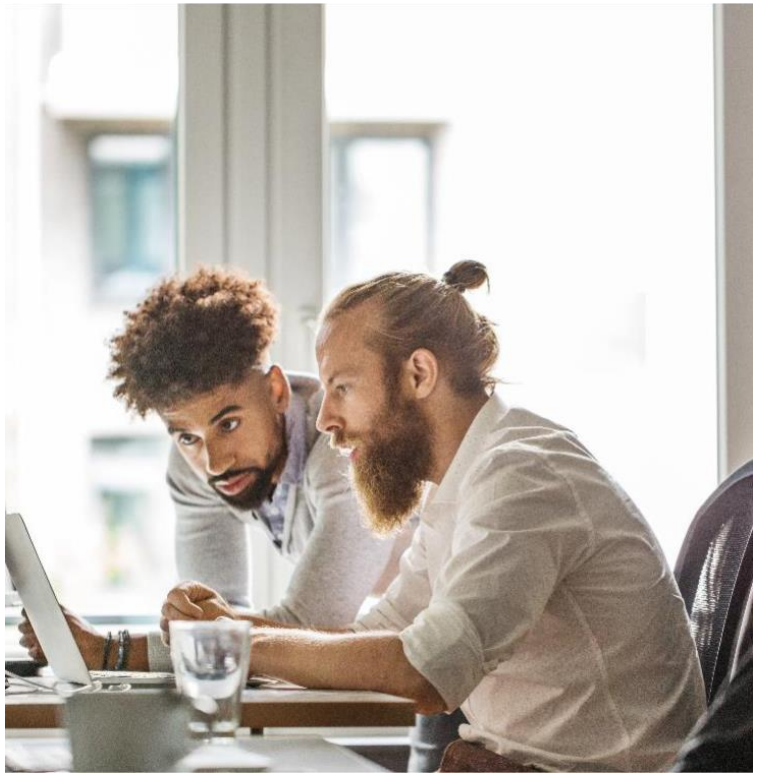
Implémentez un modèle de scoring

Julien NICOLAS

Data scientist / Open Classroom / projet 7

Ordre du jour



1. Contexte / problématique et données
2. Modélisation: démarche, tracking, résultats
3. Analyse du data drift
4. Déploiement
5. Démo: dashboard et API
6. Limites et points d’amélioration
7. Q&A session

Contexte Problématique

Mise en œuvre d'un outil de notation

La société octroie des crédits à la consommation et souhaite développer un algorithme de classification pour évaluer l'octroi (ou non) du crédit en fonction de la probabilité de défaut du client

Demande de transparence d'octroi des crédits

Les clients souhaitent comprendre les raisons d'accord ou de refus du prêt, c'est pourquoi il est demandé de mettre à disposition des chargés de clientèle un tableau de bord interactif leur permettant d'argumenter les décisions.

