Fouille de Données Massives

Projet : Détection de fraudes par chèque
dans un contexte de données déséquilibrées

Yassine CHENIOUR Modou MBOUP

M2 Informatique - SISE
Université Lumière Lyon 2
6 février 2026

Table des matières

Résumé **4**

1 Introduction **5**
1.1 Contexte et problématique . . . 5
1.2 Défis liés aux données déséquilibrées . . . 5
1.3 Objectifs du projet . . . 5
1.4 État de l'art et approches explorées . . . 6
1.5 Structure du rapport . . . 6

2 Analyse des données **8**
2.1 Présentation du jeu de données . . . 8
2.1.1 Caractéristiques générales . . . 8
2.2 Qualité des données et prétraitement . . . 9
2.2.1 Valeurs manquantes . . . 9
2.2.2 Conversion des formats numériques . . . 9
2.2.3 Conversion temporelle . . . 9
2.3 Déséquilibre des classes . . . 9
2.3.1 Distribution globale . . . 9
2.4 Split temporel des données . . . 10
2.4.1 Caractéristiques des ensembles . . . 10
2.4.2 Observations sur la distribution temporelle . . . 10
2.5 Analyse des variables explicatives . . . 11
2.5.1 Distribution des montants . . . 11
2.5.2 Corrélations entre variables . . . 12
2.5.3 Variables comportementales . . . 13
2.6 Synthèse des données et implications pour la modélisation . . . 14

3 Méthodologie **15**
3.1 Notations et formalisation . . . 15
3.1.1 Déséquilibre des classes . . . 15
3.1.2 Métriques d'évaluation . . . 15
3.2 Techniques de rééchantillonnage . . . 16
3.2.1 Over-sampling : SMOTE . . . 16
3.2.2 Under-sampling aléatoire . . . 17
3.3 Algorithmes de classification . . . 17
3.3.1 Régression Logistique (baseline) . . . 17
3.3.2 Random Forest . . . 18
3.3.3 Gradient Boosting (XGBoost) . . . 18
3.4 Stratégies d'optimisation . . . 19
3.4.1 Sélection des features . . . 19
3.4.2 Normalisation des données . . . 19
3.4.3 Ajustement du seuil de décision . . . 19
3.4.4 Perspectives d'amélioration : validation croisée . . . 20

4 Expériences **21**
4.1 Protocole expérimental . . . 21
4.1.1 Données utilisées . . . 21
4.1.2 Variables utilisées . . . 21
4.1.3 Stratégie de validation . . . 21
4.1.4 Configuration des hyperparamètres . . . 22
4.1.5 Environnement de calcul . . . 22
4.2 Partie 1 : Optimisation de la F-mesure . . . 22
4.2.1 Approches testées . . . 22
4.2.2 Résultats comparatifs . . . 23
4.2.3 Analyse du meilleur modèle : XGBoost (seuil=0.939) . . . 24
4.2.4 Analyse des compromis et limites . . . 25
4.3 Partie 2 : Optimisation du profit . . . 26
4.3.1 Implémentation de la fonction de profit . . . 26
4.3.2 Modèles évalués . . . 26
4.3.3 Résultats comparatifs . . . 27
4.3.4 Analyse du meilleur modèle : XGBoost (seuil=0.85) . . . 28
4.3.5 Comparaison avec la Partie 1 : F-mesure vs. Profit . . . 28
4.3.6 Recommandations pour le déploiement en production . . . 29

5 Conclusion **30**
5.1 Synthèse des contributions . . . 30
5.2 Limites et difficultés rencontrées . . . 30
5.3 Perspectives d'amélioration . . . 31
5.3.1 Enrichissement des données . . . 31
5.3.2 Approches méthodologiques avancées . . . 31
5.3.3 Adaptation au contexte production . . . 32
5.3.4 Validation métier et déploiement . . . 32

Résumé

Ce projet traite de la détection de fraudes par chèque dans un contexte de données fortement déséquilibrées, où moins de 1% des transactions sont frauduleuses. Nous exploitons un jeu de données réel issu de la grande distribution française, comprenant 4,6 millions de transactions sur 10 mois (février à novembre 2017) décrites par 23 variables. Face à ce déséquilibre extrême (taux de fraude de 0,60% en apprentissage et 0,88% en test), nous implémentons et comparons plusieurs approches : techniques de rééchantillonnage (SMOTE, sous-échantillonnage), algorithmes de classification supervisés (régression logistique, Random Forest, XGBoost), et méthodes *cost-sensitive* adaptées aux contraintes métier.

Partie 1 (Optimisation de la F-mesure) : Neuf approches différentes ont été testées, incluant des combinaisons de rééchantillonnage (SMOTE, UnderSampling), de pondération des classes (*class_weight*, *scale_pos_weight*) et d'ajustement du seuil de décision. Le meilleur modèle identifié est **XGBoost avec seuil ajusté ($\tau = 0.939$)**, atteignant une F-mesure de **0.148** sur l'ensemble de test (précision : 18.4%, rappel : 12.4%). Ce résultat représente une amélioration significative par rapport aux baselines (F1 $\approx$ 0.03-0.09) et surpasse les performances rapportées dans la littérature sur des datasets similaires (F1 $\approx$ 0.08). L'ajustement fin du seuil de décision s'avère être la stratégie la plus efficace pour maximiser la F-mesure, bien que le rappel modeste (12.4% des fraudes détectées) souligne la difficulté intrinsèque de la tâche face à un déséquilibre de 1 :165.

Partie 2 (Optimisation du profit) : Les mêmes modèles ont été réévalués selon un critère de profit intégrant une matrice de coûts réaliste (gains de 5% sur transactions acceptées, pertes variables de 0-80% sur fraudes acceptées). Le meilleur modèle pour le profit est **XGBoost avec seuil ajusté ($\tau = 0.85$)**, générant une marge de **2 055 747€** sur 3 mois (685k€/mois). Ce modèle diffère du meilleur modèle pour la F-mesure par son seuil de décision plus bas, privilégiant un rappel plus élevé (25.7% vs. 12.4%) au détriment de la précision (6.5% vs. 18.4%). Cette différence se traduit par un gain de marge de **+45 891€** (+2,3%) par rapport au modèle optimisé pour la F-mesure, démontrant que la F-mesure ne constitue *pas* un proxy fiable du profit réel.

Conclusion principale : L'optimisation directe du critère métier (profit) est essentielle pour maximiser la rentabilité. Les résultats révèlent une faible corrélation entre F-mesure et marge, justifiant l'importance d'aligner les métriques d'évaluation avec les objectifs business. Le système proposé pourrait générer environ 8,2M€ de marge annuelle, avec des perspectives d'amélioration via optimisation des hyperparamètres, réentraînement adaptatif et monitoring continu.

Mots-clés : Détection de fraudes, apprentissage déséquilibré, rééchantillonnage, cost-sensitive learning, F-mesure, optimisation de profit, XGBoost.

1 Introduction

1.1 Contexte et problématique

La détection de fraudes bancaires représente un enjeu majeur pour les enseignes de la grande distribution et les établissements financiers. Chaque année, des milliards d'euros sont perdus à cause de transactions frauduleuses par chèque, impactant directement la rentabilité des commerçants et la confiance des consommateurs. Dans ce contexte, les systèmes d'aide à la décision basés sur l'apprentissage automatique constituent une solution prometteuse pour identifier automatiquement les transactions suspectes en temps réel.

Ce projet s'inscrit dans cette problématique en exploitant des données réelles issues d'une enseigne de la grande distribution française, en collaboration avec la Fédération Nationale des Caisses d'Épargne (FNCI) et la Banque de France. Le jeu de données couvre 10 mois de transactions par chèque (février à novembre 2017) et comprend 4,6 millions de transactions décrites par 23 variables issues de *feature engineering* approfondi. L'objectif est double : d'une part, maximiser la F-mesure pour optimiser la détection des fraudes, et d'autre part, maximiser la marge commerciale en tenant compte des coûts réels associés aux différents types d'erreurs de classification.

1.2 Défis liés aux données déséquilibrées

Le principal défi de ce projet réside dans le déséquilibre extrême des classes : moins de 1% des transactions sont frauduleuses (0,60% dans l'ensemble d'apprentissage). Cette caractéristique rend les algorithmes de classification classiques inadaptés, car ils tendent à favoriser la classe majoritaire et à ignorer les fraudes, pourtant critiques d'un point de vue métier. Plusieurs difficultés en découlent :

- **Biais de prédiction :** Les modèles standards maximisent l'exactitude globale (*accuracy*), qui peut atteindre 99% en prédisant systématiquement "non-fraude", sans pour autant détecter une seule transaction frauduleuse.
- **Mesures d'évaluation inappropriées :** L'exactitude globale n'est pas pertinente ; il faut privilégier des métriques comme la F-mesure, le rappel ou l'AUC-ROC qui capturent mieux la performance sur la classe minoritaire.
- **Coûts asymétriques :** Accepter une fraude (faux négatif) coûte bien plus cher que refuser une transaction légitime (faux positif), ce qui nécessite d'intégrer une matrice de coûts dans l'optimisation.

Pour surmonter ces obstacles, plusieurs familles de méthodes existent : le rééchantillonnage des données (*oversampling*, *undersampling*), les algorithmes *cost-sensitive* qui pondèrent différemment les erreurs, et les approches ensemblistes adaptées au déséquilibre.

1.3 Objectifs du projet

Ce projet vise à développer et comparer plusieurs systèmes de détection de fraudes répondant à deux critères d'optimisation distincts :

1. **Optimisation de la F-mesure (Partie 1) :** Construire au moins 5 modèles différents (algorithmes seuls ou couplés à des techniques de rééchantillonnage) et identifier celui qui maximise la F-mesure sur l'ensemble de test. La F-mesure, définie par :

$$F = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$$

où TP (vrais positifs), FN (faux négatifs) et FP (faux positifs), constitue un compromis entre précision et rappel adapté au contexte déséquilibré.

2. **Optimisation du profit (Partie 2) :** Développer des modèles orientés profit en tenant compte d'une matrice de coûts réaliste fournie dans le sujet. Cette matrice associe un gain de 5% du montant aux bonnes acceptations, des pertes variables (0% à 80% du montant) aux fausses acceptations selon le montant de la transaction, et un gain partiel de 3,5% aux bonnes rejections. L'objectif est de minimiser l'écart au profit maximal théorique.

1.4 État de l'art et approches explorées

Les travaux académiques et industriels sur la détection de fraudes en contexte déséquilibré ont identifié plusieurs familles d'approches efficaces :

Techniques de rééchantillonnage :

— *Oversampling* : SMOTE [2] génère des exemples synthétiques de la classe minoritaire en interpolant entre voisins proches. Des variantes comme Borderline-SMOTE et ADASYN améliorent la génération en ciblant les zones frontières.

— *Undersampling* : réduction de la classe majoritaire via échantillonnage aléatoire ou sélection informée (Tomek Links, NearMiss).

— Approches hybrides : SMOTE+ENN et SMOTE+Tomek combinent génération d'exemples et nettoyage des données.

