Détection de Fraudes sur les Transactions par Chèque

Projet de Fouille de Données Massives

Abdourahmane Timera

Master 2 SISE
Université Lumière Lyon 2

Année universitaire 2025-2026

Résumé

Ce projet s'inscrit dans le cadre du Master 2 SISE et porte sur la détection de fraudes dans les transactions par chèque. Les données proviennent du Fichier National des Chèques Irréguliers (FNCI), géré par la Banque de France.

Objectifs :
— **Partie 1** : Maximiser le F-mesure (F1-Score) pour détecter le maximum de fraudes
— **Partie 2** : Maximiser la marge du commerçant en utilisant une matrice de coûts asymétrique

Données :
— 4 646 773 transactions (23 variables)
— Split temporel : entraînement (février-août 2017) / test (septembre-novembre 2017)
— Fort déséquilibre : 0.60% de fraudes dans l'ensemble d'entraînement

Résultats principaux :

Critère	Meilleur modèle	Performance
F1-Score	LightGBM + ADASYN	0.107 (seuil optimisé)
Marge	XGBoost optimisé	+68 181 € vs référence

Conclusion : Le système de détection de fraude permet d'améliorer la marge du commerçant de +3.5% par rapport à une stratégie sans détection.

Mots-clés : Détection de fraude, Classification déséquilibrée, Machine Learning, SMOTE, ADASYN, Random Forest, XGBoost, LightGBM, Matrice de coûts

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte

Le chèque reste un moyen de paiement largement utilisé en France, notamment pour les transactions de montants élevés. Cependant, ce mode de paiement est également sujet à des fraudes, causant des pertes significatives pour les commerçants et les institutions financières.

Le **Fichier National des Chèques Irréguliers (FNCI)**, géré par la Banque de France, recense les incidents de paiement liés aux chèques. Ce fichier constitue une source précieuse d'informations pour développer des systèmes de détection de fraude.

1.2 Problématique

La détection de fraude par chèque présente plusieurs défis :

1. **Déséquilibre des classes** : Les fraudes représentent moins de 1% des transactions, ce qui rend la classification difficile.
2. **Coûts asymétriques** : Les conséquences d'une fraude non détectée (faux négatif) sont bien plus graves que celles d'une fausse alerte (faux positif).
3. **Évolution temporelle** : Les patterns de fraude évoluent dans le temps, nécessitant une validation temporelle des modèles.

1.3 Objectifs du projet

Ce projet vise à développer un système de détection de fraude en deux parties :

1.3.1 Partie 1 : Optimisation du F1-Score

L'objectif est de maximiser le **F-mesure** (F1-Score), qui représente la moyenne harmonique entre la précision et le rappel :

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{1.1}$$

Cette métrique permet d'équilibrer la détection des fraudes (rappel) et la limitation des fausses alertes (précision).

1.3.2 Partie 2 : Optimisation de la Marge

L'objectif est de maximiser la **marge du commerçant** en tenant compte des coûts réels associés à chaque type d'erreur :

— **Vrai Négatif (TN)** : Gain de 5% du montant (marge commerciale)
— **Faux Positif (FP)** : Perte de 70% de la marge (manque à gagner)
— **Faux Négatif (FN)** : Perte variable selon le montant (0% à 80%)
— **Vrai Positif (TP)** : Ni gain ni perte

1.4 Méthodologie

La méthodologie adoptée suit les étapes classiques d'un projet de Machine Learning :

1. **Exploration des données** : Analyse descriptive et visualisation
2. **Prétraitement** : Nettoyage, sélection de variables, gestion du déséquilibre
3. **Modélisation** : Test de plusieurs algorithmes et techniques de rééchantillonnage
4. **Évaluation** : Comparaison des modèles selon les critères définis
5. **Optimisation** : Ajustement des seuils de décision

1.5 Organisation du rapport

Ce rapport est organisé comme suit :
— **Chapitre 2** : Description des données
— **Chapitre 3** : Prétraitement des données
— **Chapitre 4** : Modélisation - Partie 1 (F1-Score)
— **Chapitre 5** : Modélisation - Partie 2 (Marge)
— **Chapitre 6** : Conclusion et perspectives

Chapitre 2

Description des données

2.1 Source des données

Les données utilisées dans ce projet proviennent du **Fichier National des Chèques Irréguliers (FNCI)**, géré par la Banque de France. Ce fichier recense les informations relatives aux transactions par chèque effectuées en France.

