IMPLEMENTEZ UN MODELE DE SCORING

Samira MAHJOUB

03/09/2024

SOMMAIRE

- Contexte
- Présentation des données
- Modélisation
- Pipeline de déploiement
- Analyse de Data Drift
- Conclusion

CONTEXTE

Objectif

- **Créer un modèle prédictif fiable et interprétable qui permet d'évaluer la probabilité de remboursement d'un crédit par un client.**
- **Minimiser les erreurs coûteuses.**
- **Mettre en production le modèle avec une solution MLOps complète.**

Mission

- **Développer un modèle de scoring crédit pour "Prêt à dépenser" afin de prédire le risque de défaut de paiement et classer les demandes en crédit accordé ou refusé.**

Problématique

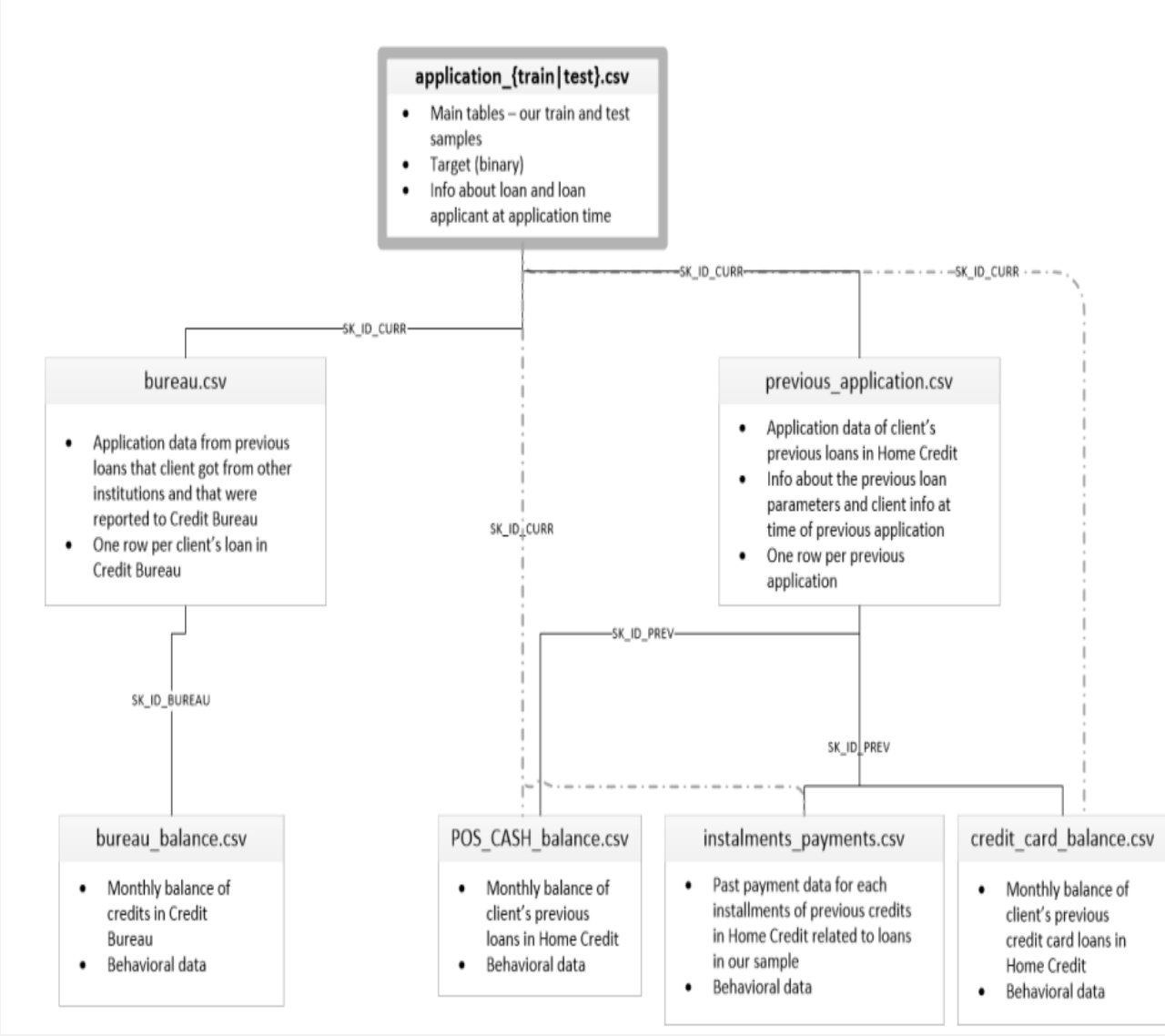
- **Comment concevoir un modèle de scoring qui réduit les risques financiers tout en restant transparent et facile à déployer ?**