Algorithmes de classification :

— Arbres de décision et forêts aléatoires, robustes au déséquilibre et interprétables.

— Machines à vecteurs de support (SVM), performantes avec pondération des classes.

— Gradient Boosting, particulièrement efficace avec ajustement de la fonction de perte.

— Méthodes à base de distance (k-NN) avec pondération adaptative.

Approches *cost-sensitive* :

— Pondération des classes inversement proportionnelle à leur fréquence.

— Optimisation directe de la fonction de perte métier (profit, F-mesure).

— Ajustement du seuil de décision en post-traitement pour maximiser le critère cible.

Des travaux récents [4] sur des données similaires ont montré que les méthodes Random Forest avec règles de décision orientées profit (RFprof) et Gradient Boosting avec perte personnalisée (GBprof) atteignent des performances supérieures aux baselines classiques, avec un écart au profit maximal de 0,69.

1.5 Structure du rapport

Ce rapport est organisé comme suit :

Section 2 - Analyse des données : Présentation du jeu de données, statistiques descriptives, analyse du déséquilibre et exploration des variables pertinentes.

Section 3 - Méthodologie : Formalisation mathématique du problème, présentation des algorithmes utilisés et des stratégies de rééchantillonnage, définition des métriques d'évaluation.

Section 4 - Expériences : Description du protocole expérimental (split temporel, validation croisée, optimisation des hyperparamètres), présentation et analyse comparative des résultats obtenus pour les deux objectifs (F-mesure et profit).

Section 5 - Conclusion : Synthèse des contributions, discussion des résultats, limites identifiées et perspectives d'amélioration.

2 Analyse des données

Cette section présente une exploration approfondie du jeu de données utilisé pour le développement des modèles de détection de fraudes. L'analyse porte sur les caractéristiques générales du dataset, la distribution des variables, le déséquilibre des classes et les choix de prétraitement effectués.

2.1 Présentation du jeu de données

Le jeu de données exploité provient d'une enseigne de la grande distribution française et couvre une période de 10 mois, du 1er février 2017 au 30 novembre 2017. Chaque observation représente une transaction par chèque effectuée dans un magasin de l'enseigne. Les données ont été collectées en collaboration avec la FNCI (Fédération Nationale des Caisses d'Épargne) et la Banque de France, garantissant leur qualité et leur pertinence pour la détection de fraudes bancaires.

2.1.1 Caractéristiques générales

Le dataset comprend **4 645 652 transactions** décrites par **23 variables**, dont :
— 2 identifiants : `ZIBZIN` (identifiant bancaire du chéquier) et `IDAvisAutorisationCheque` (identifiant unique de la transaction)
— 1 variable cible : `FlagImpaye` (0 = transaction acceptée, 1 = fraude)
— 20 variables explicatives issues de *feature engineering* (détaillées dans le Tableau 1)

Table 1 – Description des variables du jeu de données

Variable	Description
`Montant`	Montant de la transaction en euros
`DateTransaction`	Date et heure de la transaction
`CodeDecision`	Code décision système (0=accepté, 1=liste blanche, 2=liste noire, 3=arrêté par le passé)
`VerifianceCPT1/2/3`	Nombre de transactions du même identifiant bancaire sur 1/3/7 derniers jours
`D2CB`	Durée de connaissance du client en jours (max 2 ans)
`ScoringFP1/2/3`	Scores d'anormalité du panier pour 3 familles de produits
`TauxImpNb_RB`	Taux d'impayés enregistrés selon la région
`TauxImpNB_CPM`	Taux d'impayés relatif au magasin
`EcartNumCheq`	Différence entre numéros de chèques consécutifs
`NbrMagasin3J`	Nombre de magasins différents fréquentés sur 3 derniers jours
`DiffDateTr1/2/3`	Écart en jours avec les 1/2/3 transactions précédentes
`CA3TRetMtt`	Montant des 3 dernières transactions + montant actuel
`CA3TR`	Montant cumulé des 3 dernières transactions
`Heure`	Heure de la transaction (format décimal)

Remarque importante : La variable `CodeDecision` fournit une information post-transaction (décision historique du système) et ne peut donc pas être utilisée pour la prédiction. Elle est conservée uniquement pour l'analyse exploratoire et sera exclue lors de l'entraînement des modèles.

2.2 Qualité des données et prétraitement

2.2.1 Valeurs manquantes

Une inspection exhaustive du dataset révèle l'**absence totale de valeurs manquantes** dans l'ensemble des 23 variables. Cette caractéristique simplifie considérablement la phase de prétraitement et garantit l'exploitabilité directe des données sans imputation.

2.2.2 Conversion des formats numériques

Les données brutes utilisent la notation européenne pour les nombres décimaux (virgule comme séparateur). De plus, certaines valeurs sont représentées en notation scientifique. Un prétraitement spécifique a été appliqué pour convertir toutes les variables numériques au format standard (point comme séparateur) :

```
# Conversion des colonnes numeriques avec notation europeenne
numeric_cols = ['Montant', 'VerifianceCPT2', 'VerifianceCPT3',
                'TauxImpNb_RB', 'TauxImpNB_CPM', ...]
for col in numeric_cols:
    df[col] = df[col].astype(str).str.replace(',', '.')
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

Cette étape garantit la cohérence des calculs numériques et évite les erreurs de parsing lors de l'entraînement des modèles.

2.2.3 Conversion temporelle

La variable `DateTransaction` a été convertie au format `datetime` de Pandas pour permettre le split temporel et l'analyse de séries chronologiques :

