Implémentation d’un modèle de scoring

https://github.com/herospiki/OC

Prêt à dépenser

Le contexte

‘Credit scoring’

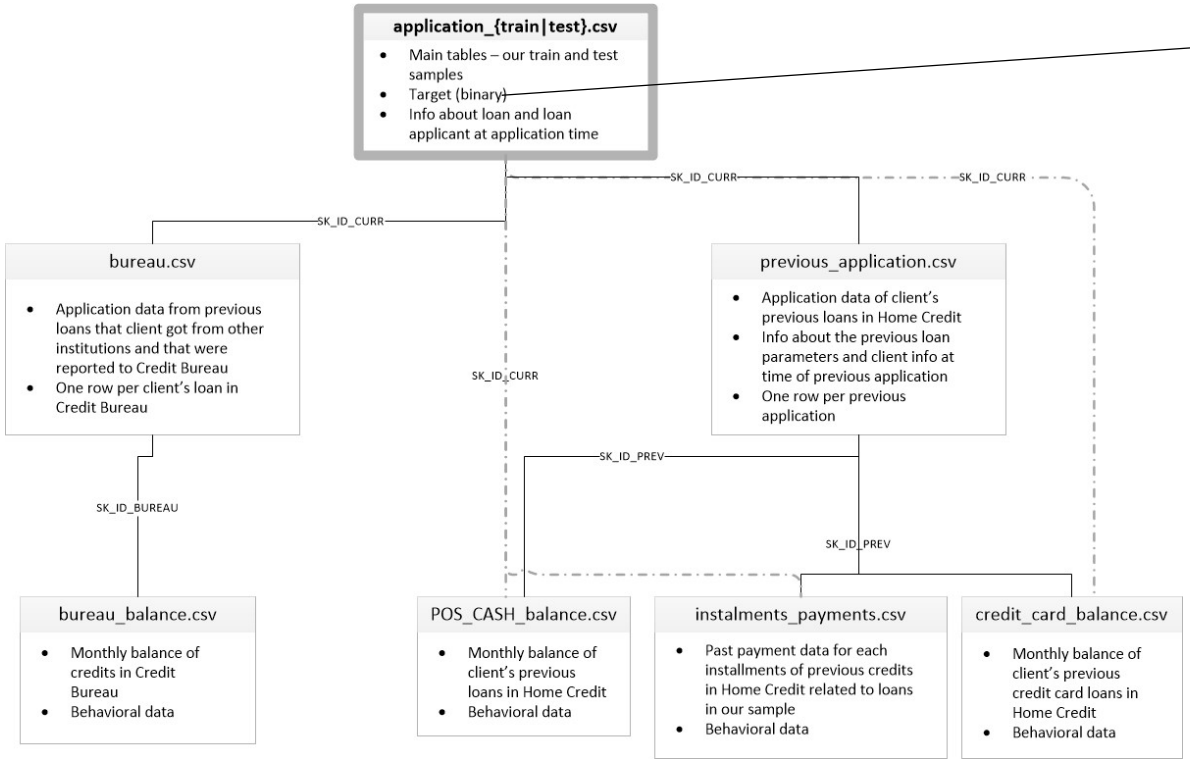
"Prêt à dépenser", propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

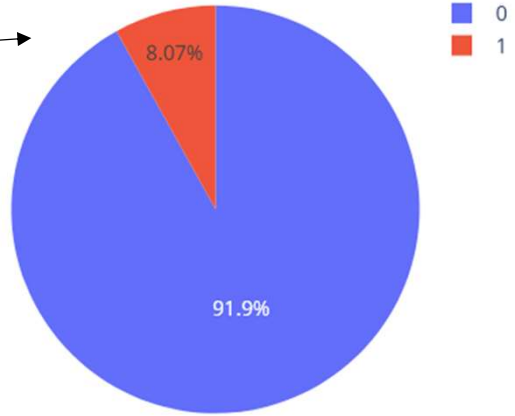
Objectif

- Mise en œuvre un outil de “scoring crédit” pour calculer la probabilité qu’un client rembourse son crédit
- Classification de la demande en crédit accordé ou refusé.
- Transparence vis-à-vis des décisions d’octroi de crédit.
- Un dashboard interactif pour :
  - expliquer les décisions d’octroi de crédit,
  - permettre l’exploration des informations personnelles du client

Le jeu de données

Le jeu de données est téléchargeable ici : https://www.kaggle.com/competitions/home-credit-default-risk






```
0.0    282686
1.0     24825
Name: TARGET,
dtype: int64
```

