Implémentez un modèle de scoring

David Marchand – Apprenant OpenClassrooms

Rappel de la problématique

- Mise en œuvre d'un outil de « scoring crédit »
- Développement d'un algorithme de classification sur le défaut / réussite de remboursement de crédit
    - Traitement dans le cadre d'un dataset déséquilibré
- Développement d'un dashboard interactif s'appuyant sur de la transparence (algorithme, utilisation et interprétabilité des données)
- Mise en production de la solution développée

Présentation du jeu de données

- Jeu de données proposé à l'occasion de la compétition Kaggle « Home Credit Default Risk »

- Séparation en multiples .csv :
  - application_{train|test}.csv
  - bureau.csv
  - bureau_balance.csv
  - POS_CASH_balance.csv
  - credit_card_balance.csv
  - previous_applications.csv
  - installments_payments.csv

Présentation de la modélisation

Vue générale

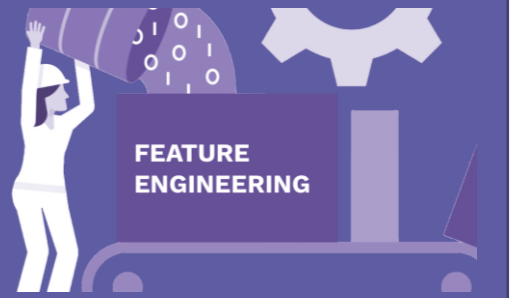
Pré-traitement (fusion .csv)

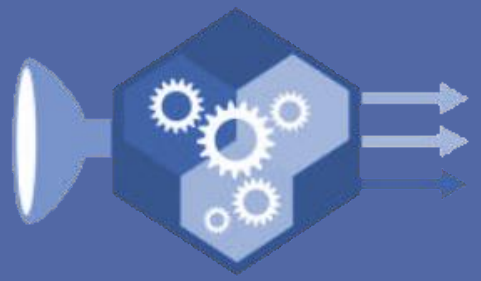
Séparation initiale selon la présence de la cible

Feature Engineering

Imputation valeurs manquantes

Séparation finale jeux de données





Définition des espaces de recherche par modèle

RandomSearchCV (pipeline imblearn)

Création de statistiques (AUC, Fbeta Score, …)

