Projet 7 :

Implémenter un modèle de scoring

Septembre 2024

Lokman AALIOUI

Contexte

L’entreprise prêt à dépenser souhaite mettre en place un outil de scoring crédit qui permet de calculer la probabilité qu’un client rembourse son crédit ou non, et donc de décider automatiquement si le crédit est accordé ou refusé.

Mission

1

Développement d’un modèle de scoring :
Le développement d'un modèle de scoring vise à créer un système capable d'évaluer et de prédire le risque d'un individu, en utilisant les données historiques et les méthodes de Machine Learning.

2

Déploiement et gestion :
Le déploiement du modèle de scoring implique l'intégration du modèle dans une infrastructure cloud, permettant un accès en temps réel aux analyses, une mise à jour dynamique des données.

Feuille de route

1

Développement d’un modèle de scoring :

- Mise en place de l'environnement
- Préparation des données
    - Présentation du jeu de données
    - Analyse exploratoire
    - Feature Engineering
- Création d’un score métier
- Création et comparaison des modèles
    - Modélisation
    - Détermination du meilleur seuil de décision
    - Analyse des feature importance locale et globale

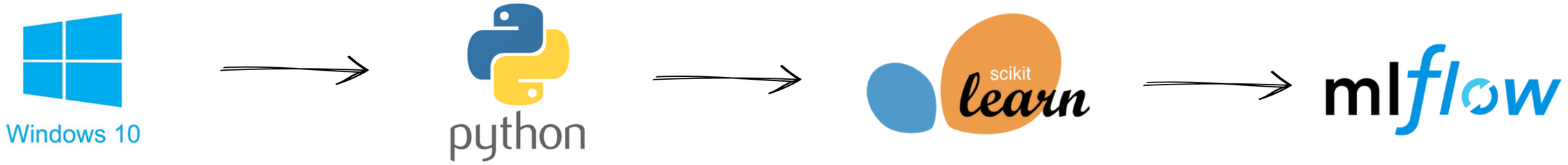
2

Déploiement et gestion :

- Etude du Data Drift
- Déploiement de l’API dans le cloud
    - Via Github
    - Création des fonctions de l’API
    - Tests Pytest
- Création de l’interface de scoring client

1ère partie :
Développer un modèle de scoring

Mise en place de l’environnement



- Création d’un répertoire local
- stockage des notebooks
- stockage des fichiers de données
- stockage des fichiers de configuration

- Création d’un environnement virtuel python 3.12.6
- installation des librairies inter-compatibles

- Activation de mlflow.sklearn.autolog pour enregistrer lesentrainements des modèles
- les hyperparamètres
- les métriques d’évalutaion
- les matrices de confusion
- les scores métiers

- Exécution de MLflow via la commande “mlflow ui”
- connexion à l’interface mlflow sur naviguateur via http://127.0.0.1:5000
- visualisation des modèles

Sélection des données

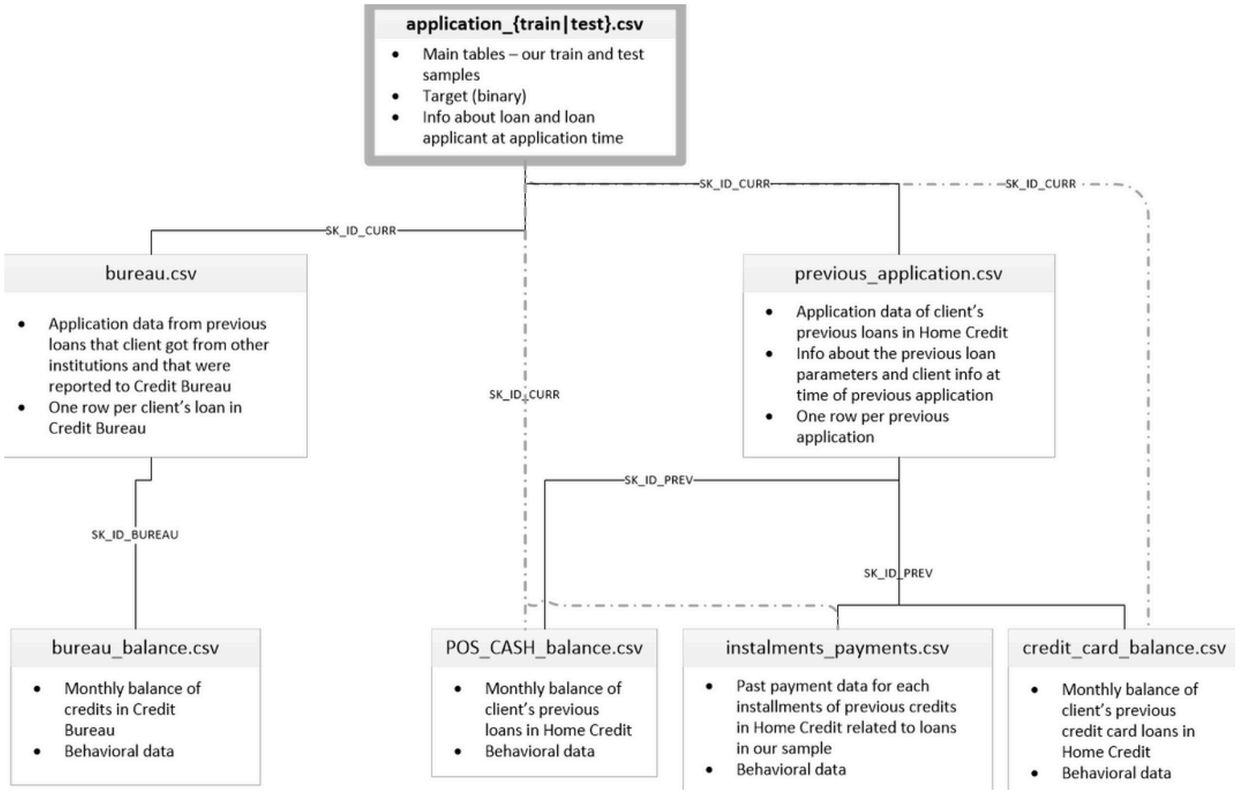
Les données :