1 - client en difficulté de paiement
0 - tous les autres cas

Préparation des données

Cleaning et feature engineering

J'ai repris le feature engineering de ce kernel, et l'ai légèrement adapté
https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features

- Suppression des variables les moins bien renseignées
- Regroupement de certaines valeurs catégorielles pour éviter trop de dimensions à l'encoding
- One-hot-encoding avec NAN comme catégorie
- Imputation par la médiane pour les variables numériques de la tablea Application
- Reprise du feature engineering déjà effectué dans le kernel de référence

Après merge des dataframes :

- Imputation des valeurs vides par 0 après merge et encoding.
- Sélection des 100 variables les plus « importantes » (selectkbest – ANOVA -)

La contrainte métier

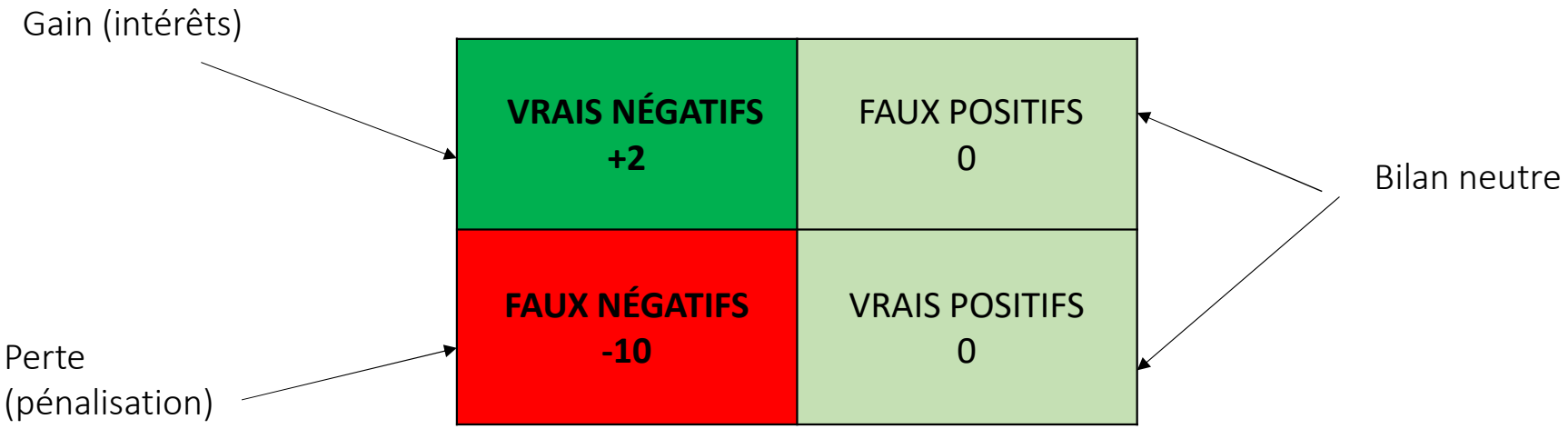
Gain métier

et

métrique d'évaluation du modèle

Le gain métier sur un ensemble test

La banque doit éviter de perdre de l'argent mais aussi en gagner !



$$GAIN = 2 \times Vrais\ Négatifs - 10 \times Faux\ Négatifs$$

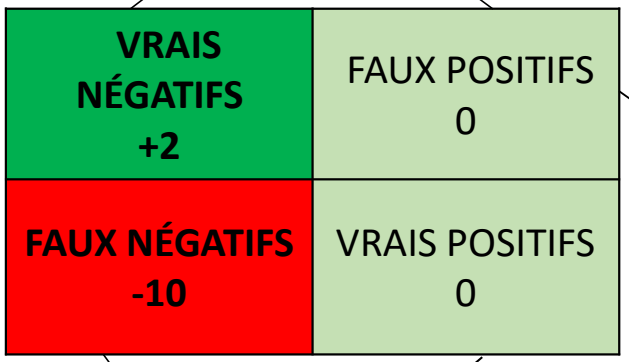
Quelle métrique pour l'évaluation du modèle ?

Fbeta_score ? AUC ? Scorer customisé ?

$$Spécificité = \frac{Vrais\ Négatifs}{(Vrais\ Négatifs + Faux\ Positifs)}$$

AUC : aire sous la courbe ROC (sensibilité en fonction de spécificité) = un bon candidat

?