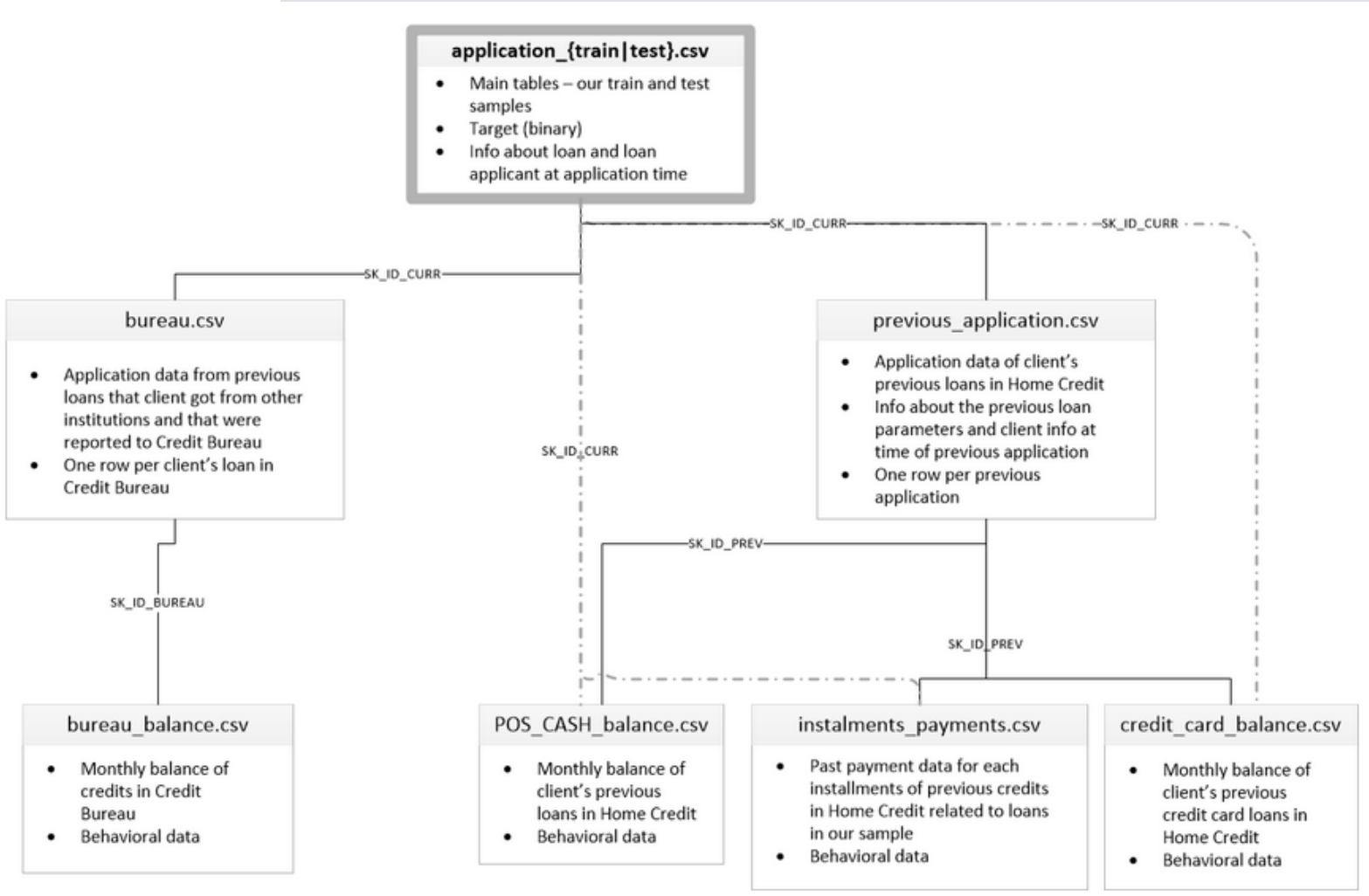
Les données

Home Credit Default Risk

Il s'agit d'un jeu de données très connu sur Kaggle car ayant fait l'objet de nombreuses compétitions en termes de prédiction.

Le diagramme ci-contre présente l'ensemble des données et leur lien grâce à un identifiant unique pour chaque client/prêt (« SK_ID_CURR/PREV »). Source: https://www.kaggle.com/code/willkoehrsen/start-here-a-gentle-introduction

Les clients se composent en 2 datasets: application train et application test. Seul application train sera utilisée pour la modélisation et le dashboard. La partie « test » ne sera utilisée que pour l'analyse du data drift



Modélisation: features engineering

Nettoyage, fusion et création de features additionelles

Le kernel LightGBM with Simple Features a été utilisé comme base pour effecture la plupart des taches de feature engineering.

A travers un code synthétique, il permet d'obtenir un jeu de données relativement propre avec:

- One-hot encoding des colonnes de catégories du dataframe principal (application)
- Pre processing (nettoyage de certaines valeurs aberrantes) et fusion de tous les datasets en un unique dataframe niveau client en supprimant les categorical features des fichiers « satellites »
- Création de nombreux indicateurs pertinents : calculs supplémentaires sur les colonnes existantes : min max, sum, mean… Ajout de pourcentage pour « standardiser » les valeurs importantes (revenu par personne dans la famille, part de jours travaillés depuis la naissance, montant obtenu / montant demandé sur le crédit précédent,..)
- Le résultat est un dataframe de 797 features et 356 254 lignes / clients dont on réduit la mémoire utilisée en convertissant au maximum le type de valeur numérique utilisé (int8 ou float16 si possible), ce qui permet de réduire la mémoire de 65%.

Pour gérer les valeurs NaN ou infinies restantes après cela, un remplacement des valeurs par la médiane de la colonne concernée a été utilisée pour ne pas déformer le résultat et le rendre compatible aux modèles utilisés.