Target : 0 = Aucun problème de remboursement | 1 = Défaut

DONNEES FOURNIES

7 Datasets décrivant les historiques bancaires des clients issus de « prêts à dépenser ».

1 Dataset définissant toutes les features.



DATA PREPROCESSING

Analyse des 7 datasets séparément

✓ suppression des features non pertinentes

✓ Analyse des variables manquantes ou aberrantes

✓ Suppression de features fortement corrélées

Analyse exploratoire

✓ Compréhension de la structure des données

✓ Identification des valeurs manquantes et des anomalies

✓ Visualisation des données

✓ Distribution des variables clés

✓ Analyse des corrélations

Preprocessing des Données

✓ Gestion des valeurs manquantes

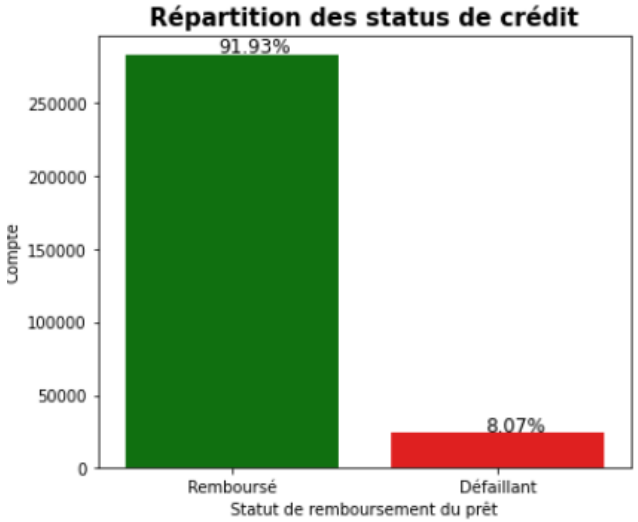
✓ Encodage One Hot des variables catégorielles

✓ Normalisation des variables numériques

✓ Création d'un dataset unifié pour l'entraînement

✓ Traitement des déséquilibres des classes

REPATITION DU TARGET



91.9 % **NON-défaillants**

La plupart des clients remboursent leur crédit

8.1 % **Défaillants**

Seulement 8% de clients sont défaillants

Distribution déséquilibrée

MODELISATIONS

Démarche:

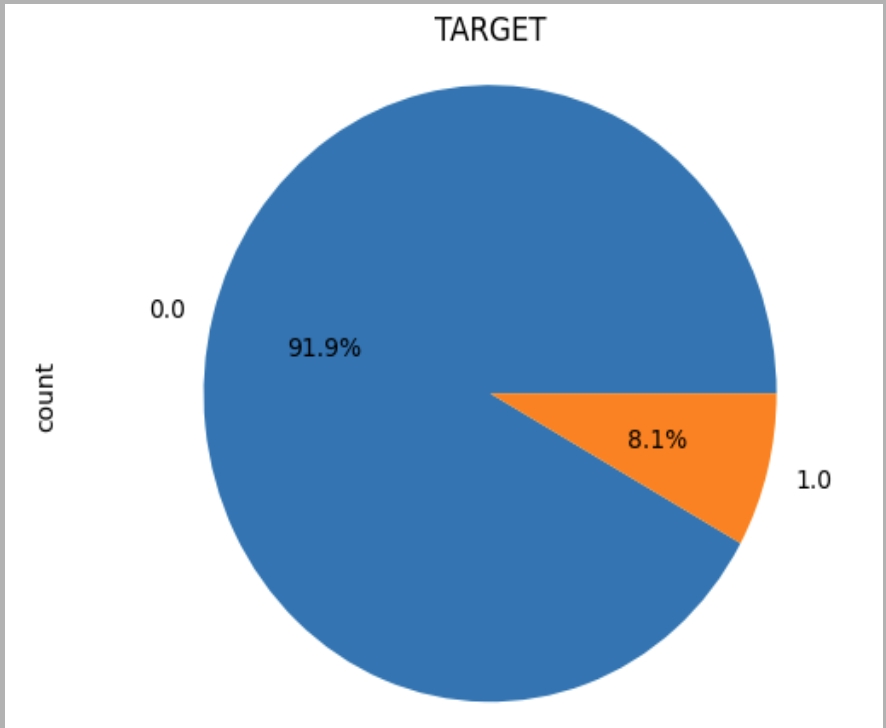
- Rééquilibrage des classes:

✓ Identification du déséquilibre entre les classes (bons vs mauvais clients).

✓ Utilisation de technique **SMOTE** pour rééquilibrer les classes.

- Sampling et séparation des données (80% train, 20% test)
  Prétraitement (Imputation des NaN et Normalisation)
- Entraînement des modèles
- Optimisation du modèle sélectionné d'un point de vue métier
  Interprétabilité (SHAP)

MODELISATIONS



Traitement du déséquilibre des classes:

Problème : Déséquilibre des Classes

- Sous-représentation des clients en difficulté de paiement (8,1%)
- Impact : Risque d'obtenir des résultats biaisés avec des scores trop optimistes.

Solutions :

- Niveau Données : Rééquilibrage via suréchantillonnage (SMOTE) ou sous-échantillonnage.
- Niveau Algorithme : Pénalisation des erreurs sur la classe minoritaire via ajustement de la fonction de perte.

Méthode Adoptée : Rééquilibrage des classes avec SMOTE.

METRIQUES & SCORE METIER

MATRICE DE CONFUSION		
Classe Réelle \ Classe Prédite	**Positif**	**Négatif**
Positif	Vrai Positif (VP)	Faux Négatif (FN) - Erreur de Type II
Négatif	Faux Positif (FP) - Erreur de Type I	Vrai Négatif (VN)

Evaluation des Performances et Score Métier

Métriques Classiques :

- **Accuracy** : % de bonnes prédictions, sensible au déséquilibre des classes.
- **Rappel** : % de la classe positive détectée, utile pour éviter les faux négatifs.
- **Précision** : % de vrais positifs parmi les positifs détectés, critique pour minimiser les faux positifs.
- **F1-score** : Moyenne harmonique de la précision et du rappel, à privilégier en cas de déséquilibre des classes.