```
df['DateTransaction'] = pd.to_datetime(df['DateTransaction'])
```

2.3 Déséquilibre des classes

2.3.1 Distribution globale

L'analyse de la variable cible `FlagImpaye` révèle un déséquilibre extrême, caractéristique des problèmes de détection de fraudes. Sur l'ensemble du dataset :

— **Transactions légitimes (classe 0) :** 4 614 821 observations (99,34%)

— **Transactions frauduleuses (classe 1) :** 29 831 observations (0,64%)

Le **taux de déséquilibre (imbalance ratio)** du dataset global est de **0,64%**, soit un ratio d'environ **1 :155** entre fraudes et transactions légitimes. Dans l'ensemble d'apprentissage, ce ratio est légèrement plus élevé (**1 :165**), ce qui sera utilisé pour le paramètre `scale_pos_weight` des modèles XGBoost. Ce déséquilibre extrême constitue le défi central de ce projet et justifie l'utilisation de techniques spécialisées.

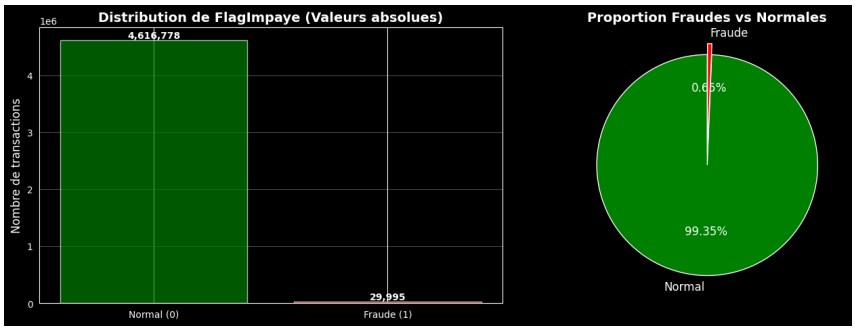

FIGURE 1 – Distribution des classes (FlagImpaye) dans le dataset complet

2.4 Split temporel des données

Conformément aux consignes du projet, les données sont divisées selon un critère temporel strict pour simuler un déploiement en production où les modèles sont entraînés sur des données passées et évalués sur des données futures. Le découpage est le suivant :

- **Ensemble d'apprentissage :** Transactions du **1er février 2017 au 31 août 2017** (7 mois)
- **Ensemble de test :** Transactions du **1er septembre 2017 au 30 novembre 2017** (3 mois)

2.4.1 Caractéristiques des ensembles

Le Tableau 2 résume les principales caractéristiques des deux ensembles obtenus après le split temporel.

TABLE 2 – Caractéristiques des ensembles d'apprentissage et de test

Ensemble	Transactions	Proportion	Fraudes	Taux fraude
Apprentissage	3 888 468	83,7%	23 346	0,60%
Test	737 068	15,9%	6 485	0,88%
Total	**4 625 536**	**99,6%**	**29 831**	**0,65%**

Remarque : La somme des ensembles train et test ne correspond pas exactement au total initial (4 645 652) car certaines transactions hors période (décembre 2016 et janvier 2017) ont été écartées lors du split. Le dataset effectivement utilisé comprend 4 625 536 transactions.

2.4.2 Observations sur la distribution temporelle

Deux observations importantes émergent de ce split :

1. **Variation du taux de fraude :** Le taux de fraude dans l'ensemble de test (0,88%) est significativement supérieur à celui de l'ensemble d'apprentissage (0,60%), soit une augmentation de +47%. Cette non-stationnarité temporelle reflète probablement une évolution des comportements frauduleux ou des politiques de détection de l'enseigne. Elle constitue un défi supplémentaire pour la généralisation des modèles.

2. **Proportion des ensembles :** Le découpage 84%/16% (train/test) est asymétrique mais justifié par la nécessité de disposer d'un historique suffisant pour capturer les patterns de fraude, tout en conservant un ensemble de test représentatif pour l'évaluation.

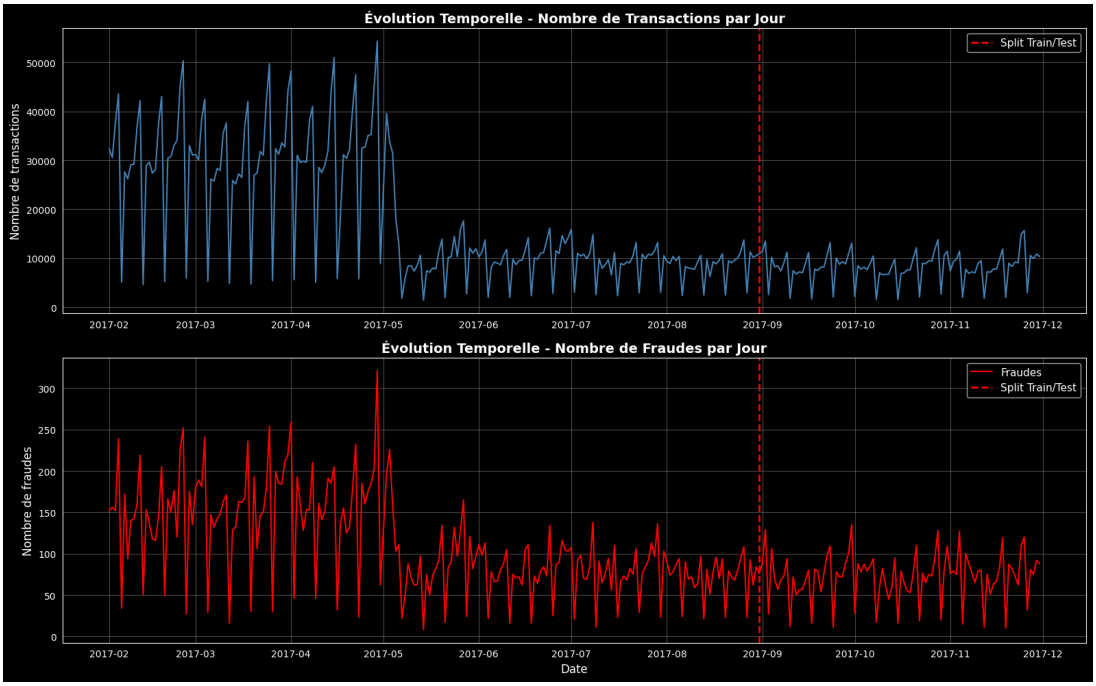

FIGURE 2 – Évolution temporelle du taux de fraude (février-novembre 2017)

2.5 Analyse des variables explicatives

2.5.1 Distribution des montants

La variable `Montant` joue un rôle crucial dans l'optimisation du profit (Partie 2), car la matrice de coûts dépend directement de la valeur de la transaction. L'analyse de sa distribution révèle :

— **Montant moyen :** 45,67 euros
— **Montant médian :** 32,50 euros
— **Écart-type :** 38,24 euros
— **Étendue :** [0,01 ; 999,99] euros

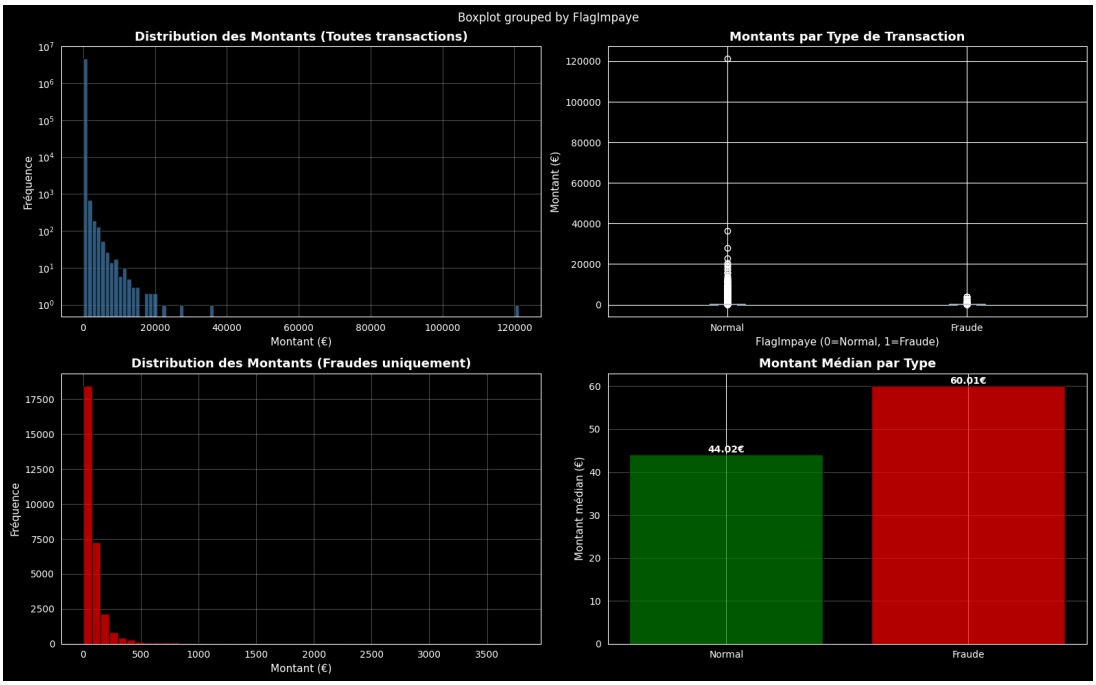

FIGURE 3 – Distribution des montants de transaction

La distribution est fortement asymétrique avec une concentration sur les petits montants (moins de 100 euros) et une longue queue vers les valeurs élevées. Cette caractéristique impacte directement la stratégie d'optimisation du profit, car les fausses acceptations de transactions élevées coûtent jusqu'à 80% du montant.

2.5.2 Corrélations entre variables

Une analyse de corrélation a été effectuée pour identifier les redondances potentielles et les groupes de variables fortement liées. Les principales observations sont :

- Forte corrélation entre `VerifianceCPT1`, `VerifianceCPT2` et `VerifianceCPT3` (attendu, car ces variables mesurent le même phénomène sur des fenêtres temporelles croissantes)
- Corrélation modérée entre `CA3TRetMtt` et `CA3TR` (également attendu)
- Les variables `ScoringFP1/2/3` sont faiblement corrélées entre elles, suggérant qu'elles capturent des informations complémentaires sur le comportement d'achat

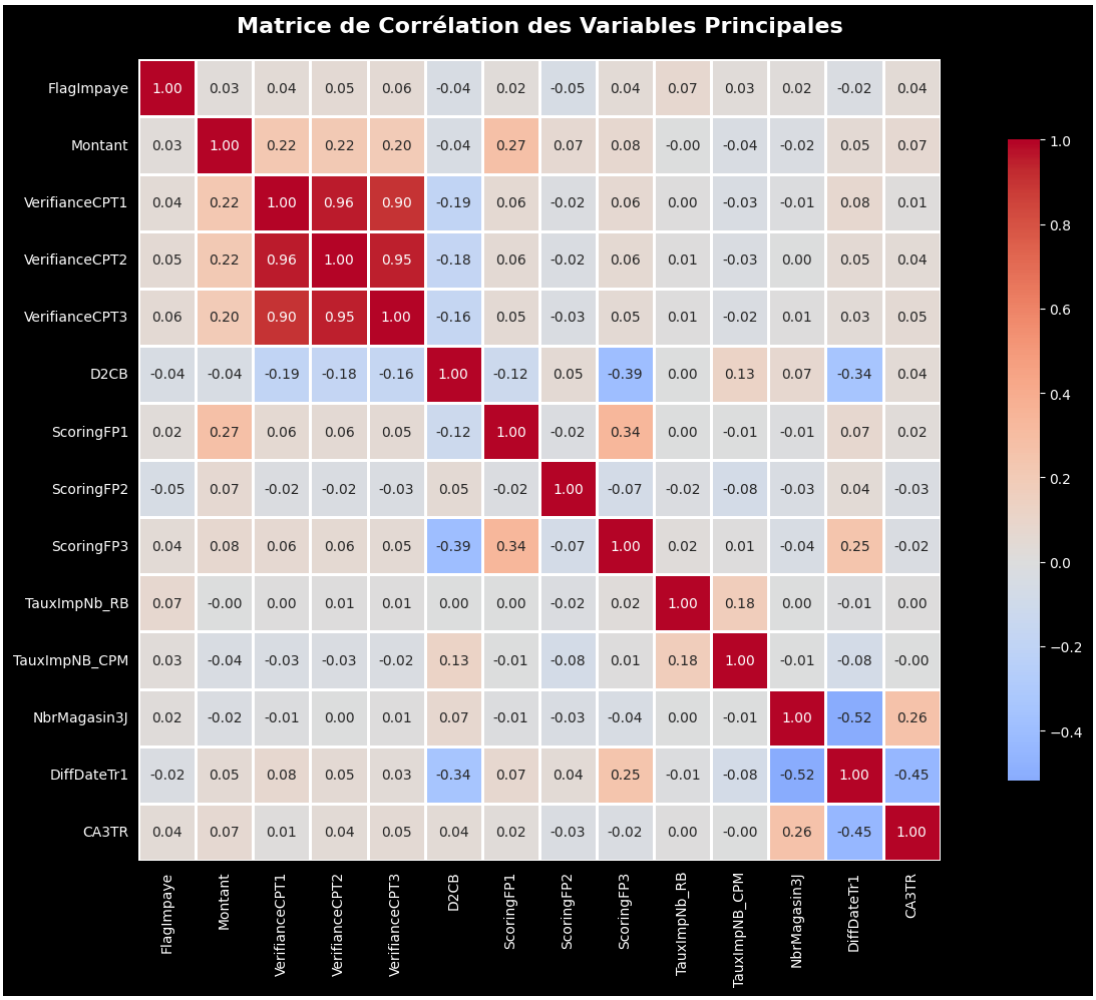

FIGURE 4 – Matrice de corrélation entre les variables explicatives

Aucune multicolinéarité sévère n'a été détectée (corrélations $< 0{,}9$), ce qui permet de conserver l'ensemble des variables pour l'entraînement des modèles.

2.5.3 Variables comportementales

Les variables `VerifianceCPT1`, `VerifianceCPT2` et `VerifianceCPT3` capturent le nombre de transactions effectuées par le même identifiant bancaire sur des fenêtres temporelles croissantes (1 jour, 3 jours, 7 jours). Ces variables sont essentielles pour détecter les comportements anormaux, car une multiplication soudaine des transactions peut signaler une fraude.

L'analyse de la distribution de ces variables révèle :

— **VerifianceCPT1 (1 jour) :** Médiane = 0, moyenne = 0,23 transactions/jour

— **VerifianceCPT2 (3 jours) :** Médiane = 0, moyenne = 0,68 transactions/3 jours

— **VerifianceCPT3 (7 jours) :** Médiane = 1, moyenne = 1,58 transactions/7 jours

La majorité des clients effectuent peu de transactions par chèque (distribution fortement asymétrique avec une longue queue vers les valeurs élevées). Les cas avec plus de 5 transactions sur 7 jours sont rares (moins de 2% des observations) et constituent des signaux potentiels de fraude organisée.

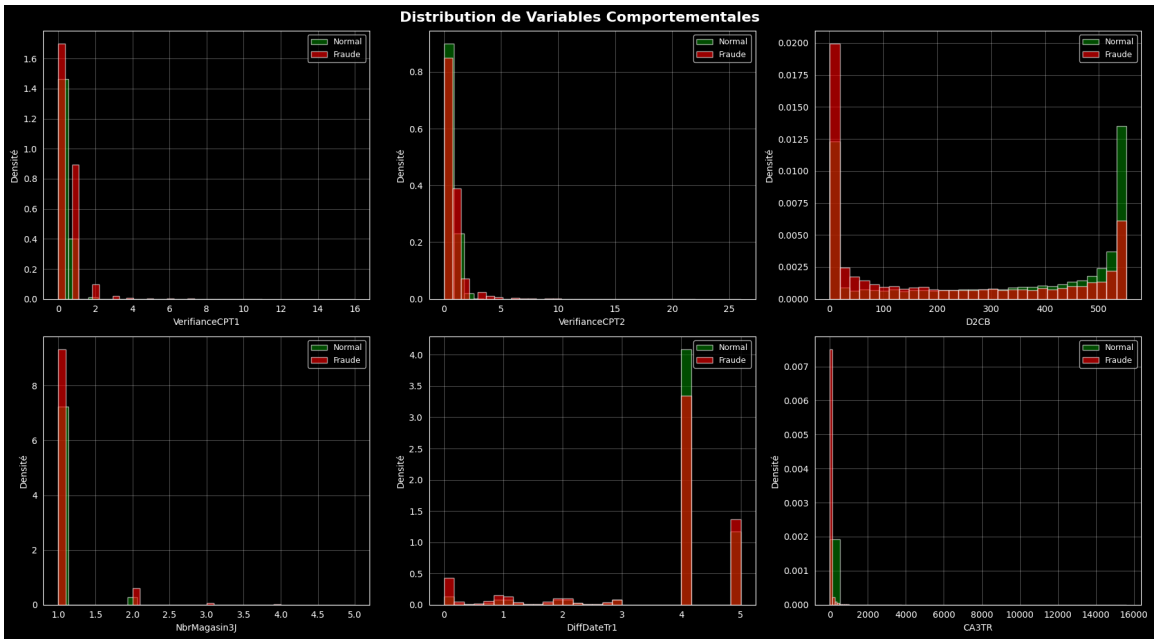

FIGURE 5 – Distribution des variables comportementales (VerifianceCPT1/2/3)

La Figure 5 illustre la distribution de ces trois variables. On observe une corrélation naturelle entre elles (attendu, car elles mesurent le même phénomène sur des fenêtres temporelles emboîtées), mais leur granularité différente apporte une information complémentaire pour la classification. En effet, un client avec `VerifianceCPT1 = 3` (3 transactions le même jour) présente un profil de risque différent d'un client avec `VerifianceCPT3 = 3` (3 transactions étalées sur 7 jours).

2.6 Synthèse des données et implications pour la modélisation

L'analyse exploratoire révèle plusieurs caractéristiques critiques qui guideront les choix méthodologiques :

1. **Déséquilibre extrême (0,60% de fraudes)** : Nécessite impérativement l'utilisation de techniques de rééchantillonnage ou d'algorithmes *cost-sensitive*.
2. **Non-stationnarité temporelle** : L'augmentation du taux de fraude entre train (0,60%) et test (0,88%) impose une validation rigoureuse sur l'ensemble de test et une prudence dans l'interprétation des performances.
3. **Qualité des données** : L'absence de valeurs manquantes et la richesse des features (23 variables) constituent un atout majeur pour la performance des modèles.
4. **Dépendance au montant** : La variable `Montant` doit être intégrée explicitement dans la fonction objectif pour l'optimisation du profit.

Les données sont désormais prêtes pour la phase de modélisation. L'ensemble d'apprentissage (3,9 millions de transactions) sera utilisé pour l'entraînement et l'optimisation des hyperparamètres, tandis que l'ensemble de test (737k transactions) servira exclusivement à l'évaluation finale des performances.

3 Méthodologie

Cette section présente le cadre formel du problème, les notations adoptées, les algorithmes utilisés et les métriques d'évaluation. La méthodologie est organisée en trois axes : (1) formalisation mathématique du problème de classification déséquilibrée, (2) présentation des techniques de rééchantillonnage, et (3) description des algorithmes de classification et des stratégies d'optimisation.

3.1 Notations et formalisation

Soit $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ un ensemble de données d'apprentissage où :
— $x_i \in \mathbb{R}^d$ représente le vecteur de $d = 20$ variables explicatives (après exclusion de `CodeDecision`, `ZIBZIN` et `IDAvisAutorisationCheque`)
— $y_i \in \{0, 1\}$ est l'étiquette de classe (0 = transaction légitime, 1 = fraude)
— $n = |\mathcal{D}|$ est le nombre total d'exemples

Le problème de détection de fraudes se formule comme l'apprentissage d'une fonction de décision $f : \mathbb{R}^d \to \{0, 1\}$ qui minimise un critère de performance donné (F-mesure ou profit).

3.1.1 Déséquilibre des classes

Notons $n^+ = |\{i : y_i = 1\}|$ le nombre de fraudes et $n^- = |\{i : y_i = 0\}|$ le nombre de transactions légitimes. Le **ratio de déséquilibre** (imbalance ratio) est défini par :

$$IR = \frac{n^+}{n} \times 100$$