2.2 Vue d'ensemble

TABLE 2.1 – Caractéristiques générales du jeu de données

Caractéristique	**Valeur**
Nombre total de transactions	4 646 773
Nombre de variables	23
Variable cible	FlagImpaye (0 = normal, 1 = fraude)
Période couverte	Février 2017 - Novembre 2017

2.3 Description des variables

Le jeu de données contient 23 variables réparties en plusieurs catégories :

2.3.1 Variable cible

— **FlagImpaye** : Indicateur de fraude (1 = impayé/fraude, 0 = normal)

2.3.2 Variables d'identification

— **Identifiant** : Identifiant unique de la transaction
— **CodeDecision** : Code de décision associé à la transaction

2.3.3 Variables temporelles

— **mois** : Mois de la transaction (2 = février à 11 = novembre)

2.3.4 Variables de montant

— **Montant** : Montant de la transaction en euros
— **CA3TR** : Chiffre d'affaires sur 3 mois (transactions)
— **CA3TRetMtt** : Chiffre d'affaires sur 3 mois (montants)

2.3.5 Variables de scoring

— **ScoringFP1** : Score de risque FP1
— **ScoringFP2** : Score de risque FP2
— **ScoringFP3** : Score de risque FP3
— **ScoringFP4** : Score de risque FP4

2.3.6 Variables Verifinance

— **VerifianceCPT1** à **VerifianceCPT5** : Indicateurs de vérification comptable

2.3.7 Autres variables

— **Enseigne** : Identifiant de l'enseigne commerciale
— **Secteur** : Secteur d'activité
— Et autres variables caractérisant la transaction

2.4 Split temporel

Conformément aux bonnes pratiques pour les données temporelles, nous avons effectué un split temporel :

TABLE 2.2 – Répartition train/test

Ensemble	Période	Transactions	Taux de fraude
Entraînement	Février - Août 2017	3 899 362	0.60%
Test	Septembre - Novembre 2017	747 411	0.88%

Remarque importante : Le taux de fraude est plus élevé dans l'ensemble de test (0.88%) que dans l'ensemble d'entraînement (0.60%), ce qui suggère une évolution temporelle des patterns de fraude.

2.5 Déséquilibre des classes

Le problème majeur de ce jeu de données est le **fort déséquilibre des classes** :

TABLE 2.3 – Distribution de la variable cible

Classe	Entraînement	Test
Normal (0)	3 875 940 (99.40%)	740 838 (99.12%)
Fraude (1)	23 422 (0.60%)	6 573 (0.88%)
Ratio	1 :166	1 :114

Ce déséquilibre extrême (1 fraude pour 166 transactions normales) nécessite l'utilisation de techniques spécifiques :

— Rééchantillonnage (SMOTE, ADASYN, sous-échantillonnage)
— Pondération des classes
— Métriques adaptées (F1-Score plutôt qu'accuracy)

2.6 Analyse exploratoire

2.6.1 Distribution des montants

Les montants des transactions présentent une distribution fortement asymétrique :

— Moyenne : 62.31 €
— Médiane : 35.00 €
— Maximum : plus de 10 000 €
— Forte concentration sur les petits montants

2.6.2 Variables les plus discriminantes

L'analyse exploratoire a permis d'identifier les variables les plus discriminantes pour la détection de fraude :

TABLE 2.4 – Variables les plus discriminantes (différence de moyenne fraude vs normal)

Variable	Différence (%)
ScoringFP1	+410%
CA3TR	+290%
ScoringFP2	-284%
ScoringFP3	+200%
VerifianceCPT3	+169%

2.6.3 Corrélations

L'analyse des corrélations a révélé :

— Forte corrélation entre les variables VerifianceCPT
— Corrélation entre Montant et CA3TRetMtt
— Corrélations faibles avec la variable cible (maximum 0.15)

Ces corrélations faibles avec la cible expliquent en partie la difficulté de la tâche de classification.

Chapitre 3

Prétraitement des données

3.1 Vue d'ensemble du prétraitement

Le prétraitement des données a suivi plusieurs étapes essentielles pour préparer les données à la modélisation :

1. Gestion des valeurs manquantes
2. Correction des types de données
3. Suppression des variables non pertinentes
4. Sélection des features
5. Split temporel train/test

3.2 Gestion des valeurs manquantes

L'analyse des valeurs manquantes a révélé :

TABLE 3.1 – Variables avec valeurs manquantes

Variable	**Valeurs manquantes**	**Pourcentage**
CodeDecision	4 590 905	98.80%
Autres variables	0	0%

La variable `CodeDecision` présentant 98.80% de valeurs manquantes a été supprimée car elle n'apporte aucune information exploitable.