Sélection du meilleur modèle

Fine Tuning

Interprétabilité du modèle

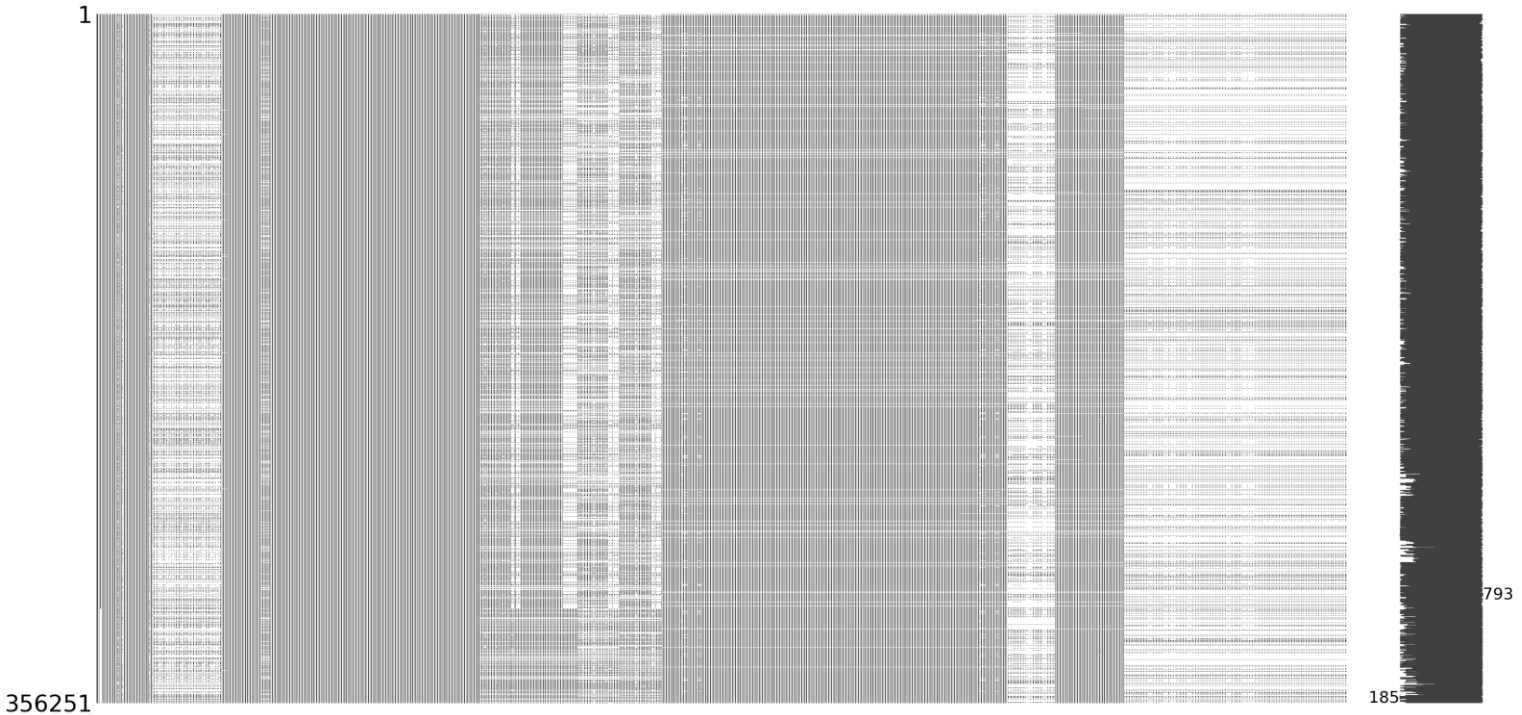
Sérialisation

Analyse Data Drift

Présentation de la modélisation

Pré-traitement

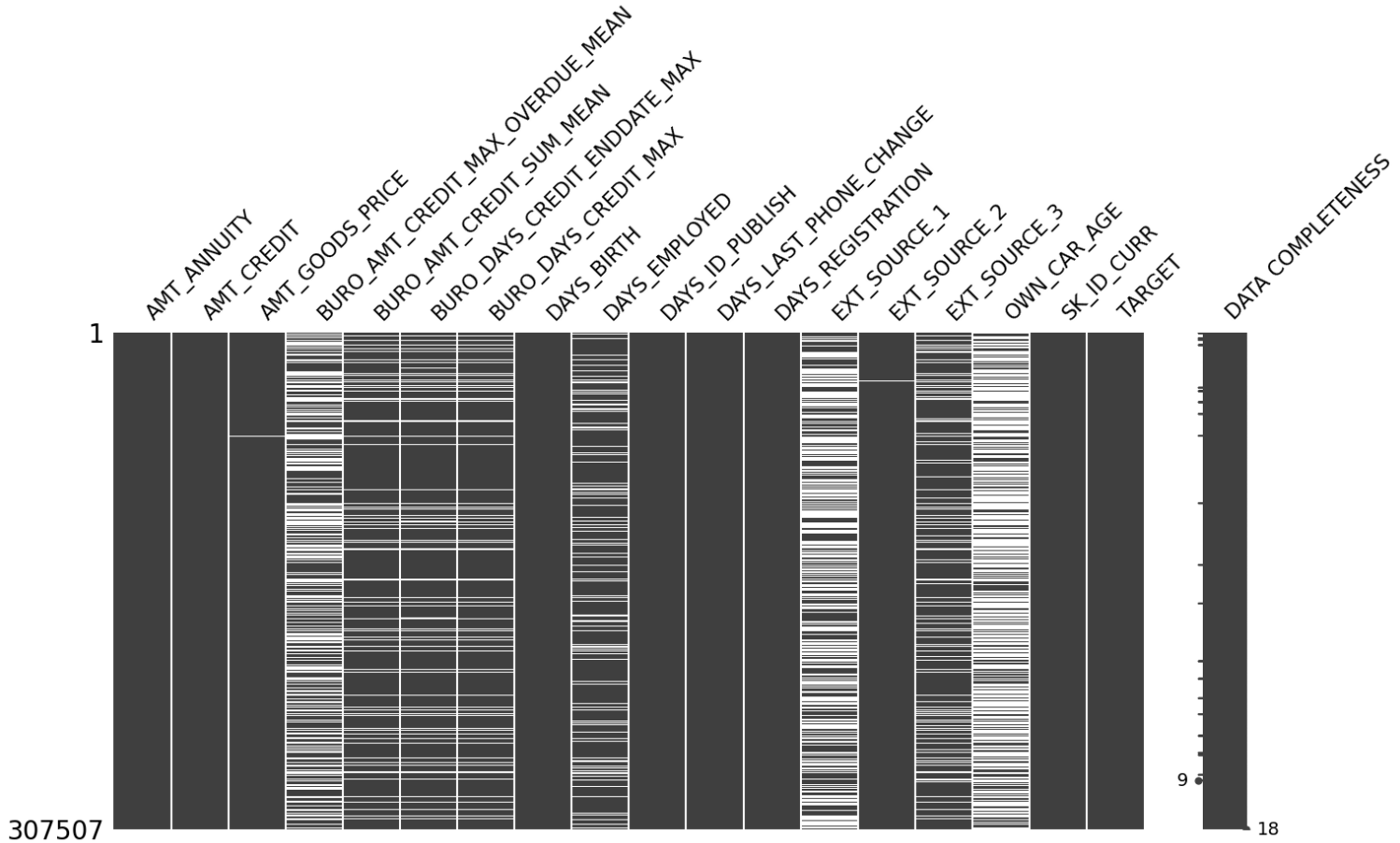
- Fusion de la majorité des .csv sur la base de liens entre fichiers
- Obtention d'un dataframe de 356251 lignes x 793 colonnes
- Taux variable de valeurs manquantes (25 % en moyenne)
- Séparation en deux dataframes selon la présence ou non d'une valeur sur la colonne cible



