entrée.

6.2. Procédure d'entraînement

Les réseaux de neurones sont implémentés à l'aide de PyTorch et sont optimisés par l'algorithme ADAM avec un taux d'apprentissage $\alpha = 0.001$ et une taille de lot de 128. Nous utilisons l'implémentation de OC-SVM offerte par la librairie Scikit-Learn le noyau RBF le paramètre ν par défaut. Les modèles sont entraînés exclusivement sur 50% des données normales. Les anomalies et le reste des transactions légitimes sont intégrées dans l'ensemble de test. Cette stratégie suit la tradition de détection d'anomalies où on assume que nos données sont largement normales [17, 12]. Pour minimiser l'effet du hasard, on répète les entraînements 10 fois et on rapporte la moyenne et l'écart-type des résultats obtenus. On évite ainsi le risque de tomber sur un bon résultat causé par la formation aléatoire d'un échantillon biaisé. On ne suit pas cette procédure pour OC-SVM car sa procédure d'optimisation est beaucoup plus stable que celle des réseaux de neurones.

Finalement, nous considérons les métriques precision, recall, f1-score, AUC et AUPR pour comparer la performance des algorithmes. Le seuil τ pour les trois premières mesures est déterminé itérativement. On le fait varier autour du taux d'anomalies ρ et on conserve le résultat qui maximise le f1-score. Cette procédure tente d'imiter le cas empirique où, après un déploiement, on devrait constamment ajuster le seuil pour maximiser la détection des anomalies. Aussi, elle donne une idée des meilleures performances possibles pour nos modèles. Les métriques AUC et AUPR, quant à elles, permettent d'évaluer les modèles sans prendre en considération la notion de seuil. On rapporte l'AUPR, car elle porte plus d'importance sur les prédictions de la classe positive contrairement à l'AUC qui considère un poids identique pour les deux classes. Comme nous sommes intéressés par les capacités prédictives de nos algorithmes sur les fraudes, l'AUPR devient très pertinente. Les résultats sont présentés dans le tableau 5.

6.3. Résultats et interprétation

Les résultas obtenus sont particulièrement intéressants. Premièrement, les modèles ne performent pas tous de manière similaire entre les différents sous-ensembles. Le modèle OC-SVM performe le mieux en terme de f1-score sur v1 et v2. Toutefois, ce résultat est causé directement par notre procédure d'optimisation du f1-score qui a priorisé le seuil générant un recall parfait. Dans notre situation, le recall mesure la sensibilité des algorithmes aux faux négatifs, qui sont représentés par les transactions légitimes. Lorsqu'on regarde les capacités prédictives sur les fraudes, OC-SVM est supplanté par les réseaux de neurones profonds. De plus, lorsqu'on ignore ces métriques, DeepSVDD performent particulièrement bien en termes de AUC et d'AUPR. Il réussit à s'imposer sur v2 et plus particulièrement sur v1 avec une différence de plus de 10 points. Il se fait cependant supplanter de justesse par DAGMM sur v3.

Ensuite, lorsqu'on compare les résultats entre les différents sous-ensembles, la stratégie de sous-échantillonnage ne semble pas avoir été payante. Cette interprétation est fortement suggérée par les écarts énormes qui séparent les performances respectives des modèles entre v1, v2 et v3. Ce résultat n'est pas surprenant considérant que les algorithmes sont conçus pour être entrainés sur un maximum de données normales possibles. Notons cependant une baisse importance de performance pour OC-SVM sur l'ensemble v3. Ce problème est probalement causé par le trop grand nombre d'observations (589 099) comparativement au deux autres ensembles (41 326).

7. Conclusion

En conclusion, IEEE-CIC Fraud Detection constitue une base de données assez complexe pour le problème de détection de fraudes. Son grand débalancement de classe, ses attributs hétérogènes et le grand nombre de valeurs manquantes rendent son traitement particulièrement difficile. De plus, l'absence de titres significatifs pour tous les attributs rend toute interprétation difficile. Nous avons tout de même pu proposer trois processus de nettoyages assez différents. Les résultats sur ceux-ci nous indiquent que le type de nettoyage a un effet différent sur tous les modèles. En effet, le sous-ensemble v3, qui présente le nettoyage le plus pointu et qui conserve le plus de données possibles, permet d'améliorer grandement les performances des réseaux de neurones, mais rend le modèle OC-SVM peu pratique. À l'inverse, ce dernier performe mieux en terme de recall en présence de moins d'observations. Dans tous les cas cependant, les modèles issus de l'apprentissage profonds supplantent OC-SVM en termes de AUPR de manière consistante ce qui en font, en vertu de l'analyse actuelle des modèles plus adaptés à la détection de fraudes. Dans une autre étude, il serait intéressant d'analyse ce problème en termes de clients. En ce moment, nous avons adopté une perspective centrée sur les transactions individuelles, mais peut-être que la tâche d'identification des clients frauduleux serait plus garante de succès.

Références

[1] Adewumi A.O. and Akinyelu A.A. "A survey of machine-learning and nature-inspired based credit card fraud detection techniques". In: ed. by Pat Langley. 2017.