Split des données pour retrouver le périmètre train / test grâce à l'identifiant

Modélisation: démarche

Etapes ayant mené au choix du modèle et de ses paramètres

- Train / test split (taille du jeu de test=25%)
- Sous échantillonnage pour équilibrer le jeu d'entrainement:

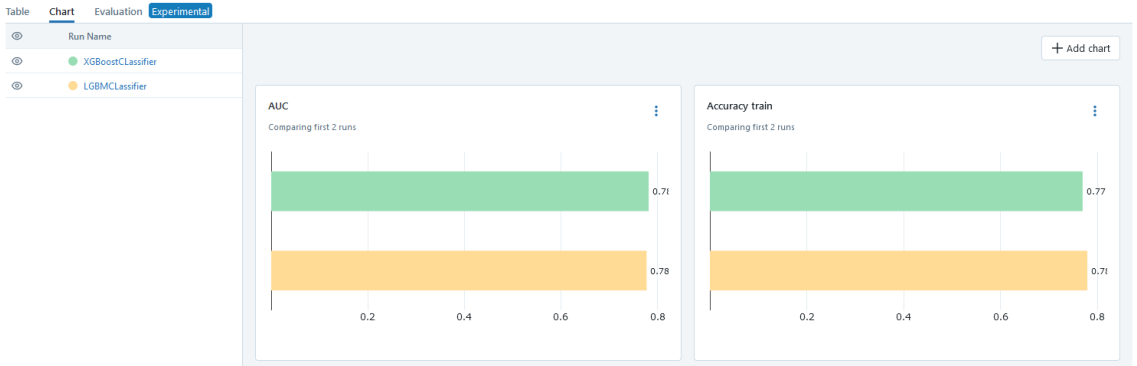
Nombre d'éléments « target »	Jeu d'entrainement original	Jeu après sous-échantillonage
Valeur 0 (crédit accepté)	212 061	18 569
Valeur 1 (crédit refusé)	18 569	18 569

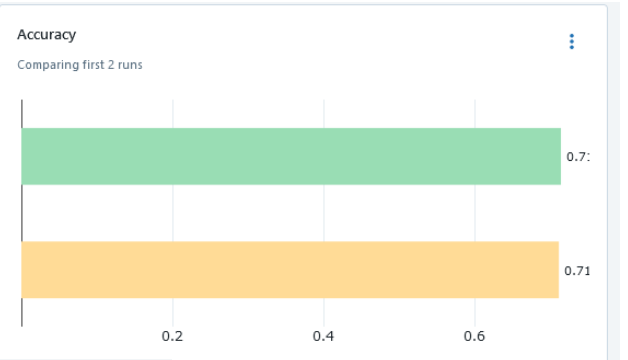
- Définition d'une fonction de coût pour prendre en compte le fait qu'un faux négatif coute 10 fois plus qu'un faux positif :
  - Création d'une matrice de confusion à partir de la valeur réelle et de la valeur prédite
  - Création de la fonction cost = (10 x faux négatif + faux positif) / taille du dataframe
  - Cette fonction est ensuite utilisée avec make scorer pour être minimisé dans la recherche d'hyperparamètres de chaque modèle ainsi que dans une fonction permettant de trouver le seuil optimal de probabilité qui permettra de déterminer le point d'équilibre pour la décision d'octroi de crédit.
- Prédiction des résultats avec un modèle naïf et une régression logistique : donner une première évaluation de notre score, évaluer la pertinence de nos modèles futurs.
- Choix des modèles utilisés: les modèles XG Boost Classifier et LightGBM Classifier sont ceux qui ont obtenu les meilleurs résultats sur ce jeu de données.
- Recherche d'hyperparamètres via GridSearch Cv sur ces modèles en jouant sur la vitessse d'apprentisage (learning rate), le nombre d'arbres à entrainer (n_estimators) , la profondeur de l'arbre de décision (max depth) et la taille du jeu de données utilisé (en nombre d'éléments et de colonnes).
- Choix du modèle et des paramètres retenus selon les métriques établis sur jeu de test et d'entrainement (score custom, AUC, accuracy, …) et le temps nécessaire.

