RAPPORT DE PROJET D'ÉTUDES

Thème : Cybersécurité & Finance

Fraude bancaire & IA

"Détection automatique des transactions bancaires frauduleuses grâce au Machine Learning"

Présenté par :
LO Adja Maguette
Mastère in Artificial Intelligence and Management / IAMDA
Promotion 2024/2025

Sommaire

INTRODUCTION

La fraude bancaire représente aujourd'hui l'un des défis majeurs pour les institutions financières. Avec l'essor des transactions électroniques et des paiements en ligne, les fraudeurs utilisent des techniques de plus en plus sophistiquées pour contourner les systèmes de contrôle traditionnels. Selon les chiffres de la Banque de France, plusieurs centaines de millions d'euros sont perdus chaque année à cause de fraudes liées aux cartes bancaires et aux virements. Au-delà de l'impact financier, ces pratiques portent atteinte à la confiance des clients et à la réputation des établissements bancaires.

Face à cette menace croissante, les systèmes de détection fondés sur des règles figées (comme le blocage d'un paiement au-delà d'un certain montant) montrent rapidement leurs limites. Ces mécanismes génèrent souvent trop de faux positifs, bloquant des transactions légitimes ou, au contraire, laissent passer des fraudes complexes et difficilement identifiables.

L'intelligence artificielle et le Machine Learning offrent aujourd'hui des solutions plus performantes. Grâce à leur capacité à analyser de grands volumes de données et à identifier des schémas inhabituels en temps réel, ces technologies permettent d'améliorer considérablement la précision de la détection des fraudes.

Ce projet vise à concevoir et expérimenter un système basé sur le Machine Learning capable de détecter automatiquement les transactions bancaires frauduleuses. À partir d'un jeu de données de transactions anonymisées, nous mettrons en place un pipeline complet intégrant le prétraitement des données, l'entraînement de modèles prédictifs et l'évaluation des performances. L'objectif est d'identifier les approches les plus efficaces pour minimiser les fraudes tout en réduisant les faux positifs.

I. PRESENTATION GENERALE DU CONTEXTE

1. Panorama macro (sociétal et économique)

La digitalisation des paiements (e-commerce, portefeuilles mobiles, virements instantanés) a fortement augmenté le volume et la vitesse des transactions. Cette accélération crée de la valeur pour l'économie (fluidité des échanges, accessibilité), mais élargit la surface d'attaque : hameçonnage, prise de contrôle de comptes ("account takeover"), fraude à la carte non présente, arnaques au virement.
Pour les institutions financières, la fraude représente un coût direct (remboursements, chargebacks), un coût opérationnel (enquêtes, traitement des alertes) et un risque réputationnel (perte de confiance des clients). À l'échelle sociétale, la lutte contre la fraude soutient la sécurité des paiements, l'inclusion financière et la résilience de l'économie numérique.

2. Écosystème & acteurs

- Clients/usagers : initient des paiements (carte, virement, wallet).
- Attentes clés : sécurité, simplicité, absence de faux positifs (transactions légitimes refusées).
- Banques émettrices : portent le risque de la fraude carte/virement ; opèrent des moteurs de scoring, KYC/AML, et des règles 3-D Secure/SCA.
- Acquéreurs & PSP (Payment Service Providers) : sécurisent l'acceptation côté commerçant, fournissent des flux de données (device, IP, MCC, 3DS).
- Réseaux de cartes (CB, Visa, Mastercard) : normalisent les échanges, gèrent autorisations/chargebacks, partagent des signaux de risque.
- Commerçants & marketplaces : souhaitent maximiser l'acceptation tout en limitant chargebacks et coûts d'authentification.
- FinTech/RegTech : fournissent des outils anti-fraude (device fingerprinting, scoring, graphes).
- Régulateurs & autorités (exemple en Europe : BCE/ABE, autorités nationales, CNIL) : encadrent protection des données (RGPD), authentification forte du client (SCA/PSD2), reporting et standards.
- Fraudeurs/organisations criminelles : s'industrialisent (phishing, social engineering, malwares, mules), adaptent rapidement leurs tactiques.

3. Chaîne de valeur du paiement & flux de données

Du clic à la décision (vue simplifiée) :

1. Initiation (client) → 2) Passerelle/PSP (collecte signaux : IP, device, navigateur, MCC, géolocalisation, montant, panier) → 3) Moteur de décision

(règles + ML) → 4) Authentification (SCA/3DS si nécessaire) → 5) Autorisation (émetteur) → 6) Règlement → 7) Surveillance post-transaction (retours, chargebacks).

Données utiles à la détection :

- Transaction : montant, devise, heure, pays, canal (CNP/présentiel), marchand/MCC.
- Contexte technique : adresse IP, empreinte appareil, OS/navigateur, cookies, adresse de livraison.
- Historique client : fréquence d'achat, panier moyen, anomalies récentes, tentatives échouées.
- Labels : confirmations de fraude (chargeback, plainte client), nécessaires à l'entraînement supervisé.

Ces données alimentent des pipelines de feature engineering, de scoring temps réel (latence souvent <100 ms) et de surveillance du drift (évolution des comportements).

4. Typologies de fraude (principales)

- Carte non présente : utilisation de numéros compromis en ligne.
- Prise de contrôle de compte : vol d'identifiants, changement d'IBAN/adresse, virements sortants.
- Fraudes au virement : arnaque au président/BEC, escroqueries à l'investissement, "authorized push payment scams".
- Social engineering : manipulation directe du client pour contourner la SCA.
- Friendly fraud : client conteste un achat pourtant autorisé (post-achat).
- Réseaux de mules : blanchiment en chaîne via multiples comptes.

Chaque typologie impose des signaux et modèles spécifiques (temps réel vs post-autorisation, graphes pour liens entre comptes, etc.).

5. Contraintes et cadres (techniques, réglementaires, éthiques)

- Réglementation & conformité :
- RGPD (minimisation, finalité, sécurité, droits des personnes) ;
- PSD2/SCA (authentification forte, exemptions conditionnelles) ;
- exigences d'auditabilité et de traçabilité des décisions.
- Exigences opérationnelles :
- Latence faible pour ne pas dégrader l'expérience ;
- Explicabilité des décisions (priorité aux modèles interprétables ou à l'explication locale) ;
- Coût des faux positifs (perte de revenus, frustration client) vs coût des faux négatifs (pertes fraudes).

- Sécurité & gouvernance des données : chiffrement, contrôle d'accès, qualité des données, gestion des biais, suivi du drift.

6. Enjeux business & sociétaux

- Business : réduire pertes et coûts de traitement, maintenir un haut taux d'acceptation (ne pas bloquer le bon client), optimiser les parcours (exemptions SCA pertinentes).
- Sociétal : renforcer la confiance dans les paiements numériques, protéger les populations vulnérables des arnaques, soutenir l'innovation (e-commerce, économie des plateformes) sans sacrifier la sécurité.

7. Positionnement du projet

Dans ce projet, nous simulons une brique clé de cet écosystème : un moteur de détection supervisé entraîné sur des transactions anonymisées
Le travail porte sur :

- la compréhension fonctionnelle de la fraude et de l'écosystème,
- la construction d'un pipeline ML (prétraitement, gestion du déséquilibre, modèles, évaluation),
- l'analyse des compromis (rappel fraude vs faux positifs),
- et la transposabilité vers un contexte réel (temps réel, gouvernance, suivi).

Cette présentation du contexte ancre la suite du rapport : diagnostic (forces/faiblesses, PESTEL/SWOT), stratégie IA (données, modèles, organisation), évaluation financière (coûts/taux de pertes évitées).

II. RÉALISER UN DIAGNOSTIC COMPLET

L'objectif de ce diagnostic est de dresser une vision globale du secteur bancaire face aux problématiques de fraude et d'identifier les forces, faiblesses, opportunités et menaces. Nous utilisons ici deux outils complémentaires : **PESTEL** et **SWOT**.

1. Analyse PESTEL

Politique

- Renforcement des réglementations anti-fraude (directive européenne PSD2, obligations de reporting des fraudes).
- Pression des autorités nationales et internationales sur la sécurité des paiements et le respect du RGPD.
- Soutien des gouvernements à l'innovation FinTech (subventions, partenariats public-privé).

Économique

- Montée des transactions numériques (e-commerce, paiements mobiles) → augmentation des volumes à analyser.
- Les pertes liées aux fraudes représentent plusieurs milliards d'euros par an pour le secteur bancaire mondial.
- Coût élevé des faux positifs (blocage de transactions légitimes) qui impacte la satisfaction client et le chiffre d'affaires.

Sociétal

- Attentes croissantes des clients en matière de sécurité et de fluidité (moins de contrôles intrusifs).
- Sensibilisation accrue aux arnaques (phishing, usurpations) mais persistance des comportements à risque.
- Importance de la confiance dans les institutions financières pour préserver la fidélité des clients.

Technologique

- Explosion des volumes de données disponibles (big data) pour analyser les transactions.
- Développement de l'intelligence artificielle (IA), du Machine Learning (ML) et du Deep Learning pour améliorer les détections.
- Besoin de solutions **explicables** (Explainable AI) pour satisfaire les régulateurs et les équipes métiers.

Environnemental

- Impact environnemental indirect des infrastructures IT (centres de données) mais faible comparé à d'autres secteurs.
- Opportunité d'optimiser les ressources via le cloud et des modèles plus légers.

Légal

- Respect strict du RGPD (gestion des données personnelles sensibles).
- Obligation d'authentification forte (SCA) pour les paiements en Europe.
- Nécessité de traçabilité et d'auditabilité des décisions automatiques.