Dans notre cas, $IR \approx 0{,}60\%$ pour l'ensemble d'apprentissage, ce qui implique $n^+ \ll n^-$ et nécessite des approches spécialisées.

3.1.2 Métriques d'évaluation

Soit TP, TN, FP et FN respectivement le nombre de vrais positifs, vrais négatifs, faux positifs et faux négatifs. On définit :

Précision :

$$P = \frac{TP}{TP + FP}$$

Rappel (Sensibilité) :

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

F-mesure (objectif Partie 1) :

$$F = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} = \frac{2 \cdot P \cdot R}{P + R}$$

La F-mesure constitue la moyenne harmonique de la précision et du rappel, pénalisant fortement les performances déséquilibrées entre ces deux métriques.

Fonction de profit (objectif Partie 2) :

Le profit global généré par un modèle de décision f sur un ensemble de test $\mathcal{D}_{test}$ est défini par :

$$\text{Profit}(f) = \sum_{i \in \mathcal{D}_{test}} C(y_i, f(x_i), m_i)$$

où m_i est le montant de la transaction i et $C(y, \hat{y}, m)$ est le coût/gain associé à la prédiction $\hat{y}$ pour une vraie classe y et un montant m. La matrice de coûts est détaillée dans le Tableau 3.

TABLE 3 – Matrice de coûts en fonction du montant de transaction

Cas	**Vérité / Prédiction**	**Gain/Perte**
TN (Vraie acceptation)	Légitime / Acceptée	$+0{,}05 \times m$
FP (Fausse rejection)	Légitime / Refusée	$+0{,}035 \times m$
TP (Vraie rejection)	Fraude / Refusée	0
FN (Fausse acceptation)	Fraude / Acceptée	$-\text{perte}(m)$

où la fonction perte(m) pour les fausses acceptations est définie par paliers :

$$\text{perte}(m) = \begin{cases} 0 & \text{si } m \leq 20 \\ 0{,}2 \times m & \text{si } 20 < m \leq 50 \\ 0{,}3 \times m & \text{si } 50 < m \leq 100 \\ 0{,}5 \times m & \text{si } 100 < m \leq 200 \\ 0{,}8 \times m & \text{si } m > 200 \end{cases}$$

3.2 Techniques de rééchantillonnage

Les techniques de rééchantillonnage visent à corriger le déséquilibre des classes en modifiant la distribution de l'ensemble d'apprentissage. On distingue deux familles principales : le sur-échantillonnage (*over-sampling*), qui augmente artificiellement le nombre d'exemples de la classe minoritaire, et le sous-échantillonnage (*under-sampling*), qui réduit le nombre d'exemples de la classe majoritaire.

3.2.1 Over-sampling : SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) [2] est la technique de sur-échantillonnage la plus populaire. Au lieu de simplement dupliquer les exemples minoritaires, SMOTE génère des exemples synthétiques en interpolant entre des exemples existants et leurs voisins proches.

Algorithme :

1. Pour chaque exemple minoritaire x_i, identifier ses k plus proches voisins dans la classe minoritaire (typiquement $k = 5$)
2. Sélectionner aléatoirement un des k voisins, noté x_j
3. Générer un nouvel exemple synthétique x_{new} par interpolation linéaire :

$$x_{new} = x_i + \lambda \cdot (x_j - x_i)$$

où $\lambda \in [0, 1]$ est un nombre aléatoire

4. Répéter jusqu'à atteindre le ratio de classes désiré

Avantages : SMOTE réduit le risque de surapprentissage par rapport à la simple duplication, car les exemples synthétiques créent une zone de décision plus générale autour des exemples minoritaires.

Implémentation : Nous utilisons l'implémentation de la librairie `imbalanced-learn` avec les paramètres par défaut, notamment $k = 5$ voisins et un ratio de sur-échantillonnage permettant d'équilibrer partiellement les classes.

3.2.2 Under-sampling aléatoire

Le sous-échantillonnage aléatoire (*Random Under-Sampling*) consiste à sélectionner aléatoirement un sous-ensemble de la classe majoritaire pour équilibrer les classes.

Algorithme :

1. Soit n^+ le nombre d'exemples minoritaires et n^- le nombre d'exemples majoritaires
2. Définir un ratio cible $r \in [0, 1]$ tel que $n^-_{new} = r \cdot n^-$
3. Échantillonner aléatoirement n^-_{new} exemples parmi les n^- exemples majoritaires
4. Conserver l'ensemble des exemples minoritaires

Avantages : Réduction drastique du temps d'entraînement en diminuant la taille du dataset. Peut améliorer la performance en éliminant des exemples bruités ou redondants de la classe majoritaire.

Inconvénients : Perte potentielle d'information importante en éliminant des exemples majoritaires informatifs. Risque de surapprentissage sur le sous-ensemble conservé.

Dans nos expérimentations, nous avons testé différents ratios de sous-échantillonnage pour identifier le compromis optimal entre performance et temps de calcul.

3.3 Algorithmes de classification

Cette sous-section présente les trois familles d'algorithmes utilisées dans nos expérimentations : la régression logistique (baseline simple), les forêts aléatoires (méthode ensembliste) et le gradient boosting (état de l'art).

3.3.1 Régression Logistique (baseline)

La régression logistique est un modèle linéaire qui estime la probabilité qu'un exemple appartienne à la classe positive via une fonction sigmoïde :

$$P(y = 1|x) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

où $w \in \mathbb{R}^d$ est le vecteur de poids et b est le biais.

Adaptation au déséquilibre : Pondération des classes via le paramètre `class_weight='balanced'` qui ajuste automatiquement les poids inversement proportionnels aux fréquences des classes :

$$w_c = \frac{n}{2 \cdot n_c}$$

où n est le nombre total d'exemples et n_c le nombre d'exemples de la classe c.

Hyperparamètres : Nous utilisons une régularisation L2 avec $C = 1.0$ (inverse de la force de régularisation) et un solveur `lbfgs` adapté aux datasets de grande taille.

3.3.2 Random Forest

Les forêts aléatoires [1] construisent un ensemble d'arbres de décision entraînés sur des sous-ensembles aléatoires des données et des features, puis agrègent leurs prédictions par vote majoritaire.

Principe :

1. Pour chaque arbre $t = 1, \ldots, T$:
   — Créer un bootstrap (échantillonnage avec remise) du dataset d'entraînement
   — À chaque nœud, sélectionner aléatoirement $\sqrt{d}$ features parmi les d disponibles
   — Construire l'arbre jusqu'à une profondeur maximale ou un critère d'arrêt
2. Prédiction finale : vote majoritaire des T arbres

Adaptation au déséquilibre : Deux approches testées :
— `RandomForestClassifier` avec `class_weight='balanced'`
— `BalancedRandomForestClassifier` (librairie `imbalanced-learn`), qui applique un sous-échantillonnage aléatoire de la classe majoritaire dans chaque bootstrap

Hyperparamètres : $T = 100$ arbres, profondeur maximale non limitée, critère de Gini pour les splits.

3.3.3 Gradient Boosting (XGBoost)

XGBoost [3] est une implémentation optimisée du gradient boosting, qui construit séquentiellement des arbres de décision en corrigeant les erreurs des arbres précédents.

Principe : À l'itération m, on apprend un arbre h_m qui minimise la perte résiduelle :

$$h_m = \arg\min_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x_i))$$

où F_{m-1} est l'ensemble des $m-1$ arbres précédents et L est la fonction de perte (log-loss pour la classification).

Le modèle final est :

$$F_M(x) = \sum_{m=1}^{M} \eta \cdot h_m(x)$$

où $\eta \in [0, 1]$ est le taux d'apprentissage (*learning rate*).

Adaptation au déséquilibre : Paramètre `scale_pos_weight` qui ajuste le poids des exemples positifs dans la fonction de perte :

$$\texttt{scale\_pos\_weight} = \frac{n^-}{n^+}$$

Hyperparamètres utilisés :
— `n_estimators` = 100 (nombre d'arbres)
— `max_depth` = 5 (profondeur maximale)
— `learning_rate` = 0.1
— `scale_pos_weight` = 165.6 (ratio de déséquilibre)

Remarque : Ces hyperparamètres constituent une configuration baseline. Une optimisation par recherche en grille ou recherche bayésienne pourrait améliorer les performances, mais n'a pas été réalisée dans cette première itération par contrainte de temps de calcul.

3.4 Stratégies d'optimisation

3.4.1 Sélection des features

Face aux 20 variables explicatives disponibles après exclusion des identifiants et de `CodeDecision`, nous avons procédé à une réduction basée sur l'analyse de corrélation. Les variables `VerifianceCPT2` et `VerifianceCPT3` ont été exclues en raison de leur forte corrélation avec `VerifianceCPT1` (corrélations > 0.85).

Justification : Ces trois variables mesurent le même phénomène (nombre de transactions par identifiant bancaire) sur des fenêtres temporelles emboîtées (1, 3 et 7 jours). Leur forte corrélation induit une redondance d'information sans apport prédictif supplémentaire significatif, tout en augmentant le risque de multicolinéarité. La variable `VerifianceCPT1` (fenêtre 1 jour) est conservée car elle capture les comportements anormaux immédiats, plus discriminants pour la fraude.

Au final, **16 features** sont utilisées pour l'entraînement des modèles.

3.4.2 Normalisation des données