3.3 Correction des types de données

Plusieurs variables étaient stockées avec des types incorrects :

TABLE 3.2 – Corrections des types de données

Variable	Type original	Type corrigé
mois	float64	int64
Enseigne	float64	int64
Secteur	float64	int64
VerifianceCPT1-5	float64	int64

3.4 Suppression des variables non pertinentes

Les variables suivantes ont été supprimées :
— **Identifiant** : Variable d'identification unique, non prédictive
— **CodeDecision** : 98.80% de valeurs manquantes
— **mois** : Utilisé uniquement pour le split temporel
— **FlagImpaye** : Variable cible (séparée)

3.5 Sélection des features

Après suppression des variables non pertinentes, **18 features** ont été retenues pour la modélisation :

TABLE 3.3 – Features retenues pour la modélisation

Catégorie	Variables
Montant	Montant, CA3TR, CA3TRetMtt
Scoring	ScoringFP1, ScoringFP2, ScoringFP3, ScoringFP4
Verifinance	VerifianceCPT1, VerifianceCPT2, VerifianceCPT3, VerifianceCPT4, VerifianceCPT5
Autres	Enseigne, Secteur, et autres

3.6 Split temporel

Le split a été réalisé de manière temporelle pour respecter la chronologie des données :

Listing 3.1 – Split temporel

```
# Ensemble d'entrainement : mois 2 a 8 (fevrier - aout)
train_mask = data['mois'] <= 8
X_train = data[train_mask].drop(columns=['FlagImpaye', 'mois'])
y_train = data[train_mask]['FlagImpaye']

# Ensemble de test : mois 9 a 11 (septembre - novembre)
test_mask = data['mois'] > 8
X_test = data[test_mask].drop(columns=['FlagImpaye', 'mois'])
y_test = data[test_mask]['FlagImpaye']
```

Résultat :

— Entraînement : 3 899 362 transactions
— Test : 747 411 transactions

3.7 Analyse des corrélations

L'analyse de la matrice de corrélation a révélé plusieurs points importants :

3.7.1 Multicolinéarité

Forte corrélation entre certaines variables :
— VerifianceCPT1 et VerifianceCPT2 : corrélation > 0.8
— Montant et CA3TRetMtt : corrélation significative
Cette multicolinéarité n'est pas problématique pour les modèles basés sur les arbres (Random Forest, XGBoost, LightGBM) que nous utilisons.

3.7.2 Corrélation avec la cible

Les corrélations avec la variable cible `FlagImpaye` sont faibles :
— Corrélation maximale : environ 0.15
— Cela indique que la tâche de classification est intrinsèquement difficile

3.8 Résumé du prétraitement

TABLE 3.4 – Résumé du prétraitement

Étape	**Résultat**
Variables initiales	23
Variables supprimées	5 (Identifiant, CodeDecision, mois, FlagImpaye, autres)
Features finales	18
Transactions train	3 899 362 (0.60% fraudes)
Transactions test	747 411 (0.88% fraudes)

Les données prétraitées ont été sauvegardées au format pickle pour être utilisées dans les étapes de modélisation.

Chapitre 4

Modélisation - Partie 1 : Optimisation du F1-Score

4.1 Objectif

L'objectif de cette première partie est de maximiser le **F1-Score**, défini comme :

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{4.1}$$

où :
— TP = Vrais Positifs (fraudes correctement détectées)
— FP = Faux Positifs (fausses alertes)
— FN = Faux Négatifs (fraudes manquées)

4.2 Défi du déséquilibre des classes

Avec un ratio de 1 :166 entre fraudes et transactions normales, un classificateur naïf qui prédirait toujours "normal" obtiendrait :
— Accuracy : 99.40%
— F1-Score : 0%
Pour contrer ce déséquilibre, nous avons utilisé plusieurs techniques de rééchantillonnage.

4.3 Techniques de rééchantillonnage

4.3.1 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE génère des échantillons synthétiques de la classe minoritaire en interpolant entre des exemples existants proches dans l'espace des features.
Principe :

1. Pour chaque exemple de la classe minoritaire, identifier les k plus proches voisins
2. Créer des exemples synthétiques sur le segment reliant l'exemple à ses voisins

4.3.2 ADASYN (Adaptive Synthetic Sampling)

ADASYN est une extension de SMOTE qui génère plus d'échantillons synthétiques pour les exemples de la classe minoritaire qui sont plus difficiles à classifier (proches de la frontière de décision).