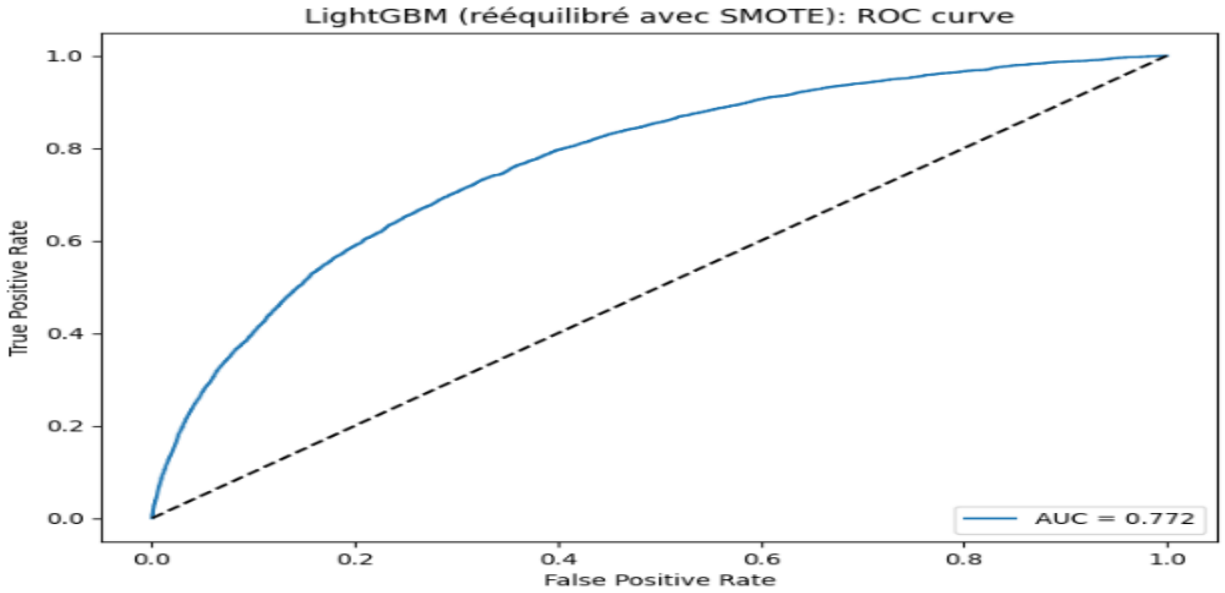
Score Métier :

- **Contexte Métier** : Les faux négatifs (mauvais client prédit bon) sont 10 fois plus coûteux que les faux positifs.
- **Formule** : Création d'un score métier normalisé qui intègre ces coûts différenciés pour optimiser le modèle.

MODELISATIONS

Aspect	Détails
Algorithmes Testés	**1. DummyClassifier (baseline):** Classificateur simple qui sert de référence. **2. Régression Logistique :** Modèle statistique pour la classification binaire. **3. RandomForestClassifier :** Ensemble d'arbres de décision pour améliorer la précision. **4. Light Gradient Boosting Machine :** algorithme de Boosting de gradient basé sur des arbres de décision, conçu pour être à la fois rapide et performant
Gestion du Déséquilibre	- Entraînement sur données déséquilibrées - Rééquilibrage par **SMOTE** suivi d'un nouvel entraînement **SMOTE (Synthetic Minority Over-sampling Technique) :** Crée de nouvelles données pour équilibrer les classes dans un ensemble de données.
Métriques d'Évaluation	- Évaluation des modèles avec l'**AUC** (Area Under the ROC Curve) et la **matrice de confusion**. - Comparaison des performances des modèles sur les données déséquilibrées et rééquilibrées pour sélectionner le meilleur modèle final. - Utilisation de la **précision**, **du rappel**, et du **F1-score** pour une évaluation complète des performances du modèle.

Choix du meilleur modèle



Business score = 41120
AUC: 0.772

	0	1	accuracy	macro avg	weighted avg
precision	0.932993	0.361479	0.906361	0.647236	0.886856
recall	0.967632	0.208661	0.906361	0.588146	0.906361
f1-score	0.949997	0.264589	0.906361	0.607293	0.894665
support	56537.000000	4965.000000	0.906361	61502.000000	61502.000000

	Modèle	Score Business	Temps	Précision	Recall	F-1 Score	Score AUC
0	DummyClassifier (déséquilibré)	49650	0.02	0.919271	0.500000	0.478969	0.500000
1	DummyClassifier (rééquilibré avec SMOTE)	49650	0.00	0.919271	0.500000	0.478969	0.500000
2	Régression Logistique (déséquilibré)	48646	5.43	0.919206	0.510253	0.500479	0.764863
3	Régression Logistique (rééquilibré avec SMOTE)	32629	5.53	0.752187	0.687736	0.567414	0.758483
4	LightGBM (déséquilibré)	48155	9.03	0.919873	0.515209	0.509920	0.776752
5	LightGBM (rééquilibré avec SMOTE)	41120	7.90	0.906361	0.588146	0.607293	0.772133
6	RandomForest (déséquilibré)	49600	177.89	0.919352	0.500504	0.479995	0.726803
7	RandomForest (rééquilibré avec SMOTE)	46602	152.45	0.912637	0.531666	0.539542	0.731165

MODELISATIONS

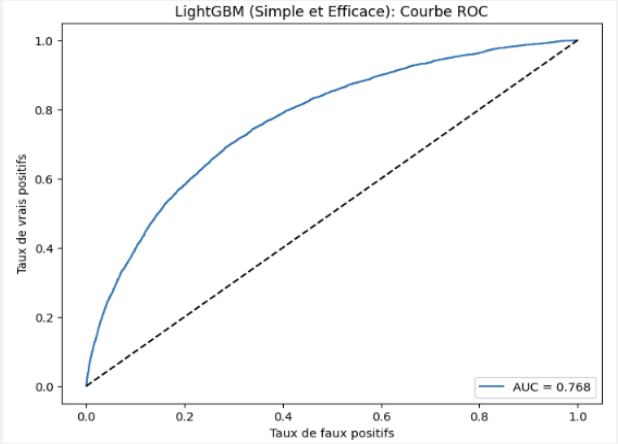
Résultat : LightGBM rééquilibré avec SMOTE donne les meilleurs résultats donc à privilégier pour la production.