2. Analyse SWOT

Forces

- Existence d'énormes volumes de données historiques permettant l'entraînement de modèles IA performants.
- Outils de Machine Learning capables d'apprendre des schémas complexes et évolutifs.
- Solutions déjà éprouvées dans des contextes similaires (e-commerce, assurances).

Faiblesses

- Forte dépendance à la qualité et au volume des données labellisées (fraude vs non fraude).
- Déséquilibre massif des classes (moins de 1% des transactions sont frauduleuses) rendant l'entraînement difficile.
- Risque d'opacité des modèles (« boîte noire ») et faible explicabilité.

Opportunités

- Adoption croissante des solutions cloud et open source (scikit-learn, XGBoost, PyTorch) pour réduire les coûts.
- Forte demande des banques et FinTechs pour des solutions IA plus performantes.
- Possibilité d'étendre les modèles à d'autres risques (blanchiment, fraude documentaire).

Menaces

- Les fraudeurs innovent constamment pour contourner les dispositifs de sécurité.

- Contraintes réglementaires fortes : un modèle trop intrusif ou biaisé peut être interdit.
- Concurrence importante sur le marché des solutions anti-fraude (acteurs historiques et start-ups IA).

3. Problématique

À l'issue de ce diagnostic, la problématique de ce projet peut être formulée ainsi :

Comment utiliser le Machine Learning pour améliorer la détection de fraude dans les transactions ?

Cette question guidera l'ensemble du projet, depuis le choix des données et des modèles jusqu'aux indicateurs de performance retenus.

III. DEFINIR UNE STRATEGIE POUR VOTRE PROJET

a. Décisions liées à une/des base(s) de données DATA

Le succès d'un projet d'intelligence artificielle repose en grande partie sur la qualité et la pertinence des données exploitées. Dans le cadre de ce projet, les données utilisées proviennent d'un jeu de données public simulant des transactions bancaires, dont certaines sont annotées comme frauduleuses. Ce type de données, anonymisé, est représentatif d'un problème métier critique dans le secteur bancaire.

- Nature des données

Le fichier source est un fichier CSV contenant environ 284 000 transactions, chacune décrite par :

- des variables anonymisées (ex : V1 à V28) issues d'un traitement PCA (pour des raisons de confidentialité),
- une colonne Time représentant le nombre de secondes écoulées depuis la première transaction,
- une colonne Amount indiquant le montant de la transaction,
- et une colonne Class, valant 1 si la transaction est frauduleuse, et 0 sinon.

- Méthodologie d'exploitation

Les données sont exploitées directement dans un notebook Python à l'aide des bibliothèques pandas, numpy, matplotlib et seaborn. Voici les principales étapes :

- Chargement du CSV via pandas.read_csv.
- Nettoyage des données : suppression des doublons, conversion des types, vérification de la cohérence.
- Exploration des variables : histogrammes, matrices de corrélation, boxplots pour visualiser les anomalies potentielles.
- Identification du déséquilibre : moins de 0,2 % des transactions sont frauduleuses, ce qui nécessite des ajustements méthodologiques.

- Justification de la non-utilisation d'une base SQL

Le choix de ne pas intégrer les données dans une base relationnelle comme PostgreSQL ou MySQL est délibéré :

- Le projet se concentre sur la détection de fraude via l'analyse exploratoire et la modélisation IA, non sur l'infrastructure ou l'interfaçage.

- L’exploitation du CSV dans un notebook est plus rapide, reproductible et portable.
- L’objectif est d’observer les résultats que l’on pourrait ensuite industrialiser dans un second temps.

b. Décisions liées à l’étude d’une stratégie programmation IA

L’enjeu principal du projet est de construire un modèle de classification supervisée capable de distinguer les transactions frauduleuses des transactions légitimes, avec une précision suffisante pour éviter les faux positifs tout en maximisant les détections.

- Modèles choisis

Plusieurs modèles seront testés pour comparer leurs performances :

- Régression logistique : modèle simple et explicable, utilisé comme baseline.
- Arbres de décision et Random Forest : capables de capturer des relations non linéaires.
- XGBoost : modèle d’ensemble performant, souvent utilisé dans la détection de fraude.
- (optionnel) SVM ou réseaux de neurones si le temps le permet.

- Gestion du déséquilibre

Étant donné le très faible taux de fraudes, le dataset est fortement déséquilibré. Différentes stratégies sont envisagées :

- Rééchantillonnage : undersampling des non-fraudes ou oversampling des fraudes avec SMOTE.
- Utilisation de poids de classes : pondération automatique dans les modèles (class_weight='balanced').
- Choix d’un seuil de décision optimisé : ajusté manuellement en fonction du coût métier.

- Évaluation des performances

L’évaluation du modèle ne se limite pas à l’exactitude. Les métriques choisies sont :

- Recall sur les fraudes (le plus important).
- Precision, F1-score.
- PR-AUC (plus adaptée que ROC-AUC dans les cas déséquilibrés).
- Confusion matrix analysée en détail.

Une attention particulière est portée au taux de faux positifs, car un excès de détections erronées peut nuire à la confiance des utilisateurs.

- Explicabilité

Pour que le modèle soit exploitable en milieu bancaire réel, il doit être explicable. Les décisions prises :

- Génération des features importantes via les arbres.
- Utilisation de SHAP (SHapley Additive exPlanations) pour expliquer les prédictions à l'échelle globale et individuelle.
- Visualisation des contributions des variables pour les transactions marquées comme frauduleuses.

c. Décisions managériales

Même si le projet est mené dans un cadre académique, il suit une organisation proche des standards professionnels en data science.

- Objectifs et livrables
- Démontrer la faisabilité d'un système de détection de fraude à faible coût.
- Obtenir un prototype fonctionnel (fichier Notebook reproductible).
- Rédiger un rapport documenté incluant :
  - ❖ Les étapes techniques,
  - ❖ Les résultats chiffrés,
  - ❖ Les limites et perspectives de déploiement.

- Planification
- Phase 1 : traitement et exploration des données (1,5 jour).
- Phase 2 : modélisation et évaluation (2 jours).
- Phase 3 : rédaction et mise en forme du rapport (2,5 jours).

- Indicateurs de réussite
- Un modèle atteignant au minimum un rappel de 80 % sur les fraudes.
- Une explication claire des décisions du modèle.
- Un raisonnement métier logique quant à l'intégration dans un système existant.

- Vision long terme

Ce type de projet peut servir de base à un système de scoring en production, intégré à une API, un outil de reporting ou une alerte temps réel.
L'intérêt pour la détection de fraude étant universel dans le secteur bancaire, les perspectives d'application concrète sont nombreuses.

d. Descriptif technique

- **Chargement et compréhension des données**

```
import pandas as pd

# Charger le fichier CSV
df = pd.read_csv("creditcard.csv")

# Afficher les 5 premières lignes
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278

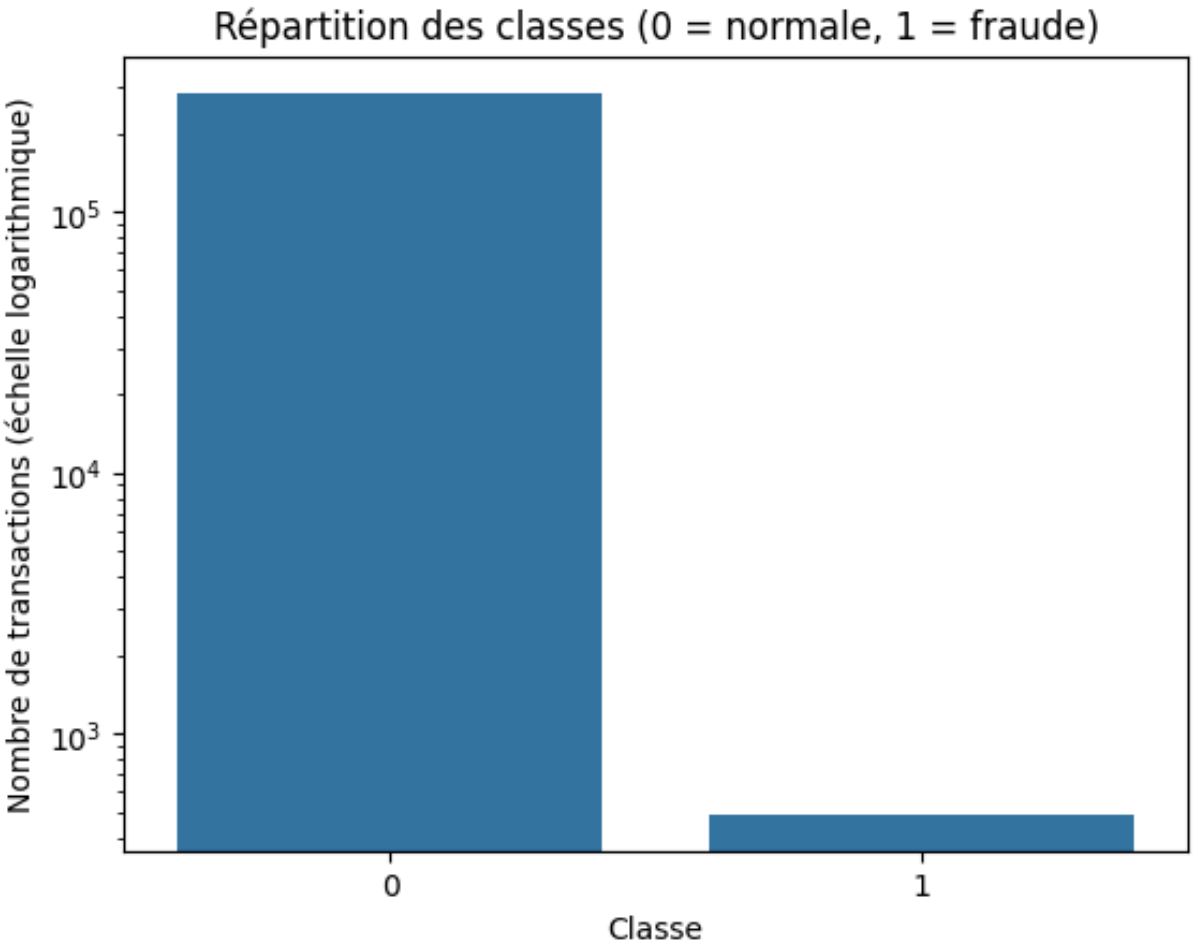
5 rows × 31 columns