$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

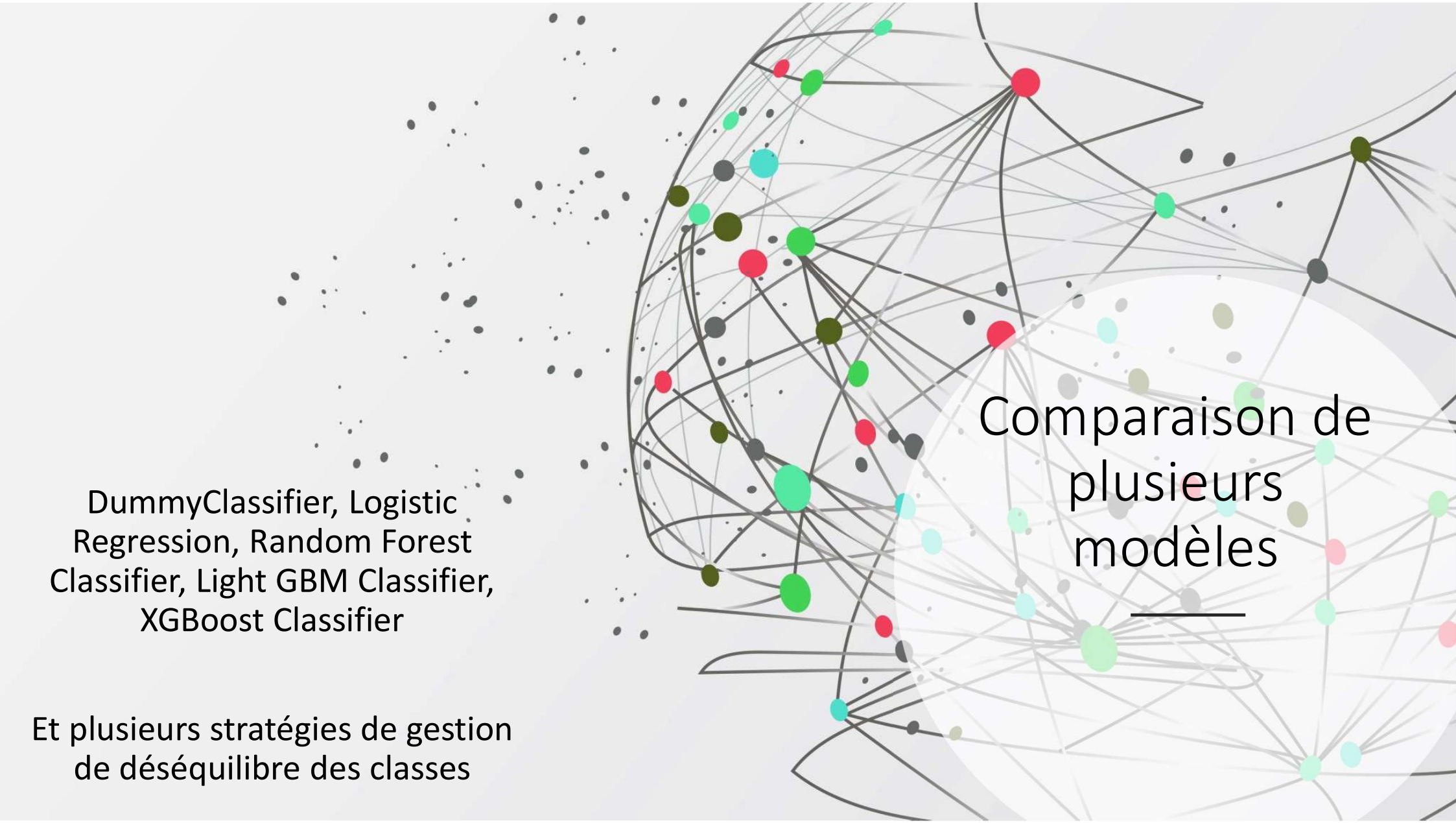
$$Précision = \frac{Vrais\ Positifs}{(Vrais\ Positifs + Faux\ Positifs)}$$

$$Recall\ (sensibilité) = \frac{Vrais\ Positifs}{(Vrais\ Positifs + Faux\ Négatifs)}$$

Gain obtenu sur la prédiction

$$Gain\ (score) = \frac{2 \times Vrais\ Négatifs - 10 \times Faux\ Négatifs}{2 \times Négatifs - 10 \times Positifs}$$