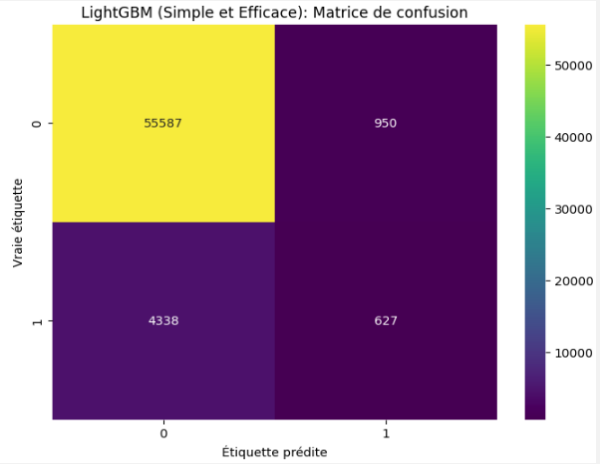
ÉVALUATION DU MODÈLE « LIGHTGBM »

- **Utilisation des hyperparamètres prédéfinis simples pour équilibrer la performance et la rapidité d'entraînement.**
  - Nombre de feuilles (num_leaves):31
  - Le taux d'apprentissage (learning_rate) :0,1
  - Le nombre d'estimateurs (n_estimators): 100

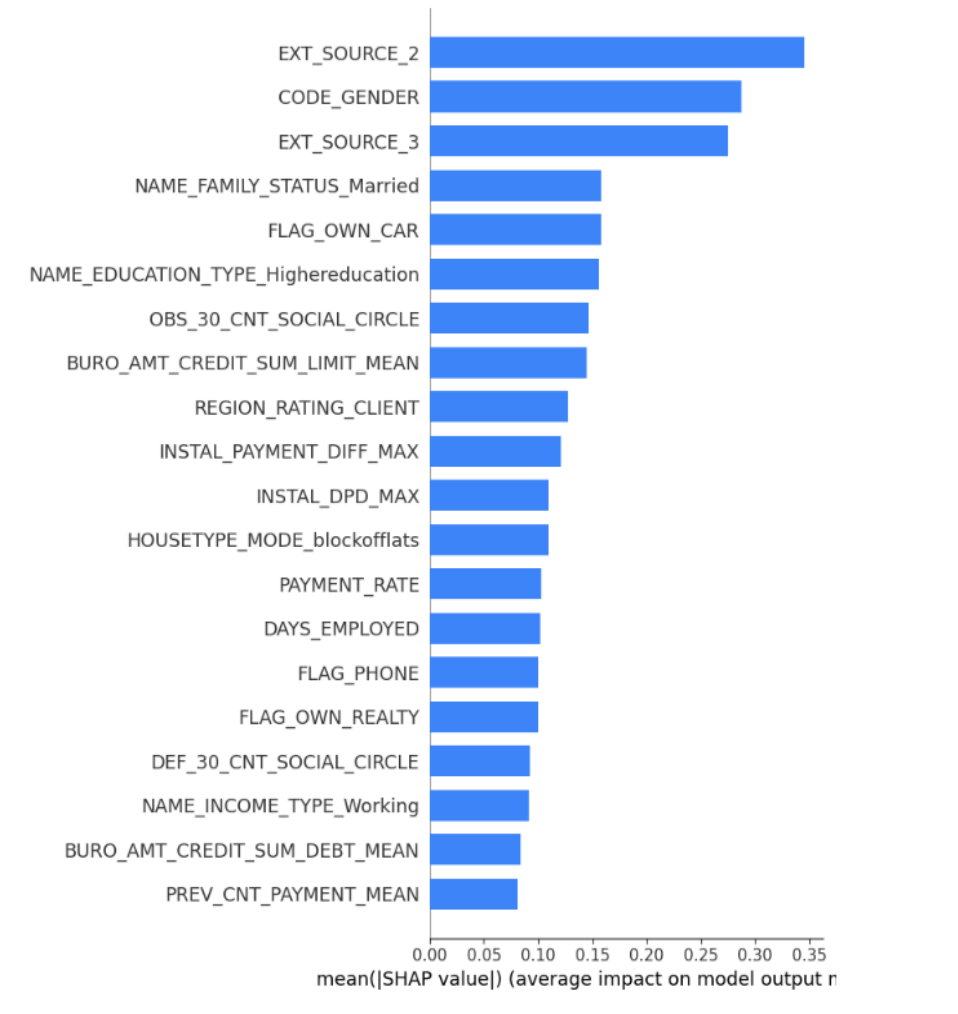
- **Métriques de performance:**

Évaluation	Description
Courbe ROC	AUC (Surface sous la courbe) : Un score de 0,768 indique que le modèle a une bonne capacité à discriminer entre les classes.
Score Métier (Business Score)	Est égal à 44 330, prenant en compte les coûts associés aux faux négatifs et aux faux positifs.
Matrice de Confusion	La matrice de confusion montre la capacité du modèle à distinguer entre les classes (clients défaillants et non-défaillants).
Précision (Precision)	Précision pour les deux classes : 0,9276 pour la classe 0 et 0,3976 pour la classe 1.
Rappel (Recall)	Rappel pour les deux classes : 0,9832 pour la classe 0 et 0,1263 pour la classe 1.
F1-Score	F1-Score pour les deux classes : 0,9546 pour la classe 0 et 0,1917 pour la classe 1.
Accuracy	Accuracy globale : 0,9140, indiquant que le modèle prédit correctement environ 91,4% des échantillons.