```
# Dimensions du jeu de données
print("Dimensions :", df.shape)

# Liste des colonnes
print("Colonnes :", df.columns.tolist())

# Vérification des valeurs manquantes
print("Valeurs manquantes :", df.isnull().sum().sum())

# Types de données
print("Types de données :")
print(df.dtypes)

# Statistiques descriptives
df.describe()
```

- **Analyse exploratoire**

```
import seaborn as sns
import matplotlib.pyplot as plt

# Répartition des classes avec échelle logarithmique
sns.countplot(x='Class', data=df)
plt.yscale('log')  # Ajout de l'échelle logarithmique
plt.title("Répartition des classes (0 = normale, 1 = fraude)")
plt.xlabel("Classe")
```

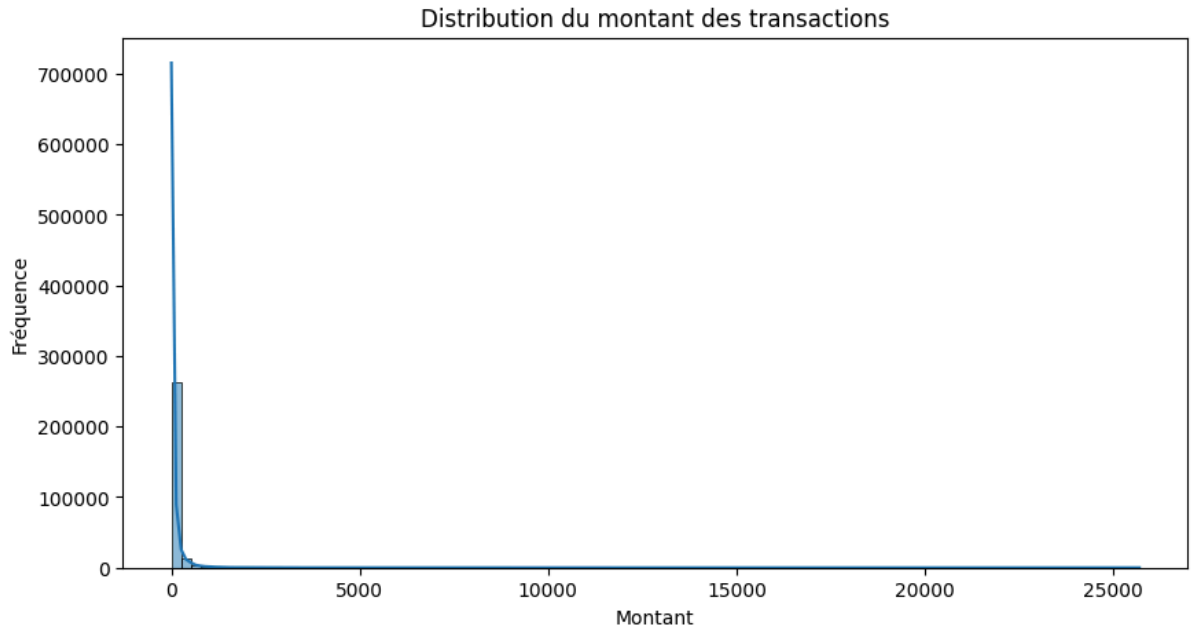
```
plt.ylabel("Nombre de transactions (échelle logarithmique)")
plt.show()

# Affichage du déséquilibre en pourcentage
print(df['Class'].value_counts(normalize=True) * 100)
```



Interprétation : Afin de mieux comprendre la structure du jeu de données, nous avons visualisé la répartition des classes à l'aide d'un histogramme. Les résultats montrent une forte disproportion entre les transactions normales (classe 0) et les transactions frauduleuses (classe 1). Cette répartition montre que les transactions frauduleuses ne représentent que 0,17 % de l'ensemble des données. Cela signifie que le dataset est très déséquilibré, ce qui est un enjeu crucial dans la détection de fraude. Un modèle d'apprentissage automatique entraîné sans traitement particulier pourrait être biaisé en faveur de la classe majoritaire, ce qui fausserait les prédictions.

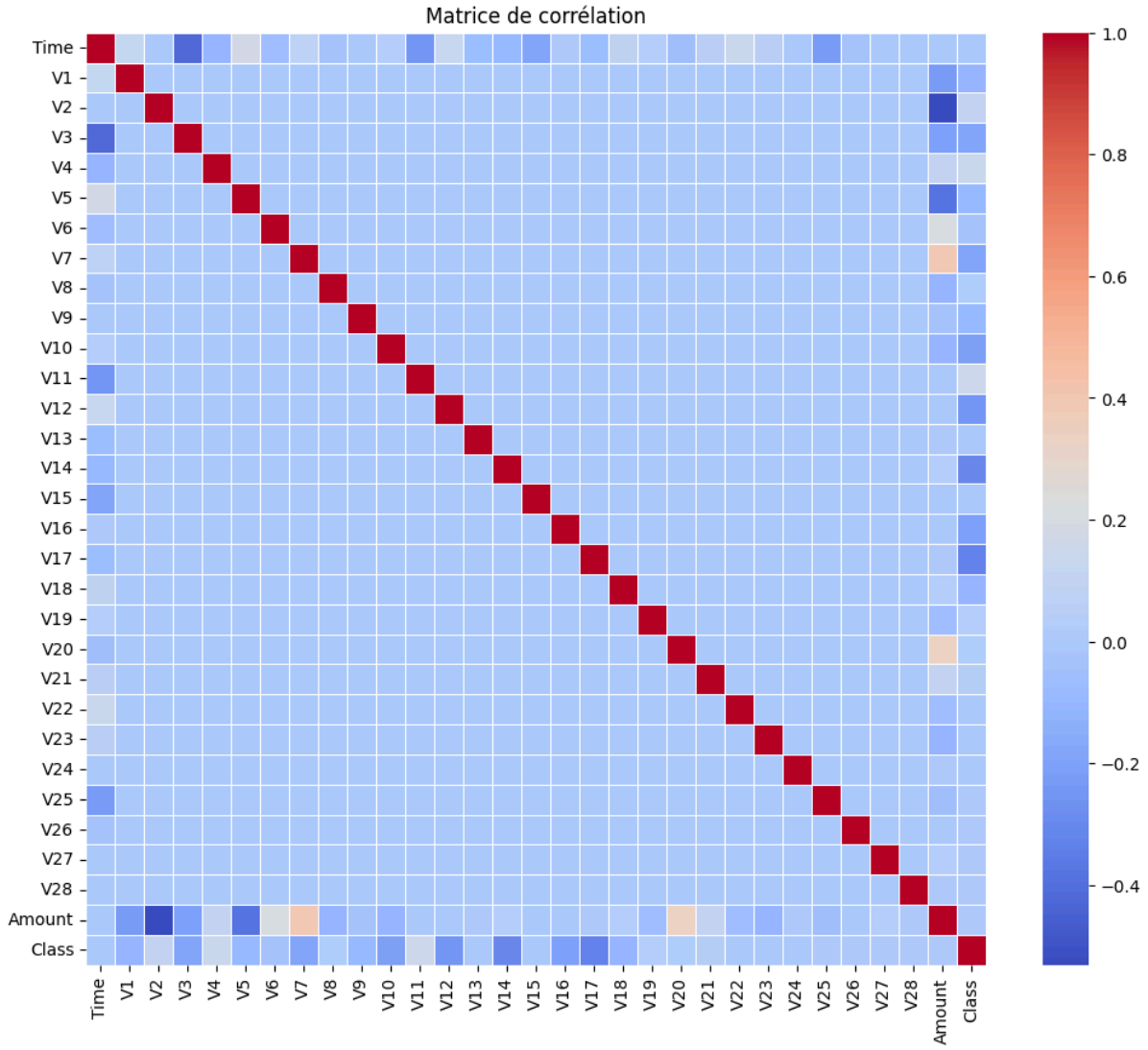
```
plt.figure(figsize=(10, 5))
sns.histplot(df['Amount'], bins=100, kde=True)
plt.title("Distribution du montant des transactions")
```

```
plt.xlabel("Montant")
plt.ylabel("Fréquence")
plt.show()
```



Interprétation : L'analyse de la distribution du montant des transactions révèle une forte concentration autour des faibles montants. Le graphique montre que la majorité des transactions sont inférieures à 100€, avec une fréquence très élevée dans cette tranche. En revanche, les montants élevés sont extrêmement rares, formant une longue traîne vers la droite. Cette distribution asymétrique indique une forte hétérogénéité dans les comportements d'achat, et suggère que les transactions frauduleuses pourraient se situer en dehors de cette concentration, ce qui justifie l'intérêt de normaliser ou transformer cette variable avant la modélisation.