Gain potentiel maximal



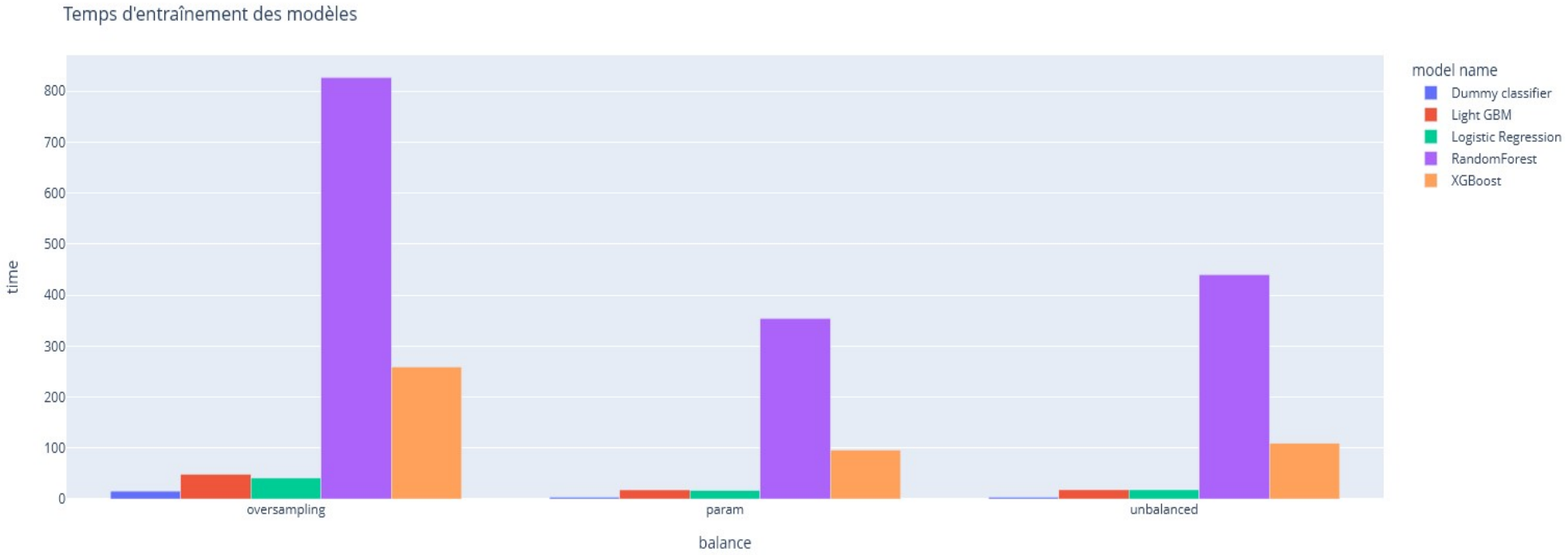
Comparaison de plusieurs modèles

DummyClassifier, Logistic Regression, Random Forest Classifier, Light GBM Classifier, XGBoost Classifier