Le jeu de données présente 10 fichiers, seuls 5 fichiers sur les 10 seront utilisés pour construire notre modèle.

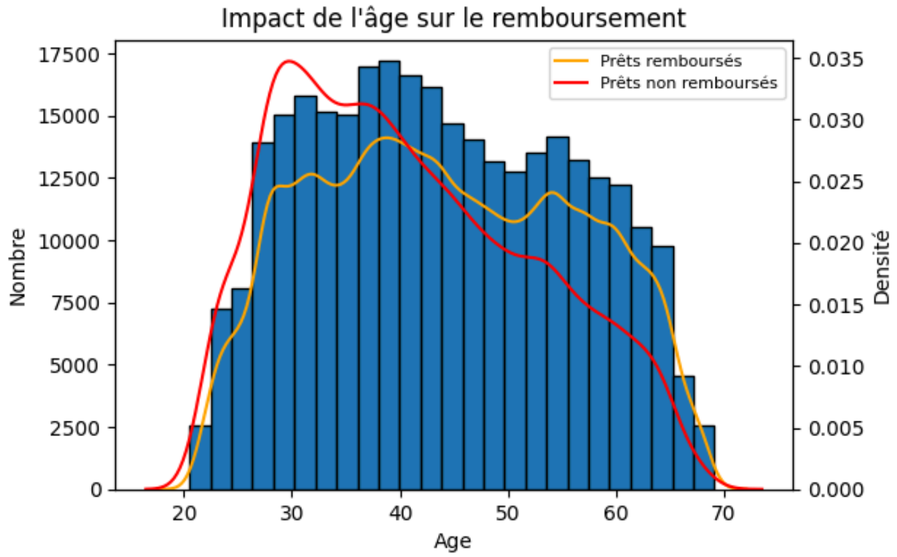
- application_train.csv : données avec TARGET
- bureau.csv : infos d'autres institutions financières
- bureau_balance.csv : complément d'infos d'autres institutions financières
- credit_card_balance.csv : solde mensuel des crédits conso
- POS_CASH_balance.csv : solde mensuel des moyens de paiement

application_train.csv sera notre fichier principal, les autres seront fusionnés à ce dernier.

Relations entre les données :

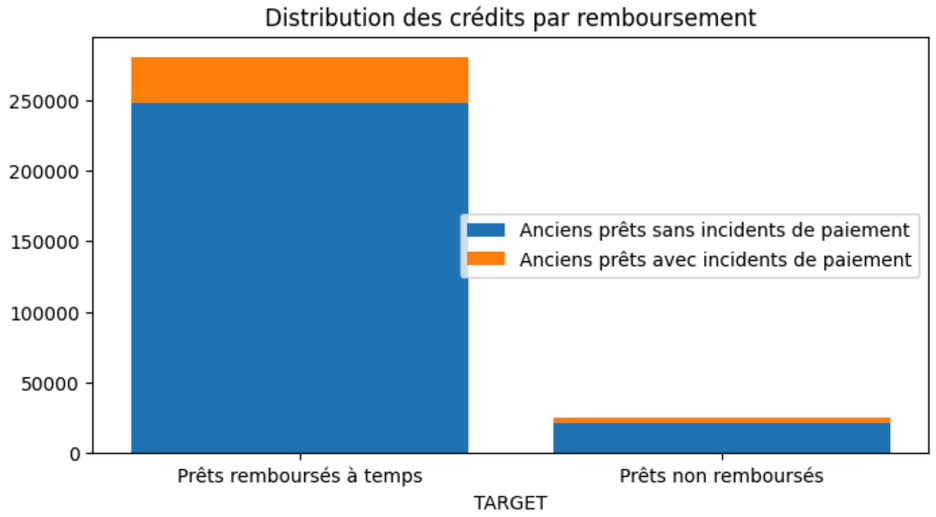


Analyse Exploratoire



Impact de l’âge sur le remboursement :

Plus les clients sont jeunes, plus la probalité qu’ils ne remboursent pas leurs prêts est élevée.



Impact de l’historique sur le remboursement :

L’historique des incidents de paiement n’a que peu d’incidence sur les remboursements futurs.