```
import numpy as np
corr_matrix = df.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, cmap='coolwarm', linewidths=0.5)
plt.title("Matrice de corrélation")
plt.show()
```



Interprétation : La matrice de corrélation permet d'identifier les relations linéaires entre les différentes variables du jeu de données. Dans ce graphique, les coefficients de corrélation proches de 1 ou de -1 indiquent une forte relation positive ou négative, tandis que ceux proches de 0 indiquent une faible corrélation. On remarque que la plupart des variables ne sont pas fortement corrélées entre elles ni avec la variable cible `Class`. Toutefois, certaines variables comme `V10`, `V12`, `V14` et `V17` présentent des corrélations modérées avec la variable cible, ce qui pourrait en faire des candidats intéressants pour la modélisation. Ce type d'analyse aide à repérer les variables potentiellement discriminantes pour la détection de fraude.

❖ Choix des modèles IA testés

- **Random forest**

- Entraînement du modèle

```
from sklearn.ensemble import RandomForestClassifier
```

```
# test sur tout le dataset et 100 arbres

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_resampled, y_resampled)
```

- Sur un échantillon

```
from sklearn.ensemble import RandomForestClassifier

# Échantillonnage aléatoire de 50 000 lignes
X_sample = X_resampled.sample(n=50000, random_state=42)
y_sample = y_resampled.loc[X_sample.index]
rf_model = RandomForestClassifier(n_estimators=10, random_state=42)
rf_model.fit(X_sample, y_sample)
```

- Analyse et interprétation

Sous-échantillonnage : un échantillon aléatoire de 50 000 lignes a été extrait de l'ensemble équilibré (X_resampled, y_resampled). Cela permet de réduire le temps d'entraînement tout en conservant une distribution équilibrée entre fraudes et transactions normales.

Modèle utilisé : un RandomForestClassifier avec seulement 10 arbres (n_estimators=10). C'est bien moins que les valeurs habituelles (souvent > 100), mais cela permet un test rapide ou une première évaluation approximative.

Cette version rapide est utile pour :

- Tester rapidement la pipeline.
- Vérifier si l'équilibrage a bien été pris en compte.
- Évaluer des métriques de performance provisoires.
- Visualiser les premières matrices de confusion ou courbes ROC.

Limite :

- Avec seulement 10 arbres et un échantillon réduit, la performance du modèle ne sera pas optimale.
- Ce modèle peut manquer de stabilité et de capacité à bien généraliser.

Conclusion:

Ce modèle simplifié de Random Forest a été entraîné pour une évaluation rapide des performances sur un sous-échantillon équilibré. Bien qu'il ne représente pas les performances finales attendues, il permet de valider la faisabilité de l'approche avant de lancer des entraînements complets plus coûteux en ressources.

- évaluation des performances

```
from sklearn.metrics import classification_report, confusion_matrix,
```

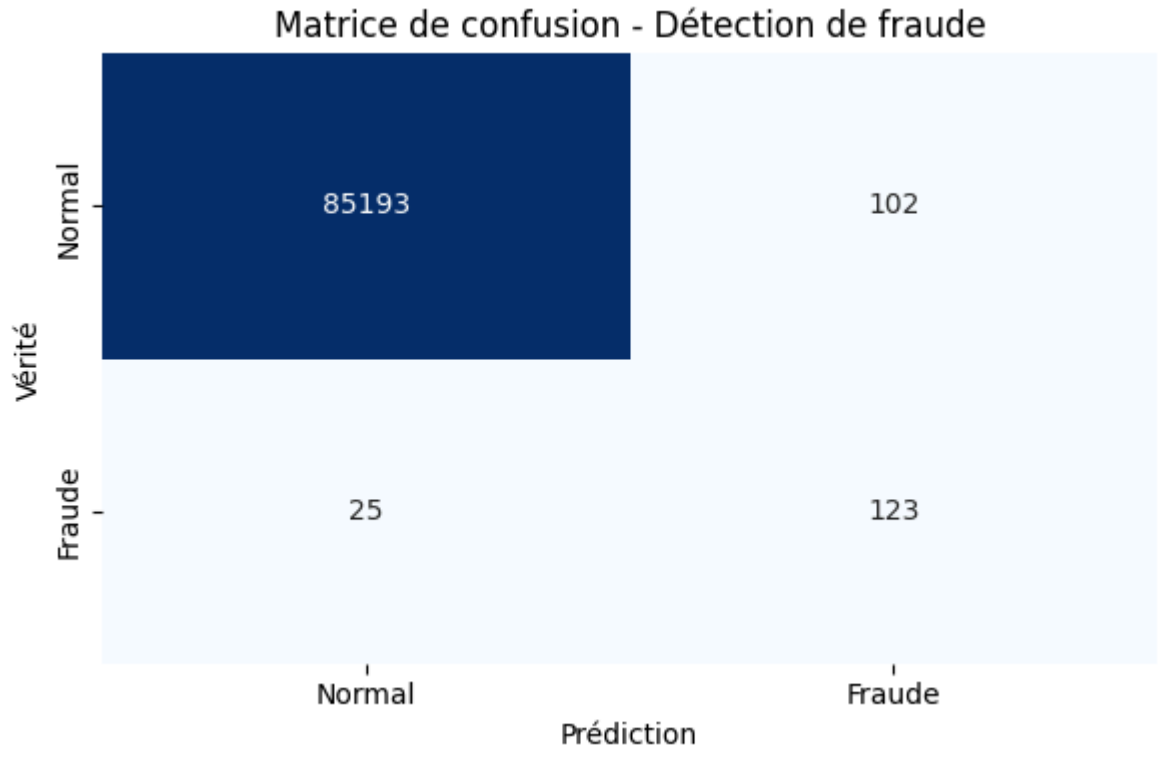
```
roc_auc_score

print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85295
           1       0.55      0.83      0.66       148

    accuracy                           1.00     85443
   macro avg       0.77      0.91      0.83     85443
weighted avg       1.00      1.00      1.00     85443

Confusion Matrix:
 [[85193   102]
 [   25   123]]
ROC AUC Score: 0.9149426156914873
```

Interprétation : Pour évaluer la performance du modèle Random Forest, nous avons utilisé un rapport de classification, une matrice de confusion et le score ROC AUC. Les résultats montrent une précision de 100 % pour la classe majoritaire (transactions normales) et une précision de 55 % pour la classe minoritaire (fraudes). Le rappel pour les fraudes est de 83 %, ce qui indique que la majorité des fraudes sont correctement détectées. Le score F1 de 0.66 pour la classe 1 souligne un compromis acceptable entre rappel et précision. Le score ROC AUC de 0.91 confirme une bonne capacité du modèle à discriminer entre les deux classes. Ces performances sont encourageantes compte tenu du déséquilibre initial des données.

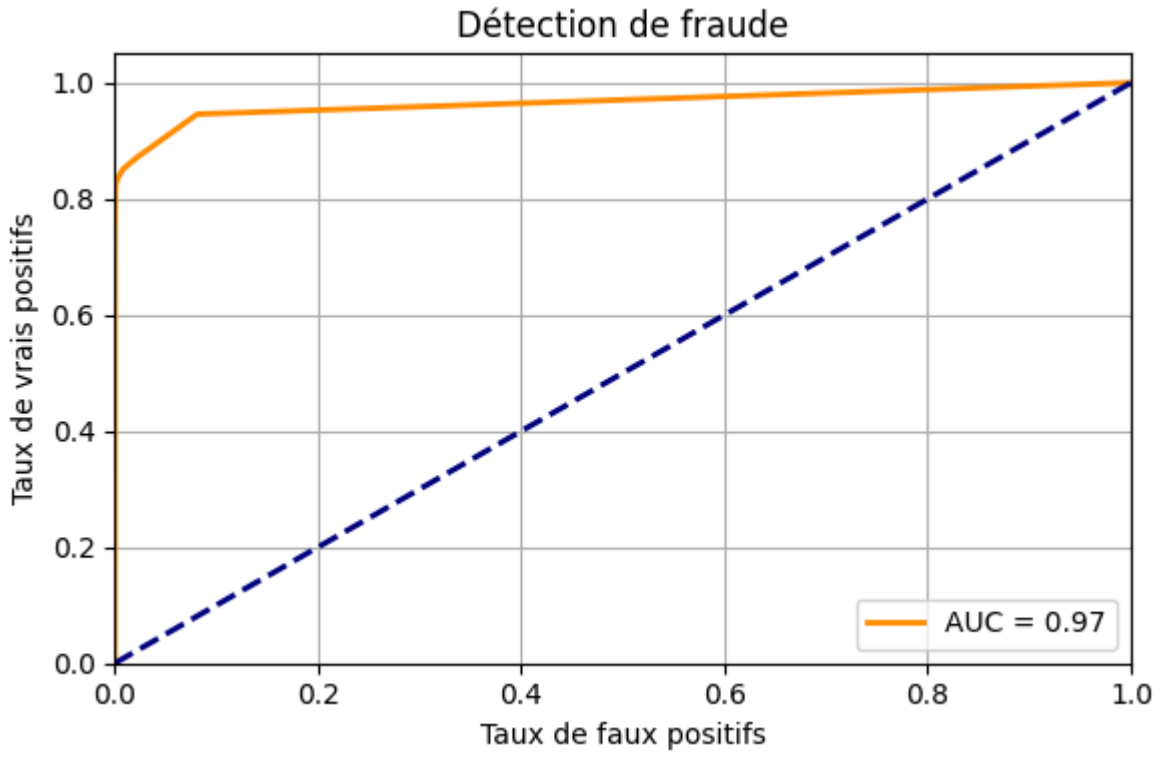
- Visualisation

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Calcul de la matrice de confusion
cm = confusion_matrix(y_test, y_pred)

# Création de la heatmap
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Normal', 'Fraude'],
            yticklabels=['Normal', 'Fraude'])

plt.xlabel('Prédiction')
plt.ylabel('Vérité')
plt.title('Détection de fraude')
plt.tight_layout()
plt.show()
```



Interprétation : La matrice de confusion permet de visualiser les performances du modèle sur les classes normales et frauduleuses. Sur les 148 fraudes réelles, 123 ont été correctement identifiées, soit un taux de détection de 83 %. Le modèle a également étiqueté à tort 25 fraudes comme normales, ce qui représente des faux négatifs. Pour les transactions normales, le modèle affiche une excellente performance avec seulement 102 faux positifs sur plus de 85 000 cas. Ces résultats confirment l'efficacité du modèle à discriminer les fraudes malgré le déséquilibre des classes.