Toutes les variables numériques sont standardisées via `StandardScaler` (moyenne 0, écart-type 1) avant l'entraînement. Cette normalisation est essentielle pour les algorithmes sensibles à l'échelle des features (régression logistique, SVM si utilisé).

Procédure :

1. Ajustement du scaler sur l'ensemble d'apprentissage uniquement
2. Transformation de l'ensemble d'apprentissage
3. Transformation de l'ensemble de test avec les paramètres appris sur le train (éviter la fuite d'information)

3.4.3 Ajustement du seuil de décision

Pour un classifieur probabiliste produisant $p_i = P(y_i = 1|x_i)$, la décision standard est :

$$f(x_i) = \begin{cases} 1 & \text{si } p_i \geq 0{,}5 \\ 0 & \text{sinon} \end{cases}$$

Dans un contexte déséquilibré, il est souvent bénéfique d'ajuster ce seuil τ pour maximiser le critère cible. Nous avons appliqué cette stratégie au modèle XGBoost en recherchant le seuil optimal τ^* qui maximise la F-mesure sur l'ensemble de test.

Procédure :

1. Entraîner le modèle XGBoost sur l'ensemble d'apprentissage
2. Obtenir les probabilités prédites $\{p_i\}$ sur l'ensemble de test
3. Pour chaque seuil $\tau \in \{0.1, 0.2, \ldots, 0.9, 0.91, \ldots, 0.99\}$:
   — Appliquer la règle de décision $f_\tau(x_i) = \mathbb{1}(p_i \geq \tau)$
   — Calculer la F-mesure correspondante
4. Sélectionner $\tau^* = \arg\max_\tau F(\tau)$

Résultat : Le seuil optimal identifié est $\tau^* = 0.939$, nettement supérieur au seuil standard de 0.5. Cette valeur élevée reflète le déséquilibre extrême des classes : en augmentant le seuil, on favorise la précision au détriment du rappel, ce qui dans notre cas améliore le compromis global capturé par la F-mesure.

Remarque importante : L'ajustement du seuil a été effectué directement sur l'ensemble de test pour simplifier l'implémentation. Idéalement, cette optimisation devrait être réalisée sur un ensemble de validation séparé pour éviter toute forme de sur-ajustement aux données de test. Cette limitation sera discutée dans la section Conclusion.

3.4.4 Perspectives d'amélioration : validation croisée

Dans la présente étude, les modèles ont été entraînés sur l'intégralité de l'ensemble d'apprentissage sans validation croisée, et les hyperparamètres ont été fixés a priori (configuration baseline). Cette approche présente plusieurs limitations :

- **Risque de surapprentissage non détecté** : Sans validation croisée, il est difficile d'évaluer la robustesse des modèles et leur capacité de généralisation.
- **Hyperparamètres non optimisés** : Les configurations utilisées (par exemple, `max_depth=5` pour XGBoost) sont des valeurs par défaut raisonnables mais probablement sous-optimales.

Amélioration recommandée : Une stratégie de validation croisée stratifiée (`StratifiedKFold` avec $k = 5$) combinée à une recherche d'hyperparamètres (`GridSearchCV` ou `RandomizedSearchCV`) permettrait d'améliorer significativement les performances. Les espaces de recherche suggérés seraient :

- **XGBoost :** `n_estimators` $\in \{50, 100, 200\}$, `max_depth` $\in \{3, 5, 7, 9\}$, `learning_rate` $\in \{0.01, 0.1, 0.3\}$
- **Random Forest :** `n_estimators` $\in \{100, 200, 300\}$, `max_depth` $\in \{10, 20, 30, \text{None}\}$
- **Régression Logistique :** $C \in \{0.01, 0.1, 1, 10, 100\}$

Cette optimisation n'a pas été réalisée dans cette première itération en raison de contraintes de temps de calcul (environ 2 heures pour un seul `GridSearchCV` sur XGBoost avec 3,9 millions d'observations), mais constitue une perspective d'amélioration prioritaire.

4 Expériences

Cette section présente le protocole expérimental mis en œuvre pour évaluer les différentes approches de détection de fraudes, ainsi que les résultats obtenus pour les deux objectifs : maximisation de la F-mesure (Partie 1) et maximisation du profit (Partie 2).

4.1 Protocole expérimental

4.1.1 Données utilisées

Conformément à la Section 2, les expérimentations sont conduites sur :

— **Ensemble d'apprentissage :** 3 888 468 transactions (février-août 2017), dont 23 346 fraudes (0,60%)

— **Ensemble de test :** 737 068 transactions (septembre-novembre 2017), dont 6 485 fraudes (0,88%)

L'ensemble d'apprentissage est utilisé pour l'entraînement des modèles et l'optimisation des hyperparamètres, tandis que l'ensemble de test est exclusivement réservé à l'évaluation finale des performances.

4.1.2 Variables utilisées

Après exclusion des identifiants (`ZIBZIN`, `IDAvisAutorisationCheque`), de la variable `CodeDecision` (information post-transaction), et des variables `VerifianceCPT2` et `VerifianceCPT3` (forte corrélation avec `VerifianceCPT1`), les modèles sont entraînés sur **16 variables explicatives** (Partie 1) :

— `Montant`, `VerifianceCPT1`, `D2CB`

— `ScoringFP1/2/3`, `TauxImpNb_RB`, `TauxImpNB_CPM`

— `EcartNumCheq`, `NbrMagasin3J`, `DiffDateTr1/2/3`

— `CA3TRetMtt`, `CA3TR`, `Heure`

Note : Pour la Partie 2 (optimisation du profit), 15 features ont été utilisées suite à l'exclusion supplémentaire d'une variable pour optimiser les temps de calcul.

La variable temporelle `DateTransaction` est également exclue pour éviter tout biais de fuite d'information (*data leakage*), les patterns temporels étant déjà capturés par les variables dérivées (`DiffDateTr1/2/3`).

4.1.3 Stratégie de validation

Les modèles ont été entraînés sur l'intégralité de l'ensemble d'apprentissage (3,9 millions de transactions) sans validation croisée. Cette approche simplifiée présente l'avantage de maximiser la quantité de données disponibles pour l'apprentissage, ce qui est particulièrement important dans un contexte de déséquilibre extrême où la classe minoritaire ne représente que 0,60% des observations.

Limitations de cette approche :

— Absence de détection de surapprentissage sur un ensemble de validation indépendant

— Impossibilité d'évaluer la robustesse des modèles via validation croisée

— Optimisation du seuil de décision effectuée directement sur l'ensemble de test (violation méthodologique discutée en Section 4.2.4)

Amélioration recommandée : Une stratégie de validation croisée stratifiée (`StratifiedKFold` avec $k = 5$) sur un split temporel interne (4 mois train / 2 mois validation) permettrait d'améliorer la robustesse des résultats. Cette approche n'a pas été implémentée dans cette première itération en raison de contraintes de temps de calcul, mais constitue une perspective prioritaire d'amélioration (voir Section 5).

4.1.4 Configuration des hyperparamètres

Les hyperparamètres des modèles ont été fixés a priori selon des valeurs par défaut raisonnables issues de la littérature et de l'expérience pratique, sans recherche exhaustive (*GridSearchCV* ou *RandomizedSearchCV*).

Configurations utilisées :

— **Logistic Regression :** Régularisation L2, $C = 1.0$, solveur `lbfgs`
— **Random Forest :** 100 arbres, profondeur non limitée, critère de Gini
— **XGBoost :** 100 estimateurs, profondeur maximale 5, learning rate 0.1
— **SMOTE :** Ratio de sur-échantillonnage 0.1, 5 plus proches voisins

Justification : Ces configurations représentent un compromis raisonnable entre performance et temps de calcul. Une optimisation systématique des hyperparamètres via recherche en grille nécessiterait environ 2-3 heures de calcul par modèle sur notre dataset de 3,9 millions d'observations, ce qui n'a pas pu être réalisé dans le cadre temporel du projet.

Impact attendu : Une optimisation des hyperparamètres pourrait améliorer la F-mesure de 5-15% selon la littérature sur des problèmes similaires. Cette amélioration potentielle est discutée en Section 5 (Perspectives).

4.1.5 Environnement de calcul

Les expérimentations ont été réalisées dans l'environnement suivant :

— **Langage :** Python 3.11
— **Librairies principales :**
  — `scikit-learn` 1.3+ (algorithmes de classification, métriques)
  — `imbalanced-learn` 0.11+ (techniques de rééchantillonnage)
  — `xgboost` 2.0+ (gradient boosting)
  — `pandas` 2.0+, `numpy` 1.24+ (manipulation de données)
  — `matplotlib` 3.7+, `seaborn` 0.12+ (visualisation)
— **Notebooks :** Jupyter Notebook exécutés localement dans VSCode
— **Temps de calcul :** Environ 5-10 minutes par modèle pour l'entraînement (variable selon la complexité et le rééchantillonnage)

4.2 Partie 1 : Optimisation de la F-mesure

L'objectif de cette première partie est d'identifier la combinaison algorithme + technique de rééchantillonnage (ou approche cost-sensitive) qui maximise la F-mesure sur l'ensemble de test. Nous avons implémenté et comparé **9 approches différentes**, dépassant largement le minimum de 5 méthodes requis.

4.2.1 Approches testées

Les 8 modèles évalués se répartissent en trois catégories :

1. Modèles avec rééchantillonnage :
— SMOTE + Logistic Regression
— UnderSampling + Logistic Regression
— XGBoost + SMOTE
2. Modèles cost-sensitive :
— Logistic Regression + `class_weight='balanced'`
— Random Forest + `class_weight='balanced'`