Préparation des données

Transformations sur le fichier principal :

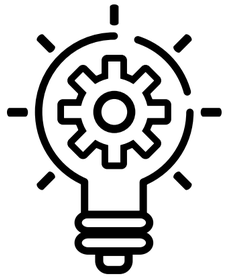
- Remplacer les valeur positives d'ancienneté au travail par NaN
- Remplacer les valeurs XNA par NaN
- Remplacer les valeurs numériques manquantes par la moyenne
- Remplacer les valeurs catégorielles manquantes par la plus fréquente
- Convertir les variables catégorielles avec Label Encoder et OneHotEncoder
- Création d'une variable polynomiale pour les 4 variables les plus corrélées (EXT_SOURCE x3 et DAYS_BIRTH)

Transformations sur les fichiers secondaires :

- Supprimer les données inutiles
- Convertir les variables qualitatives avec OneHotEncoder
- Indiquer les incidents de paiement pour tous les crédits
- Fusionner les variables avec moyenne, médiane, somme, etc...
- Fusionner les fichiers secondaires au principal

Feature Engineering :

- DAYS_EMPLOYED_PERCENT : pourcentage de jours employés par rapport à l'âge
- CREDIT_INCOME_PERCENT : pourcentage du montant du crédit par rapport au revenu
- INCOME_PER_PERSON : pourcentage du montant du crédit par individus composant le foyer
- ANNUITY_INCOME_PERCENT : pourcentage de la rente du prêt par rapport au revenu
- PAYMENT_RATE : pourcentage du montant du crédit remboursé annuellement



Création d’un score métier

1. Objectif :

Évaluer et comparer des modèles de classification pour gérer le déséquilibre entre bons et mauvais clients, en tenant compte des coûts métier associés aux erreurs.

2. Modèles de Classification

- LGBMClassifier
- LinearSVC
- RandomForestClassifier
- LogisticRegression

3. Techniques de Rééchantillonnage

Utilisées sur 80 % des données initiales :

- Random Under-Sampling : Réduction de la classe majoritaire.
- Random Over-Sampling : Augmentation de la classe minoritaire.
- SMOTE : Génération d’échantillons synthétiques pour la classe minoritaire.

4. Optimisation des Hyperparamètres

GridSearchCV de scikit-learn pour déterminer les meilleurs hyperparamètres.

5. Évaluation des Modèles

- Utilisation des 20 % restants pour tester les modèles.

Score métier calculé à partir de la matrice de confusion

$$\text{score métier} = \frac{3 \cdot TN - 10 \cdot FN - FP + TP}{TN + FN + FP + TP}$$

6. Importance des Erreurs

Les faux négatifs (FN) sont pénalisés sévèrement (-10) car ils entraînent des pertes en capital.

Type d'Erreur	Coût
FN (Mauvais prédit bon)	-10
FP (Bon prédit mauvais)	-1
TP (Bon prédit bon)	+1
TN (Mauvais prédit mauvais)	+3

Simulation des Modèles

1. Conclusion des Modèles

Meilleur modèle : LGBMClassifier avec données équilibrées (SMOTE)

2. Préparation du Jeu de Données Final

Déploiement : L'API sera déployée sur une plateforme cloud via une WebApp (Heroku)

3. Stratégies de Réduction de Taille

- Sélectionner les données :

80 % pour l'entraînement, 10 % pour le jeu final

- Feature Selection :

Utiliser les features importance du LGBMClassifier

Conserver 90% des features importance, réduisant le jeu à 115 variables

- Compression :

Compresser le jeu de données final en ZIP

4. Préparation du Modèle Pré-Entraîné

Cohérence des Données : Réentraîner le modèle sur des données réduites.

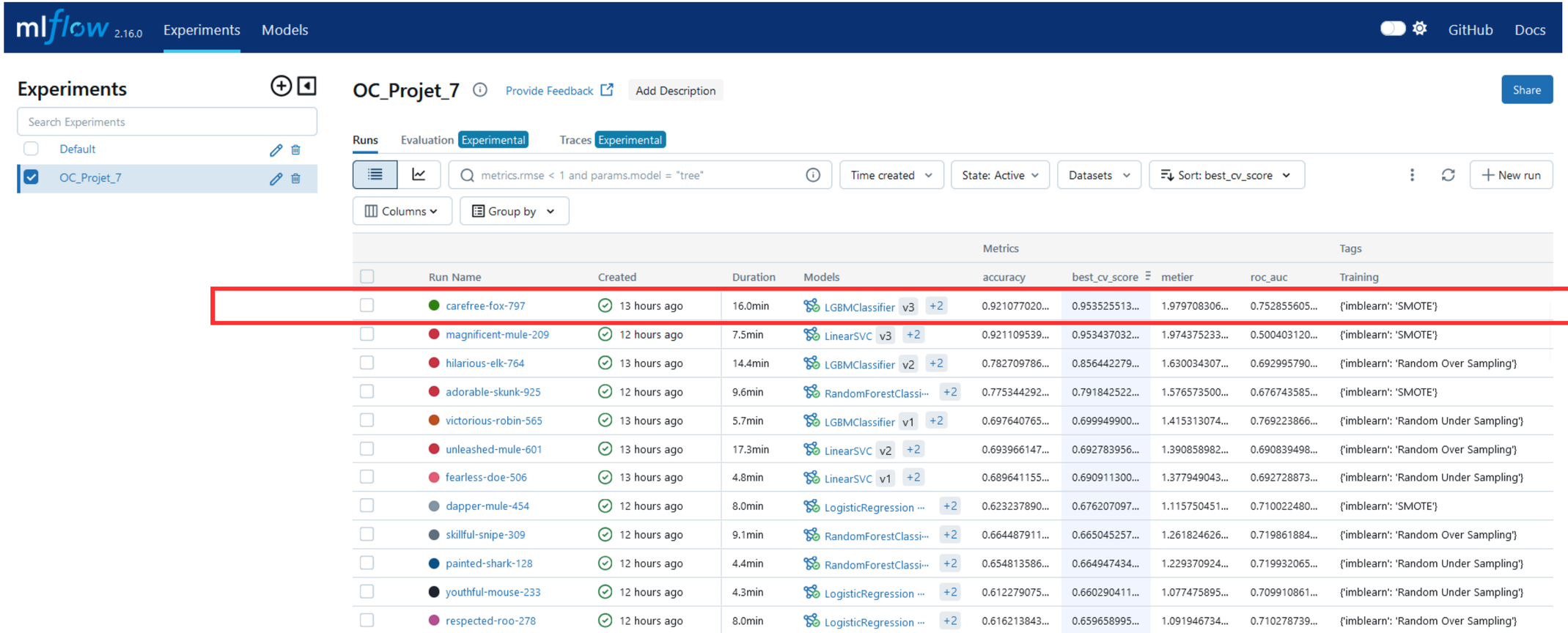
Étapes :

