Détection de fraude bancaire

Projet de certification – Implémentation d'un modèle de ML pour la néobanque Fluzz

Contexte & Objectifs

Augmentation des transactions frauduleuses dans les néobanques
Les clients sont facturés pour des achats qu'ils n'ont pas réalisés. Le coût final est supporté par la banque.

Objectifs du projet

- **Analyser** les transactions historiques pour détecter des **schémas frauduleux**
- **Concevoir** un modèle prédictif **robuste** et **éthique**
- **Déployer** un service de détection avec **tableau de bord**

Cycle de vie des données

- **Acquisition & Stockage** : collecte des données, base interne sécurisée
- **Préparation** : nettoyage, normalisation, gestion valeurs manquantes
- **Transformation** : PCA déjà appliquée, feature engineering
- **Entraînement** : orchestration via Airflow pipeline
- **Déploiement** : FastAPI, Docker / Kubernetes
- **Supervision** : Prometheus & Grafana
- **Réentraînement** : intégration de nouveaux jeux de données, amélioration continue

Jeu de données

Jeu de données

- Nombre de lignes: 284807
- Nombre de colonnes: 31

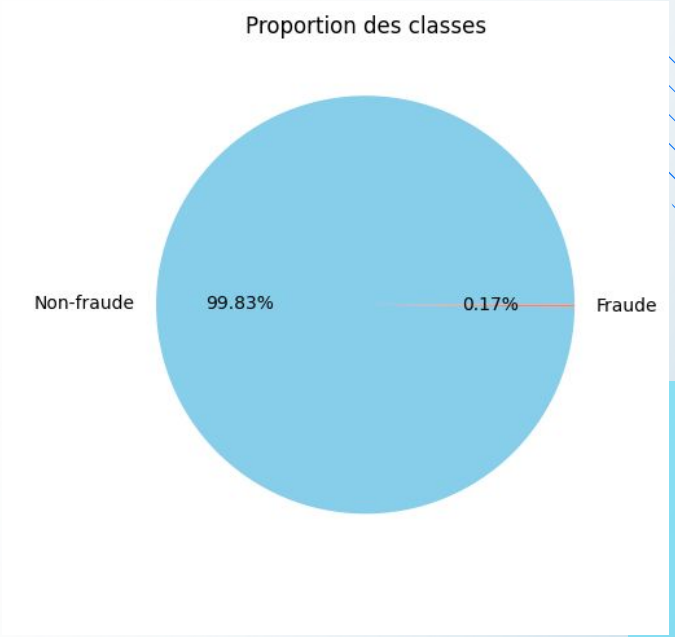
- **Time** contient le montant et l'horodatage de chaque transaction depuis la première du dataset
- **Amount** représente le coût de la transaction
- **V1** à **V28** sont des data pré-process
- **Aucune valeurs null**