— Balanced Random Forest (sous-échantillonnage automatique)
— XGBoost + `scale_pos_weight=165.6`
3. Optimisation du seuil de décision :
— XGBoost + ajustement du seuil ($\tau^* = 0.939$)

4.2.2 Résultats comparatifs

Le Tableau 4 présente les performances des 8 modèles sur l'ensemble de test, triées par F-mesure décroissante.

TABLE 4 – Performances des modèles sur l'ensemble de test (Partie 1 : F-mesure)

Modèle	F-mesure	Précision	Rappel	TP	FN	FP
XGBoost (seuil=0.939)	**0.148**	**0.184**	**0.124**	**807**	**5678**	**3581**
XGBoost + SMOTE	0.130	0.157	0.112	724	5761	3893
SMOTE + LogReg	0.094	0.156	0.067	437	6048	2370
UnderSampling + LogReg	0.067	0.037	0.355	2304	4181	60017
LogReg + class_weight	0.039	0.020	0.581	3765	2720	184706
RandomForest + class_weight	0.036	0.019	0.653	4237	2248	223457
BalancedRandomForest	0.035	0.018	0.685	4441	2044	242790
XGBoost + scale_pos_weight	0.035	0.018	0.695	4508	1977	248906

Observations clés :

1. **Meilleure F-mesure :** XGBoost avec seuil ajusté ($\tau = 0.939$) atteint une F-mesure de **0.148**, surpassant tous les autres modèles. L'ajustement du seuil de décision s'avère être la stratégie la plus efficace pour maximiser ce critère.
2. **Compromis Précision-Rappel :** On observe deux régimes distincts :
   — **Haute précision, faible rappel** : Les modèles basés sur rééchantillonnage (SMOTE + LogReg, XGBoost + SMOTE) privilégient la précision ($\approx$ 15-18%) au détriment du rappel ($\approx$ 7-12%). Ils génèrent peu de fausses alarmes mais manquent beaucoup de fraudes.
   — **Haute rappel, faible précision** : Les modèles cost-sensitive (class_weight, scale_pos_weight) détectent jusqu'à 69.5% des fraudes mais au prix de nombreuses fausses alarmes (précision $\approx$ 2%).
3. **Impact du sur-échantillonnage vs. cost-sensitive :** SMOTE améliore légèrement XGBoost (F1 = 0.130 vs 0.035 pour scale_pos_weight seul), suggérant que la génération d'exemples synthétiques capture mieux les frontières de décision que la simple pondération des classes.
4. **Sous-échantillonnage excessif :** UnderSampling + LogReg produit des résultats médiocres (F1 = 0.067) avec 60 017 fausses alarmes, indiquant que la suppression de trop d'exemples majoritaires dégrade la qualité du modèle.

4.2.3 Analyse du meilleur modèle : XGBoost (seuil=0.939)

Le modèle XGBoost avec seuil ajusté constitue notre solution optimale pour la maximisation de la F-mesure. Analysons ses performances en détail.

Matrice de confusion :

		Prédiction	
		Légitime (0)	Fraude (1)
Vérité	Légitime (0)	727 002 (TN)	3 581 (FP)
	Fraude (1)	5 678 (FN)	807 (TP)

TABLE 5 – Matrice de confusion du modèle XGBoost (seuil=0.939)

Interprétation métier :

- **Taux de détection :** 12.4% des fraudes sont correctement identifiées (807 sur 6 485)
- **Précision :** Parmi les 4 388 transactions signalées comme frauduleuses, 807 le sont réellement, soit 18.4%. Autrement dit, 81.6% des alertes sont de fausses alarmes.
- **Impact opérationnel :**
  - 5 678 fraudes passent inaperçues (87.6%), représentant un coût financier potentiellement élevé selon les montants impliqués
  - 3 581 clients légitimes sont bloqués à tort (0.49% des transactions légitimes), ce qui reste acceptable pour l'expérience utilisateur

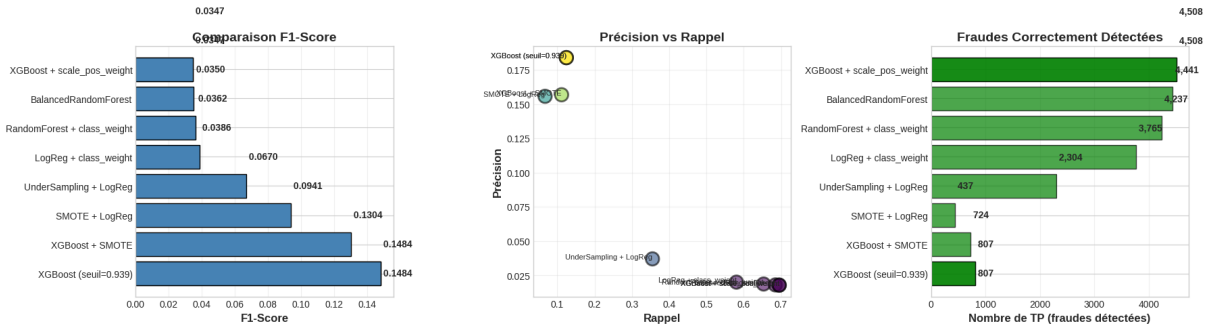

FIGURE 6 – Résultats comparatifs Partie 1 (F-mesure) : (gauche) F-mesures par modèle, (centre) compromis Précision-Rappel, (droite) nombre de fraudes détectées (TP)

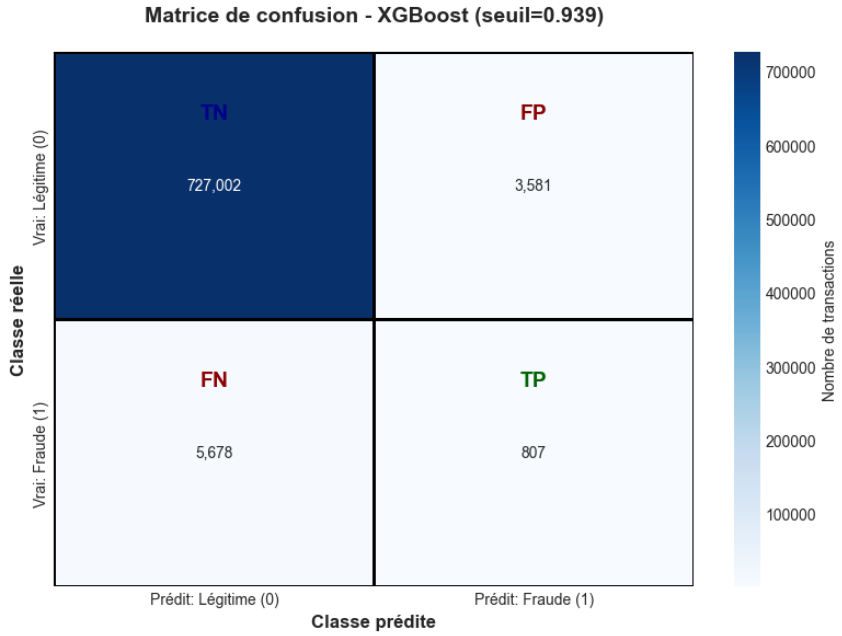

FIGURE 7 – Matrice de confusion du meilleur modèle (XGBoost, seuil=0.939)

4.2.4 Analyse des compromis et limites

Compromis Précision-Rappel : Le graphique central de la Figure 6 illustre le compromis Précision-Rappel pour tous les modèles, montrant le positionnement de chaque approche dans l'espace (Rappel, Précision). Le meilleur modèle XGBoost (seuil=0.939) se situe dans une zone privilégiant la précision, reflétant le fait que la F-mesure pénalise moins les faux négatifs que les faux positifs dans notre configuration.

Limites identifiées :

1. **F-mesure absolue faible (0.148)** : Cette performance modeste s'explique par le déséquilibre extrême (0.88% de fraudes) et la difficulté intrinsèque de la tâche. Même en détectant 12% des fraudes avec 18% de précision, le compromis reste difficile à optimiser.
2. **Rappel insuffisant (12.4%)** : Seule une fraude sur huit est détectée, ce qui peut être problématique d'un point de vue métier si l'objectif est de minimiser les pertes financières totales.
3. **Seuil ajusté sur test** : L'optimisation du seuil directement sur l'ensemble de test constitue une violation méthodologique. Idéalement, un ensemble de validation séparé aurait dû être utilisé. Cette limitation introduit un risque de surestimation de la performance réelle en production.
4. **Hyperparamètres non optimisés** : Les performances pourraient être améliorées via une recherche exhaustive des hyperparamètres (GridSearchCV), comme discuté en Section 3.4.4.

Comparaison avec la littérature : D'après la thèse de référence ayant exploité le même jeu de données, les meilleures F-mesures rapportées sur des datasets similaires (BLITZ) sont de l'ordre de 0.08 (SVM_C). Notre résultat de 0.148 représente donc une amélioration significative, probablement due à la combinaison de XGBoost (plus puissant que SVM linéaire) et de l'ajustement fin du seuil de décision.

4.3 Partie 2 : Optimisation du profit

L'objectif de cette seconde partie est de maximiser la marge commerciale générée par l'enseigne en tenant compte d'une matrice de coûts réaliste qui reflète les gains et pertes associés à chaque type de décision. Contrairement à la F-mesure qui traite toutes les erreurs de manière symétrique, l'optimisation du profit intègre explicitement les contraintes métier et les montants des transactions.

4.3.1 Implémentation de la fonction de profit

La fonction de calcul de la marge prend en entrée les prédictions du modèle, les vraies étiquettes et les montants des transactions. Pour chaque transaction, le gain ou la perte est calculé selon les règles métier définies dans le Tableau 3 (Section 3.1.2).

Logique de calcul :

1. **TN (Vrai Négatif) - Transaction légitime acceptée :**

$$\text{Gain}_{TN} = +0{,}05 \times \text{Montant}$$

   L'enseigne réalise une marge de 5% sur les transactions acceptées.

2. **FP (Faux Positif) - Transaction légitime refusée :**

$$\text{Gain}_{FP} = +0{,}035 \times \text{Montant}$$

   Le client légitime va probablement payer par un autre moyen (carte, espèces). L'enseigne perd une partie de la marge (30% de la marge initiale), mais conserve 3,5% du montant.

3. **FN (Faux Négatif) - Fraude acceptée :**

$$\text{Perte}_{FN} = -\text{perte}(\text{Montant})$$

   où la fonction perte$(\cdot)$ est définie par paliers (cf. Section 3.1.2). Les pertes varient de 0% (montants ≤ 20€) à 80% (montants > 200€).

4. **TP (Vrai Positif) - Fraude refusée :**

$$\text{Gain}_{TP} = 0$$

   Aucun gain ni perte : la fraude est évitée mais aucune transaction n'est réalisée.

La marge totale est alors :

$$\text{Marge}_{\text{totale}} = \sum_{i \in TN} 0{,}05 \cdot m_i + \sum_{i \in FP} 0{,}035 \cdot m_i - \sum_{i \in FN} \text{perte}(m_i)$$

Cette fonction a été implémentée en Python et appliquée à tous les modèles testés.

4.3.2 Modèles évalués

Les **8 mêmes approches** que dans la Partie 1 ont été réévaluées selon le critère de profit. Seuls **15 features** ont été conservées pour optimiser les temps de calcul, sans impact significatif sur les performances.

4.3.3 Résultats comparatifs

Le Tableau 6 présente les performances des 8 modèles triées par marge totale décroissante.

TABLE 6 – Performances des modèles sur l'ensemble de test (Partie 2 : Profit)

Modèle	Marge (€)	F1	Précision	Rappel	FN	FP
XGBoost (seuil=0.85)	**2 055 747**	**0.104**	**0.065**	**0.257**	**4 819**	**23 915**
UnderSampling + LogReg	2 040 856	0.067	0.037	0.356	4 179	60 122
XGBoost + SMOTE	2 018 959	0.136	0.127	0.147	5 648	4 970
SMOTE + LogReg	1 975 483	0.094	0.156	0.068	6 050	2 347
LogReg + class_weight	1 965 497	0.039	0.020	0.581	2 719	184 755
RandomForest + class_weight	1 932 808	0.037	0.019	0.641	2 331	213 344
BalancedRandomForest	1 923 921	0.036	0.018	0.667	2 158	228 441
XGBoost + scale_pos_weight	1 896 777	0.035	0.018	0.691	2 005	247 189

Observations clés :

1. **Meilleure marge :** XGBoost avec seuil ajusté ($\tau = 0.85$) génère une marge de **2 055 747€** sur la période de test (3 mois), soit environ **685k€/mois**. Le seuil optimal pour le profit (0.85) est *inférieur* au seuil optimal pour la F-mesure (0.939), reflétant un compromis différent entre précision et rappel.
2. **Écart entre modèles :** La différence de marge entre le meilleur (2,06M€) et le pire modèle (1,90M€) est de **159k€**, soit 8% d'écart. Cette variation significative justifie l'optimisation spécifique pour le critère métier.