1. Appliquer la même feature selection aux données d'entraînement
2. Réentraîner le modèle
3. Enregistrer via MLflow (mlflow.sklearn.save_model)

5. Interface MLflow :

http://127.0.0.1:5000

Interface MLflow

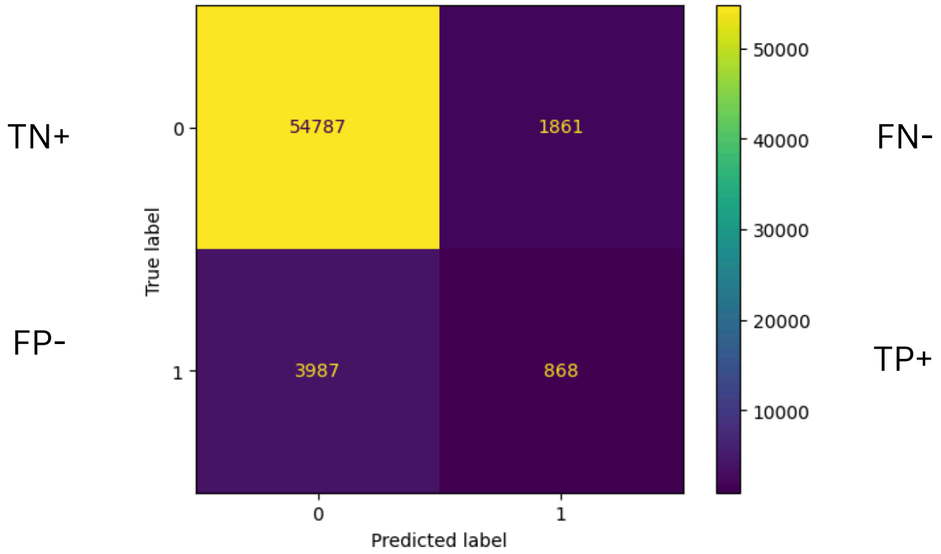


Détermination du seuil de décision optimal

Méthodologie :

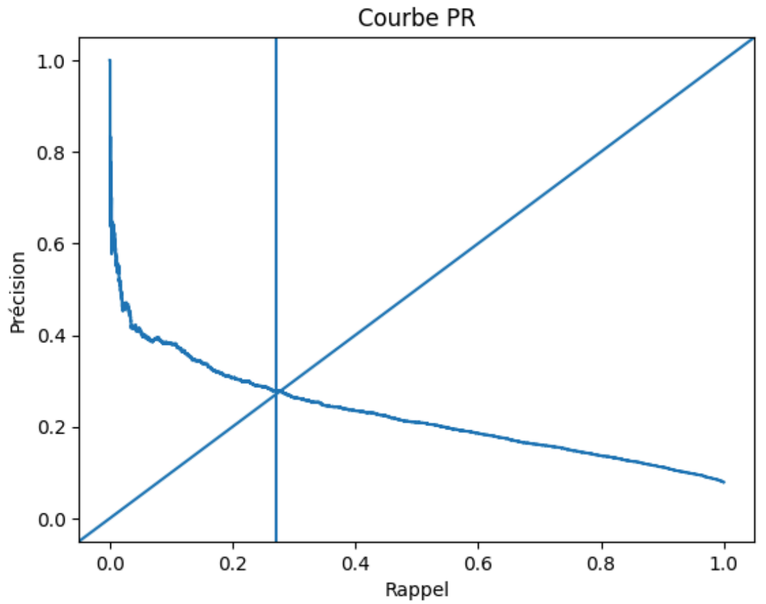
Le choix du seuil de décision se fait en priorisant les objectifs suivants, par ordre d'importance :

1. Minimiser les Faux Négatifs (FN) : car perte en capital
2. Minimiser les Faux Positifs (FP) : car manque à gagner
3. Maximiser les Vrais Positifs (TP) : identifier correctement les bons clients
4. Maximiser les Vrais Négatifs (TN) : identifier correctement les mauvais clients



Indicateurs de Performance :

L'objectif est de minimiser les Faux Négatifs (FN) tout en maximisant les Vrais Négatifs (TN)



Seuil de décision optimal :