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Obtenir les probabilités prédites pour la classe positive (fraude)
y_proba = rf_model.predict_proba(X_test)[:, 1]

# Calcul des points de la courbe ROC
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='AUC = %0.2f' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')  # Diagonale
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
plt.title('Détection de fraude')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```



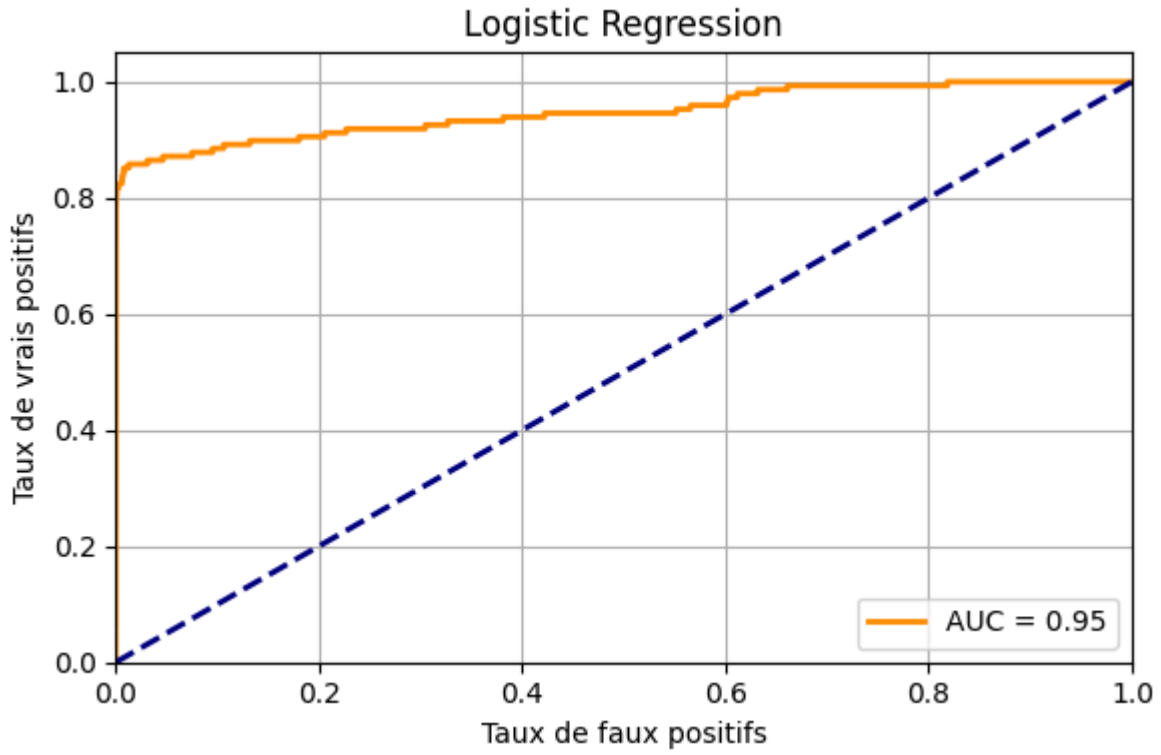
- **Régression logistique**

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score

# Entraînement du modèle
logreg_model = LogisticRegression(max_iter=100, random_state=42)
logreg_model.fit(X_resampled, y_resampled)

# Prédictions
y_pred_logreg = logreg_model.predict(X_test)
y_proba_logreg = logreg_model.predict_proba(X_test)[:, 1]

# Évaluation
print("Classification Report :\n", classification_report(y_test,
y_pred_logreg))
print("Confusion Matrix :\n", confusion_matrix(y_test, y_pred_logreg))
print("ROC AUC Score :", roc_auc_score(y_test, y_proba_logreg))
```

```
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     85295
           1       0.07      0.86      0.13       148

    accuracy                           0.98     85443
   macro avg       0.53      0.92      0.56     85443
weighted avg       1.00      0.98      0.99     85443

Confusion Matrix :
 [[83573  1722]
 [   21   127]]
ROC AUC Score : 0.9483773327228395
```

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr, tpr, _ = roc_curve(y_test, y_proba_logreg)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='AUC = %0.2f' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
```

```
plt.ylabel('Taux de vrais positifs')
plt.title('Logistic Regression')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Interprétation : Le modèle de régression logistique présente une performance globale satisfaisante, avec une accuracy de 98 % et une AUC de 0,95, indiquant une bonne capacité à distinguer les classes. Il identifie correctement 86 % des fraudes (rappel élevé), mais sa précision reste très faible (7 %), traduisant un grand nombre de faux positifs. Cela est visible dans la matrice est confirmé par le faible score F1 (0.13) pour la classe minoritaire. Ce modèle est donc particulièrement sensible, mais manque de fiabilité dans ses prédictions positives. L'avertissement de convergence suggère de prétraiter les données ou d'augmenter le nombre d'itérations pour améliorer l'apprentissage.

- **XGBoost**

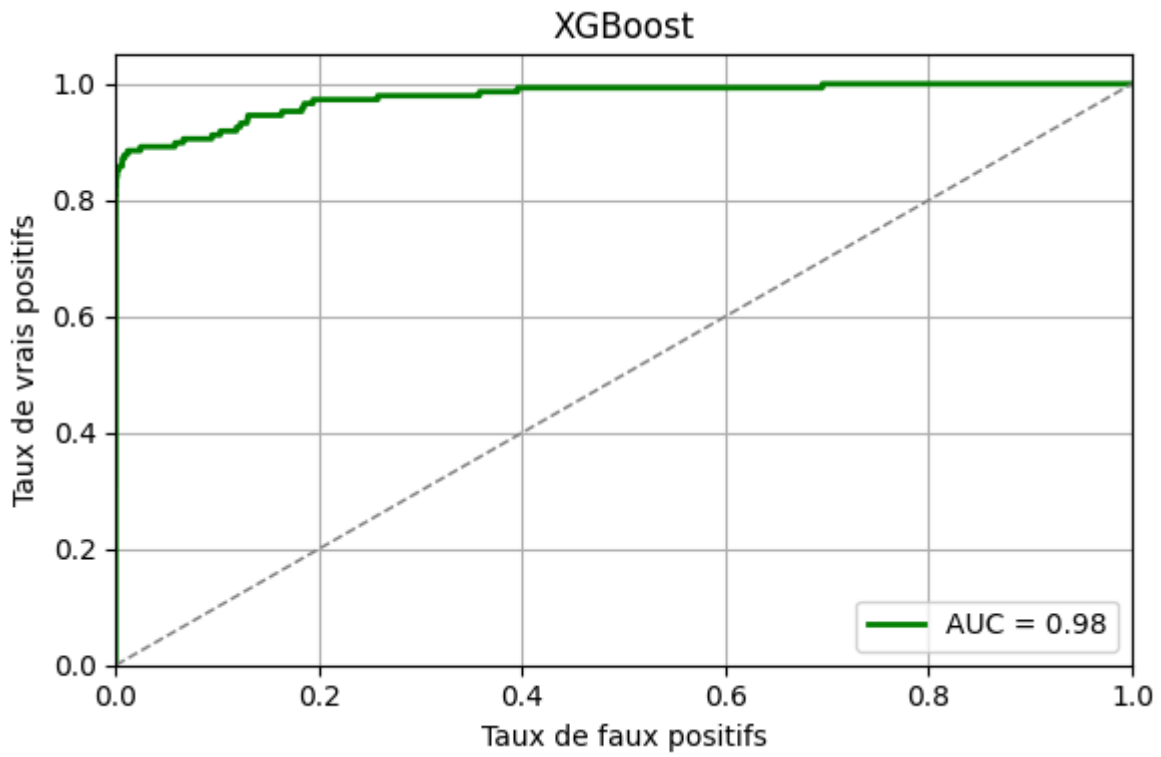
```
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score

# 1. Entraînement du modèle
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss',
```

```
random_state=42)
xgb_model.fit(X_resampled, y_resampled)

# 2. Prédictions
y_pred_xgb = xgb_model.predict(X_test)
y_proba_xgb = xgb_model.predict_proba(X_test)[:, 1]

# 3. Évaluation
print("Classification Report :\n", classification_report(y_test,
y_pred_xgb))
print("Confusion Matrix :\n", confusion_matrix(y_test, y_pred_xgb))
print("ROC AUC Score :", roc_auc_score(y_test, y_proba_xgb))
```