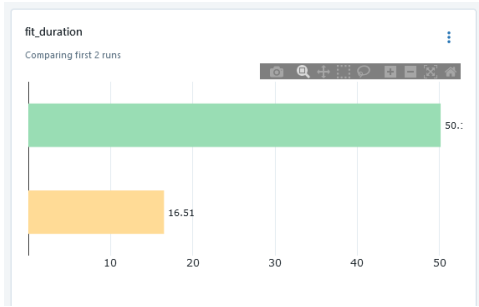
Modélisation: tracking et résultats

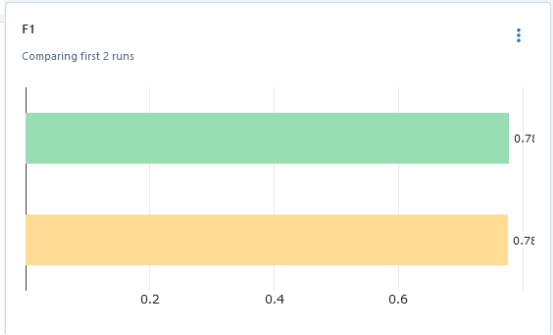
ML Flow

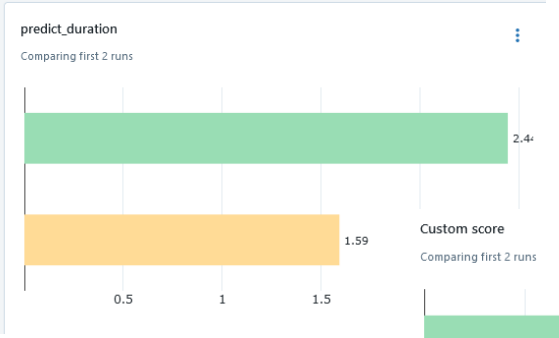
- Après obtention des meilleurs hyperparamètres de chaque modèle (LightGBM et XG Boost), on log les résultats de chaque modèle avec ses hyperparamètres dans ML Flow pour pouvoir faire une comparaison des modèles
- Log des paramètres utilisés et du modèle