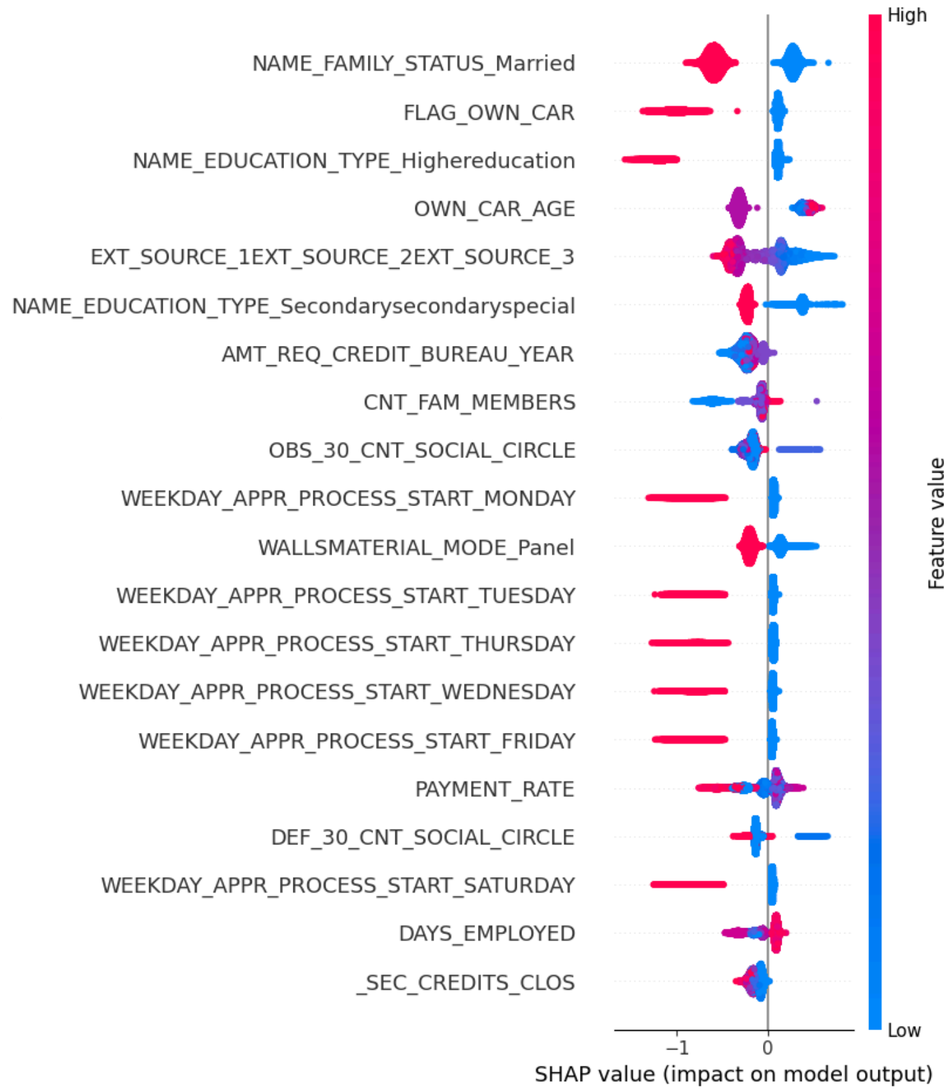
Cela conduit à un seuil de décision recommandé de 0,27

Feature Importance Globale

Méthode SHAP :

SHAP (SHapley Additive exPlanations) est une approche basée sur la théorie des jeux qui permet d'expliquer les prédictions des modèles en attribuant à chaque caractéristique une explication proportionnelle à son importance pour la prédiction.

Features les plus explicatives par SHAP :



Top 3 des features les plus explicatives :

- NAME_FAMILY_STATUS_Married : Le statut marié réduit relativement la probabilité de faire défaut
- FLAG_OWN_CAR : Posseder une voiture réduit la probabilité de faire défaut
- NAME_EDUCATION_TYPE_... : Avoir fait des études supérieur réduit fortement la probabilité de faire défaut

Feature Importance Locale

Méthode :

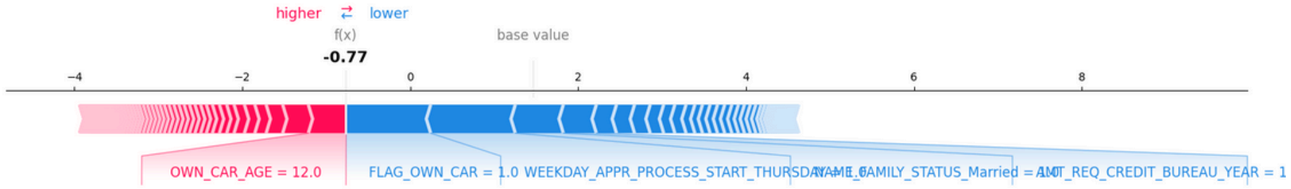
Nous visualisons la Feature Importance pour 2 cas, le premier cas avec une TARGET = 1 et pour une TARGET =2. La valeur de base est identique pour les 2 classes car les données sont équilibrées.

Base value = 1.4628 :

Les valeurs SHAP de chaque variables, influent dans le sens des flèches à partir de la valeur de base jusqu'à la prédiction.