Avantage : Meilleure adaptation aux zones difficiles de l'espace des features.

4.3.3 Sous-échantillonnage (Random Under-Sampling)

Cette technique réduit le nombre d'exemples de la classe majoritaire pour équilibrer les classes.

Inconvénient : Perte d'information potentielle.

4.3.4 SMOTETomek

Combinaison de SMOTE et de la suppression des liens Tomek (paires d'exemples de classes différentes qui sont plus proches voisins l'un de l'autre).

4.3.5 Pondération des classes (Cost-Sensitive Learning)

Plutôt que de modifier les données, on attribue des poids différents aux classes dans la fonction de perte du modèle.

4.4 Méthodes testées

Nous avons testé 5 méthodes différentes :

TABLE 4.1 – Méthodes testées pour l'optimisation du F1-Score

#	**Méthode**	**Technique**
1	Random Forest Baseline	Aucun rééchantillonnage
2	Random Forest + SMOTE	Sur-échantillonnage synthétique
3	XGBoost Weighted	Pondération des classes
4	LightGBM + ADASYN	Sur-échantillonnage adaptatif
5	Random Forest + Under-Sampling	Sous-échantillonnage

4.5 Résultats

4.5.1 Comparaison des méthodes

TABLE 4.2 – Résultats des 5 méthodes (seuil par défaut = 0.5)

Méthode	F1-Score	Precision	Recall	ROC-AUC	Accuracy
LightGBM + ADASYN	**0.0685**	0.0388	0.2947	0.7219	92.95%
RF + SMOTE	0.0483	0.0257	0.4059	0.7182	85.94%
RF + UnderSampling	0.0465	0.0243	0.5425	0.7459	80.45%
RF Baseline	0.0359	0.6836	0.0184	0.7426	99.13%
XGBoost Weighted	0.0346	0.0178	0.6983	0.7517	65.75%

4.5.2 Analyse des résultats

Observations clés :

1. **LightGBM + ADASYN** obtient le meilleur F1-Score (0.0685), offrant le meilleur compromis entre précision et rappel.
2. **RF Baseline** a une précision très élevée (68.36%) mais un rappel catastrophique (1.84%). Le modèle est trop conservateur.
3. **XGBoost Weighted** a un rappel élevé (69.83%) mais une précision très faible (1.78%), générant trop de fausses alertes.
4. Les techniques de rééchantillonnage (SMOTE, ADASYN) améliorent significativement le F1-Score par rapport au baseline.

4.6 Optimisation du seuil de décision

Par défaut, un classificateur utilise un seuil de 0.5 : si la probabilité prédite est ≥ 0.5, on prédit "fraude".

Pour les données déséquilibrées, ce seuil n'est pas optimal. Nous avons recherché le seuil qui maximise le F1-Score.

4.6.1 Méthode

Pour chaque seuil $s \in [0.01, 0.99]$:

1. Calculer les prédictions : $\hat{y} = 1$ si $P(fraude) \geq s$, sinon $\hat{y} = 0$
2. Calculer le F1-Score correspondant
3. Retenir le seuil donnant le meilleur F1-Score

4.6.2 Résultat

Pour le meilleur modèle (LightGBM + ADASYN) :

TABLE 4.3 – Impact de l'optimisation du seuil

Configuration	Seuil	F1-Score
Par défaut	0.50	0.0685
Optimisé	0.74	**0.1070**
Amélioration		**+56%**

L'optimisation du seuil permet d'améliorer le F1-Score de **56%**, passant de 0.0685 à 0.1070.

4.7 Analyse du meilleur modèle

4.7.1 Performance avec seuil optimisé

TABLE 4.4 – Détail des performances - LightGBM + ADASYN (seuil = 0.74)

Métrique	Valeur
F1-Score	0.1070
Precision	0.0388
Recall	0.2947
ROC-AUC	0.7219

4.7.2 Matrice de confusion

TABLE 4.5 – Matrice de confusion - LightGBM + ADASYN (seuil = 0.74)

	Prédit Normal	Prédit Fraude
Réel Normal	737 094 (TN)	3 744 (FP)
Réel Fraude	5 990 (FN)	583 (TP)

