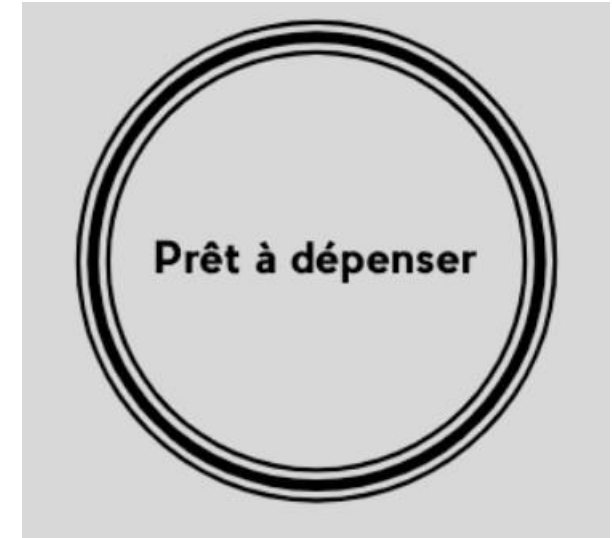


#7 Implémentez un modèle de scoring



Soutenance Emilie Groschêne le 12/06/2023
Evalueur: Alexandre Landi
Mentor: Léa Naccache

Sommaire

I

Problématique et jeu de données

II

Modélisation

III

Pipeline de déploiement

IV

Analyse de data drift

V

Dashboard

VI

Conclusions

I. PROBLEMATIQUE ET JEU DE DONNEES

I. Présentation de la problématique



La société "Prêt à dépenser" propose des **crédits à la consommation** pour des personnes ayant **peu ou pas du tout d'historique de prêt**.

❑ Mission:

Prêt à dépenser souhaite mettre en œuvre un outil de **scoring crédit** pour:

- Calculer la **probabilité de faillite** d'un client de façon automatique
- Classifier la demande en **crédit accordé** ou **refusé**
- Développer un **algorithme de classification** en s'appuyant sur des sources de données variées

❑ Objectifs :

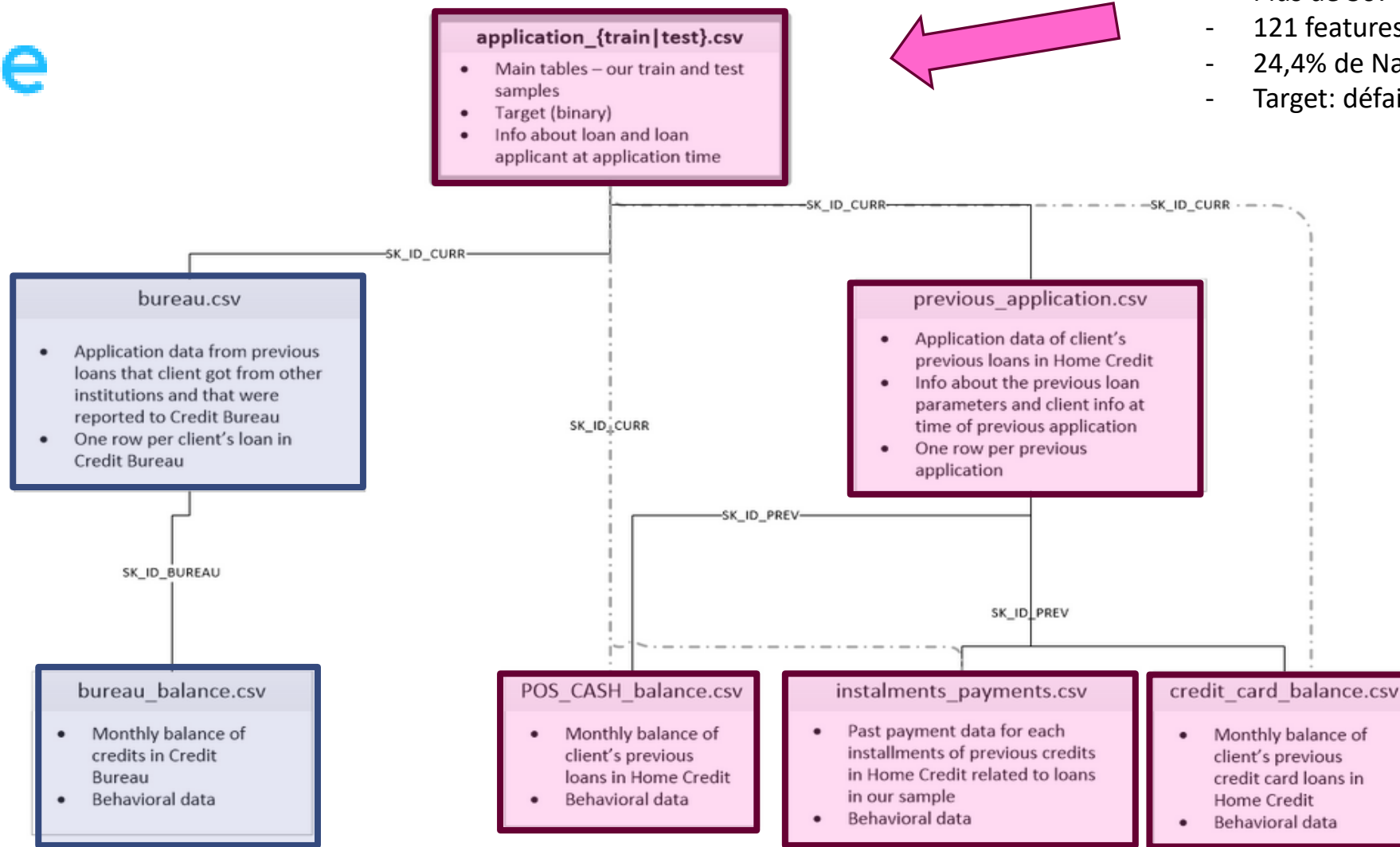
- Pouvoir expliquer de façon la plus **transparente** possible les décisions d'octroi de crédit
- Permettre aux clients de disposer de leurs **informations personnelles** et de les explorer facilement

I. Présentation du jeu de données

kaggle

Dataset principal:

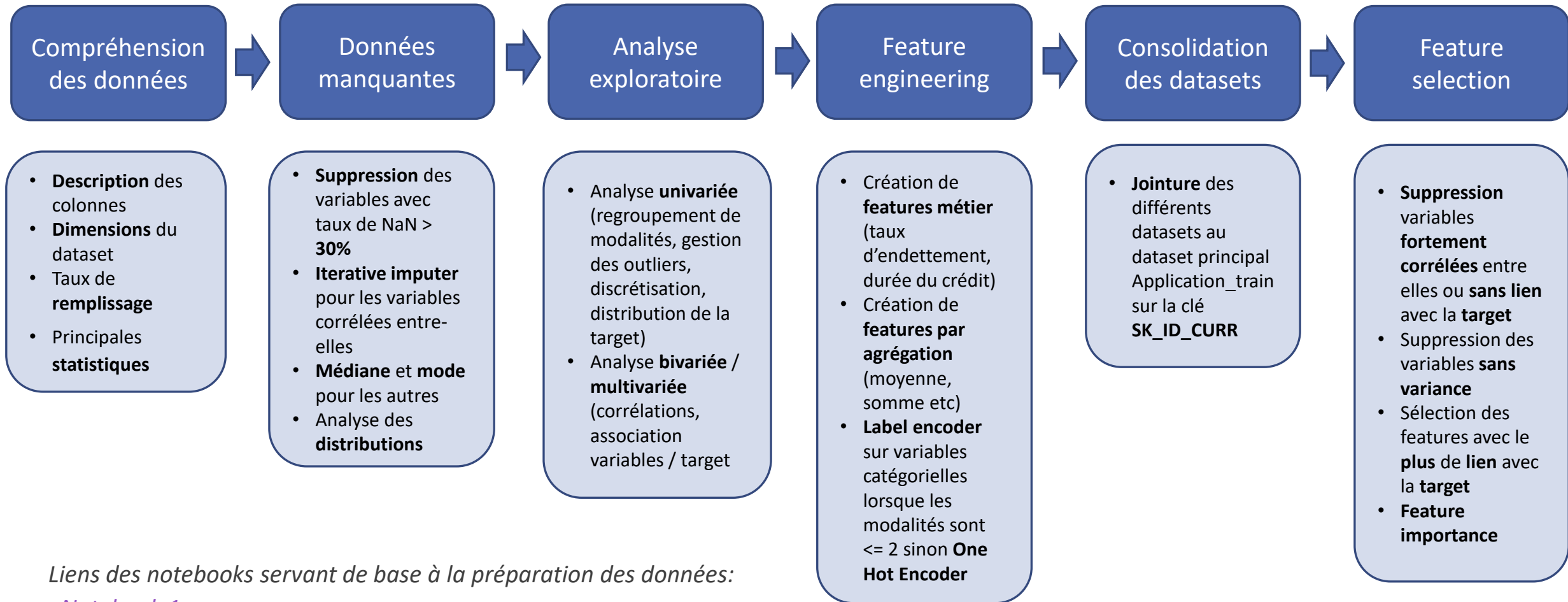
- Plus de 307 000 clients
- 121 features
- 24,4% de NaN
- Target: défaillant / non défaillant



Prêt à dépenser

Autres institutions financières

I. Preprocessing des données



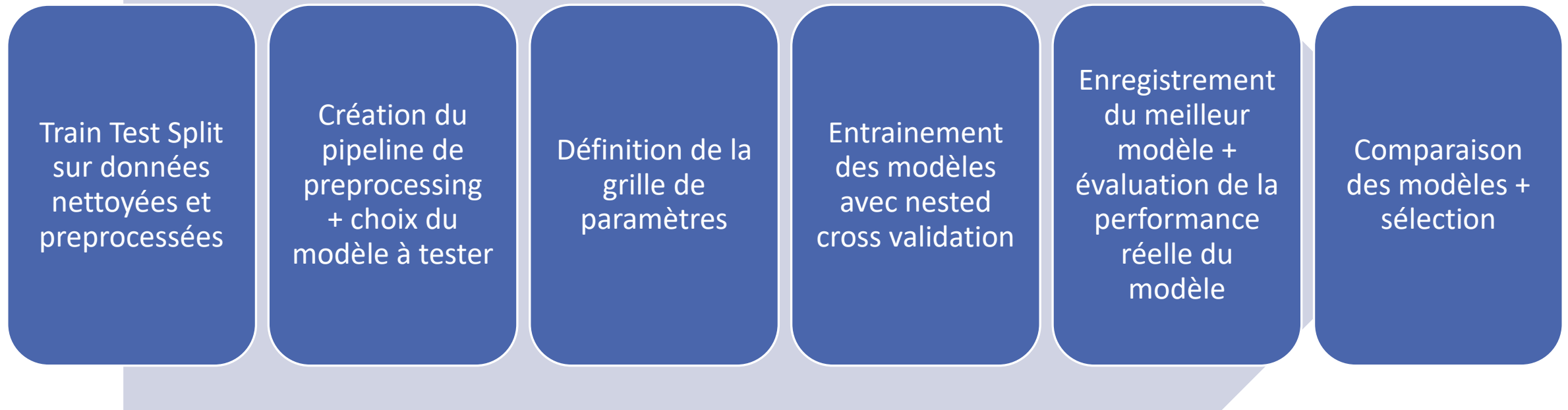
Liens des notebooks servant de base à la préparation des données:

- [Notebook 1](#)

- [Notebook 2](#)

II. MODELLISATION

II. Modélisation: démarche



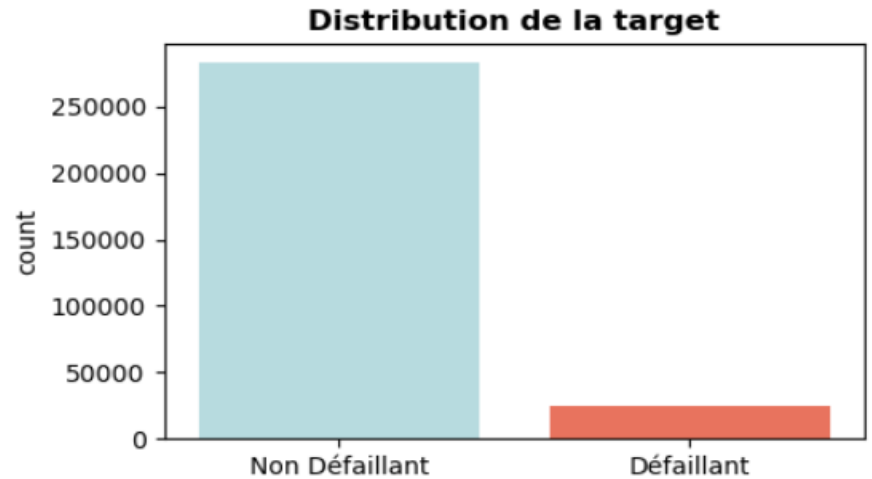
II. Modélisation: traitement des données déséquilibrées

Non défaillants surreprésentés (>91%).