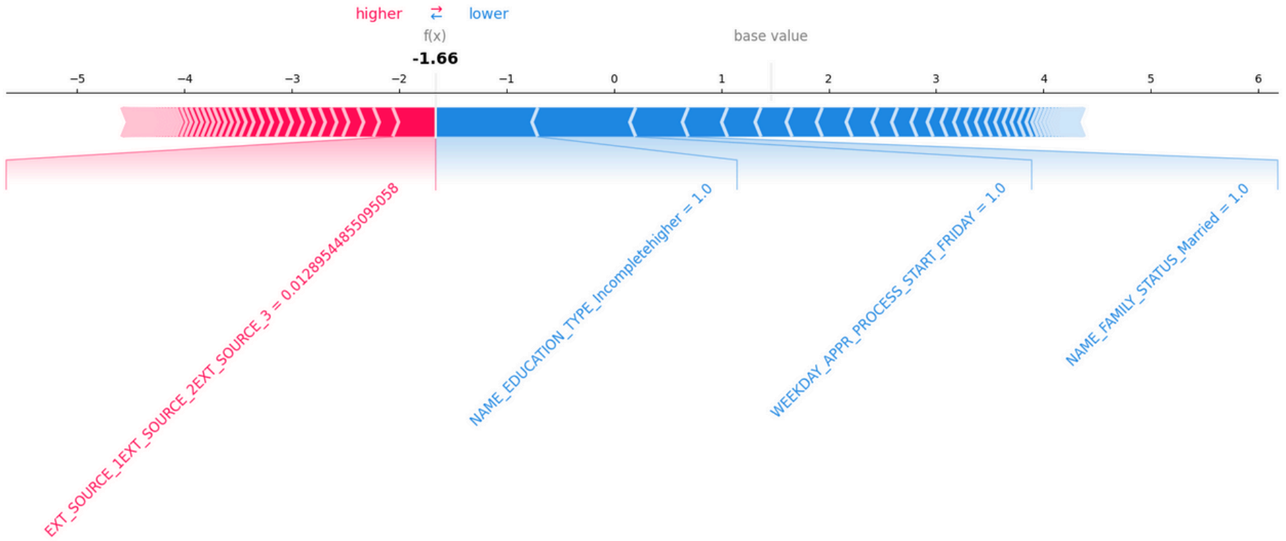
Exemple Target = 1

Posséder une voiture influe dans un sens sur la prédiction, tandis que l'âge de la voiture influe dans l'autre sens sur la prédiction



Exemple Target = 0

Avoir fait des études supérieur influe positivement sur la prédiction, tandis que les valeurs externes influent négativement

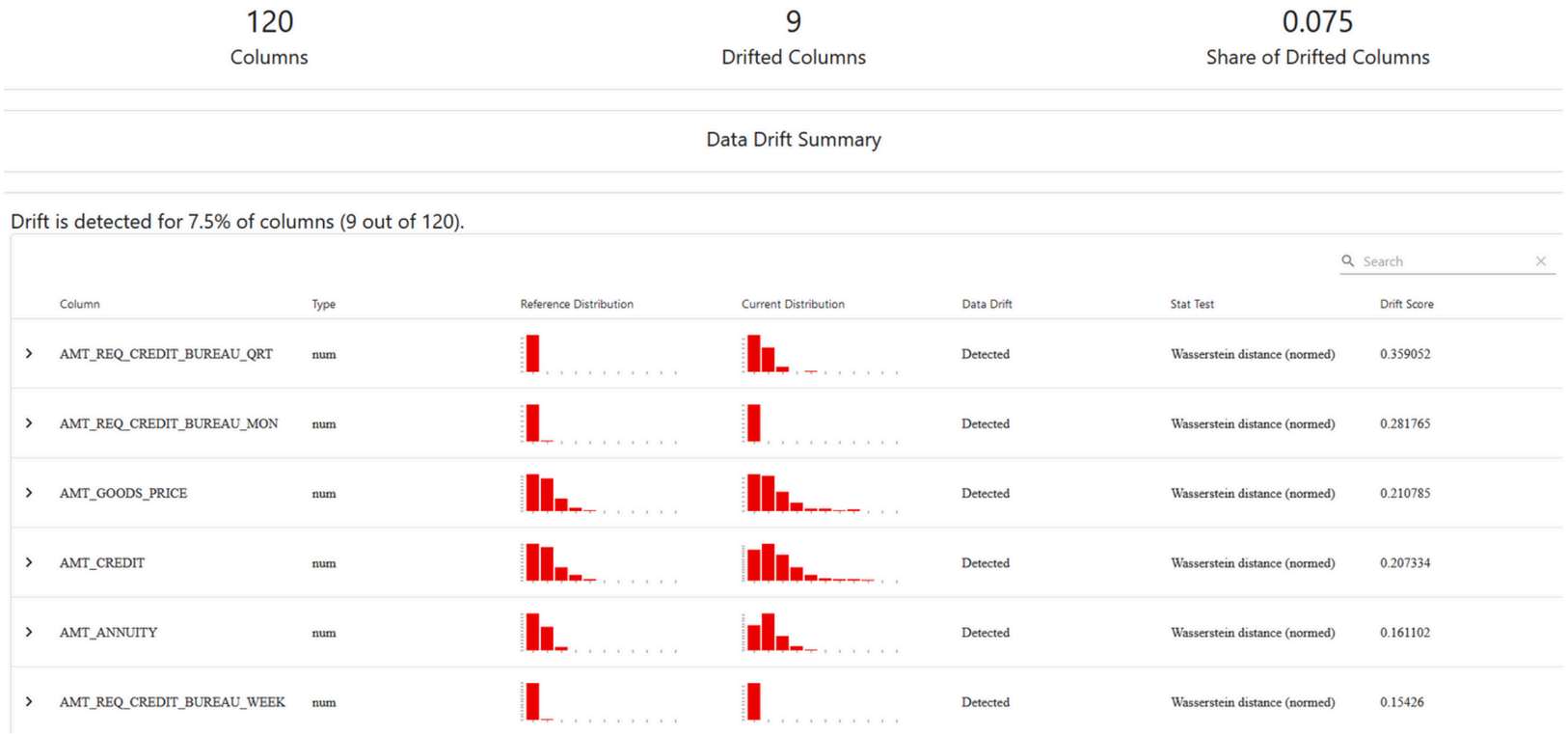


2ème partie :
Déployer le modèle de scoring

Analyse du Data Drift

Evidently :