Colonne	Null	Type
Time	0	Float
V1 - V28	0	Float
Amount	0	Float
Class	0	Int

Jeu de données

Proportion des classes

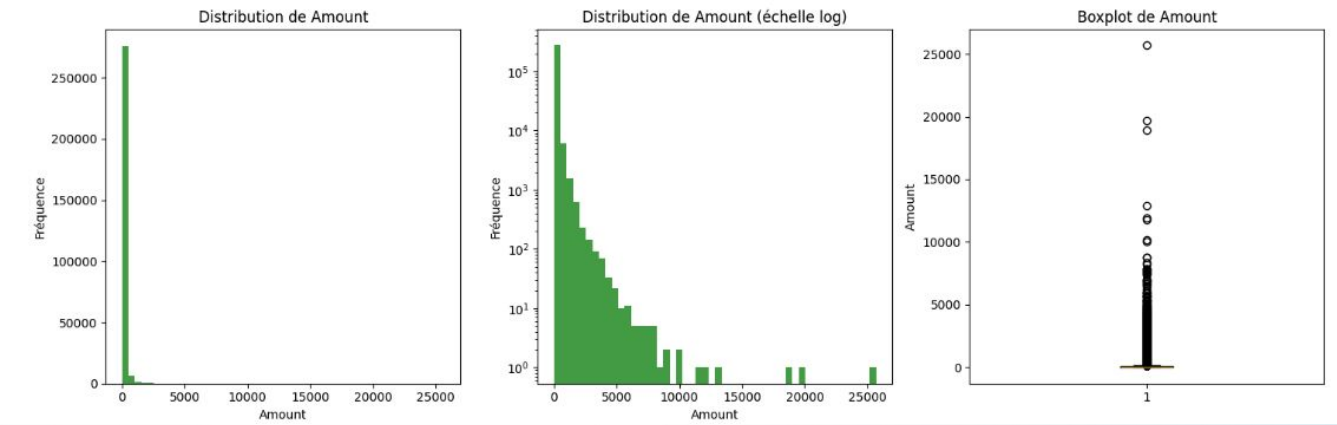
- **284 807 transactions**, **31 variables**, données anonymisées
- Extrêmement déséquilibré : ~**0,17 %** de fraudes



Jeu de données

Analyse sur Amount

- Présente de valeurs aberrantes (box écraser vers le bas)
- La plupart des valeurs sont petites
- Utilisation d'un StandardScaler pour atténuer les données aberrantes



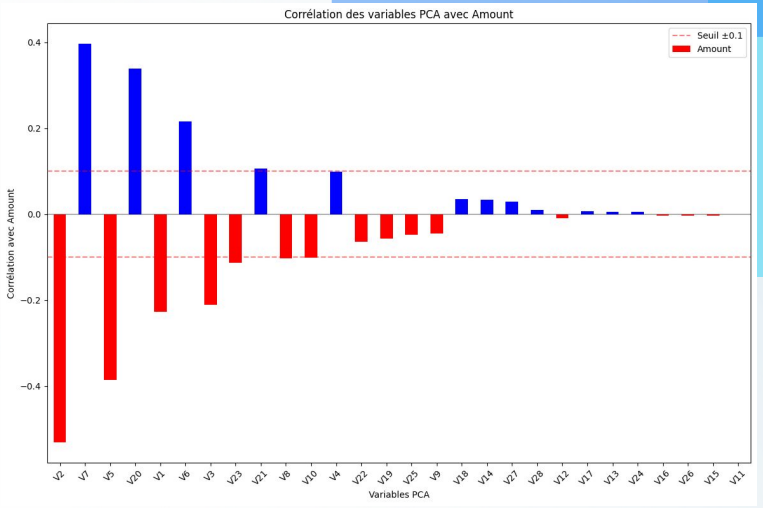
Jeu de données

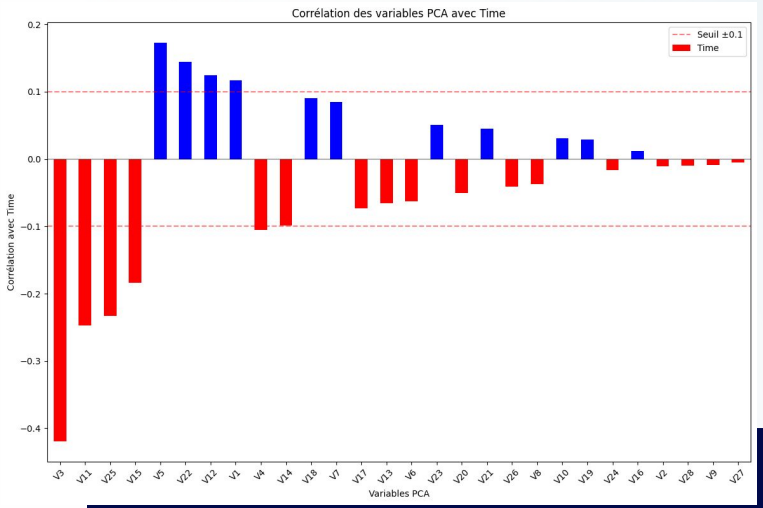
Corrélation avec les valeurs V1 à V28

- Les valeurs **V1** à **V28** n'ont aucune corrélation entre-elle ce qui découle d'une bonne transformation préalable
- Les variables **Amount** et **Time** ont une forte corrélation avec certaines variables (Vxx)