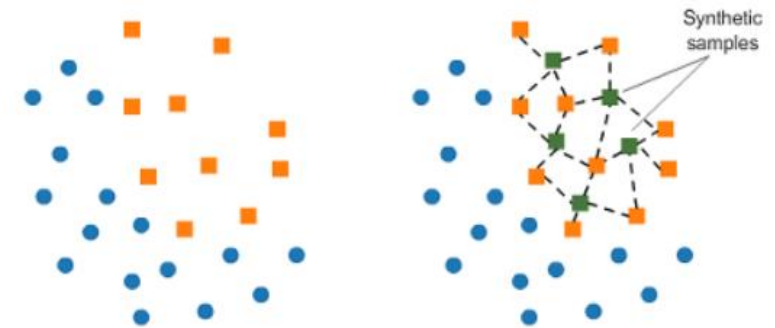
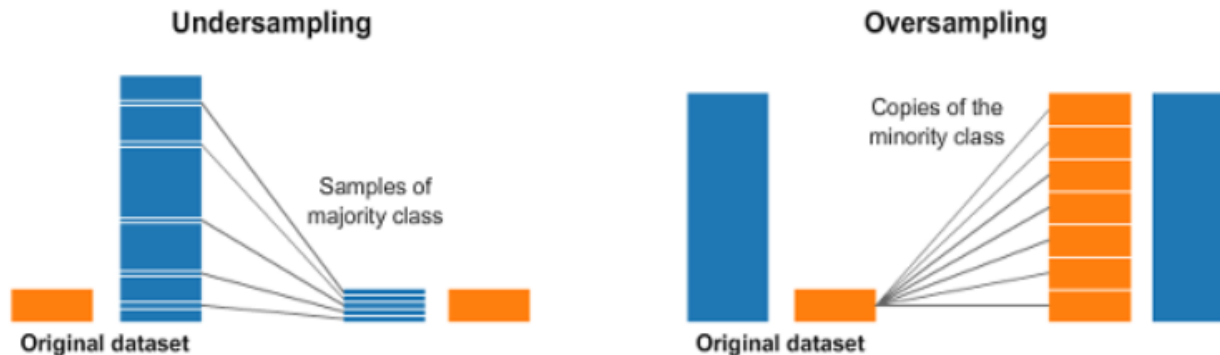
Si le déséquilibre des classes n'est pas traité, le modèle va ignorer la classe minoritaire et avoir des **performances médiocres**.

2 méthodes testées:

- pondération de classes via le paramètre **class_weight** des algorithmes de classification sklearn
- **sous-échantillonnage aléatoire + SMOTE**



Avec SMOTE, de **nouveaux exemples** sont synthétisés à partir des exemples existants:



II. Modélisation: choix des mesures

❑ Problématique:

- **Coût important** pour la Banque **d'accorder un crédit** à un **client défaillant prédit non défaillant** (risque de ne pas rembourser son crédit ou en partie) => faux négatif
- **Manque à gagner** si un **crédit** n'est **pas accordé** à un client qui l'aurait remboursé (risque de perte de clients, de manque à gagner) => faux positif

❑ Compromis à trouver entre:

- Minimiser les faux négatifs => maximiser les **recall**
- Minimiser les faux positifs => maximiser la **précision**

❑ En prenant en compte que:


- Le **coût** des **faux négatifs** **supérieur** à celui des **faux positifs**

		Prédiction	
		Y=1 (Défaillant)	Y=0 (Non défaillant)
Réalité	Y=1 (Défaillant)	Vrais positifs => à maximiser	Faux négatifs => à minimiser
	Y=0 (Non défaillant)	Faux positifs => à minimiser	Vrais négatifs => à maximiser

Rappel: $\frac{VP}{VP + FN}$

Précision: $\frac{VP}{VP + FP}$

} **A maximiser**



- **F-beta score** (moyenne harmonique pondérée de la précision et du rappel avec bêta = poids du rappel dans le score) à maximiser avec beta = 9
- **Score métier** à minimiser = $10 \cdot FN + 1 \cdot FP$

II. Modélisation: visualisation du tracking

« MLflow est une plateforme open-source qui permet de gérer le cycle de vie des modèles de Machine Learning, y compris l'expérimentation, la reproductibilité, le déploiement et un registre central de modèles. » - mlflow.org

MLflow Tracking est une API et une interface utilisateur pour **l'enregistrement des paramètres**, des **versions de code**, des **métriques** et des **fichiers de sortie** lors de l'exécution des modèles de Machine Learning et pour la **visualisation ultérieure des résultats**.

MLflow Tracking permet de logger les expériences à chaque fois qu'elles sont lancées.



mlflow-default-risk Provide Feedback

Experiment ID: 571819232420284944 Artifact Location: file:///C:/Users/mlie/01_PYTHON/4.%20OPEN_CLASSROOMS/07_PROJET_7/mlruns/571819232420284944

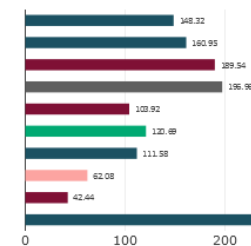
> Description Edit

Table view Chart view metrics.rmse < 1 and params.model = "tree" Sort: Created Columns

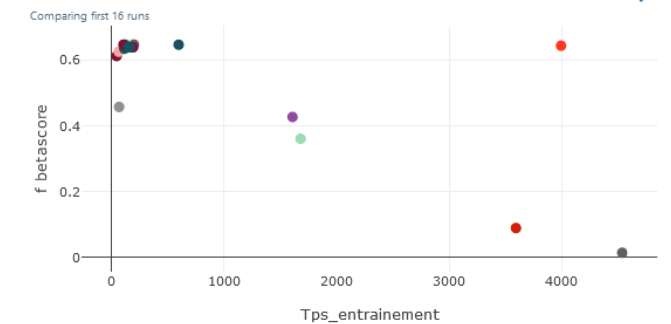
Time created: All time State: Active

	Run Name	Created	Duration	User	Source	Version	Models	Metrics
								Tps_entraineme accuracy auc
	LightGBM 30 - FLAML	21 days ago	3.2min	mlie	C:\Users...	aee49e	sklearn	148.3 0.716 0.691
	Lgb - no pipeline	22 days ago	3.8min	mlie	C:\Users...	efa0cc	LightGBM_n.../1	160.9 0.716 0.691
	LightGBM 30 - FLAML	23 days ago	5.5min	mlie	C:\Users...	efa0cc	LightGBM/1	189.5 0.716 0.691
	LightGBM 30 - Beta = 2	23 days ago	3.5min	mlie	C:\Users...	efa0cc	sklearn	197 0.703 0.689
	LightGBM 30 - GridSearchCV	23 days ago	1.8min	mlie	C:\Users...	efa0cc	sklearn	103.9 0.703 0.689
	LightGBM - Feature Importance *30	23 days ago	2.1min	mlie	C:\Users...	e226a2	sklearn	120.7 0.703 0.689
	LightGBM - Feature Importance *20	23 days ago	2.0min	mlie	C:\Users...	e226a2	sklearn	111.6 0.696 0.679
	LightGBM - Feature Importance *10	23 days ago	1.1min	mlie	C:\Users...	e226a2	sklearn	62.08 0.687 0.67
	LightGBM - Feature Importance *5	23 days ago	48.6s	mlie	C:\Users...	e226a2	sklearn	42.44 0.677 0.659
	LightGBM	23 days ago	10.0min	mlie	C:\Users...	e226a2	sklearn	594.7 0.715 0.694
	Régression Logistique - SMOTE 10/60	26 days ago	26.9min	mlie	C:\Users...	e226a2	sklearn	1607.5 0.829 0.649
	Régression Logistique - SMOTE 20/30	26 days ago	1.0h	mlie	C:\Users...	e226a2	sklearn	3595 0.911 0.536
	Régression Logistique - SMOTE 10/50	26 days ago	28.1min	mlie	C:\Users...	e226a2	sklearn	1679.1 0.857 0.632
	Régression Logistique - Class Weight	26 days ago	1.1h	mlie	C:\Users...	e226a2	sklearn	3996.6 0.69 0.68
	Régression Logistique	26 days ago	1.3h	mlie	C:\Users...	e226a2	sklearn	4540.8 0.919 0.506
	Baseline - Dummy classifier	26 days ago	1.3min	mlie	C:\Users...	e226a2	sklearn	65.22 0.498 0.493