- Le data drift désigne un changement dans les données utilisées par un modèle entre la phase d'entraînement et celle de production. Cela peut dégrader les performances du modèle, car il n'est plus adapté aux nouvelles données.
- Le rapport d'Evidently fournit une analyse détaillée des changements dans la distribution des données sur une période, permettant de détecter des divergences entre les données d'entraînement et les données actuelles pour prévenir les dégradations de performance des modèles. Ici, nous comparerons les données application_train et les données application_test.



120	9	0.075
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary

Drift is detected for 7.5% of columns (9 out of 120).

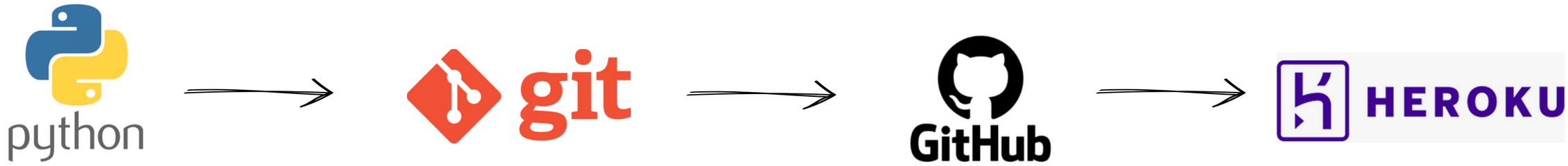
Search

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
AMT_REQ_CREDIT_BUREAU_QRT	num			Detected	Wasserstein distance (normed)	0.359052
AMT_REQ_CREDIT_BUREAU_MON	num			Detected	Wasserstein distance (normed)	0.281765
AMT_GOODS_PRICE	num			Detected	Wasserstein distance (normed)	0.210785
AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.207334
AMT_ANNUITY	num			Detected	Wasserstein distance (normed)	0.161102
AMT_REQ_CREDIT_BUREAU_WEEK	num			Detected	Wasserstein distance (normed)	0.15426

Conclusions du rapport :

- Ce rapport indique que sur 120 colonnes, 9 présentent un datadrift plus ou moins important avec une dérive pouvant allé jusqu'à 36%.
- Il s'agira de refaire l'entrainement du modèle régulièrement afin de s'assurer de sa pertinence.

Mise en place de l’environnement



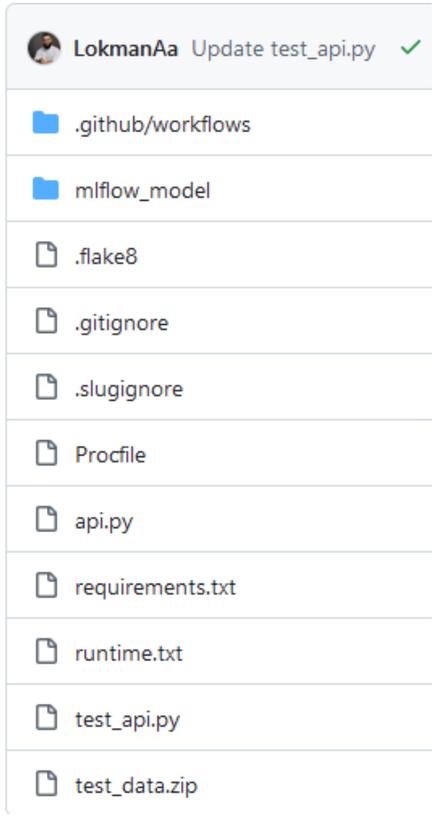
- FastAPI est le framework principal
- La route GET est définie avec @app.get()
- Pydantic est utilisé pour la validation des données

- Initialisation du dépot Git avec git init
- Commit avec git add .

- Création d’un dépot Github main
- Lier le dépot local et le Github
- Poussée du code avec git push

- Connexion avec heroku login sur le terminal
- Connexion avec le repository Github via l’interface Heroku
- Déploiement

Déploiement de l’API



Repository Github :

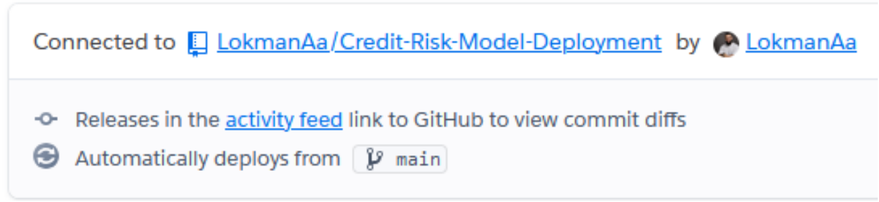
- Script Python de l'API
- Script de test de l'API avec Pytest
- Modèle
- Fichier de données de test