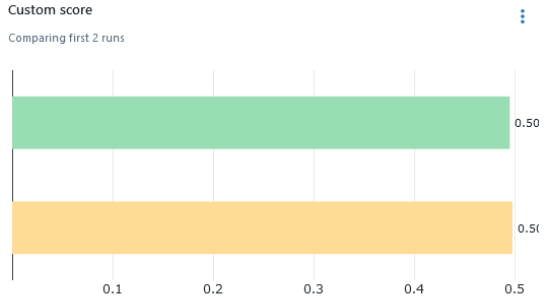










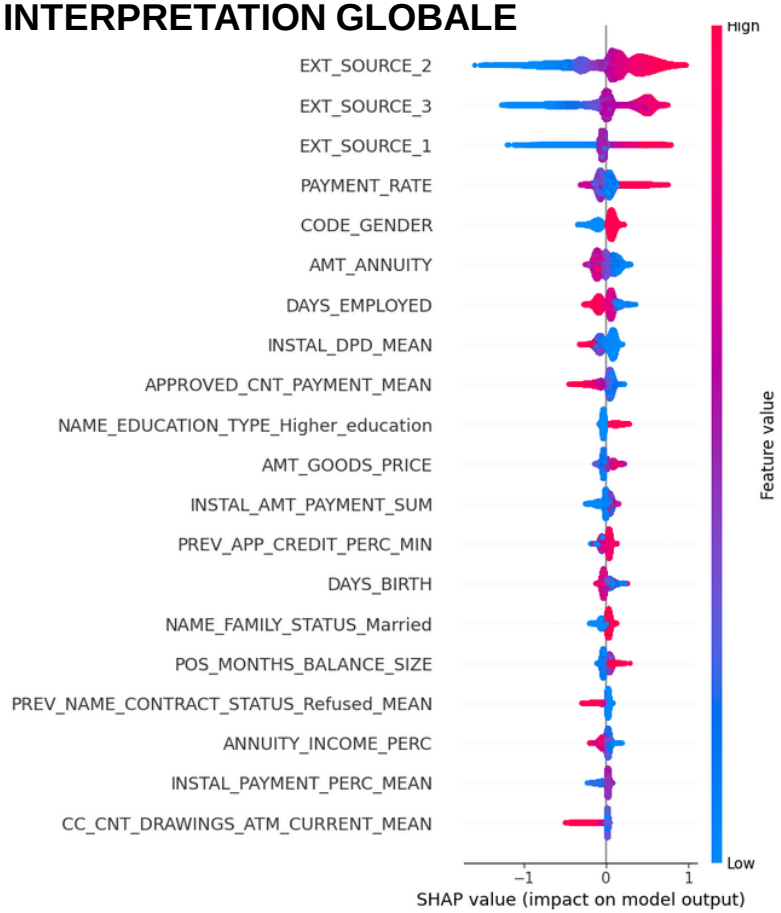


Modélisation: features importance

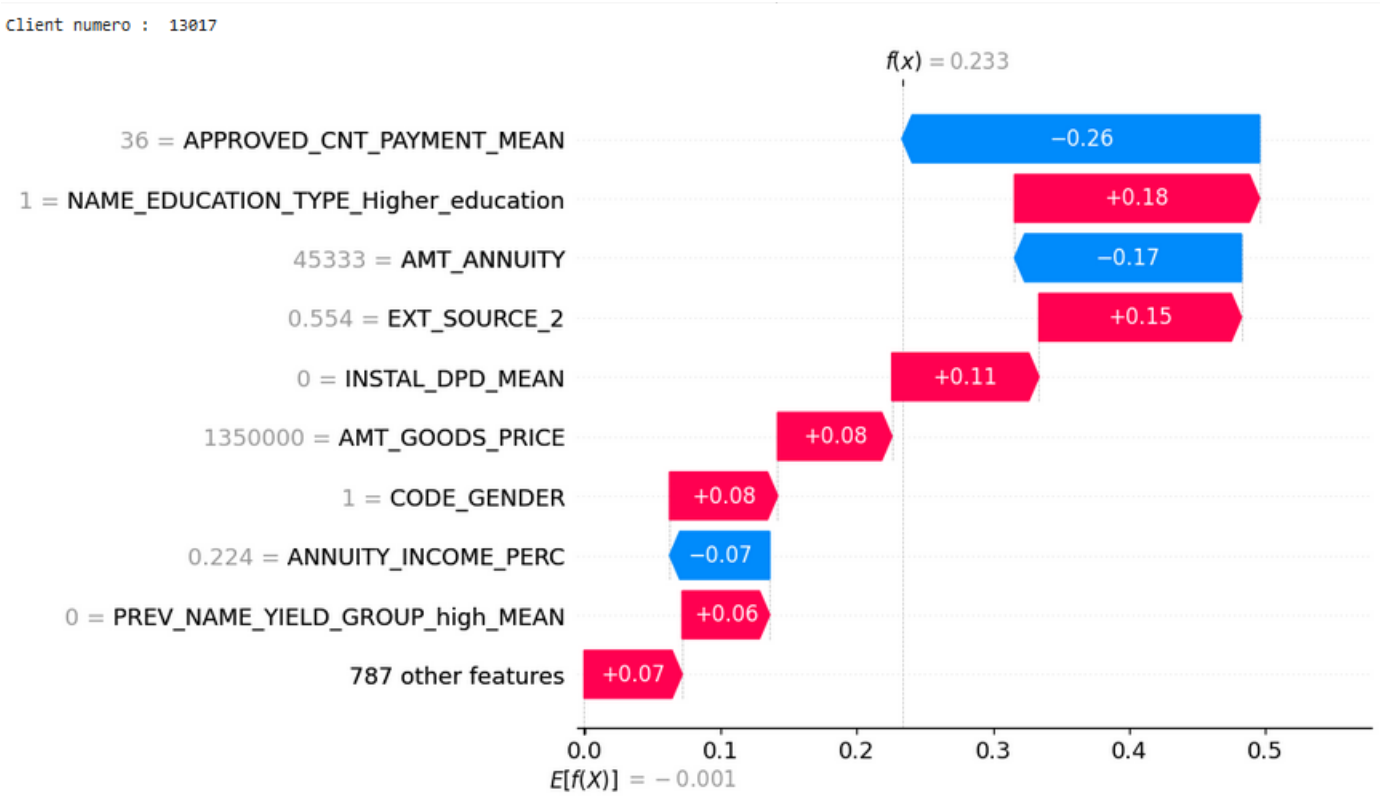
SHAP

- Mesure des features importances globale et locale avec la librairie SHAP qui permet de calculer et mettre en forme les shapley values des features utilisés par notre modèle

INTERPRETATION GLOBALE



INTERPRETATION LOCALE



Analyse du data drift

Data drift avec Evidently

- Métriques et seuil
  - Colonnes numériques: divergence de Kullback-Leibler / seuil=0,1 (défaut)
  - Colonnes catégoriques: indice de stabilité de population (PSI) / seuil=0,2

- Résultats:

798	17	0.0213
Columns	Drifted Columns	Share of Drifted Columns

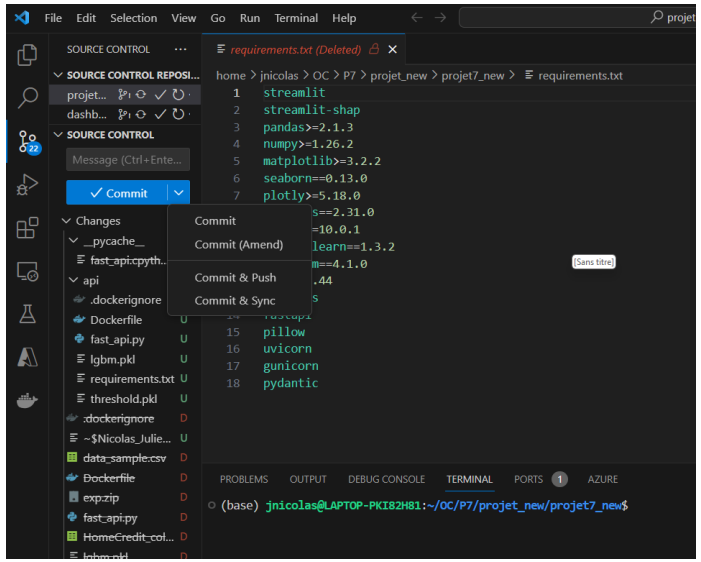
- Interprétation
  - Le data drift est limité entre les 2 jeux de données avec seulement 2,13% des colonnes « drifted »
  - Les colonnes drifted concernent des informations très spécifiques à chaque client sur leurs crédits précédents (min / max / moyenne / somme sur le nombre de mois, les montants des mensualités,…)
  - Colonnes catégoriques ayant le plus de drift: type de prêt (cash / revolving) -> le jeu référence contient des prêts « revolving » alors que le courant n'en contient pas.

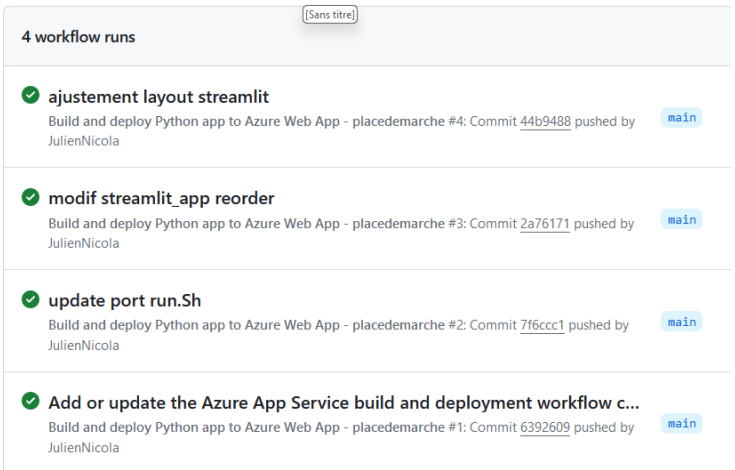
NAME_CONTRACT_TYPE_Cash loans	cat		Not Detected	PSI	0.193682
NAME_CONTRACT_TYPE_Revolving loans	cat		Not Detected	PSI	0.193682

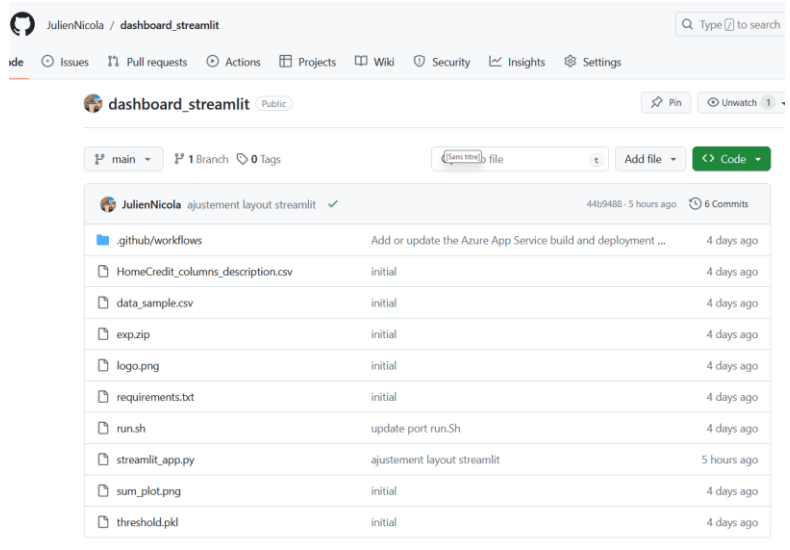
Déploiement de la solution (1/3)