Tps_entrainement
Comparing first 10 runs

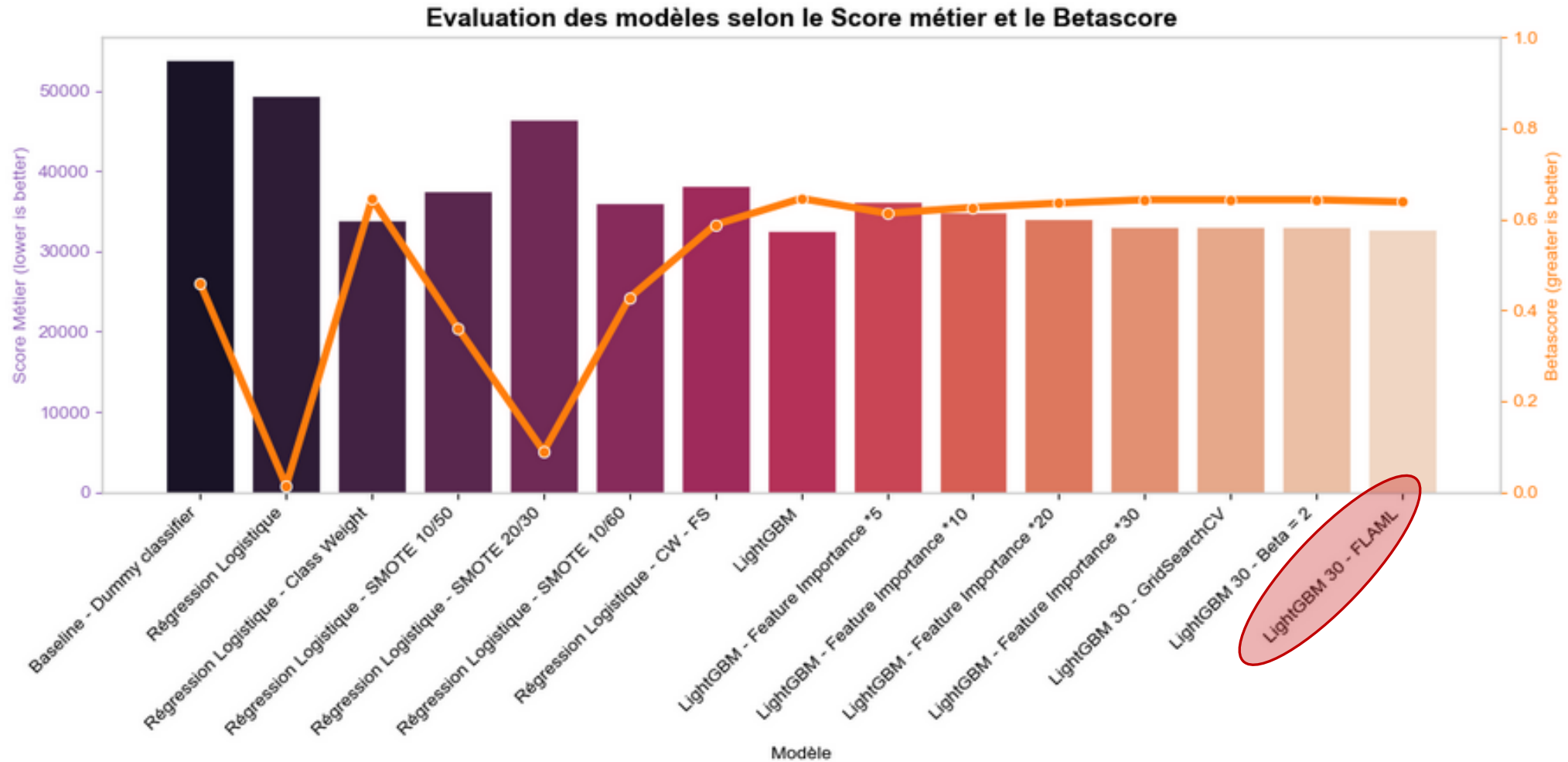


Tps_entrainement vs. f betascore

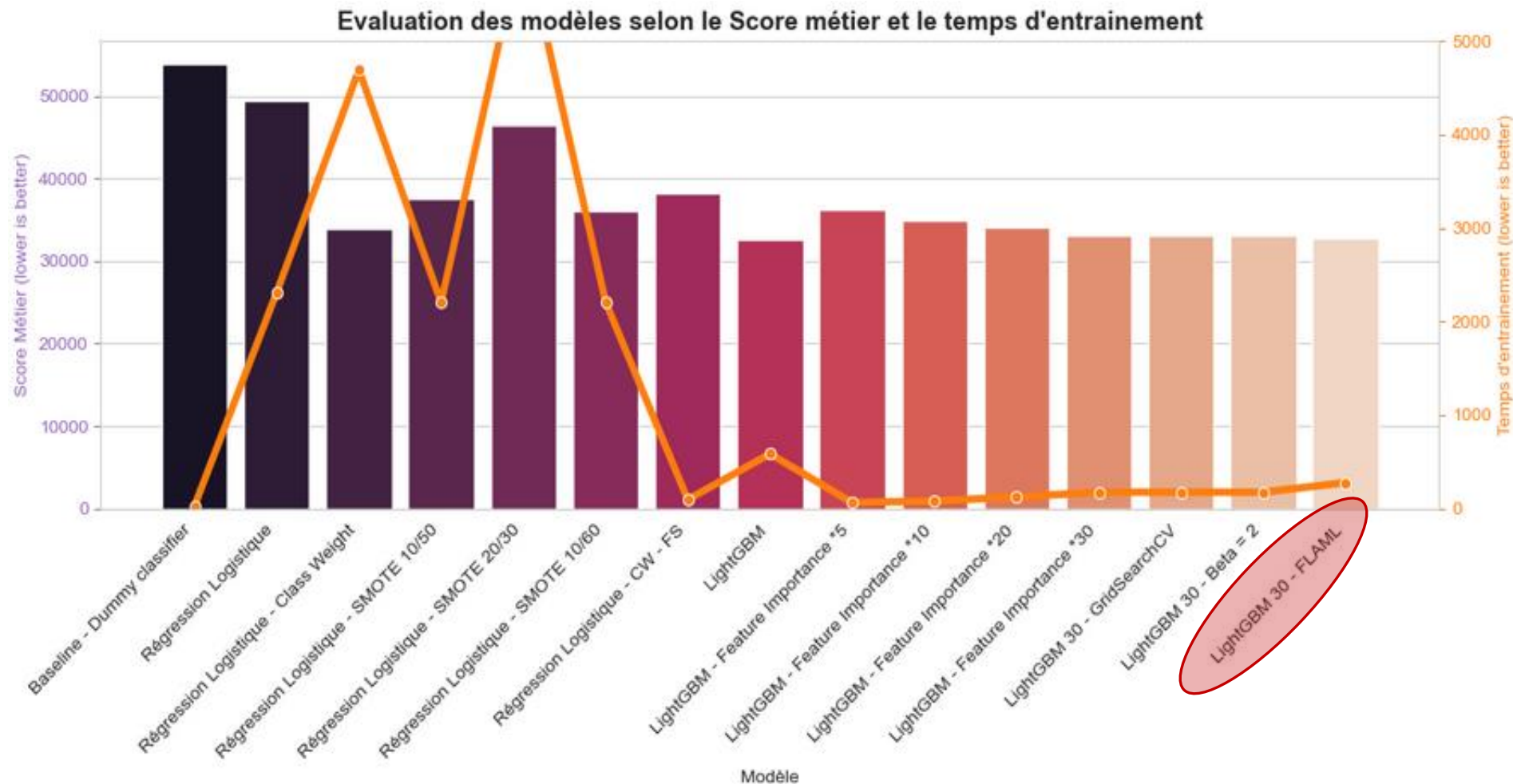


II. Modélisation: synthèse des résultats

3 modèles testés (15 versions au total)



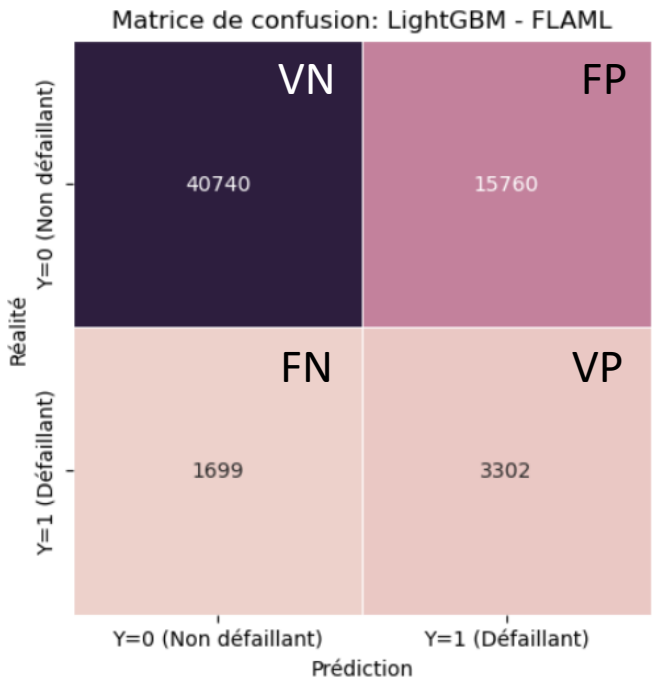
II. Modélisation: synthèse des résultats



Le LightGBM avec les 30 features les plus importantes optimisé via FLAML est le modèle qui associe:

- un **score métier faible**
- un **temps d'entrainement relativement court**