Et plusieurs stratégies de gestion de déséquilibre des classes

Temps d’entraînement des modèles

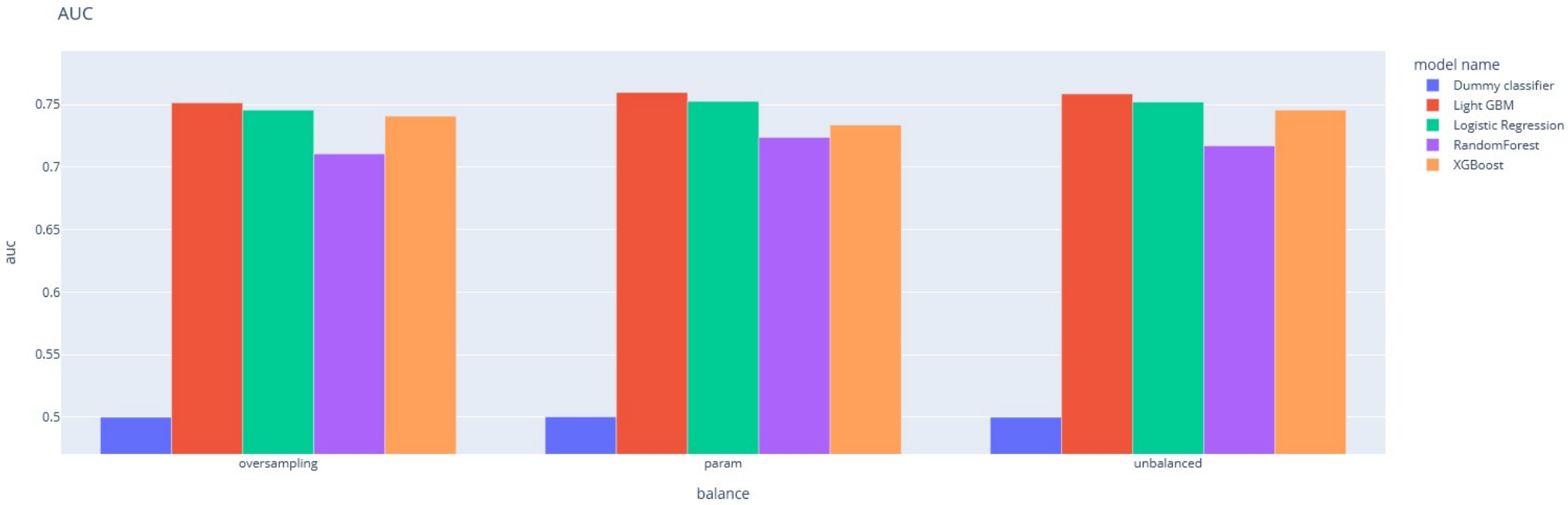
Sur 50% des données



L’oversampling est pénalisant en terme de temps et certains modèles sont très longs à entraîner.
Le classifieur LGBM est aussi rapide que la régression logistique, mais permet plus d’optimisation.

Score AUC

Comparaison entre les modèles

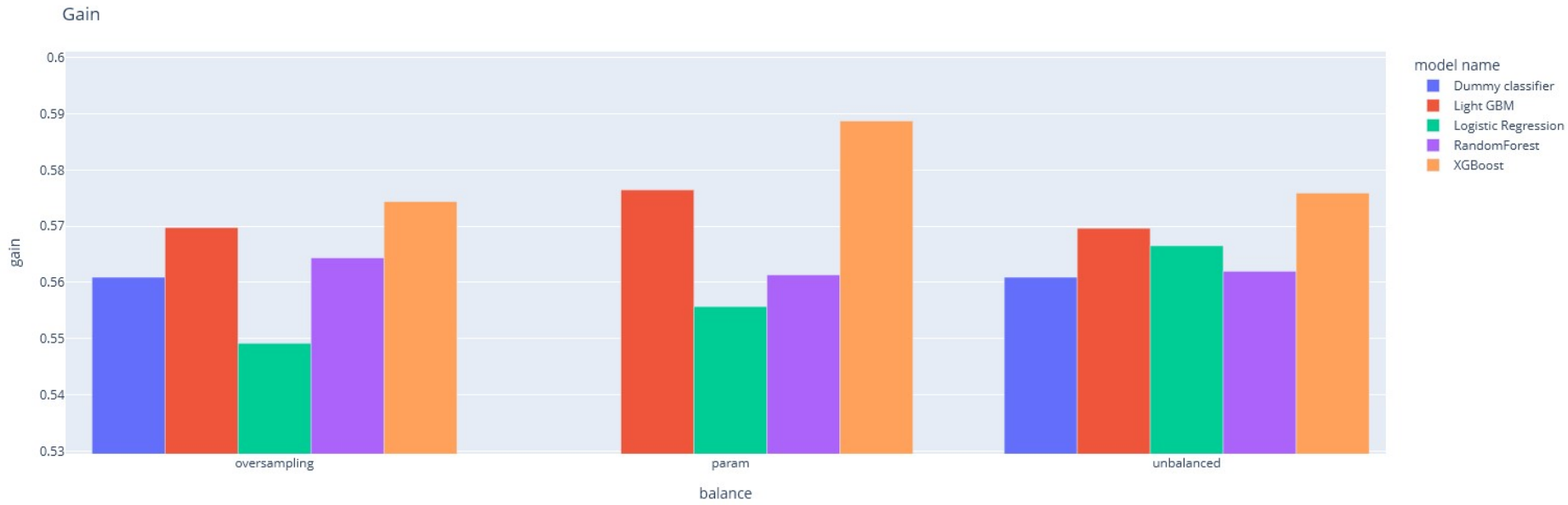


Quelque soit la stratégie, c'est le classifieur LGBM qui donne le meilleur score.