Présentation de la modélisation

Feature Engineering

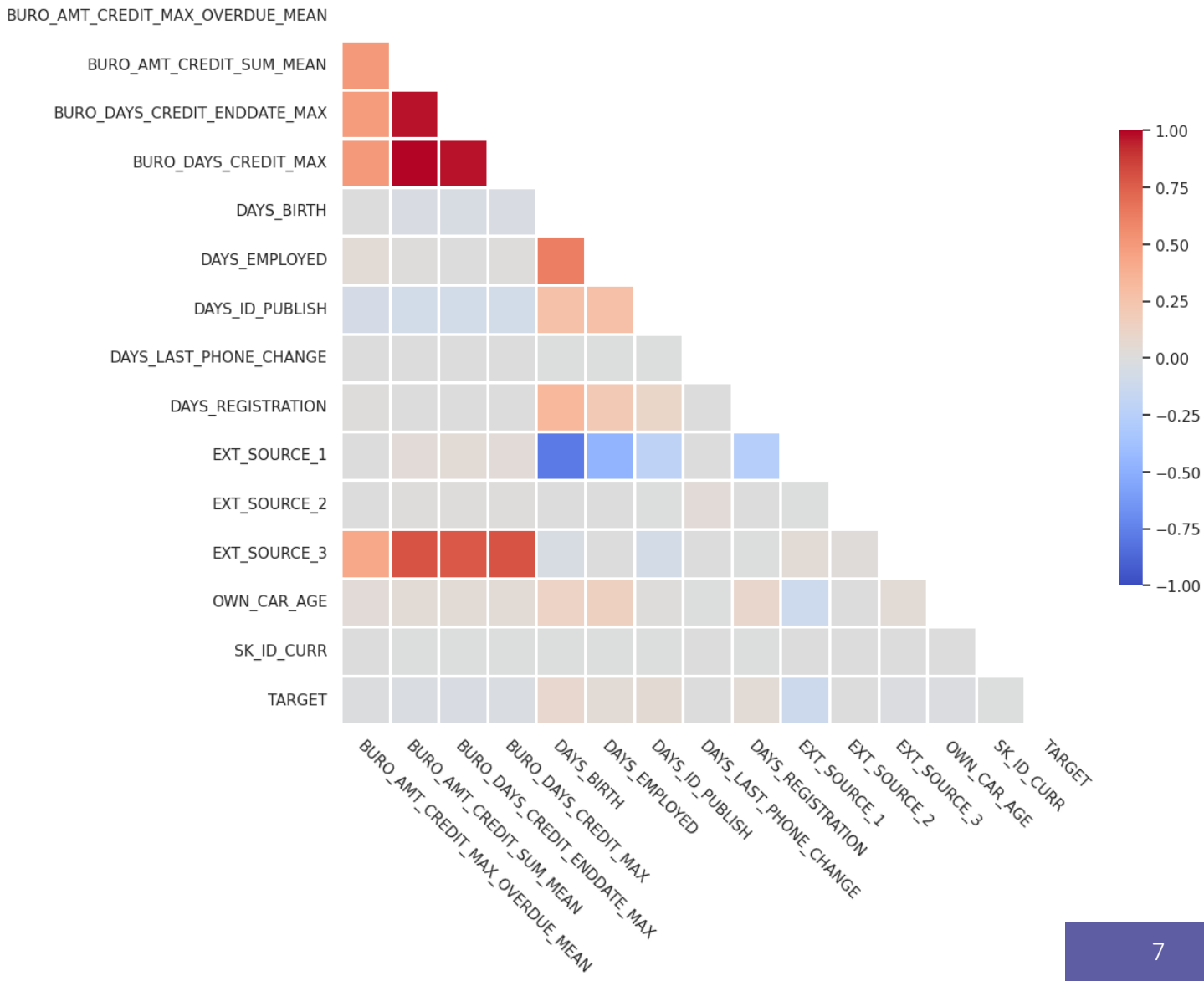
- Raffinement du dataframe via Feature Engineering
- Passage à un dataframe de 307507 lignes x 20 colonnes via Feature Importance
- 18 variables initiales retenues
- Création de nouvelles variables (3 variables initiales exploitées) :
  - AMT_CREDIT_TO_ANNUITY_RATIO
  - AMT_GOODS_PRICE_TO_ANNUITY_RATIO
  - AMT_CREDIT_GOODS_PRICE_DIFF
  - GOODS_PRICE_FAIL_RATIO
  - CREDIT_FAIL_RATIO



Présentation de la modélisation

Data Leakage - Analyse de corrélation entre variables

- Faibles corrélation avec la variable cible (TARGET)
- Prévention du Data Leakage
- Plusieurs variables possédant de multiples corrélations :
  - EXT_SOURCE_1
  - EXT_SOURCE_3
  - DAYS_REGISTRATION
  - DAYS_ID_PUBLISH