- un **f-beta score important**
- un **nombre de features relativement faible**
- des **features facilement explicables** pour quelqu'un de non expert en datascience

II. Modélisation: modèle retenu LightGBM30 - FLAML

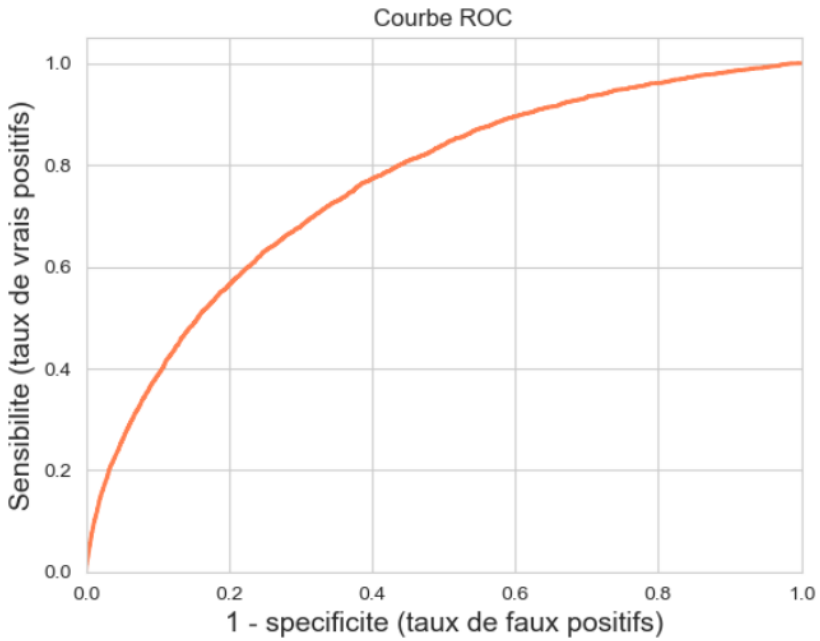


Le modèle prédit correctement 72% du total des clients (**Accuracy**)

Le modèle détecte 66% de clients défaillants (**Recall**) mais ses prédictions ne sont correctes que dans 17% des cas (**Précision**).

Les clients non défaillants sont trop souvent prédits défaillants.