EXPLICABILITÉ (FEATURE IMPORTANCE)



Feature Importance :

- Mesure l'impact de chaque caractéristique sur les prédictions du modèle.
- Identifie les variables les plus influentes qui permettent de comprendre et d'expliquer le modèle.
- EXT_SOURCE_2, EXT_SOURCE_3, CODE_GENDER sont les caractéristiques les plus importantes dans le modèle, ce qui signifie qu'elles ont le plus grand impact sur les prédictions.

Vérification du Data Leakage :

- Processus d'analyse des Feature Importance pour détecter des fuites d'informations.
- Processus consistant à s'assurer qu'aucune information du futur (par rapport à la prédiction) n'est utilisée dans les caractéristiques du modèle.

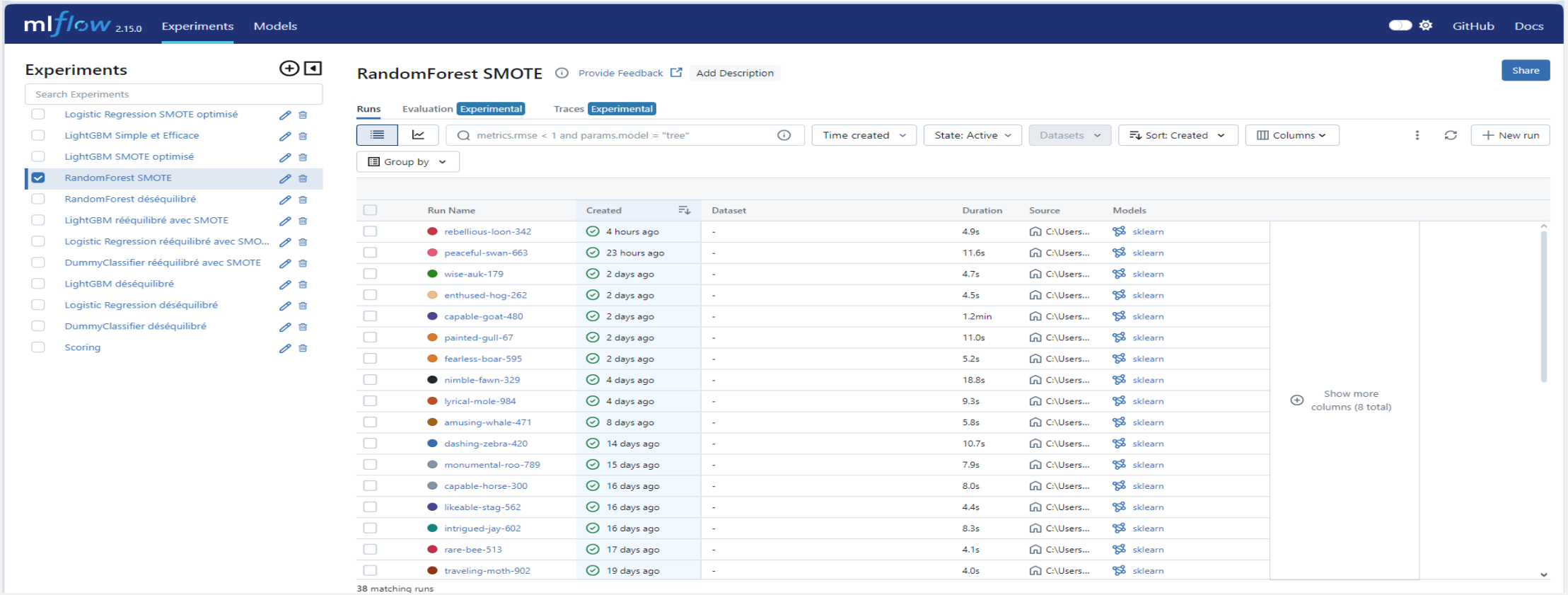
Selon l'analyse visuelle de l'importance des caractéristiques, aucune variable n'apparaît anormalement dominante, ce qui suggère qu'il n'y a pas de fuite de données (data leakage) évidente.

PIPELINE DE DÉPLOIEMENT

MLflow Tracking :

- Outil de suivi des expériences de machine learning.
- Permet de suivre, comparer et reproduire les différentes versions des modèles entraînés.
- Permet l'enregistrement des hyperparamètres ,des métriques de performance, et la gestion des différents modèles testés lors de l'entraînement. Il est directement lié au processus de développement et de déploiement continu.