Présentation de la modélisation

Recherche des meilleurs hyperparamètres

- Création d'un pipeline gérant le déséquilibre de classes :
- Entraînement du modèle sur le jeu d'entraînement (déséquilibré)
- Validation croisée sur un jeu de validation équilibré (oversampling via la méthode SMOTE)
- Scoring sur le « score métier » et le Fbeta-score
- Score métier :
  - Détection des faux positifs / faux négatifs
  - Pondération sur la valeur associée à l'état faux positif / faux négatif

Présentation de la modélisation

Fine Tuning du meilleur modèle

- Utilisation d’un pipeline similaire à la recherche des meilleurs hyperparamètres
- Approximation bayesienne du résultat obtenu pour un ensemble donné d’hyperparamètres
- Possibilité de cribler un plus large spectre de valeurs pour un coût computationnel moindre
- Idéal dans une approche de Fine Tuning

Résultats

Tableau récapitulatif

	ROC AUC	Fbeta Score	Score Métier	Accuracy	Temps d'entraînement
Dummy (Baseline)	0,5	0,31	508840	0,08	2 s (CPU)
Logistic Regression	0,52	0,31	457881	0,17	6,3 s (CPU)
XGBoost	0,65	0,38	183409	0,64	1,4 s (GPU)
LightGBM	0,68	0,42	149111	0,71	6,2 s (CPU)
Fine-Tuned XGBoost	0,68	0,42	150017	0,7	1,5 s (GPU)

- LightGBM meilleur modèle global, mais empiriquement retrouvé « instable »
- Fine-Tuned XGBoost modèle final retenu (quasi aussi performant que LightGBM mais empiriquement bien plus « stable »)