Modèle	Best_Params	Score_metier	Betascore	Recall	Precision	Accuracy	AUC	Train_Time
LightGBM 30 - FLAML	{'n_estimators': 31204, 'num_leaves': 4, 'min_child_samples': 3, 'learning_rate': 0.009033979476164342, 'log_max_bin': 10, 'colsample_bytree': 0.5393339924944204, 'reg_alpha': 15.800090067239827, 'reg_lambda': 34.82471227276953}	32750	0.64	0.66	0.17	0.72	0.69	277.58

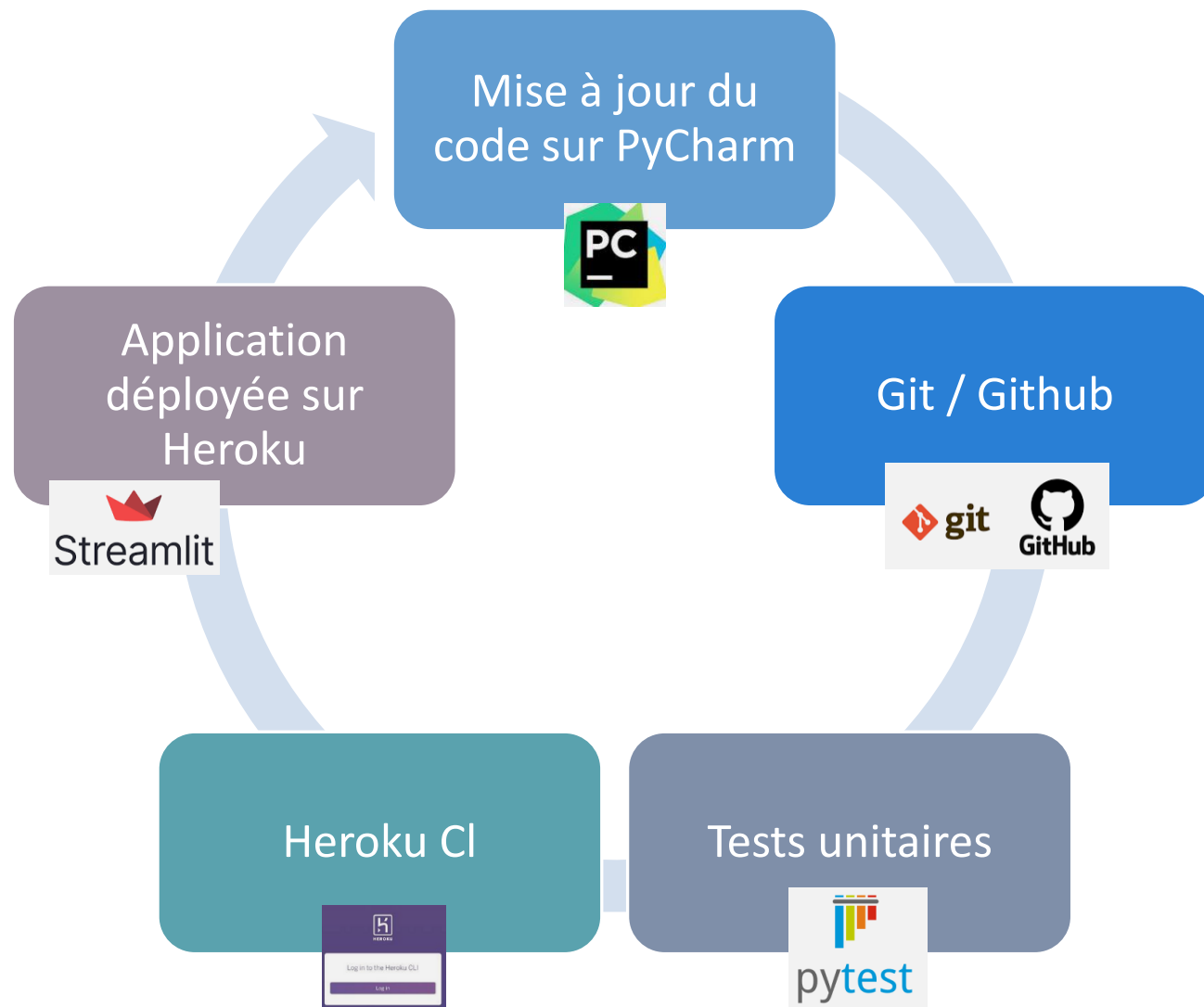


L'aire sous la courbe (**AUC**) représente la capacité d'un modèle à discriminer les classes positives et négatives.

Ici l'AUC de 0,69 signifie que le modèle est **meilleur qu'un modèle aléatoire** (aire de 0,5) mais n'est **pas parfait** (aire de 1).

III. PIPELINE DE DEPLOIEMENT

III. Pipeline de déploiement



III. Pipeline de déploiement - commits

Lien du dossier [GitHub](#)



Milie-Mimi / OC_DS_Project7_ Public

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

Settings

main 1 branch 0 tags

Go to file

Add file

<> Code

Milie-Mimi add dashboard link 6883184 4 minutes ago 96 commits

API_Heroku	Add API link	7 minutes ago
Dashboard_API	add dashboard link	4 minutes ago
Notebooks	Ajout Readme découpage dossier Notebooks	yesterday
Data_Drift_Analysis.html	Ajout tableau HTML d'analyse du data drift	3 days ago
GROSCHENE_Emilie_4_note_méthod...	Ajout note méthodologique	3 days ago
README.md	Mise en page README	3 weeks ago

III. Pipeline de déploiement - commits

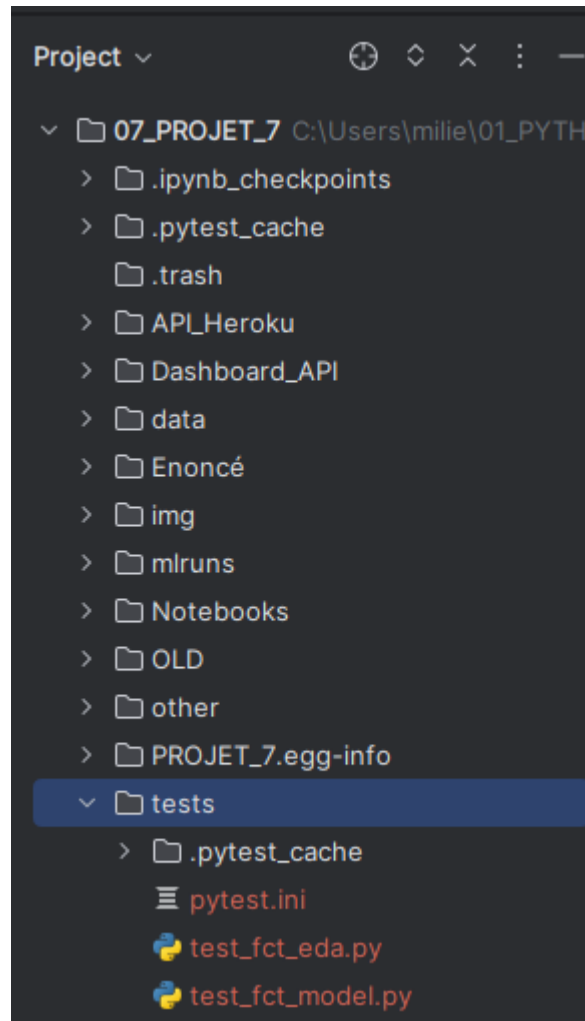
Commits

main

Commits on Jun 8, 2023

add dashboard link Milie-Mimi committed 7 minutes ago	Verified		6883184	<>
Add API link Milie-Mimi committed 10 minutes ago	Verified		b8d95d7	<>
add scaler Milie-Mimi committed 12 minutes ago	Verified		22e8409	<>
Delete fast_api.py Milie-Mimi committed 13 minutes ago	Verified		5c3602b	<>
version définitive Milie-Mimi committed 14 minutes ago	Verified		f675c25	<>
ajout numpy version Milie-Mimi committed 14 minutes ago	Verified		2ec94e4	<>
version définitive Milie-Mimi committed 15 minutes ago	Verified		31e7c01	<>
ajout logo pour affichage readme Milie-Mimi committed 16 minutes ago	Verified		335e1fb	<>