Intégration et déploiement continu avec Github / déploiement de Web APP dans Azure

- Mise en place de 3 repositories Github :
  - Fichiers utilisés pour le développement et le déploiement de l'API : fichiers docker, requirements pour l'installation des librairies, fichiers pickle pour stocker le modèle + paramètres et la valeur du seuil de probabilité et bien sur le fichier de développement de l'API avec Fast API https://github.com/JulienNicola/api
  - Fichiers utilisés pour le développement et le déploiement du dashboard: valeurs explicatives des shapley values locales, échantillon des données utilisées pour la modélisation, requirements pour l'installation des librairies, images des shapley values globales et du logo de la société, commande de démarrage de l'application (run.sh) et worklow github (sous-dossier) https://github.com/JulienNicola/dashboard_streamlit
  - Fichiers jupyter utilisés dans le cadre de la modélisation et de l'analyse du data drift ainsi que cette présentation
- Déploiement continu avec commit /push depuis Visual Studio







Déploiement de la solution (2/3)

Tests unitaires sur le fonctionnement de l'API

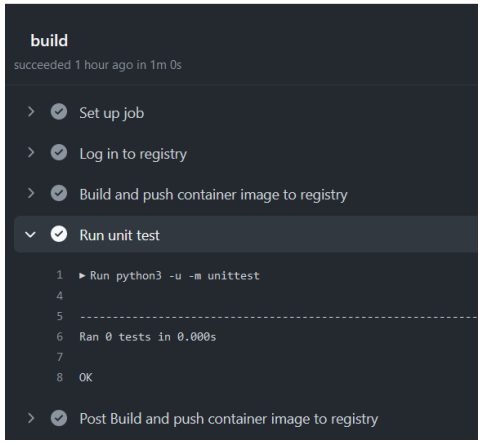
- Tests effectués sur l'API en testant les résultats obtenus pour la prédiction du score client à partir d'échantillons de données sur 2 clients:
  - Le client ID ramené est égal à celui envoyé
  - Le score est compris entre 0 et 1
  - La « décision » n'est pas vide

- En local

```
(base) jnicolas@LAPTOP-PKI82H81:~/OC/P7/projet_new/projet7_new/api/api$ python test_predict.py
/home/jnicolas/.local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle est
imator LabelEncoder from version 1.2.2 when using version 1.3.2. This might lead to breaking code or invalid results. Use
at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/home/jnicolas/.local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle est
imator LabelEncoder from version 1.2.2 when using version 1.3.2. This might lead to breaking code or invalid results. Use
at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
./home/jnicolas/.local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle es
timator LabelEncoder from version 1.2.2 when using version 1.3.2. This might lead to breaking code or invalid results. Use
 at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
.
----------------------------------------------------------------------
Ran 2 tests in 0.047s

OK
```