Interprétation :
— **Fraudes détectées** : 583 sur 6 573 (8.9%)
— **Fraudes manquées** : 5 990 sur 6 573 (91.1%)
— **Fausses alertes** : 3 744 (0.51% des transactions normales)

4.8 Importance des variables

Les variables les plus importantes pour la détection de fraude sont :

1. **ScoringFP1** : Score de risque principal
2. **CA3TR** : Chiffre d'affaires sur 3 mois
3. **ScoringFP2** : Score de risque secondaire
4. **Montant** : Montant de la transaction
5. **VerifianceCPT3** : Indicateur de vérification

4.9 Conclusion de la Partie 1

— Le meilleur modèle pour maximiser le F1-Score est **LightGBM + ADASYN**.
— Le F1-Score obtenu est de **0.107** avec un seuil optimisé de 0.74.
— La technique ADASYN est plus efficace que SMOTE car elle génère plus d'exemples dans les zones difficiles.
— L'optimisation du seuil est cruciale pour les données déséquilibrées (+56% de F1-Score).
— Malgré les optimisations, seulement 8.9% des fraudes sont détectées, ce qui s'explique par les faibles corrélations entre les features et la cible.

Chapitre 5

Modélisation - Partie 2 : Optimisation de la Marge

5.1 Objectif

L'objectif de cette seconde partie est de maximiser la **marge du commerçant** en tenant compte des coûts réels associés à chaque type de décision.

Contrairement à la Partie 1 où toutes les erreurs étaient traitées de manière égale, nous utilisons ici une **matrice de coûts asymétrique** qui reflète la réalité économique.

5.2 Matrice de coûts

Pour une transaction de montant m :

TABLE 5.1 – Matrice de coûts

Situation	**Description**	**Gain/Perte**
TN (Vrai Négatif)	Client normal accepté	$+5\% \times m$
FP (Faux Positif)	Client normal refusé	$-70\% \times 5\% \times m$
TP (Vrai Positif)	Fraude détectée	0
FN (Faux Négatif)	Fraude manquée	Variable selon m

5.2.1 Détail des coûts FN (Faux Négatifs)

La perte associée à une fraude non détectée dépend du montant de la transaction :

TABLE 5.2 – Perte FN selon le montant

Tranche de montant	**Perte**
$m \leq 20$ €	0%
$20 < m \leq 50$ €	20% de m
$50 < m \leq 100$ €	30% de m
$100 < m \leq 200$ €	50% de m
$m > 200$ €	80% de m

Interprétation : Les fraudes sur les petits montants ($\leq$ 20 €) sont couvertes par l'assurance, donc sans perte. Plus le montant augmente, plus la perte est importante.

5.3 Calcul de la marge

La marge totale est calculée comme la somme des gains/pertes sur toutes les transactions :

$$Marge_{totale} = \sum_{i=1}^{n} gain_i(y_i, \hat{y}_i, m_i) \tag{5.1}$$

où $gain_i$ est déterminé par la matrice de coûts en fonction de :
— y_i : Vraie étiquette (0 = normal, 1 = fraude)
— $\hat{y}_i$: Prédiction du modèle
— m_i : Montant de la transaction

5.4 Référence : Sans détection de fraude

Avant d'évaluer les modèles, nous calculons la marge de référence si le commerçant acceptait **toutes** les transactions sans système de détection :

TABLE 5.3 – Marge de référence (tout accepter)

Composante	**Montant**
Gain TN (clients normaux)	+2 294 459 €
Perte FP (fausses alertes)	0 €
Perte FN (fraudes manquées)	-352 608 €
Marge totale	**1 941 852 €**

Cette valeur sert de **point de comparaison** : un système de détection n'est utile que s'il améliore cette marge.

5.5 Méthodes testées

Nous avons testé plusieurs approches :

1. **LightGBM + ADASYN** (meilleur modèle de la Partie 1)
2. **LightGBM + ADASYN avec seuil optimisé** pour la marge
3. **XGBoost avec différents poids de classe**
4. **Random Forest + SMOTE avec seuil optimisé**
5. **Seuil adaptatif** (seuils différents selon le montant)
6. **Ensemble** (moyenne des probabilités de plusieurs modèles)

5.6 Optimisation du seuil pour la marge

Contrairement à la Partie 1 où on maximisait le F1-Score, ici on cherche le seuil qui maximise la marge en euros.

Méthode : Pour chaque seuil $s \in [0.01, 0.99]$, calculer la marge totale et retenir le seuil donnant la meilleure marge.