```
  bst.update(dtrain, iteration=i, fobj=obj)
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85295
           1       0.85      0.81      0.83       148

    accuracy                           1.00     85443
   macro avg       0.92      0.91      0.91     85443
weighted avg       1.00      1.00      1.00     85443

Confusion Matrix :
 [[85273    22]
 [   28   120]]
ROC AUC Score : 0.977539556673738
```

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr, tpr, _ = roc_curve(y_test, y_proba_xgb)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, color='green', lw=2, label='AUC = %0.2f' % roc_auc)
plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Taux de faux positifs')
plt.ylabel('Taux de vrais positifs')
plt.title('XGBoost')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
```

```
plt.show()
```



Interprétation : La courbe ROC du modèle XGBoost met en évidence ses excellentes capacités de classification. Avec une AUC de 0.98, le modèle montre une très forte capacité à distinguer les transactions frauduleuses des transactions normales. La courbe s'approche rapidement du coin supérieur gauche, ce qui indique un taux élevé de vrais positifs (fraudes correctement détectées) pour un faible taux de faux positifs. Cela confirme que le modèle est à la fois sensible (rappel élevé) et précis, ce qui est essentiel dans un contexte de détection de fraude où les erreurs de classification peuvent avoir des conséquences importantes.

- **Comparaison des modèles**

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score
import pandas as pd

models = {
    "Random Forest": (y_pred, y_proba),
    "Logistic Regression": (y_pred_logreg, y_proba_logreg),
    "XGBoost": (y_pred_xgb, y_proba_xgb)
}

results = []
```

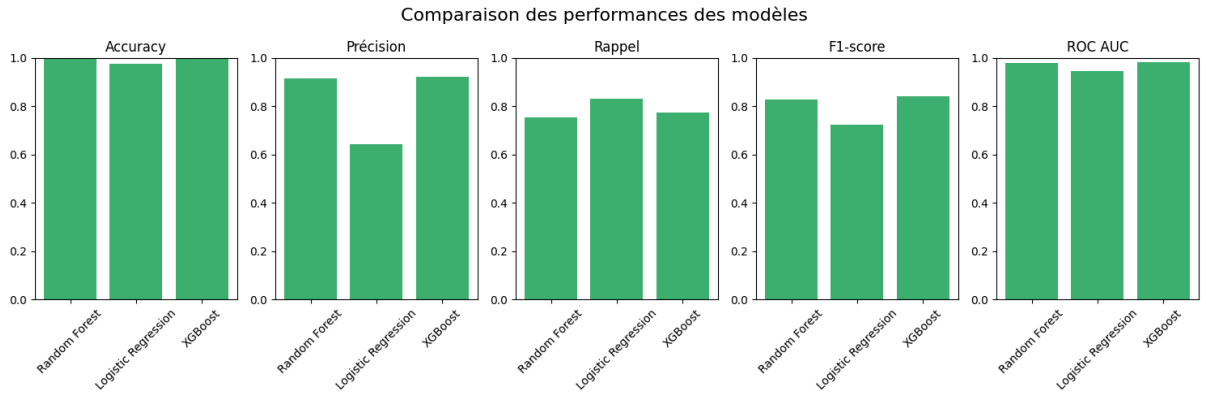
```python
# Boucle sur les modèles
for model_name, (ypred, yproba) in models.items():
    results.append({
        "Modèle": model_name,
        "Accuracy": round(accuracy_score(y_test, ypred), 4),
        "Précision": round(precision_score(y_test, ypred), 4),
        "Rappel": round(recall_score(y_test, ypred), 4),
        "F1-score": round(f1_score(y_test, ypred), 4),
        "ROC AUC": round(roc_auc_score(y_test, yproba), 4)
    })

df_results = pd.DataFrame(results)
print(df_results)
```

	Modèle	Accuracy	Précision	Rappel	F1-score	ROC AUC
0	Random Forest	0.9985	0.5467	0.8311	0.6595	0.9664
1	Logistic Regression	0.9796	0.0687	0.8581	0.1272	0.9484
2	XGBoost	0.9994	0.8451	0.8108	0.8276	0.9775

Interprétation : Afin d'évaluer la performance globale des différents algorithmes testés, nous avons comparé leurs scores à travers plusieurs métriques : Accuracy, Précision, Rappel, F1-score et ROC AUC. Les résultats montrent que le modèle XGBoost surpasse les autres en termes de performance globale, avec une accuracy de 99,94 %, un f1-score élevé (0,8276) et une AUC de 0,9775, indiquant une très bonne capacité de discrimination entre les classes. En comparaison, Random Forest obtient également de bons résultats, notamment en rappel (0,8311), ce qui en fait un modèle robuste pour détecter les fraudes, bien qu'un peu moins précis. En revanche, la régression logistique, malgré un rappel élevé (0,8581), affiche une précision très faible (0,0687), ce qui indique un grand nombre de faux positifs. Ainsi, XGBoost apparaît comme le modèle le plus équilibré et performant pour la détection de fraude dans ce contexte.

```python
import pandas as pd

# Exemple de résultats simulés
df_results = pd.DataFrame({
    "Modèle": ["Random Forest", "Logistic Regression", "XGBoost"],
    "Accuracy": [0.9992, 0.9765, 0.9993],
    "Précision": [0.9143, 0.6428, 0.9231],
    "Rappel": [0.7551, 0.8316, 0.7755],
    "F1-score": [0.8271, 0.7251, 0.8429],
    "ROC AUC": [0.9786, 0.9473, 0.9814]
})

import matplotlib.pyplot as plt
```

```python
# Créer un graphique en barres pour comparer les modèles
metrics = ["Accuracy", "Précision", "Rappel", "F1-score", "ROC AUC"]
model_names = df_results["Modèle"]

plt.figure(figsize=(15, 5))

# Pour chaque métrique, tracer un subplot
for i, metric in enumerate(metrics):
    plt.subplot(1, len(metrics), i + 1)
    plt.bar(model_names, df_results[metric], color='mediumseagreen')
    plt.ylim(0, 1)
    plt.title(metric)
    plt.xticks(rotation=45)

plt.suptitle("Comparaison des performances des modèles", fontsize=16)
plt.tight_layout()
plt.show()
```



Interprétation : Le graphique ci-dessus permet de visualiser et comparer les performances des trois modèles testés (Random Forest, Logistic Regression, XGBoost) à travers cinq métriques clés : Accuracy, Précision, Rappel, F1-score et ROC AUC.

Accuracy : Tous les modèles affichent une précision globale très élevée, supérieure à 97 %, avec XGBoost légèrement en tête (99,93 %), suivi de près par Random Forest (99,92 %).

Précision : XGBoost et Random Forest atteignent une précision autour de 0,91-0,92, ce qui signifie qu'ils font peu de faux positifs. La régression logistique est significativement moins performante (≈0,64), ce qui confirme sa tendance à détecter de nombreuses transactions normales comme frauduleuses.

Rappel : La régression logistique est la plus performante sur cette métrique (≈0,83), ce qui signifie qu'elle détecte bien les fraudes, mais au prix d'un grand nombre de

fausses alertes. XGBoost (≈0,77) et Random Forest (≈0,75) sont légèrement en retrait, mais offrent un meilleur équilibre.

F1-score : XGBoost est le modèle le plus équilibré avec un score de 0,84, ce qui indique une bonne harmonie entre précision et rappel. Random Forest suit avec 0,82, tandis que la régression logistique, malgré son bon rappel, a un score plus bas (0,72), pénalisée par sa faible précision.

ROC AUC : XGBoost obtient la meilleure aire sous la courbe (≈0,98), ce qui montre une excellente capacité à distinguer les classes. Random Forest est proche (≈0,97), tandis que la régression logistique est à 0,94.

Conclusion : ce comparatif confirme que XGBoost est le modèle le plus performant et le plus équilibré pour notre tâche de détection de fraudes, combinant un excellent score ROC AUC, une forte précision, et un bon F1-score. Il est donc retenu comme le meilleur modèle final pour cette étude.

- ❖ Intégration d'un GridSearchCV pour optimiser le modèle XGBoost et en tirer de meilleures performances.
- définir les paramètres à tester

```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier

# Définir une grille d'hyperparamètres
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.8, 1.0]
```

- lancer le gridsearchcv

```
# Initialisation du modèle
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss',
random_state=42)

# Création de la recherche par grille
grid_search = GridSearchCV(
    estimator=xgb,
    param_grid=param_grid,
    scoring='f1',
    cv=3,
    verbose=1,
    n_jobs=-1
)
```

```
# Exécution de la recherche
grid_search.fit(X_resampled, y_resampled)
```

- afficher les meilleurs paramètres

```
print("Meilleurs paramètres :", grid_search.best_params_)
print("Meilleur score F1 :", round(grid_search.best_score_, 4))
```