Tracking des expérimentations

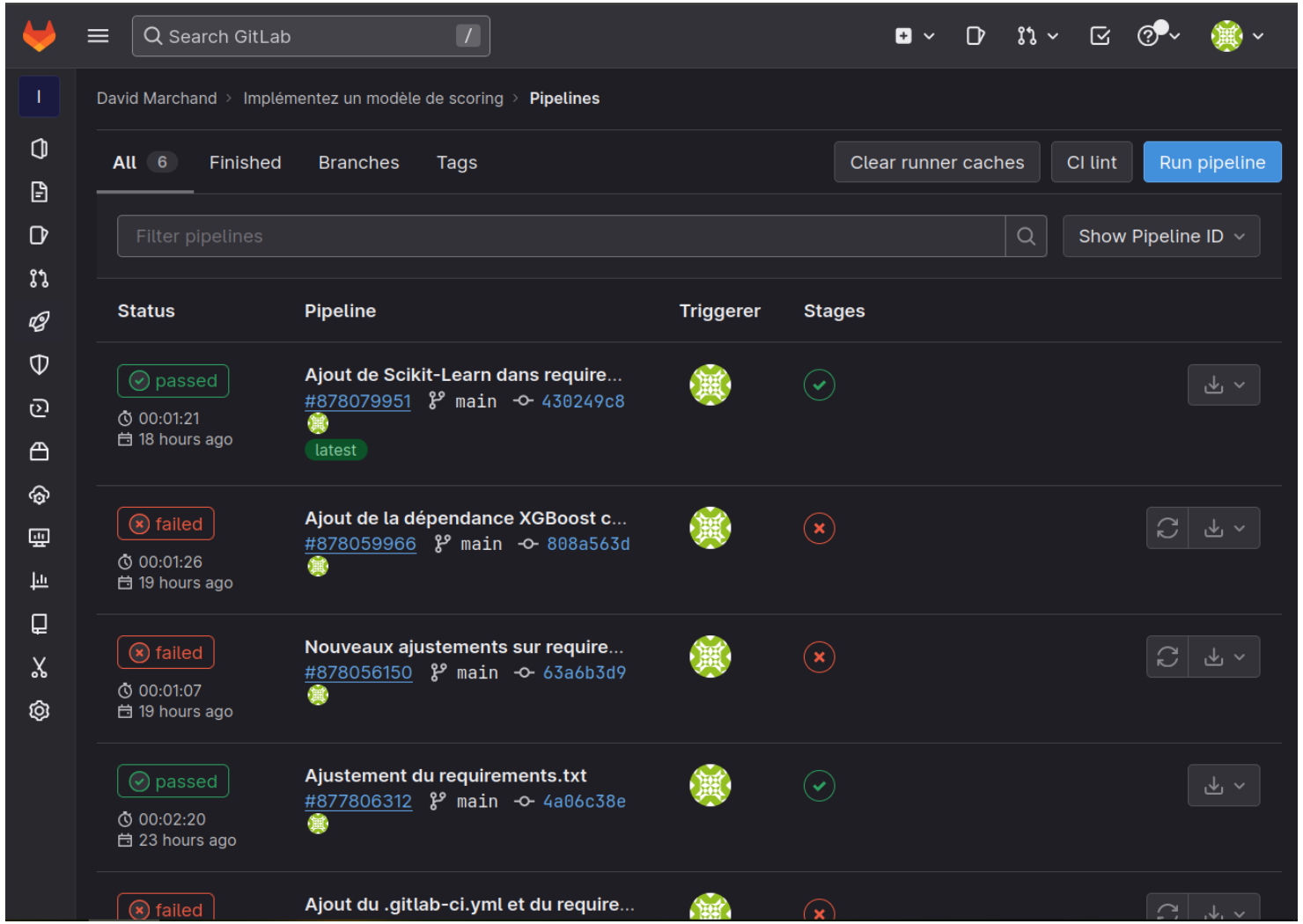
MLFlow

Courte démonstration

Pipeline de déploiement

Git, Github / Gitlab, tests unitaires

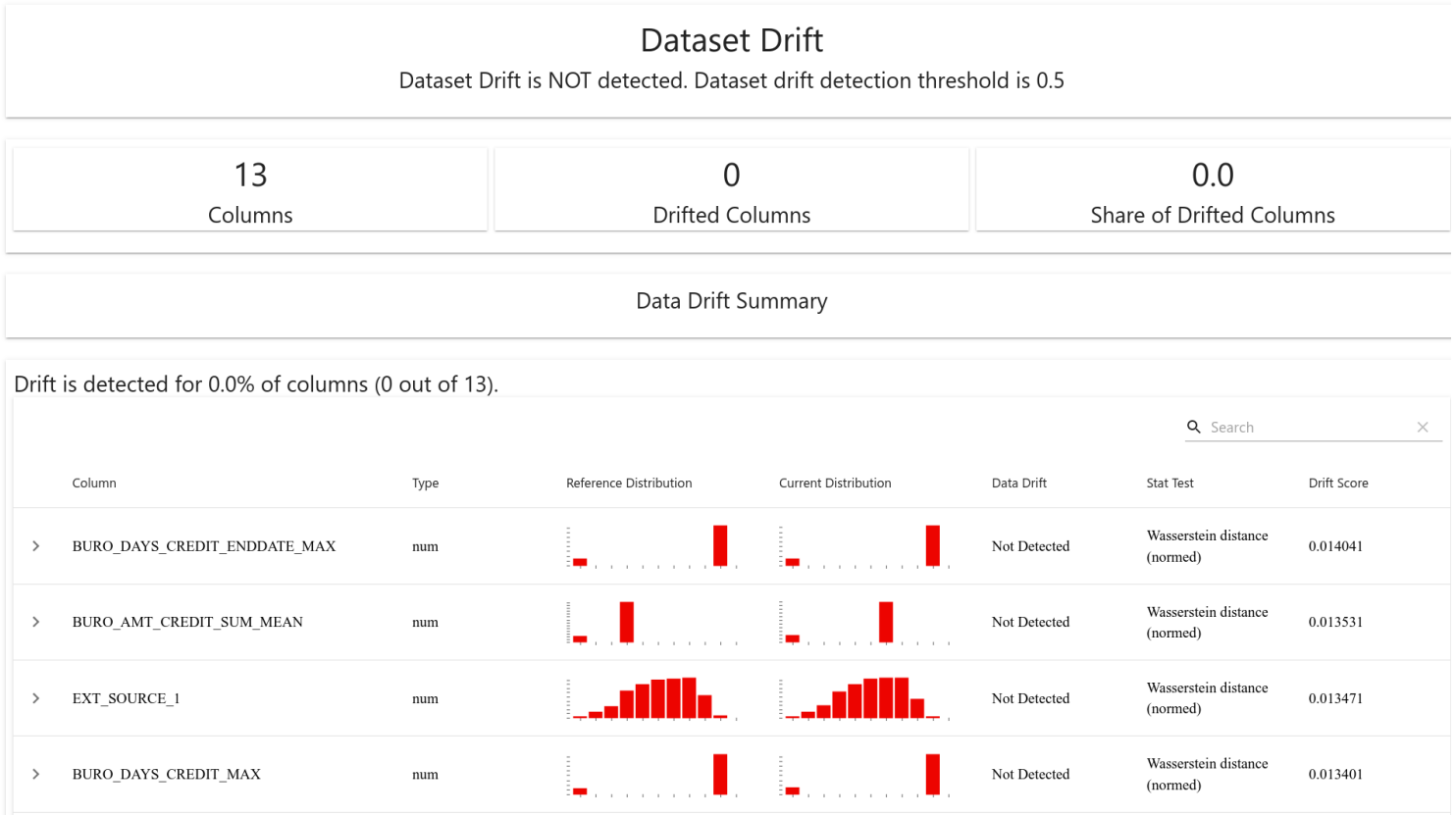
- Utilisation de PyTest pour réalisation de tests unitaires
- Gestion du projet via la plate-forme Github / Gitlab :
  - Basés sur Git
  - Complets (fonctionnalités de wiki, système de suivi de bugs, CI/CD, tracking de vulnérabilités, etc.)
- Exploitation de PyTest pour la réalisation de tests CI/CD sur Github / Gitlab



Data Drift

Analyse via Evidently AI

- Analyse du Data Drift sur les datasets :
  - Train_df vs test_df
  - Train_df vs score_df
  - Train_df vs newclients_df
- Résultats positifs pour test_df et score_df
- Résultat négatif pour newclients_df
- newclients_df issu du fichier .csv « test » du dataset initial



Dataset Drift

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

13	0	0.0
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary

Drift is detected for 0.0% of columns (0 out of 13).