5.7 Stratégie de seuil adaptatif

L'idée est d'utiliser des seuils différents selon le montant de la transaction :

— Pour les petits montants : seuil élevé (moins prudent)

— Pour les gros montants : seuil bas (plus prudent)

Justification : Le coût d'une fraude manquée est beaucoup plus élevé pour les gros montants, donc il vaut mieux être plus prudent.

5.8 Résultats

TABLE 5.4 – Comparaison des marges par méthode

Méthode	Marge (€)	Gain TN	Perte FP	Perte FN
XGBoost optimisé	**2 010 033**	2 269 681	-17 345	-242 304
Ensemble	1 998 133	2 277 594	-11 806	-267 656
LightGBM (seuil opt)	1 988 509	2 273 072	-14 971	-269 591
RF + SMOTE optimisé	1 978 646	2 289 693	-3 337	-307 710
Référence	**1 941 852**	2 294 459	0	-352 608
Seuil adaptatif	1 665 461	2 000 923	-205 476	-129 986
LightGBM (seuil 0.5)	1 641 927	1 994 041	-210 293	-141 821

5.9 Analyse des résultats

5.9.1 Meilleur modèle : XGBoost optimisé

TABLE 5.5 – Performance du meilleur modèle

Métrique	Valeur
Marge totale	2 010 033 €
Amélioration vs référence	+68 181 € (+3.5%)
Fraudes détectées (TP)	628 sur 6 573 (9.6%)
Fraudes manquées (FN)	5 945
Fausses alertes (FP)	Perte de 17 345 €

5.9.2 Observations clés

1. **Le système de détection améliore la marge** : +68 181 € par rapport à la stratégie "tout accepter".
2. **Le seuil par défaut (0.5) fait perdre de l'argent** : LightGBM avec seuil 0.5 donne une marge de 1 641 927 €, inférieure à la référence (1 941 852 €).
3. **L'optimisation du seuil est cruciale** : Le même modèle avec un seuil optimisé passe de 1 641 927 € à 1 988 509 €.
4. **Le seuil adaptatif n'est pas optimal** : Bien qu'il détecte plus de fraudes (1 349 TP), il génère trop de fausses alertes (-205 476 €).
5. **L'équilibre est clé** : Le meilleur modèle (XGBoost) trouve le bon compromis entre :
   — Réduire les pertes FN : de 352 608 € à 242 304 € (-110 304 €)
   — Limiter les pertes FP : seulement 17 345 € de manque à gagner

5.9.3 Comparaison avec la Partie 1

TABLE 5.6 – Comparaison des objectifs

Critère	**Partie 1 (F1)**	**Partie 2 (Marge)**
Meilleur modèle	LightGBM + ADASYN	XGBoost optimisé
Seuil optimal	0.74	Variable
Fraudes détectées	583 (8.9%)	628 (9.6%)
Objectif atteint	F1 = 0.107	+68 181 €

5.10 Conclusion de la Partie 2

— Le système de détection de fraude **améliore la marge** du commerçant de **+68 181 €** (+3.5%).
— Le meilleur modèle est **XGBoost avec optimisation du seuil**.
— L'optimisation du seuil est **encore plus importante** que dans la Partie 1 : un mauvais seuil peut faire perdre de l'argent par rapport à l'absence de détection.
— La stratégie de seuil adaptatif, bien qu'intuitive, n'est pas optimale dans notre cas car elle génère trop de fausses alertes.
— Le gain de 68 181 € justifie économiquement la mise en place d'un système de détection de fraude.

Chapitre 6

Conclusion

6.1 Synthèse des résultats

Ce projet avait pour objectif de développer un système de détection de fraudes sur les transactions par chèque, avec deux critères d'optimisation différents.

6.1.1 Partie 1 : Optimisation du F1-Score

TABLE 6.1 – Résumé Partie 1

Élément	**Résultat**
Meilleur modèle	LightGBM + ADASYN
F1-Score (seuil par défaut)	0.0685
F1-Score (seuil optimisé = 0.74)	0.1070
Fraudes détectées	583 / 6 573 (8.9%)
Amélioration par optimisation du seuil	+56%

6.1.2 Partie 2 : Optimisation de la Marge

TABLE 6.2 – Résumé Partie 2

Élément	**Résultat**
Meilleur modèle	XGBoost optimisé
Marge obtenue	2 010 033 €
Marge de référence (sans détection)	1 941 852 €
Amélioration	+68 181 € (+3.5%)
Fraudes détectées	628 / 6 573 (9.6%)

6.2 Enseignements clés

6.2.1 Sur le déséquilibre des classes

Le fort déséquilibre des classes (1 :166) constitue le défi majeur de ce projet. Les techniques de rééchantillonnage (SMOTE, ADASYN) et l'optimisation du seuil de décision sont essentielles pour obtenir des performances acceptables.