Feature Engineering

- Vérifier que la suppression des variables **Amount** et **Time** sur les différents modèles à analyser
- La **création de variables temporel est limitée** car l'origine temporelle du jeu de données n'est pas connue (potentiel biais)





Évaluation des models

Models

- Régression logistique
- Random Forest
- MLP

Scoring

- Jeu de données déséquilibrer, donc priorité donnée au **Recall (ne pas laisser passer une fraude)**, **Précision (ne pas accuser à tort une transaction normale)**, **F1 (équilibre entre Recall et Précision) et le PR-AUC**
- Utilisation de la **matrice de confusion** pour évaluer l'équilibre entre **faux positifs (FP)** et **faux négatifs (FN)**.

Technique de mise en place

1. Augmenter le dataset avec **SDV (GaussianCopula)**
2. Mise en place de **Pipelines**
3. Optimisation hyperparamètres **GridSearchCV + StratifiedKFold**
4. Training final avec **StratifiedKFold**
5. Vérification des différents scores
    a. Dataset déséquilibrer, donc grande importance au **Recall (ne pas laisser passer une fraude)**, **Précision (ne pas accuser à tort une transaction normale.)**, **F1 (équilibre entre Recall et Précision) et le PR-AUC**
    b. Utilisation de la **matrice de confusion** pour évaluer le compromis entre **faux positifs (FP)** et **faux négatifs (FN)**.

Utilisation de SDV

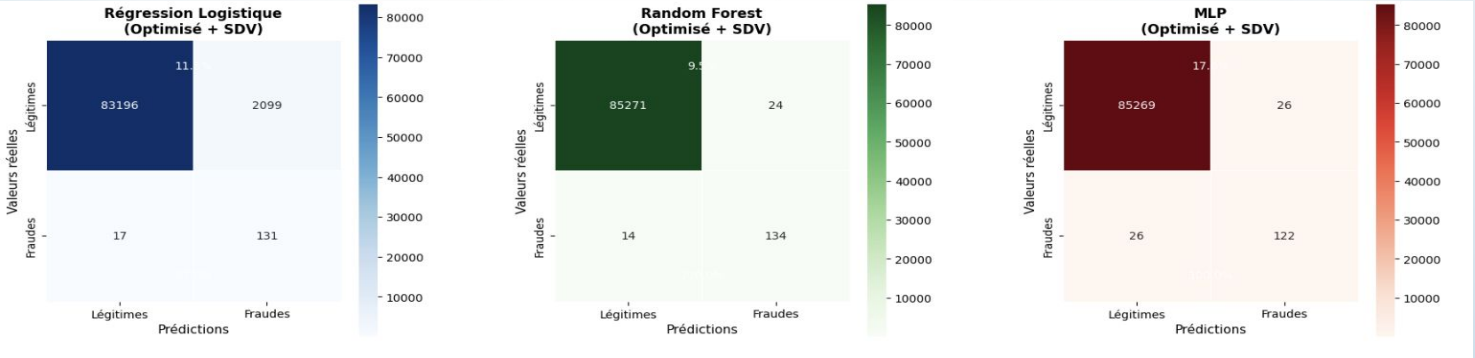
Model	F1	Précison	Rappel	PR-AUC
Régression Logistique	0.12	0.06	0.87	0.70
Random Forest	0.81	0.97	0.70	0.80
MLP	0.80	0.90	0.70	0.78
Régression Logistique + SDV	0.11	0.05	0.88	0.53
Random Forest + SDV	0.92	0.95	0.89	0.92
MLP + SDV	0.81	0.81	0.81	0.84