Search

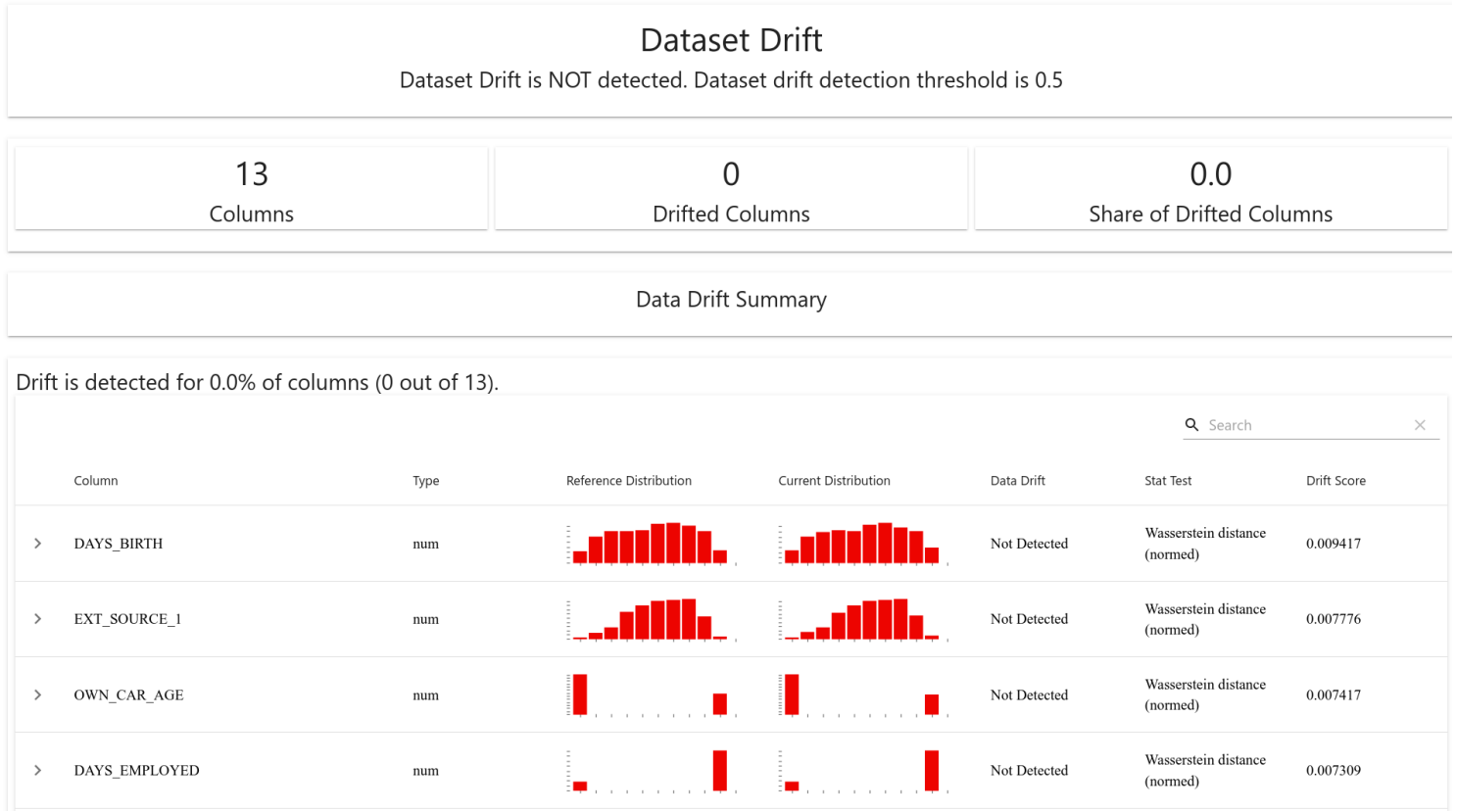
Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
BURO_DAYS_CREDIT_ENDDATE_MAX	num			Not Detected	Wasserstein distance (normed)	0.014041
BURO_AMT_CREDIT_SUM_MEAN	num			Not Detected	Wasserstein distance (normed)	0.013531
EXT_SOURCE_1	num			Not Detected	Wasserstein distance (normed)	0.013471
BURO_DAYS_CREDIT_MAX	num			Not Detected	Wasserstein distance (normed)	0.013401

train_df vs test_df

Data Drift

Analyse via Evidently AI

- Analyse du Data Drift sur les datasets :
  - Train_df vs test_df
  - Train_df vs score_df
  - Train_df vs newclients_df
- Résultats positifs pour test_df et score_df
- Résultat négatif pour newclients_df
- newclients_df issu du fichier .csv « test » du dataset initial