Visualisation du tracking via Mlflow (Vue d'ensemble)



PIPELINE DE DÉPLOIEMENT

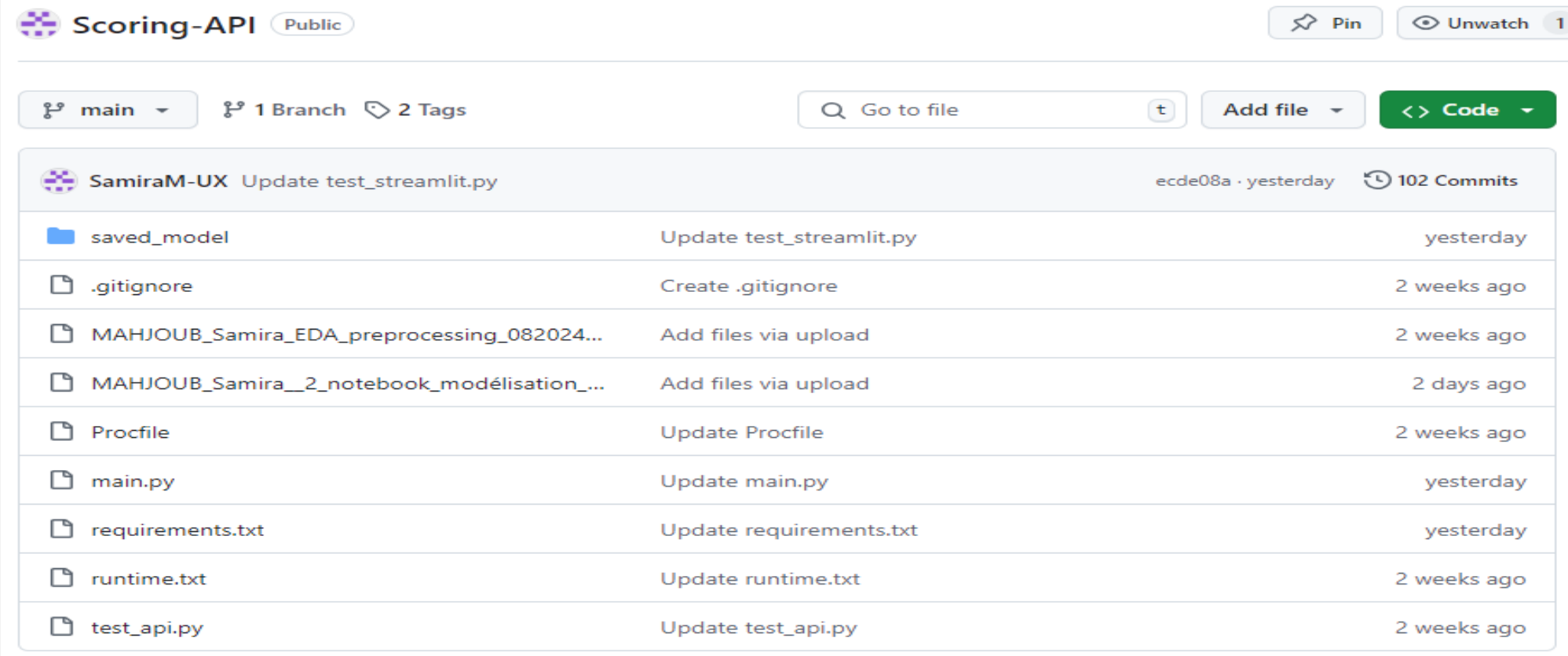
https://github.com/SamiraM-UX/Scoring-API

GitHub

- Une plateforme en ligne pour héberger des projets Git, collaborer avec d'autres développeurs et partager le code.

- **Git :** Un logiciel de gestion de versions qui permet de suivre les modifications dans le code source.

Visualisation du GITHUB (Vue d'ensemble)



PIPELINE DE DÉPLOIEMENT

Tests Unitaires

Tests pour l'API Flask

- **Flask API :** Un micro Framework en Python utilisé pour créer des APIs légères et rapides, permettant de gérer des requêtes HTTP et de renvoyer des réponses basées sur des données ou des prédictions.
- **Pytest** : Un outil de test en Python utilisé pour écrire des tests unitaires simples et efficaces, permettant de vérifier le bon fonctionnement du code.

3 tests utilisées pour l'API Flask:

Tests Unitaires Effectués :

- **Chargement du Modèle :** Vérifie que le modèle est chargé correctement depuis le répertoire spécifié.
- **Vérification des Données :** Assure que le fichier CSV sur les données d'entraînement est chargé avec succès et qu'il n'est pas vide.
- **Test de Prédiction :** Simule une requête GET à l'API avec un identifiant SK_ID_CURR.
- Vérifie que l'API retourne une probabilité valide en réponse à la prédiction.