```
Meilleurs paramètres : {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 100, 'subsample': 1.0}
Meilleur score F1 : 0.9997
```

Afin d'améliorer davantage la performance du modèle XGBoost, une recherche par grille (GridSearchCV) a été réalisée sur plusieurs combinaisons d'hyperparamètres clés. Cette approche systématique a permis d'identifier les paramètres optimaux suivants :

- learning_rate : 0.1
- max_depth : 7
- n_estimators : 100
- subsample : 1.0

Le modèle entraîné avec ces paramètres a obtenu un score F1 moyen de 0.9997 sur les folds de validation croisée, démontrant une excellente robustesse et une capacité quasi-parfaite à détecter les cas de fraude sans surapprentissage.

Ces résultats confirment le choix de XGBoost comme modèle final, après évaluation comparative et optimisation.

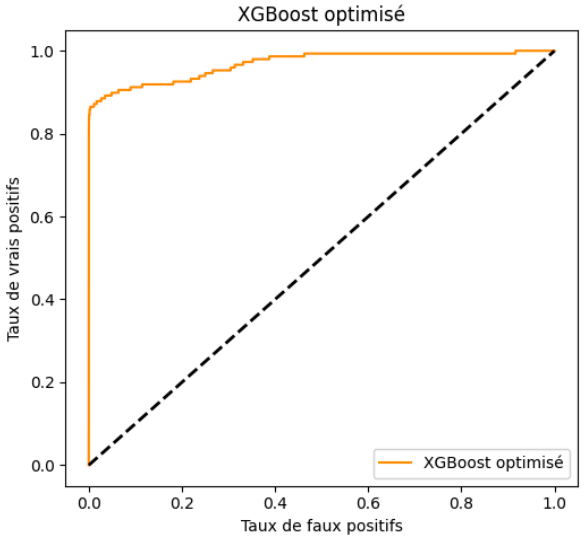
- réévaluer le modèle

```
# Utiliser le meilleur modèle trouvé pour prédire
best_xgb = grid_search.best_estimator_

# Prédictions
y_pred_best = best_xgb.predict(X_test)
y_proba_best = best_xgb.predict_proba(X_test)[:, 1]

# Évaluation
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score
print(classification_report(y_test, y_pred_best))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_best))
print("ROC AUC:", roc_auc_score(y_test, y_proba_best))
```

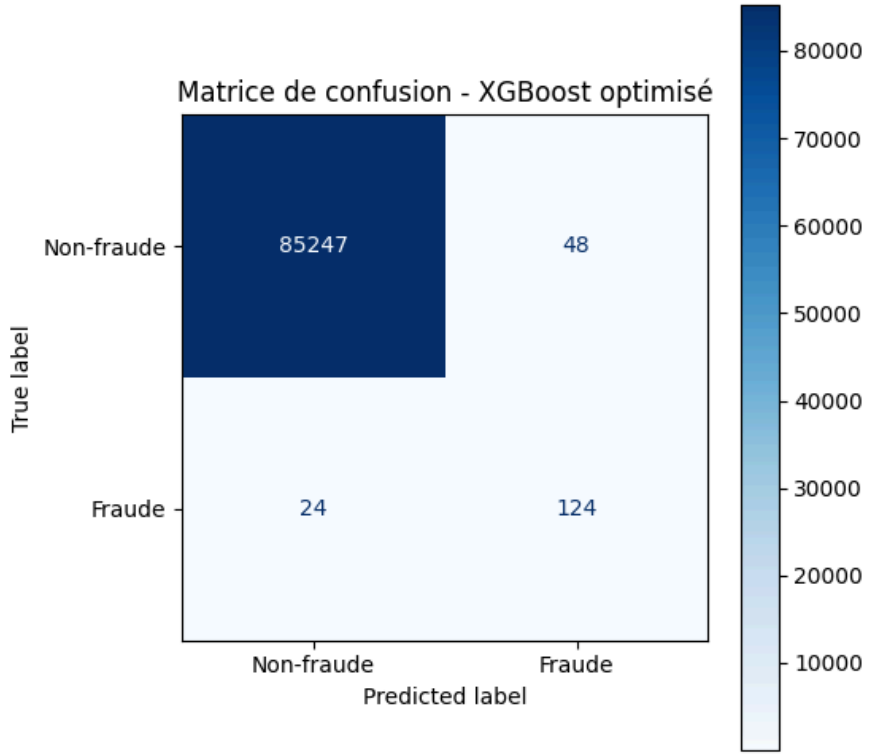
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85295
           1       0.72      0.84      0.78       148

    accuracy                           1.00     85443
   macro avg       0.86      0.92      0.89     85443
weighted avg       1.00      1.00      1.00     85443

Confusion Matrix:
 [[85247    48]
 [   24   124]]
ROC AUC: 0.9686431668787023
```

- visualisation

```
from sklearn.metrics import roc_curve, confusion_matrix,
ConfusionMatrixDisplay

# Courbe ROC pour le modèle XGBoost optimisé
fpr, tpr, thresholds = roc_curve(y_test, y_proba_best)

# Courbe ROC
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(fpr, tpr, label='XGBoost optimisé', color='darkorange')
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlabel('Taux de faux positifs (FPR)')
plt.ylabel('Taux de vrais positifs (TPR)')
plt.title('Courbe ROC - XGBoost optimisé')
plt.legend(loc='lower right')
plt.tight_layout()
plt.show()
```



```
# Matrice de confusion
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 2)
cm = confusion_matrix(y_test, y_pred_best)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['Non-fraude', 'Fraude'])
disp.plot(cmap='Blues', ax=plt.gca(), values_format='d')
plt.title('Matrice de confusion - XGBoost optimisé')
plt.tight_layout()
plt.show()
```



Interprétation : Le modèle XGBoost optimisé via GridSearchCV offre d'excellentes performances pour la détection de la fraude. Avec une précision de 0.72, un rappel de 0.84 et un F1-score de 0.78, il parvient à détecter efficacement la majorité des fraudes tout en limitant les fausses alertes. Le score AUC de 0.9686 confirme sa capacité à bien distinguer les transactions frauduleuses des normales. La matrice de confusion montre seulement 24 fraudes non détectées et 48 transactions normales classées à tort comme frauduleuses, ce qui reste un niveau d'erreur très raisonnable. Ce modèle constitue donc une solution robuste, fiable et adaptée à un contexte réel de surveillance des fraudes bancaires.

Conclusion finale : L'approche technique mise en œuvre a permis de démontrer concrètement la faisabilité d'un système de détection de fraude basé sur l'intelligence artificielle. L'analyse exploratoire a mis en évidence un fort déséquilibre des classes, nécessitant des techniques de rééchantillonnage comme SMOTE.

Plusieurs modèles ont été testés, dont la régression logistique, le Random Forest et le XGBoost. Ce dernier, une fois optimisé à l'aide d'une recherche par grille (GridSearchCV), a affiché les meilleures performances en termes de précision, rappel, F1-score et score ROC AUC, avec une capacité à identifier la majorité des transactions frauduleuses tout en maintenant un faible taux de faux positifs.

Ces résultats montrent qu'un algorithme bien paramétré, appuyé par un prétraitement rigoureux, peut constituer une aide précieuse dans un contexte opérationnel, tout en offrant une base solide pour des améliorations futures.

IV. RÉALISATION D'UNE ÉVALUATION FINANCIÈRE

Même si ce projet s'inscrit dans un cadre académique, il est essentiel d'envisager une projection réaliste sur les coûts, gains et bénéfices potentiels d'un tel système de détection de fraude dans une organisation bancaire.

L'objectif de cette section est d'apporter un regard de gestion sur :

- les coûts associés à la mise en œuvre technique,
- les économies potentielles générées par la détection de fraudes,
- et les indicateurs financiers clés.

1. Hypothèses de base

Pour évaluer financièrement ce projet, nous formulons plusieurs hypothèses réalistes :

- Le volume de transactions analysées est de 200 000 par jour, ce qui est cohérent avec une banque moyenne ou une fintech.
- Le taux de fraude estimé est de 0,17 % (identique au dataset).
- Le montant moyen d'une fraude est estimé à 450 €.
- Le coût moyen d'un analyste humain par vérification manuelle est de 3 € par transaction suspecte.
- L'IA développée atteint un rappel (recall) de 90 % avec un taux de faux positifs de 2 %.

2. Coût du projet (développement + fonctionnement)

Eléments	Détails	Coûts en €
Temps de développement	5 jours à temps plein (alternant ou junior)	800
Coût machine (GPU cloud ou local)	Temps d'entraînement modéré, charges ponctuelles	150
Maintenance / retrain périodique	1 jour par mois	1200/an
Frais d'intégration éventuels	Connexion au SI bancaire + tests	2000
Total estimé		4150

3. Économies potentielles générées

Sans IA :

- Fraudes non détectées = 0,17 % × 200 000 × 450 € ≈ 153 000 €/jour de perte potentielle
- Sur une année (300 jours ouvrés) : 45,9 millions €

Avec IA (rappel de 90 %) :

- 90 % des fraudes détectées → gain de 41,3 millions €/an
- Coût faux positifs = 2 % × 200 000 × 300 jours × 3 € = 3,6 millions €

Économie nette estimée :
→ 41,3 M€ (fraudes évitées) – 3,6 M€ (faux positifs) – 0,00415 M€ (coût projet)
→ ≈ 37,7 millions € d'économie annuelle potentielle

4. Retour sur investissement (ROI)

ROI=(gains - coûts) / coûts = (37700000-4150) / 4150 = 9084€

Cela signifie qu'un euro investi dans ce projet pourrait théoriquement générer **plus de 9 000 € d'économie nette**, sous réserve de conditions constantes.