III. Pipeline de déploiement – tests unitaires



```
from Notebooks import fct_model

def test_get_data_check_return_right_columns():
    # given
    expected = ['SK_ID_CURR', 'AGE', 'CODE_GENDER', 'NAME_EDUCATION_TYPE_Lower Secondary & Secondary',
                'YEARS_EMPLOYED', 'YEARS_ID_PUBLISH', 'YEARS_LAST_PHONE_CHANGE', 'REGION_POPULATION_RELATIVE',
                'AMT_CREDIT', 'AMT_GOODS_PRICE', 'CREDIT_GOODS_PERC', 'CREDIT_DURATION', 'AMT_ANNUITY', 'DEBT_RATIO',
                'PAYMENT_RATE', 'EXT_SOURCE_2', 'PREV_YEARS_DECISION_MEAN', 'PREV_PAYMENT_RATE_MEAN',
                'INSTAL_DAYS_BEFORE_DUE_MEAN', 'INSTAL_PAYMENT_DIFF_MEAN', 'INSTAL_DAYS_PAST_DUE_MEAN',
                'POS_MONTHS_BALANCE_MEAN', 'POS_CNT_INSTALMENT_FUTURE_MEAN', 'POS_NB_CREDIT',
                'BURO_AMT_CREDIT_SUM_SUM', 'BURO_YEARS_CREDIT_ENDDATE_MAX', 'BURO_AMT_CREDIT_SUM_DEBT_SUM',
                'BURO_YEARS_CREDIT_ENDDATE_MEAN', 'BURO_AMT_CREDIT_SUM_MEAN', 'BURO_CREDIT_ACTIVE_Active_SUM',
                'BURO_AMT_CREDIT_SUM_DEBT_MEAN']

    # when
    dataframe = fct_model.get_data(nrows=2)
    actual = list(dataframe.columns)

    # then
    assert expected == actual

def test_score_metier_check_return_calculation():
    # given
    ytest = [0, 0, 1, 1, 0, 1]
    y_pred = [0, 1, 1, 0, 0, 1]
    expected = 11

    # when
    actual = fct_model.score_metier(ytest=ytest,
                                   y_pred=y_pred)

    # then
    assert expected == actual
```

III. Pipeline de déploiement – tests unitaires

```
(projet7) PS C:\Users\milie\01_PYTHON\4. OPEN_CLASSROOMS\07_PROJET_7> python -m pytest
===== test session starts =====
platform win32 -- Python 3.9.16, pytest-7.3.1, pluggy-1.0.0
rootdir: C:\Users\milie\01_PYTHON\4. OPEN_CLASSROOMS\07_PROJET_7
plugins: anyio-3.5.0
collected 3 items
..\..\..\anaconda3\envs\projet7\lib\site-packages\shap\plots\_image.py:18
..\..\..\anaconda3\envs\projet7\lib\site-packages\shap\plots\_text.py:9
  Importing display from IPython.core.display is deprecated since IPython 7.14, please import from IPython display

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 3 passed, 4 warnings in 16.96s =====
```

IV. ANALYSE DE DATA DRIFT

IV. Analyse de data drift

Application_train

Données pour la
modélisation

Application_test

Données des nouveaux clients
une fois le modèle en production

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
PAYMENT_RATE	num			Detected	Wasserstein distance (normed)	0.582838
CREDIT_DURATION	num			Detected	Wasserstein distance (normed)	0.582654
AMT_GOODS_PRICE	num			Detected	Wasserstein distance (normed)	0.236377
AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.234577
AMT_ANNUIITY	num			Detected	Wasserstein distance (normed)	0.149195
YEARS_LAST_PHONE_CHANGE	num			Detected	Wasserstein distance (normed)	0.129542
PREV_PAYMENT_RATE_MEAN	num			Detected	Wasserstein distance (normed)	0.103929

Comparaison des distributions de chaque variable dans les deux datasets.

Le **test statistique** (ou la métrique appropriée) est automatiquement choisi en fonction du **type des données** et de son **volume**. Il s'agit ici de la **distance normalisée de Wasserstein** (méthode par défaut pour les données tabulaires numériques supérieures à 1 000 lignes).