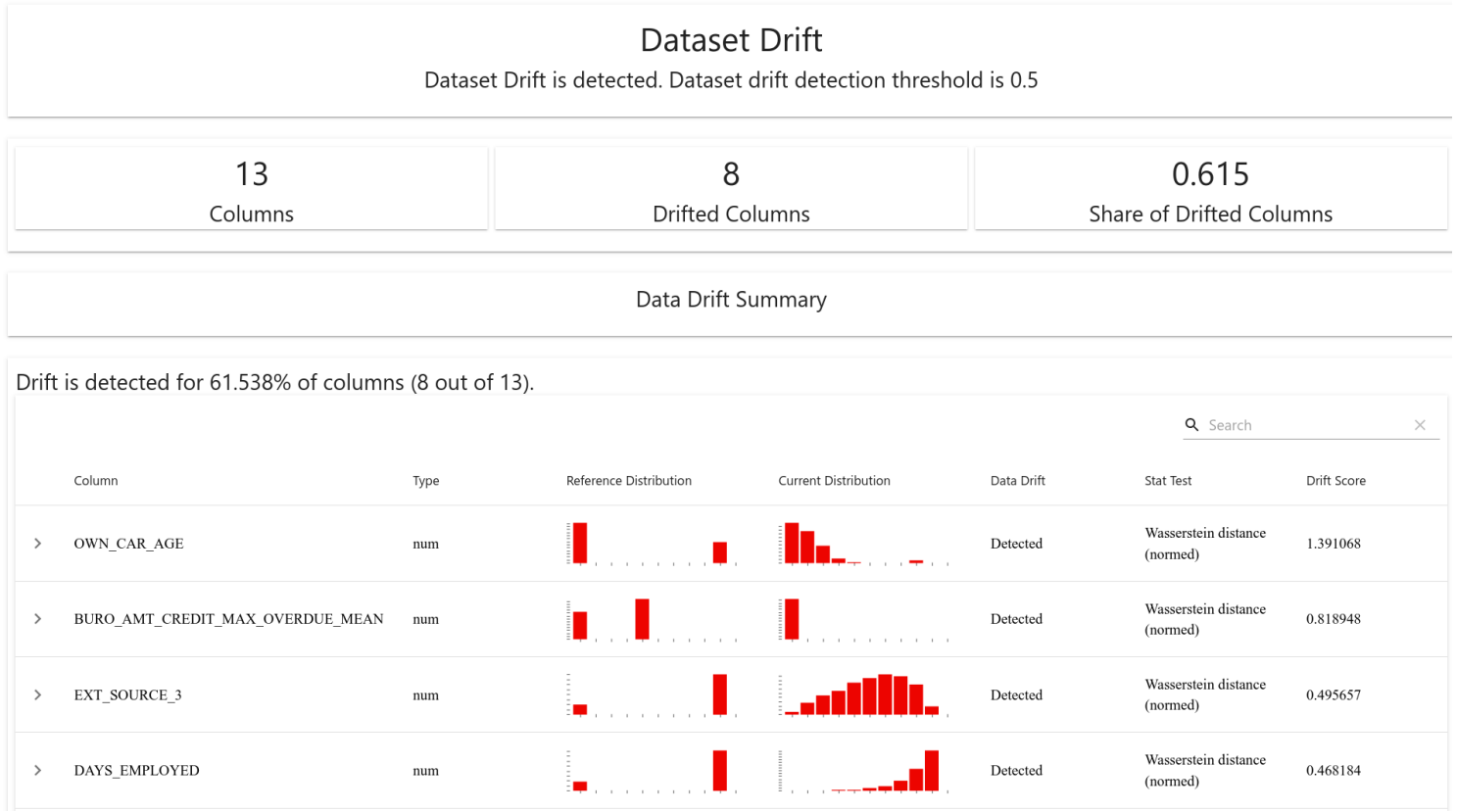


train_df vs score_df

Data Drift

Analyse via Evidently AI

- Analyse du Data Drift sur les datasets :
  - Train_df vs test_df
  - Train_df vs score_df
  - Train_df vs newclients_df
- Résultats positifs pour test_df et score_df
- Résultat négatif pour newclients_df
- newclients_df issu du fichier .csv « test » du dataset initial



Dataset Drift

Dataset Drift is detected. Dataset drift detection threshold is 0.5

13	8	0.615
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary

Drift is detected for 61.538% of columns (8 out of 13).