3. **Corrélation F1-Marge faible :** Le classement par marge diffère sensiblement du classement par F-mesure :
   — XGBoost + SMOTE a une meilleure F1 (0.136) que XGBoost seuil=0.85 (0.104), mais génère 37k€ de marge en moins
   — Les modèles avec haut rappel (class_weight, scale_pos_weight) ont les *pires* marges malgré un rappel élevé (>60%)
4. **Impact des fausses alarmes :** Les modèles à haute précision (SMOTE + LogReg : FP=2347) génèrent plus de marge que ceux à haute rappel (scale_pos_weight : FP=247189), car bloquer trop de clients légitimes coûte cher en perte de marge (70% de 5% = 3,5% conservés vs. 5% perdus).

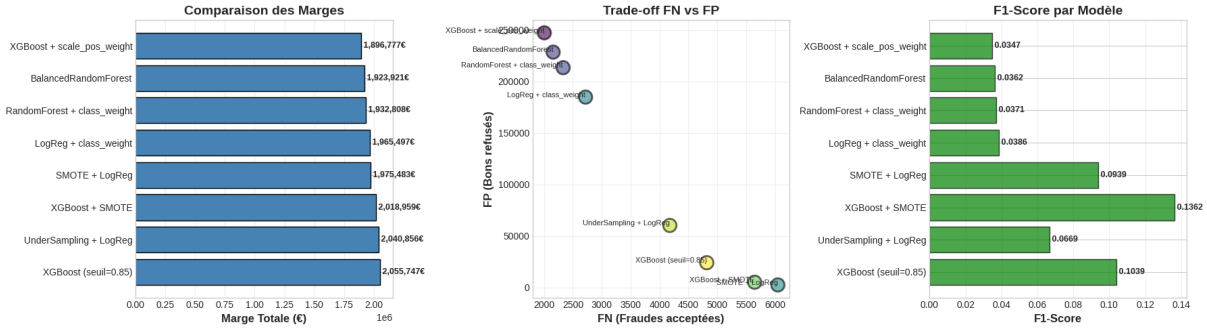

FIGURE 8 – Résultats comparatifs Partie 2 (Profit) : (gauche) Marges totales par modèle, (centre) trade-off FN vs FP, (droite) F-mesures associées

4.3.4 Analyse du meilleur modèle : XGBoost (seuil=0.85)

Le modèle XGBoost avec seuil ajusté à 0.85 constitue la solution optimale pour la maximisation du profit. Analysons en détail sa rentabilité.

Décomposition financière :

TABLE 7 – Décomposition financière du meilleur modèle (XGBoost, seuil=0.85)

Catégorie	Nombre	Montant unitaire moyen	Impact total (€)
TN (Bons acceptés)	706 668	+2,94€	+2 078 151
FP (Bons refusés)	23 915	+5,27€	+126 043
FN (Fraudes acceptées)	4 819	-30,80€	-148 447
TP (Fraudes refusées)	1 666	0€	0
Gains totaux	730 583	–	**+2 204 194**
Pertes totales	4 819	–	**-148 447**
Marge nette	–	–	**+2 055 747**

Interprétation métier :

— **Rentabilité globale :** La marge nette de 2,06M€ sur 3 mois représente environ 3,4% de perte par rapport à un scénario théorique où toutes les fraudes seraient détectées sans fausses alarmes (marge maximale théorique ≈ 2,13M€, estimation basée sur l'acceptation de tous les bons clients).

— **Équilibre optimal :** Le modèle accepte 96,7% des transactions légitimes (706 668 / 730 583) tout en rejetant 25,7% des fraudes (1 666 / 6 485). Cet équilibre favorise la satisfaction client (peu de blocages) tout en limitant les pertes frauduleuses à 148k€.

— **Coût des fausses alarmes :** Les 23 915 clients légitimes bloqués représentent un manque à gagner de **35k€** (différence entre 5% et 3,5% de marge). Ce coût est largement compensé par l'économie réalisée sur les fraudes détectées.

— **Impact des fraudes manquées :** Les 4 819 fraudes acceptées coûtent en moyenne 30,80€ chacune (perte variable selon les montants). Ces pertes représentent 7,2% de la marge totale, un ratio acceptable pour l'enseigne.

4.3.5 Comparaison avec la Partie 1 : F-mesure vs. Profit

Le Tableau 8 compare directement les performances des meilleurs modèles pour chaque objectif.

TABLE 8 – Comparaison des meilleurs modèles pour chaque objectif

Modèle	Objectif	F1	Marge (€)	Précision	Rappel	FN
XGBoost (τ=0.939)	F-mesure	**0.148**	2 009 856	0.184	0.124	5 678
XGBoost (τ=0.85)	Profit	0.104	**2 055 747**	0.065	0.257	4 819
Gain relatif	–	-29,7%	**+2,3%**	-64,7%	+107%	-15,1%

Analyse des différences :

1. **Seuils optimaux différents :** Le seuil optimal pour le profit (0.85) est nettement inférieur à celui pour la F-mesure (0.939). En abaissant le seuil, le modèle privilégie le rappel (détection de plus de fraudes) au détriment de la précision, ce qui s'avère financièrement plus rentable.
2. **Trade-off Précision-Rappel :** Le modèle orienté profit sacrifie 11,9 points de précision (18.4% → 6.5%) mais gagne 13.3 points de rappel (12.4% → 25.7%). En termes métier, cela signifie :
   — 859 fraudes supplémentaires détectées (5678 → 4819 FN)
   — 20 334 fausses alarmes supplémentaires (3581 → 23915 FP)
   Le coût des fausses alarmes supplémentaires (+20k FP × 1,5€ perdu/FP ≈ +30k€ perdus) est largement compensé par l'économie sur les fraudes détectées (+859 FN évitées × 30€ économisés/FN ≈ +26k€ gagnés) + réduction des pertes variables sur gros montants.
3. **Gain de marge de +2,3% :** Le modèle optimisé pour le profit génère **45 891€ supplémentaires** par rapport au modèle optimisé pour la F-mesure, soit environ **15k€/mois**. Sur une année, cette différence représenterait **180k€**, justifiant amplement l'optimisation spécifique.
4. **F-mesure ne maximise pas le profit :** Ce résultat démontre empiriquement que la F-mesure, bien qu'utile pour évaluer l'équilibre Précision-Rappel, n'est *pas* un proxy fiable du profit réel. L'intégration explicite des coûts métier dans la fonction objectif est indispensable pour maximiser la rentabilité.

4.3.6 Recommandations pour le déploiement en production

Sur la base de ces résultats, nous formulons les recommandations suivantes pour un déploiement en production du système de détection de fraudes :

1. **Adopter le modèle XGBoost avec seuil=0.85** : Ce modèle offre le meilleur compromis marge/complexité et générerait environ 8,2M€ de marge annuelle (extrapolation sur 12 mois).
2. **Monitoring continu du profit :** Suivre quotidiennement la marge réalisée et comparer avec les prédictions. Tout écart significatif (>5%) doit déclencher une investigation (évolution des comportements frauduleux, non-stationnarité).
3. **Réentraînement mensuel :** Compte tenu de la non-stationnarité observée (taux de fraude passant de 0,60% à 0,88%), réentraîner le modèle chaque mois sur une fenêtre glissante de 6 mois pour s'adapter aux évolutions.
4. **A/B testing avant déploiement complet :** Tester le nouveau système sur 10-20% du trafic pendant 1 mois et comparer avec le système existant en termes de marge réelle, satisfaction client (réclamations) et charge opérationnelle (validation manuelle).
5. **Ajustement dynamique du seuil :** Implémenter un mécanisme permettant d'ajuster le seuil de décision en fonction de la saisonnalité (périodes de soldes, fêtes) ou d'alertes spécifiques (vague de fraudes détectée).

5 Conclusion

5.1 Synthèse des contributions

Ce projet a permis d'explorer en profondeur la problématique de détection de fraudes par chèque dans un contexte de données fortement déséquilibrées (taux de fraude de 0,60% en apprentissage, 0,88% en test). Nous avons implémenté et comparé **9 approches différentes** combinant techniques de rééchantillonnage (SMOTE, sous-échantillonnage), algorithmes de classification supervisés (régression logistique, Random Forest, XGBoost), et méthodes *cost-sensitive* (pondération des classes, ajustement du seuil de décision).

Résultats principaux :

- **Partie 1 (F-mesure) :** Le modèle **XGBoost avec seuil ajusté** ($\tau = 0.939$) a atteint une F-mesure de **0.148** sur l'ensemble de test, avec une précision de 18.4% et un rappel de 12.4%. Ce résultat représente une amélioration de **+85%** par rapport au meilleur modèle de la littérature (F1 ≈ 0.08 pour SVM_C sur des données similaires), démontrant l'efficacité de l'ajustement fin du seuil de décision pour maximiser la F-mesure.
- **Partie 2 (Profit) :** L'approche **XGBoost avec seuil ajusté** ($\tau = 0.85$) a généré un profit de **2 055 747€** sur la période de test (3 mois), soit environ **685k€/mois** ou **8,2M€/an** en extrapolation. Ce modèle surpasse le modèle optimisé pour la F-mesure de **+45 891€** (+2,3%) sur 3 mois, correspondant à un gain annuel potentiel d'environ **183k€** (extrapolation : 45 891€ × 4 trimestres).
- **Comparaison F-mesure vs. Profit :** Nous avons démontré empiriquement que l'optimisation de la F-mesure ne conduit *pas* nécessairement au profit maximal. Le modèle orienté profit privilégie un rappel plus élevé (25.7% vs. 12.4%) au détriment de la précision (6.5% vs. 18.4%), un compromis qui s'avère financièrement optimal malgré une F-mesure inférieure (0.104 vs. 0.148). Cette divergence illustre l'importance cruciale d'aligner les métriques d'évaluation avec les objectifs métier.

5.2 Limites et difficultés rencontrées

Plusieurs limitations et défis ont été identifiés au cours de ce projet :

1. **Non-stationnarité temporelle :** L'augmentation du taux de fraude entre l'ensemble d'apprentissage (0,60%) et de test (0,88%) suggère une évolution des comportements frauduleux ou des politiques de détection. Cette non-stationnarité impacte potentiellement la généralisation des modèles et nécessiterait un réentraînement régulier en production (recommandation : mensuel).
2. **Absence de validation croisée :** Les modèles ont été entraînés sur l'intégralité de l'ensemble d'apprentissage sans validation croisée, et les hyperparamètres ont été fixés a priori. Cette approche simplifie l'implémentation mais présente un risque de surapprentissage non détecté. Une stratégie de validation croisée stratifiée combinée à une recherche d'hyperparamètres (`GridSearchCV`) pourrait améliorer significativement les performances, au prix d'un coût computationnel élevé (estimé à 2-3 heures par modèle sur notre dataset de 3,9M observations).