- Pendant le déploiement sur GitHub Actions

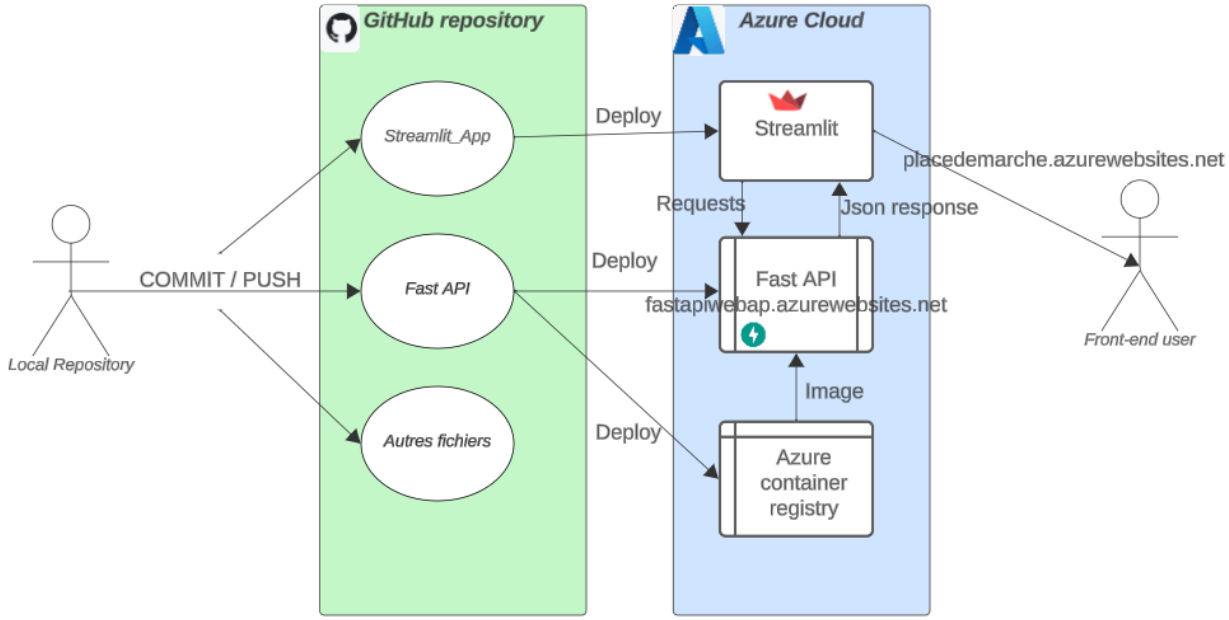


Déploiement de la solution (3/3)

Intégration et déploiement continu avec Github / déploiement de Web APP dans Azure

- Création des applications depuis le portail Azure et Azure CLI:
  - L'API utilise une image docker stockée dans un container hébergé dans Azure (ACR) que j'ai créé
  - Le Dashboard utilise une fonctionnalité simplifiée d'Azure permettant d'automatiser la création de l'image et du conteneur pour n'avoir à gérer que la partie déploiement du workflow (fichier Yaml dans Github actions)

- Schéma synthétique du déploiement



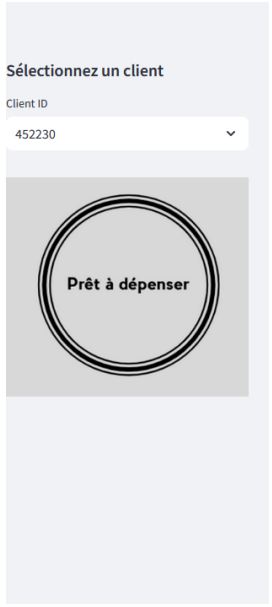
Démo

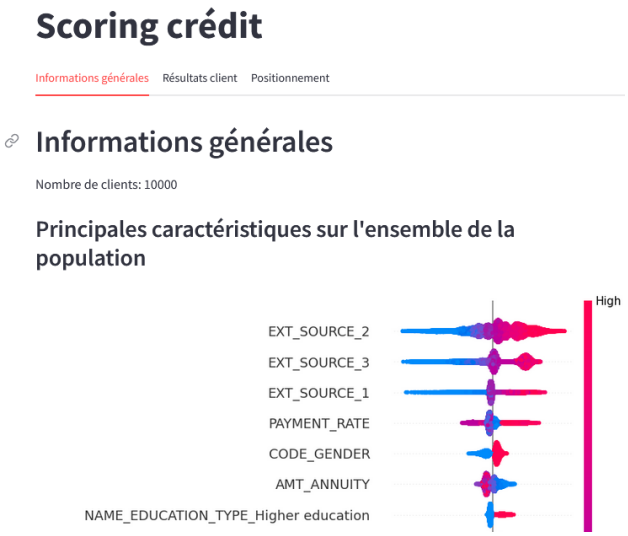
DASHBOARD: STREAMLIT

- Informations générales.
- Résultats clients.
- Positionnement
- https://placedemarche.azurewebsites.net/

FAST API

- Utilisation d'un jeu de données client spécifique
- https://fastapiwebap.azurewebsites.net/docs





Limites et points d'amélioration

Les pistes d'amélioration pour une application totalement opérationelle

- Augmentation des capacités dans le Cloud: stockage de la base de données en dehors de l'application pour pouvoir intégrer l'ensemble des données (test + train) plutôt qu'un échantillon

- Augmentation des capacités en mémoire ou réduction de dimensions du dataset afin de pouvoir optimiser la recherche d'hyperparamètres, tester d'autres modèles et tester le data drift sur le jeu de données complet

- Complément d'informations sur les descriptions des colonnes: être en capacité de fournir des explications précises sur chaque feature utilisée et les intégrer dans le dashboard afin que l'interprétation du résultat soit plus parlante pour le chargé de relations client

- Vérifier les résultats avec les managers de la société : ne met-on pas en risque le développement commercial de la société avec cette fonction « cout » très conservatrice: environ 40% des crédits non accordés…

Merci pour votre attention