Search

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
OWN_CAR_AGE	num			Detected	Wasserstein distance (normed)	1.391068
BURO_AMT_CREDIT_MAX_OVERDUE_MEAN	num			Detected	Wasserstein distance (normed)	0.818948
EXT_SOURCE_3	num			Detected	Wasserstein distance (normed)	0.495657
DAYS_EMPLOYED	num			Detected	Wasserstein distance (normed)	0.468184

train_df vs newclients_df

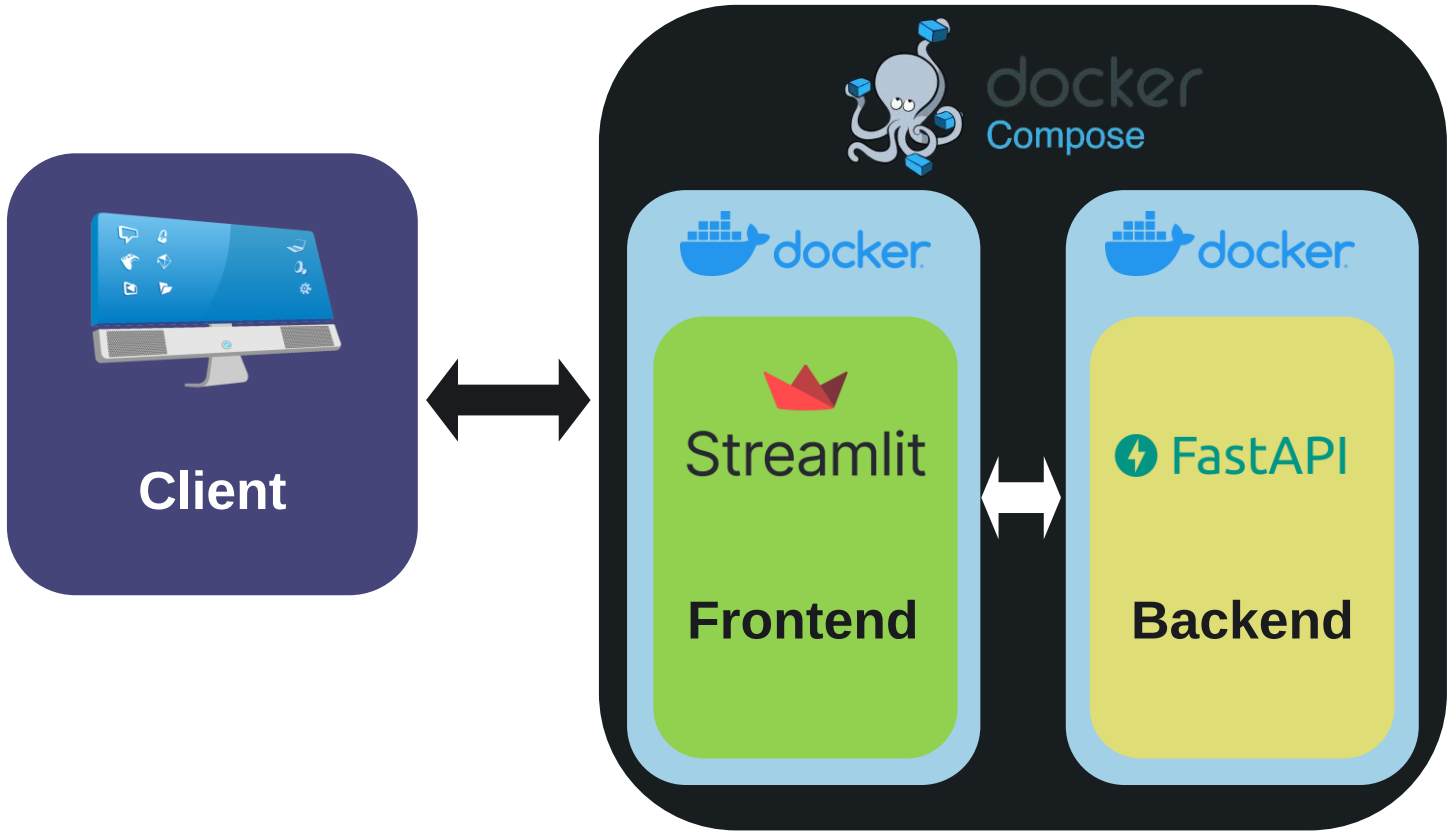
Dashboard

Présentation de l'architecture générale

- Séparation de l'architecture du dashboard en Frontend (partie visuelle) et Backend (partie traitement)
- Frontend développé autour du package Streamlit
- Envoi des requêtes faites sur le Dashboard vers le Backend via une API développée via le package FastAPI
- Utilisation de Docker Compose pour empaquetage de l'application
- Envoi temporaire sur Hostinger (VPS)
- Projection pour envoi sur Azure Web Apps

Azure Web Apps

HOSTINGER



Dashboard

Démonstration

Un peu plus longue que la précédente !

Conclusion

Limites et améliorations

Limites

- Quantité encore importante de faux négatifs
- Choix du modèle (« instabilité » de LightGBM)
- Scalabilité actuelle de la plate-forme d'hébérgement (contournement possible via Azure Web Apps, Amazon EC2 App Runner ou Heroku)

Améliorations

- Echantillonnage (passage à de l'oversampling + undersampling ?)
- Raffinement du modèle (Feature Engineering + augmentation du nombre d'hyperparamètres)
- Augmentation de l'interactivité du dashboard (transition vers Plotly)
- Amélioration globale du code
- Stabilisation de l'API

Merci de votre attention !