6.2.2 Sur l'optimisation du seuil

L'optimisation du seuil de décision s'est révélée **cruciale** dans les deux parties :
— Partie 1 : +56% de F1-Score
— Partie 2 : Différence entre gain et perte par rapport à la référence
Le seuil par défaut de 0.5 n'est **jamais optimal** pour les données déséquilibrées.

6.2.3 Sur les objectifs différents

Les deux parties illustrent l'importance de définir clairement l'objectif métier :
— Le F1-Score traite toutes les erreurs de manière égale
— La marge prend en compte les coûts réels, qui varient selon le montant
Le meilleur modèle pour le F1-Score (LightGBM + ADASYN) n'est pas le meilleur pour la marge (XGBoost optimisé).

6.3 Limites du projet

6.3.1 Taux de détection

Malgré les optimisations, le taux de détection reste faible :
— Partie 1 : 8.9% des fraudes détectées
— Partie 2 : 9.6% des fraudes détectées
Cela s'explique par les corrélations faibles entre les features et la variable cible (maximum 0.15).

6.3.2 Drift temporel

Le taux de fraude est plus élevé dans l'ensemble de test (0.88%) que dans l'ensemble d'entraînement (0.60%), suggérant une évolution des patterns de fraude dans le temps. Un système en production nécessiterait un réentraînement régulier.

6.3.3 Données limitées

Certaines informations potentiellement utiles ne sont pas disponibles :
— Historique du client
— Informations géographiques
— Comportement temporel (heure, jour de la semaine)

6.4 Perspectives

6.4.1 Améliorations techniques

1. **Feature engineering** : Créer de nouvelles variables (ratios, agrégations temporelles)
2. **Modèles plus complexes** : Réseaux de neurones, autoencoders pour la détection d'anomalies
3. **Optimisation bayésienne** : Pour le tuning des hyperparamètres
4. **Validation croisée temporelle** : Pour une évaluation plus robuste

6.4.2 Déploiement

1. **Système temps réel** : Intégration dans le processus de validation des chèques
2. **Monitoring** : Surveillance des performances et détection du drift
3. **Réentraînement automatique** : Mise à jour périodique du modèle
4. **Interface utilisateur** : Dashboard pour les analystes fraude

6.5 Conclusion finale

Ce projet démontre la faisabilité d'un système de détection de fraudes par chèque basé sur le Machine Learning. Le système développé permet d'améliorer la marge du commerçant de **+3.5%** (+68 181 €) par rapport à une stratégie sans détection.

Les principales leçons tirées sont :

1. L'importance des techniques de gestion du déséquilibre des classes
2. Le rôle crucial de l'optimisation du seuil de décision
3. La nécessité d'aligner le critère d'optimisation avec l'objectif métier

Bien que les performances de détection restent modestes (environ 10% des fraudes détectées), le gain économique justifie la mise en place d'un tel système.

Annexe A

Annexes

A.1 Environnement technique

A.1.1 Langages et outils

— **Python** 3.9+
— **Jupyter Notebook** pour l'analyse exploratoire
— **Visual Studio Code** comme IDE

A.1.2 Bibliothèques Python

TABLE A.1 – Bibliothèques utilisées

Bibliothèque	Usage
pandas	Manipulation des données
numpy	Calculs numériques
scikit-learn	Modèles ML et métriques
imbalanced-learn	Techniques de rééchantillonnage
xgboost	Modèle XGBoost
lightgbm	Modèle LightGBM
matplotlib	Visualisation
seaborn	Visualisation statistique

A.2 Structure du projet

```
fraud-detection-project/
|-- config/
|   `-- config.py
|-- data/
|   |-- raw/
|   `-- processed/
|-- notebooks/
|   |-- 01_exploration.ipynb
|   |-- 02_preprocessing.ipynb
```

```
|   |-- 03_modeling.ipynb
|   `-- 04_margin_optimization.ipynb
|-- rapport_final/
|   |-- main.tex
|   |-- sections/
|   `-- figures/
|-- reports/
|   `-- figures/
|-- src/
|   |-- data_loader.py
|   |-- preprocessing.py
|   `-- evaluation.py
|-- models/
|-- requirements.txt
`-- README.md
```