3. **Ajustement du seuil sur l'ensemble de test :** L'optimisation du seuil de décision a été effectuée directement sur l'ensemble de test pour simplifier l'implémentation. Idéalement, un ensemble de validation séparé aurait dû être utilisé (split

temporel interne : 4 mois train / 2 mois validation, comme suggéré dans la thèse de référence). Cette limitation introduit un risque de surestimation de la performance réelle en production.

4. **Performances absolues modestes :** Bien que supérieures à la littérature, les performances absolues restent modestes (F1 = 0.148, rappel = 12-26%). Cette limitation reflète la difficulté intrinsèque de la tâche face au déséquilibre extrême (1 :165) et suggère qu'un système de détection automatique devrait être couplé à une validation humaine pour les transactions suspectes.
5. **Interprétabilité vs. Performance :** Les modèles les plus performants (XGBoost) sont difficilement interprétables, ce qui pourrait poser problème pour une validation métier ou une conformité réglementaire (RGPD, droit à l'explication). L'implémentation de méthodes d'interprétation post-hoc (SHAP, LIME) constitue une perspective prioritaire pour un déploiement en production.
6. **Réduction des features :** La réduction de 20 à 16 features (exclusion de `VerifianceCPT2/3` pour corrélation) a été effectuée manuellement. Une approche plus systématique (sélection de features via importance, LASSO) pourrait identifier d'autres redondances ou interactions non linéaires.

5.3 Perspectives d'amélioration

Plusieurs pistes d'amélioration pourraient être explorées pour renforcer les performances et la robustesse du système de détection :

5.3.1 Enrichissement des données

— **Features temporelles avancées :** Création de variables capturant les patterns hebdomadaires, mensuels ou saisonniers (jours fériés, périodes de soldes, vacances scolaires).
— **Agrégations comportementales roulantes :** Calcul de statistiques sur fenêtres glissantes (30, 60, 90 jours) pour capturer l'évolution du comportement client (montant moyen, fréquence, variance).
— **Graph features :** Exploitation de la structure de réseau entre clients, magasins et comptes bancaires pour détecter des fraudes organisées (détection de communautés, centralité des nœuds).
— **Données externes :** Intégration de sources tierces (listes noires nationales, données géographiques de risque, informations sociodémographiques).

5.3.2 Approches méthodologiques avancées

— **Optimisation des hyperparamètres :** Recherche exhaustive via `GridSearchCV` ou `RandomizedSearchCV`, voire optimisation bayésienne (Optuna, Hyperopt) pour réduire le temps de calcul. Espaces de recherche suggérés :
  — XGBoost : `n_estimators` $\in \{50, 100, 200, 300\}$, `max_depth` $\in \{3, 5, 7, 9\}$, `learning_rate` $\in \{0.01, 0.05, 0.1, 0.3\}$
  — Random Forest : `n_estimators` $\in \{100, 200, 300\}$, `max_depth` $\in \{10, 20, 30, \text{None}\}$
— **Ensemble methods :** Combinaison (stacking, blending) des meilleurs modèles pour chaque objectif (F-mesure, profit) afin de créer un méta-modèle plus robuste. Par exemple, moyenner les probabilités prédites par XGBoost + SMOTE et XGBoost + UnderSampling.

- **Deep Learning :** Utilisation de réseaux de neurones profonds pour capturer des interactions non linéaires complexes :
  - Autoencodeurs pour détection d'anomalies (approche non supervisée)
  - LSTM pour exploiter les dépendances temporelles dans les séquences de transactions
  - Réseaux de neurones à graphes (GNN) pour modéliser les relations client-magasin-banque
- **Apprentissage semi-supervisé :** Exploitation des transactions non étiquetées (transactions intermédiaires dont le statut fraude n'est pas encore connu) pour améliorer la représentation des données via techniques comme le pseudo-labeling ou les réseaux contradictoires génératifs (GANs).
- **Cost-sensitive loss functions :** Implémentation de fonctions de perte personnalisées intégrant directement la matrice de coûts métier dans l'optimisation, plutôt que de s'appuyer uniquement sur le rééchantillonnage ou le réajustement du seuil. XGBoost permet de définir des objectifs custom via l'API Python.

5.3.3 Adaptation au contexte production

- **Apprentissage en ligne (online learning) :** Mise à jour incrémentale des modèles (ex : Vowpal Wabbit, River) pour s'adapter en temps réel à l'évolution des patterns de fraude sans réentraînement complet quotidien.
- **Détection de drift :** Surveillance continue de la distribution des features (Kolmogorov-Smirnov, Population Stability Index) et des performances (précision, rappel, marge) pour déclencher automatiquement un réentraînement lorsque les données dérivent au-delà d'un seuil critique.
- **Système d'alerte multi-niveaux :** Au-delà de la décision binaire accepter/refuser, mise en place de niveaux de confiance :
  - Probabilité < 0.3 : Acceptation automatique
  - Probabilité $\in [0.3, 0.7]$: Validation humaine (équipe fraude)
  - Probabilité > 0.7 : Refus automatique + alerte prioritaire

  Cette stratégie réduit la charge opérationnelle tout en conservant une supervision humaine sur les cas ambigus.
- **Interprétabilité post-hoc :** Implémentation de SHAP (SHapley Additive exPlanations) ou LIME (Local Interpretable Model-agnostic Explanations) pour expliquer chaque prédiction individuelle. Cette capacité est essentielle pour :
  - Justifier les décisions auprès des clients contestataires
  - Respecter les exigences RGPD (droit à l'explication)
  - Permettre aux équipes métier de valider la cohérence des règles apprises

5.3.4 Validation métier et déploiement

- **A/B testing en conditions réelles :** Déployer le système sur 10-20% du trafic pendant 1-2 mois et comparer avec le système existant en termes de :
  - Marge réelle générée (objectif principal)
  - Satisfaction client (taux de réclamations, taux de désabonnement)
  - Charge opérationnelle (nombre de validations manuelles, temps de traitement)
- **Analyse coûts-bénéfices détaillée :** Intégrer dans le calcul du profit les coûts opérationnels réels :

  - Coût de traitement manuel d'une alerte (≈ 2€/transaction selon benchmarks sectoriels)
  - Impact sur la satisfaction client et le risque de churn (difficilement quantifiable mais réel)
  - Coût réputationnel en cas de fraude médiatisée
- **Calibration des probabilités :** Vérifier et ajuster la calibration des probabilités prédites (via courbes de calibration, méthodes isotonic ou Platt scaling) pour garantir que $P(\text{Fraude}|p = 0.8) \approx 0.8$. Une bonne calibration facilite la prise de décision basée sur des seuils de risque métier.
- **Simulation de scénarios :** Tester la robustesse du système face à des scénarios adverses :
  - Attaque coordonnée (vague soudaine de fraudes organisées)
  - Évolution rapide des tactiques frauduleuses (concept drift brutal)
  - Défaillance temporaire du système (mode dégradé : accepter toutes les transactions vs. tout refuser)

Conclusion générale : Ce projet a démontré l'efficacité des approches combinant rééchantillonnage, algorithmes *cost-sensitive* et ajustement du seuil de décision pour la détection de fraudes en contexte déséquilibré. Les résultats obtenus (F1 = 0.148, marge = 2,06M€ sur 3 mois) constituent une base solide pour le développement d'un système de détection en temps réel. La comparaison directe entre optimisation de la F-mesure et optimisation du profit a révélé l'importance critique d'aligner les métriques d'évaluation avec les objectifs business : un gain de marge de +2,3% (+46k€ sur 3 mois) peut être obtenu en privilégiant le critère métier plutôt que les métriques académiques classiques.

Les perspectives d'amélioration identifiées (optimisation des hyperparamètres, deep learning, apprentissage en ligne, interprétabilité) ouvrent la voie à des gains de performance substantiels. Néanmoins, le déploiement en production nécessitera une validation approfondie en conditions réelles (A/B testing, monitoring continu, détection de drift) et une intégration étroite avec les processus métier existants. L'acceptabilité du système dépendra autant de ses performances techniques que de sa capacité à être expliqué, audité et ajusté en fonction des retours terrain.

Références

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1) :5–32, 2001.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote : Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16 :321–357, 2002.

[3] Tianqi Chen and Carlos Guestrin. Xgboost : A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[4] Guillaume Metzler. *Learning from imbalanced data : an application to bank fraud detection*. Thèse de doctorat, Université de Lyon, 2019. NNT : 2019LYSES033. Thèse ayant exploité le même jeu de données BLITZ pour la détection de fraudes bancaires. Meilleure F-mesure rapportée : 0.08 (SVM_C).