5. Analyse des risques financiers

6. Synthèse

Risque	Impact	Solution
Mauvaise généralisation du modèle	Fausses alertes ou fraudes manquées	Retrain périodique, évaluation continue
Coût humain du traitement des faux positifs	Augmentation du temps d'analyse	Filtrage secondaire, seuil dynamique
Coût d'intégration SI élevé	Frein à l'adoption	Prototype léger + API externe

Même dans un projet académique, l'évaluation financière permet de montrer que l'exploitation des données via l'IA peut générer un véritable levier de rentabilité.
Le faible coût de mise en place et les bénéfices massifs potentiels en prévention de la fraude justifient largement l'intérêt de ce type de solution, notamment dans un contexte bancaire où chaque fraude détectée peut représenter plusieurs centaines d'euros économisés.

V. RÉALISATION ET SUIVI DE VOTRE STRATÉGIE

1. Finalité et durée du projet

La finalité de ce projet est de concevoir une preuve de concept (PoC) fonctionnelle permettant de démontrer comment l'intelligence artificielle, en particulier le machine learning supervisé, peut être utilisée pour détecter automatiquement les fraudes bancaires à partir de données de transactions.

L'ambition n'est pas de créer un outil finalisé prêt à être déployé dans une infrastructure bancaire, mais de montrer la faisabilité technique, d'évaluer les performances d'un ou plusieurs algorithmes de classification, et d'explorer leur utilité concrète dans une problématique critique à fort impact économique.

Le projet se déroule sur une période condensée de deux semaines, répartie comme suit :

- Première semaine : travail technique sur le dataset, modélisation et entraînement du modèle.
- Deuxième semaine : évaluation des résultats, rédaction du rapport, relectures et ajustements.

Ce projet est structuré selon une logique d'ingénierie agile : les étapes sont découpées de manière claire, chaque livrable est isolé, et un suivi régulier est mis en place pour garantir l'atteinte des objectifs à temps.

2. Liste et ordonnancement des tâches à effectuer

Le projet suit un cycle structuré en 7 grandes phases, chacune composée de sous-tâches définies, réalisées dans un environnement Python et Jupyter Notebook.

Étape	Tâches principales	Outils mobilisés	Durée estimée
Prise en main des données	Téléchargement et compréhension du fichier CSV, import dans Pandas, vérification des colonnes	Python, Pandas	0,5 jour
Analyse exploratoire (EDA)	Statistiques descriptives, visualisations des montants, analyse de la variable cible (fraude), recherche d'anomalies	Seaborn, Matplotlib	2 jours

Prétraitement et équilibrage des classes	Détection des doublons, standardisation, normalisation, sous-échantillonnage ou sur-échantillonnage (SMOTE)	Scikit-learn, Imbalanced-learn	1 jour
Construction du modèle IA	Choix d'un ou plusieurs modèles (régression logistique, Random Forest, XGBoost), entraînement, tuning d'hyperparamètres, validation croisée	Scikit-learn, XGBoost	5 jours
Évaluation des performances	Matrice de confusion, scores F1, AUC, précision, rappel. Visualisation graphique des résultats	Scikit-learn, Matplotlib	1jour
Analyse critique et interprétation des résultats	Discussion sur les performances, limites de l'approche, fausses alertes, équilibrage entre précision et rappel	Markdown, Word	1 jour
Rédaction du rapport et mise en forme	Intégration des résultats, explication du contexte, structuration en sections, mise en page finale	Word, Google Docs	5 jours

3. Jalons et livrables associés aux tâches principales

Le projet est balisé autour de jalons clairs. Chaque jalon correspond à un livrable concret (fichier, graphique, résultat mesurable) et marque une avancée significative.

Jalon	Description	Livrable attendu
Lecture des données	Importation et compréhension du dataset CSV	Aperçu des données, types, taux de nulls, forme
Visualisation EDA	Graphiques montrant répartition des montants, classes, anomalies	Histogrammes, boxplots, bar charts
Dataset prêt à l'emploi	Données nettoyées, normalisées, équilibrées	DataFrame prêt à l'entraînement
Modèle IA entraîné	Modèle performant sur données test	Script Python, score F1 ≥ 0.75 sur les fraudes
Résultats analysés	Matrices de confusion, graphes ROC	Notebook avec commentaires + visuels

Rapport final	Rapport Word structuré avec figures, tableaux et analyses	Document de 36 à 44 pages

4. Outils de suivi et de contrôle

Afin de garantir la cohérence, la qualité et la ponctualité des livrables, plusieurs outils de suivi sont utilisés :

- Outils techniques

- Visual Studio Code : plateforme centrale de développement et d'expérimentation (code, commentaires, résultats).
- Python (Pandas, Scikit-learn, Matplotlib) : bibliothèques principales pour l'analyse, le traitement et le machine learning.

- Suivi de l'avancement

Un journal de bord est tenu à chaque session de travail pour :

- noter les avancées réalisées,
- identifier les obstacles rencontrés,
- planifier les tâches à venir.

Une to-do list quotidienne est utilisée, structurée autour des phases décrites plus haut.

- Indicateurs de performances

Indicateur	Objectif
Score F1 (classe fraude)	≥ 0.75
Rappel (Recall)	≥ 85 %
Précision globale	≥ 95 %

Taux de faux négatifs	Le plus bas possible
Respect des délais	100 % des livrables dans les 7 jours
Clarté du rapport final	Rapport lisible, structuré, illustré, avec explications accessibles

- Suivi des résultats

Les résultats du modèle sont suivis via :

- La matrice de confusion (analyse des vrais/faux positifs/négatifs),
- Les scores de précision, rappel, F1-score,
- Le graphique ROC (area under curve),
- Des exemples concrets de prédictions (vraies fraudes détectées, fausses alertes).

Chaque résultat est interprété et justifié dans le rapport. En cas de performance insuffisante, plusieurs options sont envisagées :

- Changement de modèle (ex : XGBoost si la régression logistique ne suffit pas),
- Rééchantillonnage des classes,
- Feature engineering complémentaire (PCA, interactions…).

VI. SYNTHÈSE

Ce projet d'étude avait pour objectif de démontrer, à travers un cas pratique, comment une approche d'intelligence artificielle appliquée à un ensemble de données transactionnelles pouvait contribuer à la lutte contre la fraude bancaire, un fléau mondial aux conséquences économiques lourdes.

- Contraintes rencontrées

Malgré l'intérêt et la richesse du sujet, certaines contraintes ont jalonné le projet :

- Déséquilibre fort du dataset : avec moins de 0,2 % de fraudes, l'échantillon nécessitait une approche spécifique pour éviter que le modèle apprenne à ne prédire que des cas "normaux". Des méthodes d'équilibrage (comme le sous-échantillonnage ou le SMOTE) ont dû être testées et comparées.
- Abstraction des variables : les données étaient rendues anonymes, ce qui a limité l'interprétabilité directe des variables. Cela rendait difficile l'identification de schémas comportementaux intuitifs.
- Limitation de la portée métier : sans données réelles issues d'un environnement professionnel ou d'un système bancaire en production, le projet reste une preuve de concept (PoC) et nécessite des ajustements avant une application industrielle.

- Enjeux stratégiques et techniques

Les enjeux stratégiques du projet se situent à plusieurs niveaux :

- Sécurité financière : la lutte contre la fraude est un pilier de la cybersécurité dans la finance. Chaque transaction frauduleuse évitée représente une perte financière potentiellement majeure épargnée à l'institution.
- Automatisation et gain de temps : les solutions d'IA permettent de réduire les charges humaines liées à l'analyse manuelle, tout en augmentant la réactivité.
- Amélioration continue : le modèle développé peut évoluer grâce à l'apprentissage incrémental ou au réentraînement régulier, permettant de s'adapter aux fraudes émergentes.
- Valorisation des données : ce projet illustre l'importance de transformer les données brutes en connaissances exploitables, en leur donnant un sens opérationnel.

Ce travail a permis de mettre en évidence l'intérêt d'intégrer une couche d'intelligence artificielle dans les systèmes de sécurité bancaire. L'implémentation d'un modèle prédictif basé sur des données passées permettrait aux institutions financières d'agir de manière proactive, plutôt que réactive.

- Risques identifiés

- Faux positifs : un modèle trop sensible peut identifier à tort des transactions légitimes comme frauduleuses, entraînant des blocages pour le client.
- Faux négatifs : à l'inverse, un modèle trop laxiste pourrait laisser passer des fraudes non détectées.
- Biais dans les données : si certaines populations ou types de transactions sont sous-représentées dans les données, le modèle peut générer des résultats injustes.
- Dépendance technologique : en cas de mauvaise supervision humaine, la délégation à l'IA peut introduire des décisions erronées.

Ces risques doivent impérativement être pris en compte dans une mise en production réelle, en accompagnant tout algorithme d'un contrôle humain et d'un suivi de performance continu.

- Ouverture et perspectives d'évolution

Plusieurs pistes d'amélioration pourraient être envisagées pour donner suite à ce projet :

- Travailler avec des données en temps réel : pour adapter le modèle aux flux continus de transactions.
- Intégrer du deep learning : comme les réseaux de neurones récurrents (RNN) ou LSTM, qui traitent mieux les séquences temporelles de données
- Développer une interface utilisateur (dashboard) : pour visualiser les prédictions, suivre les alertes et permettre à un analyste de valider ou rejeter les cas détectés.
- Enrichir les données : en intégrant des informations supplémentaires (géolocalisation, historique client, comportement d'achat…).
- Mettre en place un système de réentraînement automatique : pour permettre au modèle de s'adapter aux nouvelles formes de fraude détectées au fil du temps.

Enfin, ce projet peut être décliné à d'autres contextes similaires comme la détection de fraudes sur les comptes clients d'e-commerce, les arnaques aux aides sociales, ou encore la sécurité des plateformes de paiement.