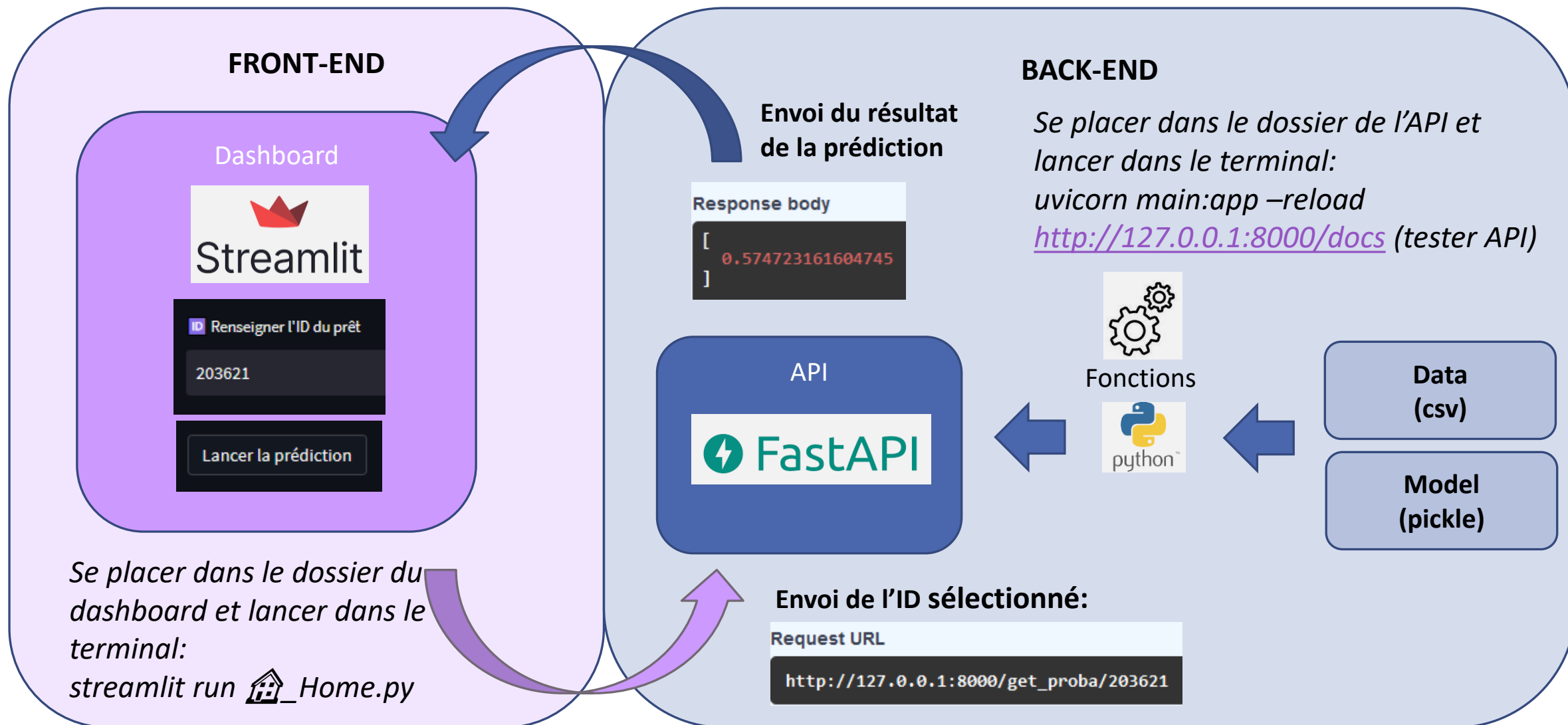


Sur les 30 features de notre modèle de scoring, **7 features** montrent du data drift (soit **23,33%** du nombre total de features).

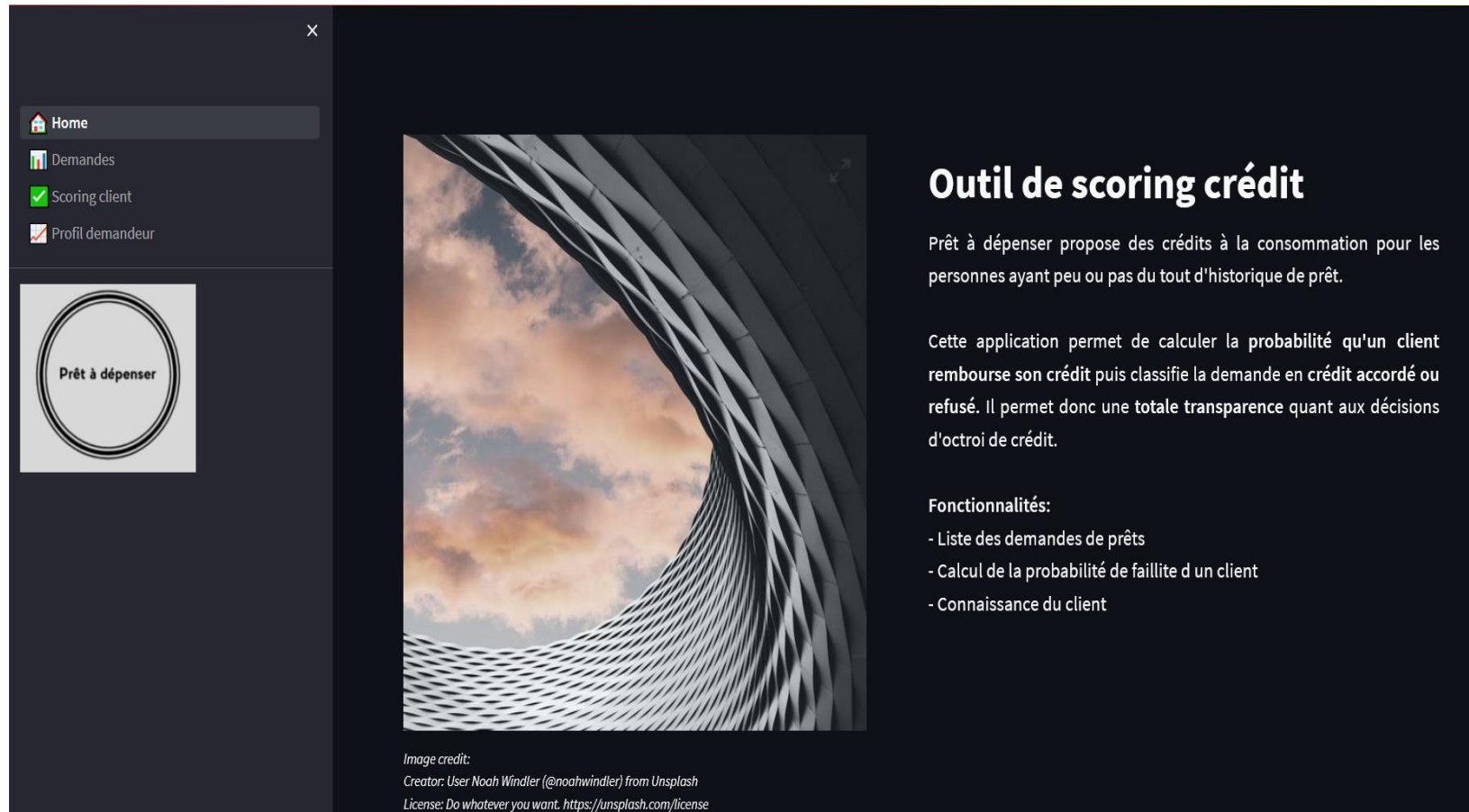
Drift score: **nombre d'écarts types en moyenne** qu'il faudrait déplacer pour chaque ID du groupe actuel pour qu'il corresponde au groupe de référence avec seuil de détection du data drift de 0,1.

V. DASHBOARD

V. Dashboard: architecture locale



V. Dashboard: démo du dashboard déployé sur Heroku



FastAPI 0.1.0 OAS3
[/openapi.json](#)

default

GET /get_glossary/ Get Glossary

GET /get_loans/ Get Loans

GET /get_proba/{loan_ID} Get Proba

GET /get_feature_importance/ Get Feature Importance

Lien API



Lien du dashboard



VI. CONCLUSIONS

VI. Conclusions



Implémentation d'un modèle de classification binaire avec classes déséquilibrées dans une application hébergée sur le cloud.

Axes d'amélioration:

- Echanges avec Prêt à dépenser pour pallier à la **méconnaissance du secteur bancaire**:
 - **Feature engineering** et **sélection de variables**
 - Choix des **métriques d'évaluation** et du **seuil de prédiction**
 - Mise en place **d'alertes** en cas d'une dégradation significative de la performance (monitoring evidently)
- Ajout de **fonctionnalités** supplémentaires dans le dashboard
- Variables **renommées**
- **Clustering** pour situer le client parmi des profils similaires
- Ajout **d'autres graphiques** (interactifs avec plotly par exemple)
- Utilisation de **st.session_state** pour garder en mémoire des paramètres et accélérer / améliorer l'affichage de la page
- Ajout de **tests unitaires** supplémentaires pour chaque fonction



MERCI