Après SDV + optimisation

Model	F1	Précison	Rappel	PR-AUC
Régression Logistique	0.11	0.05	0.88	0.53
Random Forest	0.87	0.84	0.90	0.90
MLP	0.82	0.82	0.82	0.85

Analyse

- **Random Forest** est le meilleur compromis
  - **Recall** au dessus des autres
  - **F1 score** équilibrer
  - **Moins** de détection de **faux négatif**



Détection de drift

- **Métriques surveillées** : Précision, Recall, F1 et PR-AUC
- **Déclencheurs** : baisse des performances par rapport au modèle précédent
- **Action** : alerte Grafana → réentraînement via pipeline Airflow
- **Traçabilité** : versioning du modèle (MLflow)

Biais identifiés

- **Dataset très limité** : seulement deux jours de données, ce qui réduit fortement la représentativité.
- **Données déjà traitées en amont** : application préalable de la PCA, ce qui peut entraîner une sur-utilisation ou une redondance de variables déjà exploitées.
- **Données personnelles** : le dataset est déjà dépourvu de données personnelles.

Sécurité

- Dataset déjà **anonymisé** (*PCA*) ; seules les variables **"Amount"** et **"Time"** restent exploitables
- **Datacenter interne** à l'entreprise, couvrant l'intégralité des **demandes en matière de sécurité**
- Application du **principe du moindre privilège** pour la gestion des accès et droits utilisateurs
- Conformité **RGPD** :
    - Conservation des données limitée au strict nécessaire (**minimisation des données**)
    - Droit d'accès, de rectification et de suppression garantit aux clients
    - Traçabilité et auditabilité des traitements mis en place

Pipeline Airflow

Step 1
AIRFLOW

Step 2
Prometheus

Step 3
Grafana

Airflow – Pipeline proposé

1. **Préparation des données d'entrée** : nettoyage, transformation et mise en forme des données brutes.
2. **Entraînement du modèle** : lancement du training selon les configurations définies.
3. **Sauvegarde et versioning** : stockage du modèle entraîné avec gestion des versions.

Grafana – Indicateurs de suivi

- **Taux d'échec des tâches** : proportion de jobs qui n'aboutissent pas.
- **Durée des tâches** : temps moyen/maximum d'exécution des jobs.
- **Échecs de validation des données** : par exemple en cas de détection de *drift* ou d'anomalies.

Pipeline API

Step 1
FastAPI

Step 2
Prometheus

Step 3
Grafana

FastAPI – API de vérification

- **Vérification de la transaction entrante** : chaque transaction est analysée par le modèle avant validation.
- **Blocage en cas de fraude** : si une anomalie est détectée, la transaction est automatiquement refusée.

Prometheus & Grafana – Alertes clés

- **Volumétrie des requêtes** : suivi du nombre de requêtes traitées par le système.
- **Temps de traitement du modèle** : identifier d'éventuels ralentissements ou goulets d'étranglement.
  Nombre de fraudes par heure : surveillance en temps réel pour identifier des pics inhabituels.

Industrialisation et déploiement

2 pipelines distincts :

- **Airflow** → training, scoring, monitoring (**Prometheus + Grafana**)
- **FastAPI** → service de prédiction temps réel + monitoring (**Prometheus + Grafana**)
- **Conteneurisation Docker** → isolation, portabilité, reproductibilité
- **CI/CD avec GitHub Actions** → build & push des images Docker, déploiement automatisé sur Kubernetes
- **Déploiement interne sur Kubernetes** :
  - **Rolling updates** pour assurer le **zéro downtime**
  - **Scalabilité** (adapter la charge automatiquement)
  - **Haute disponibilité** (réplicas, tolérance aux pannes)

Supervision intégrée avec Prometheus & Grafana sur un seul container

Conclusion & Perspectives

- **Prochain modèle candidat** : **XGBoost**
  a. Modèle reconnu pour son efficacité sur des jeux de données déséquilibrés avec un temps d'entraînement inférieur à RandomForest
- **Objectif** : tester XGBoost et comparer aux modèles actuels