[2] L Bontemps et al. "Collective anomaly detection based on long short-term memory recurrent neural networks". In: *Int J Adv Res Comput Commun Eng*. 2016.

[3] *Credit Card Fraud Statistics. Statistic Brain*. 2018. URL: https://www.statisticbrain.com/credit-card-fraud-statistics/.

[4] Damien Fourure et al. "Anomaly Detection: How to Artificially Increase Your F1-Score with a Biased Evaluation Protocol". In: *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track*. Ed. by Yuxiao Dong et al. Cham: Springer International Publishing, 2021, pp. 3–18. ISBN: 978-3-030-86514-6.

[5] Kang Fu et al. "Credit Card Fraud Detection Using Convolutional Neural Networks". In: *Neural Information Processing*. Ed. by Akira Hirose et al. Cham: Springer International Publishing, 2016, pp. 483–490. ISBN: 978-3-319-46675-0.

[6] László Jeni, Jeffrey Cohn, and Fernando De la Torre. "Facing Imbalanced Data - Recommendations for the Use of Performance Metrics". In: vol. 2013. Sept. 2013. DOI: 10.1109/ACII.2013.47.

[7] Johannes Jurgovsky et al. "Sequence classification for credit-card fraud detection". In: *Expert Systems with Applications* 100 (2018), pp. 234–245. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.01.037. URL: https://www.sciencedirect.com/science/article/pii/S0957417418300435.

[8] Ahmed AHE Khan MZ Pathan JD. "Credit card fraud detection using hidden markov model and K-clustering". In: *Int J Adv Res Comput Commun Eng 3*. 2014, pp. 5458–5461.

[9] Yvan Lucas and Johannes Jurgovsky. "Credit card fraud detection using machine learning: A survey". In: *CoRR* abs/2010.06479 (2020). arXiv: 2010.06479. URL: https://arxiv.org/abs/2010.06479.

[10] *Reproducible machine learning for credit card fraud detection - practical handbook*. URL: https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_3_GettingStarted/Introduction.html.

[11] Lukas Ruff et al. "A Unifying Review of Deep and Shallow Anomaly Detection". In: *CoRR* abs/2009.11732 (2020). arXiv: 2009.11732. URL: https://arxiv.org/abs/2009.11732.

[12] Lukas Ruff et al. "Deep One-Class Classification". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 4393–4402.

[13] Duman E Sahin Y. "Detecting credit card fraud by decision trees and support vector machines". In: *Proceedings of the international multiConference of engineers and computer scientists*. 2011, pp. 1–6.

Modèle	Sous-ensemble	Precision	Recall	F1-Score	AUC	AUPR
OC-SVM	v1	$.667 \pm .000$	$\mathbf{1.00 \pm .000}$	$\mathbf{.800 \pm .000}$	$.573 \pm .000$	$.702 \pm .000$
DeepSVDD	v1	$\mathbf{.732 \pm .036}$	$.664 \pm .027$	$.778 \pm .026$	$\mathbf{.692 \pm .065}$	$\mathbf{.823 \pm .031}$
DAGMM	v1	$.679 \pm .059$	$.730 \pm .007$	$.704 \pm .058$	$.543 \pm .151$	$.715 \pm .097$
OC-SVM	v2	$.667 \pm .000$	$\mathbf{1.00 \pm .000}$	$\mathbf{.800 \pm .000}$	$.578 \pm .000$	$.705 \pm .000$
DeepSVDD	v2	$.664 \pm .025$	$.714 \pm .027$	$.688 \pm .026$	$\mathbf{.559 \pm .058}$	$\mathbf{.753 \pm .035}$
DAGMM	v2	$\mathbf{.675 \pm .001}$	$.725 \pm .007$	$.699 \pm .007$	$.527 \pm .023$	$.690 \pm .023$
OC-SVM	v3	$.432 \pm .000$	$.573 \pm .000$	$.495 \pm .000$	$.410 \pm .000$	$.403 \pm .000$
DeepSVDD	v3	$.915 \pm .104$	$\mathbf{.993 \pm .018}$	$.949 \pm .056$	$.859 \pm .203$	$.940 \pm .086$
DAGMM	v3	$\mathbf{.947 \pm .074}$	$.967 \pm .054$	$\mathbf{.956 \pm .064}$	$\mathbf{.907 \pm .146}$	$\mathbf{.963 \pm .056}$

Table 5. Les résultats sur les différents sous-ensembles

[14] Bernhard Schölkopf et al. "Support Vector Method for Novelty Detection". In: vol. 12. Jan. 1999, pp. 582–588.

[15] *Single euro payments area (SEPA)*. Mar. 2021. URL: https://ec.europa.eu/info/business-economy-euro/banking-and-finance/consumer-finance-and-payments/payment-services/single-euro-payments-area-sepa_en#:~:text=SEPA.

[16] V.V Vlasselaer et al. "Apate: A novel approach for automated credit card transactions fraud detection using network-based extensions". In: *Decision support systems*. 2015.

[17] Bo Zong et al. "Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection." In: *ICLR (Poster)*. OpenReview.net, 2018. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2018.html#ZongSMCLCC18.