PIPELINE DE DÉPLOIEMENT

Déploiement de notre application sur Streamlit

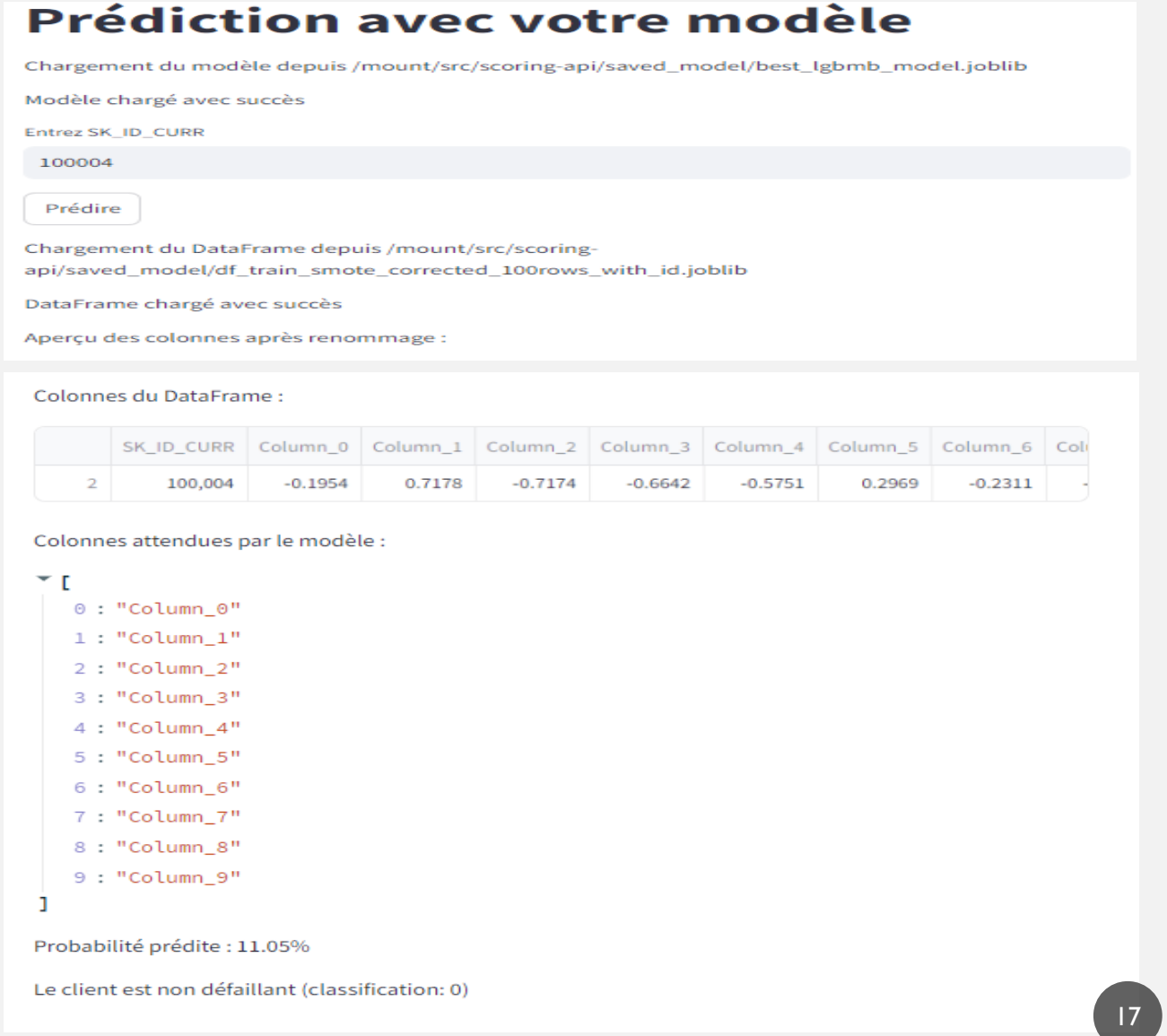
Streamlit : Une bibliothèque Python pour créer rapidement des applications web interactives.
Objectif : Rendre notre modèle de scoring accessible en ligne via une interface simple et interactive.

Étapes de déploiement avec Streamlit Sharing :

1. Préparation de notre projet avec tous les fichiers nécessaires (modèle, script Streamlit, requirements.txt).
2. Push du projet sur GitHub.
3. Connection à Streamlit Cloud et sélection de notre dépôt et configuration de l'application.
4. Lancement de l'application via Streamlit Cloud pour générer une URL publique.

Déploiement du modèle avec Streamlit :

Local : Lancement de l'application en local avec streamlit run app.py.
En ligne : Déploiement sur une plateforme Streamlit Cloud pour générer une URL publique accessible à tous.



PIPELINE DE DÉPLOIEMENT

Déploiement de notre application sur Streamlit

Accédez à l'application et testez l'API :

https://scoring-api-bgzqtnd9kebfvi7c6hipe5.streamlit.app/

ANALYSE DE DATA DRIFT

Data Drift : C'est le phénomène où les caractéristiques des données évoluent entre l'entraînement du modèle et son utilisation en production, pouvant entraîner une dégradation de la performance du modèle.