Fichiers de gestion :

- .gitignore : contrôle des fichiers GitHub
- .slugignore : contrôle des fichiers Heroku
- Procfile : gestion du lancement de l'API
- requirements.txt : dépendances Heroku
- runtime.txt : version Python Heroku

Workflows GitHub Actions :

- Déclenchement : à chaque push ou pull_request sur la branche "main"
- Dependabot Updates : surveille les vulnérabilités des dépendances
- flake8 + pause + pytest (Python application):
- Lint with flake8 : vérification conformité PEP8
- Sleep for 120 seconds : délai pour déploiement
- Test with pytest : exécution des tests de l'API



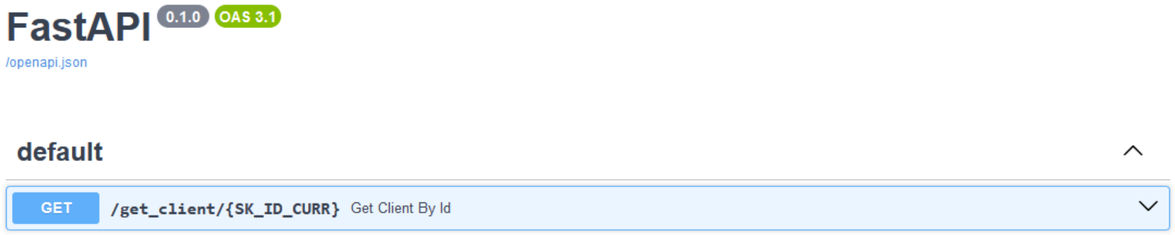
Déploiement via heroku :

- Déploiement automatique déclenchée après chaque publication dans la branche « main » du dépôt GitHub publique

Présentation de l’API

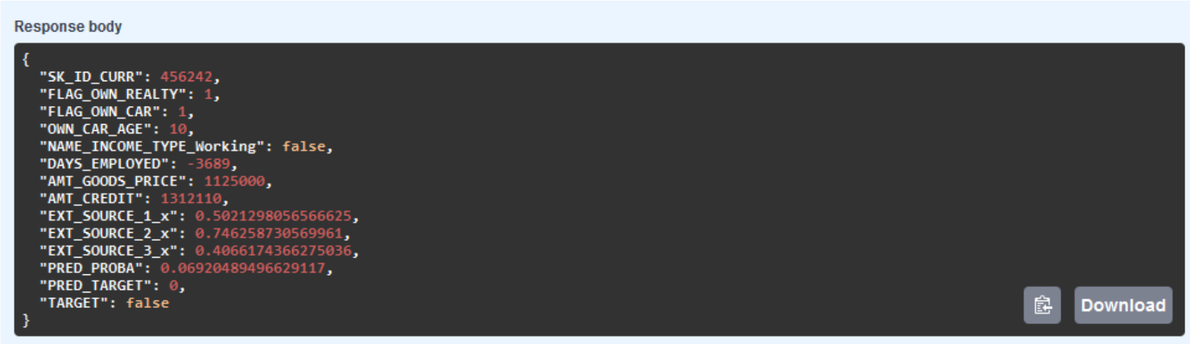
FastAPI :

L'interface de FastAPI est une interface web rapide qui permet de créer facilement des API, et permet une gestion automatique de la validation des données, et une doc interactive (Swagger UI)



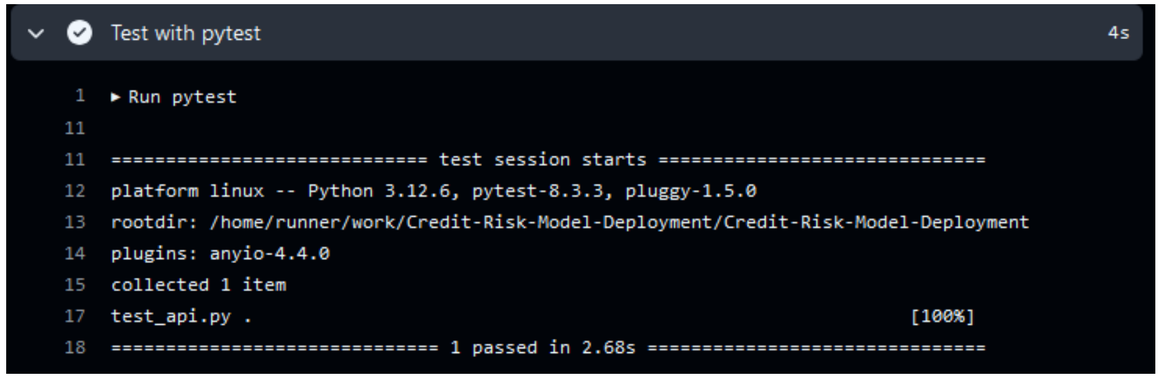
Résultats :

En entrant l’identifiant d’un client, nous obtenons quelques informations sur ce dernier ainsi que les prédictions.



Test de bon fonctionnement de l’API :

- Envoi d’une requête à l’API avec un identifiant mal formé
- Envoi d’une requête à l’API avec un identifiant inconnu
- Envoi d’une requête à l’API avec un identifiant correct



HEROKU

Interface Heroku :

https://projet7-92ff04160aea.herokuapp.com/docs#/

Merci pour votre attention !

Septembre 2024

Lokman AALIOUI