A.3 Métriques d'évaluation

A.3.1 Définitions

- **TP (True Positive)** : Fraude correctement détectée
- **TN (True Negative)** : Transaction normale correctement classée
- **FP (False Positive)** : Fausse alerte (transaction normale classée fraude)
- **FN (False Negative)** : Fraude manquée (fraude classée normale)

A.3.2 Formules

$$Precision = \frac{TP}{TP + FP} \tag{A.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{A.2}$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{A.3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{A.4}$$

$$Specificity = \frac{TN}{TN + FP} \tag{A.5}$$

A.4 Algorithmes de rééchantillonnage

A.4.1 SMOTE

Synthetic Minority Over-sampling Technique
Pour chaque exemple x_i de la classe minoritaire :

1. Trouver les k plus proches voisins de x_i dans la classe minoritaire
2. Sélectionner aléatoirement un voisin x_{nn}
3. Créer un exemple synthétique : $x_{new} = x_i + \lambda \times (x_{nn} - x_i)$ où $\lambda \in [0, 1]$

A.4.2 ADASYN

Adaptive Synthetic Sampling

Extension de SMOTE qui génère plus d'exemples synthétiques pour les exemples difficiles à classifier (proches de la frontière de décision).

Le nombre d'exemples synthétiques pour chaque x_i est proportionnel à :

$$r_i = \frac{\Delta_i / k}{\sum_j \Delta_j / k} \tag{A.6}$$

où Δ_i est le nombre de voisins de classe différente parmi les k plus proches voisins.

A.5 Code des fonctions principales

A.5.1 Calcul de la marge

Listing A.1 – Fonction de calcul de la marge

```
def compute_fn_loss(montant):
    """Perte pour un Faux Negatif selon le montant."""
    if montant <= 20:
        return 0
    elif montant <= 50:
        return 0.2 * montant
    elif montant <= 100:
        return 0.3 * montant
    elif montant <= 200:
        return 0.5 * montant
    else:
        return 0.8 * montant

def compute_transaction_margin(y_true, y_pred, montant):
    """Marge pour une transaction."""
    marge_base = 0.05 * montant

    if y_true == 0 and y_pred == 0:  # TN
        return marge_base
    elif y_true == 0 and y_pred == 1:  # FP
        return -0.70 * marge_base
    elif y_true == 1 and y_pred == 0:  # FN
        return -compute_fn_loss(montant)
    else:  # TP
        return 0
```

A.5.2 Optimisation du seuil

Listing A.2 – Recherche du seuil optimal

```
def find_best_threshold_for_margin(y_true, y_proba, montants):
    """Trouve le seuil maximisant la marge."""
    thresholds = np.arange(0.01, 0.99, 0.01)
    margins = []

    for thresh in thresholds:
        y_pred = (y_proba >= thresh).astype(int)
        margin = compute_total_margin(y_true, y_pred, montants)
        margins.append(margin['total_margin'])

    best_idx = np.argmax(margins)
    return thresholds[best_idx], margins[best_idx]
```

A.6 Tableaux de résultats détaillés

A.6.1 Partie 1 : Tous les modèles

Table A.2 – Résultats complets Partie 1

Modèle	F1	Prec.	Recall	AUC	Acc.
LightGBM + ADASYN	0.0685	0.0388	0.2947	0.7219	92.95%
RF + SMOTE	0.0483	0.0257	0.4059	0.7182	85.94%
RF + UnderSampling	0.0465	0.0243	0.5425	0.7459	80.45%
RF Baseline	0.0359	0.6836	0.0184	0.7426	99.13%
XGBoost Weighted	0.0346	0.0178	0.6983	0.7517	65.75%

A.6.2 Partie 2 : Tous les modèles

Table A.3 – Résultats complets Partie 2

Modèle	Marge (€)	TP	FN	vs Réf.
XGBoost optimisé	2 010 033	628	5 945	+68 181 €
Ensemble	1 998 133	440	6 133	+56 282 €
LightGBM (seuil opt)	1 988 509	488	6 085	+46 658 €
RF + SMOTE opt	1 978 646	179	6 394	+36 795 €
Référence	1 941 852	0	6 573	-
Seuil adaptatif	1 665 461	1 349	5 224	-276 391 €
LightGBM (seuil 0.5)	1 641 927	1 937	4 636	-299 925 €