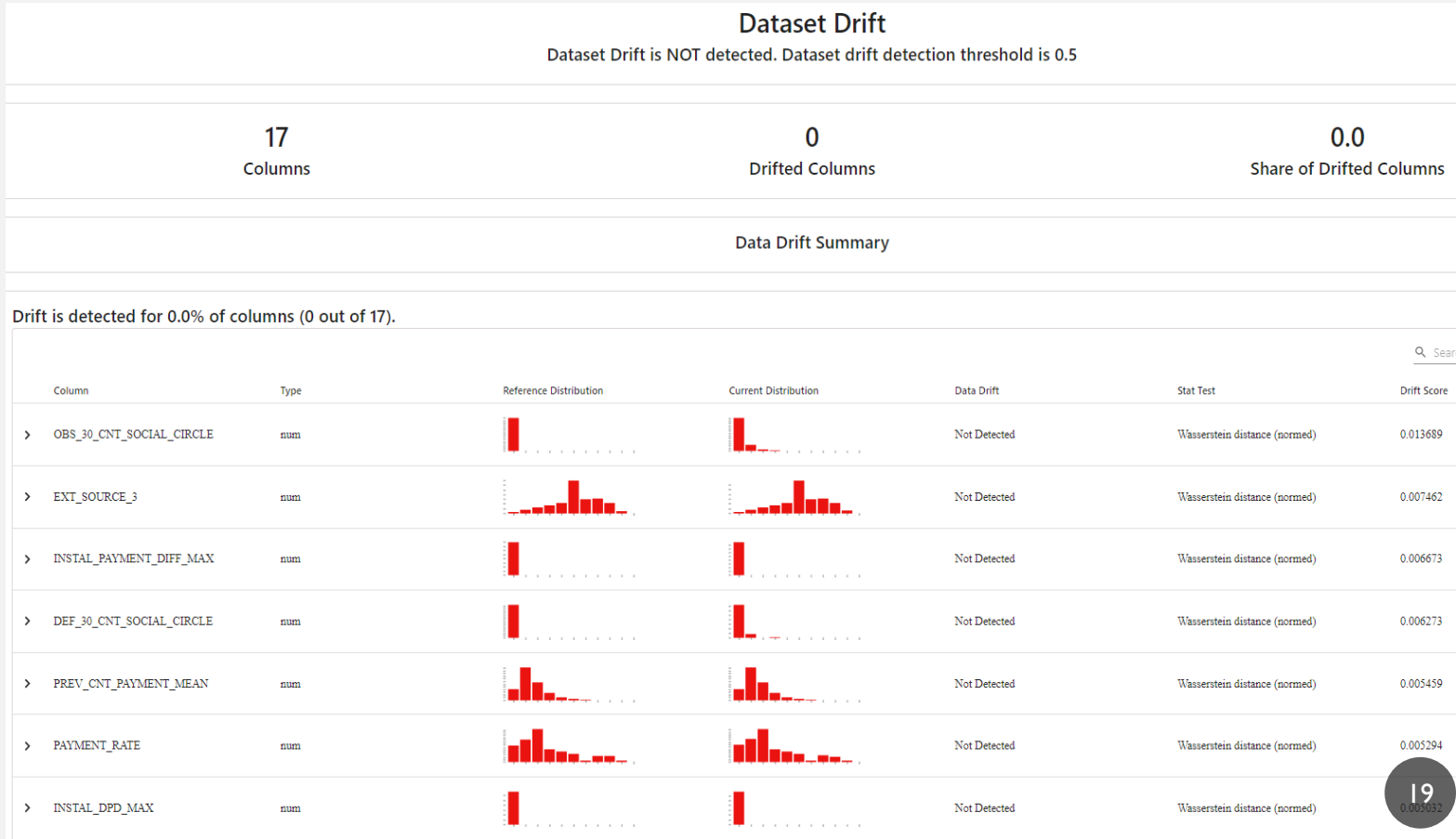
Evidently : une bibliothèque Python. Elle est conçue pour aider à surveiller et analyser la performance des modèles de machine learning en production, en particulier en détectant les dérives de données (data drift) et les dérives de concept (concept drift).

➢ **Il est crucial donc de suivre un modèle en production et notamment :**

✓ **La dérive et la qualité des données .**
✓ **La dérive de la cible ainsi que la performance du modèle**

EVIDENTLY AI

- Aucun data drift (dérive des données) n'a été détecté dans les 17 colonnes analysées.
- Cela signifie que les distributions des données actuelles sont cohérentes avec les données de référence, indiquant que le modèle devrait continuer à fonctionner de manière fiable sans dégradation de performance liée à un changement dans les données.



Dataset Drift

Dataset Drift is NOT detected. Dataset drift detection threshold is 0.5

17	0	0.0
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary

Drift is detected for 0.0% of columns (0 out of 17).

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
OBS_30_CNT_SOCIAL_CIRCLE	num			Not Detected	Wasserstein distance (normed)	0.013689
EXT_SOURCE_3	num			Not Detected	Wasserstein distance (normed)	0.007462
INSTAL_PAYMENT_DIFF_MAX	num			Not Detected	Wasserstein distance (normed)	0.006673
DEF_30_CNT_SOCIAL_CIRCLE	num			Not Detected	Wasserstein distance (normed)	0.006273
PREV_CNT_PAYMENT_MEAN	num			Not Detected	Wasserstein distance (normed)	0.005459
PAYMENT_RATE	num			Not Detected	Wasserstein distance (normed)	0.005294
INSTAL_DPD_MAX	num			Not Detected	Wasserstein distance (normed)	0.005032

CONCLUSION
AXES D'AMÉLIORATION ET PERSPECTIVES

Les données disponibles ont permis de développer un algorithme de classification performant, avec une modélisation optimisée utilisant l'algorithme LGBM. Cependant, pour aller plus loin et améliorer ces résultats, les axes suivants peuvent être explorés :

✓ Acquérir une meilleure connaissance du secteur bancaire, ce qui permettrait de raffiner le processus de traitement des données et d'ajuster les variables en fonction des spécificités du domaine.

✓ Approfondir la compréhension des variables clés, comme celles liées aux sources externes (EXT_SOURCE), afin d'expliquer et d'interpréter plus précisément le modèle.

✓ Collaborer étroitement avec les équipes métier pour affiner la métrique d'évaluation et définir une fonction de coût plus adaptée aux besoins réels de l'entreprise.

✓ Renforcer les compétences en développement logiciel pour sécuriser et automatiser entièrement le déploiement, garantissant ainsi la robustesse et la fiabilité de l'application en production.

Ces points d'amélioration permettront de renforcer la précision, la fiabilité et l'adaptabilité de la solution mise en place.

MERCI POUR VOTRE ATTENTION !