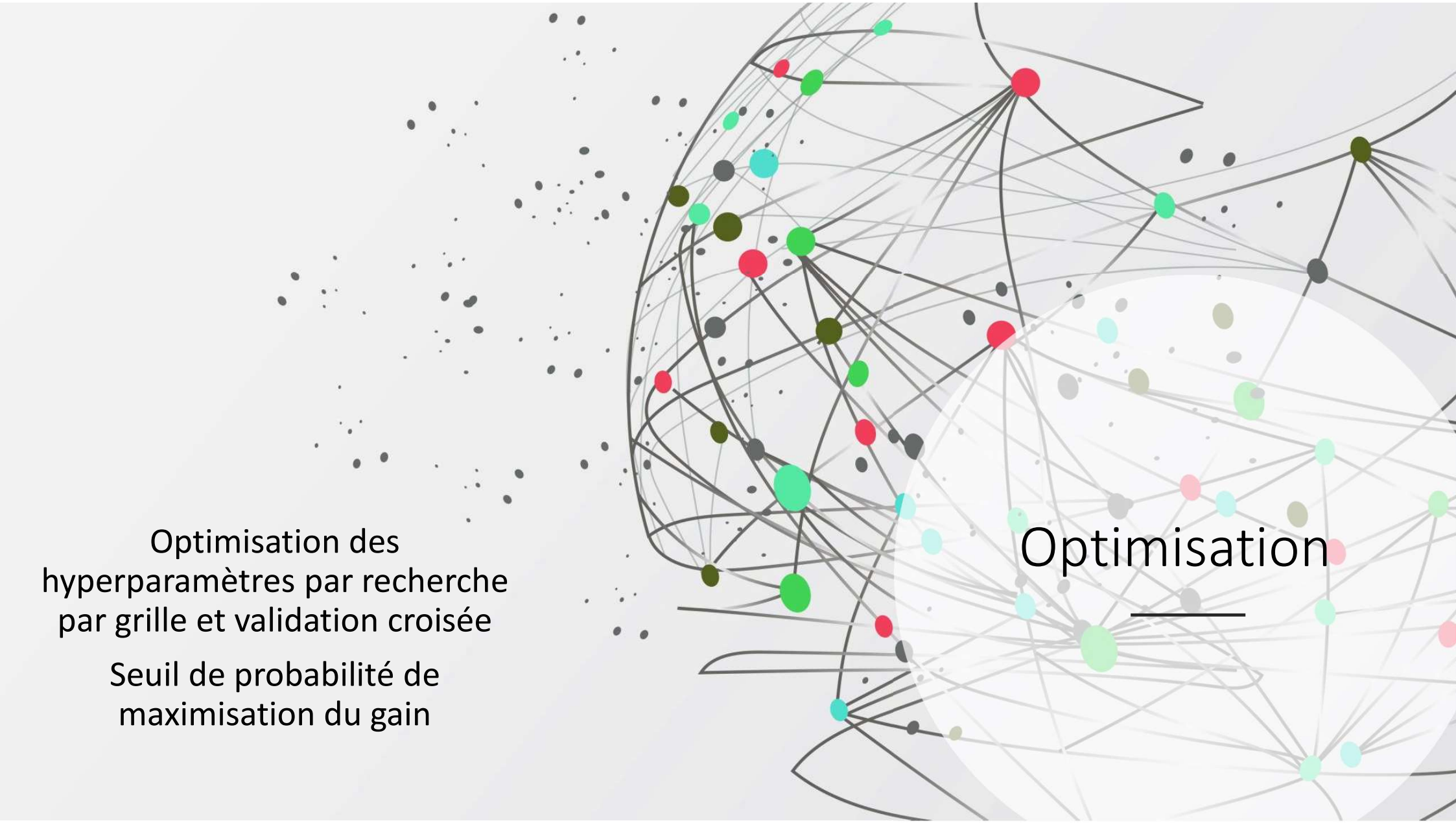
Et pour le gain ?

Le classifieur d'XGBoost est légèrement meilleur mais de peu…

Il faut zoomer pour faire apparaître les différences. Le classifieur LGBM obtient sa meilleure performance avec la paramétrisation du modèle (scale_pos_weight) tandis que l'oversampling n'influe pas sur les résultats du classifieur XGBoost par rapport à la version non rééquilibrée.



Le classifieur Light GBM l'emportait largement au temps ! On peut espérer que l'entraînement sur la totalité des données et l'optimisation amélioreront le score.



Optimisation

Optimisation des hyperparamètres par recherche par grille et validation croisée

Seuil de probabilité de maximisation du gain

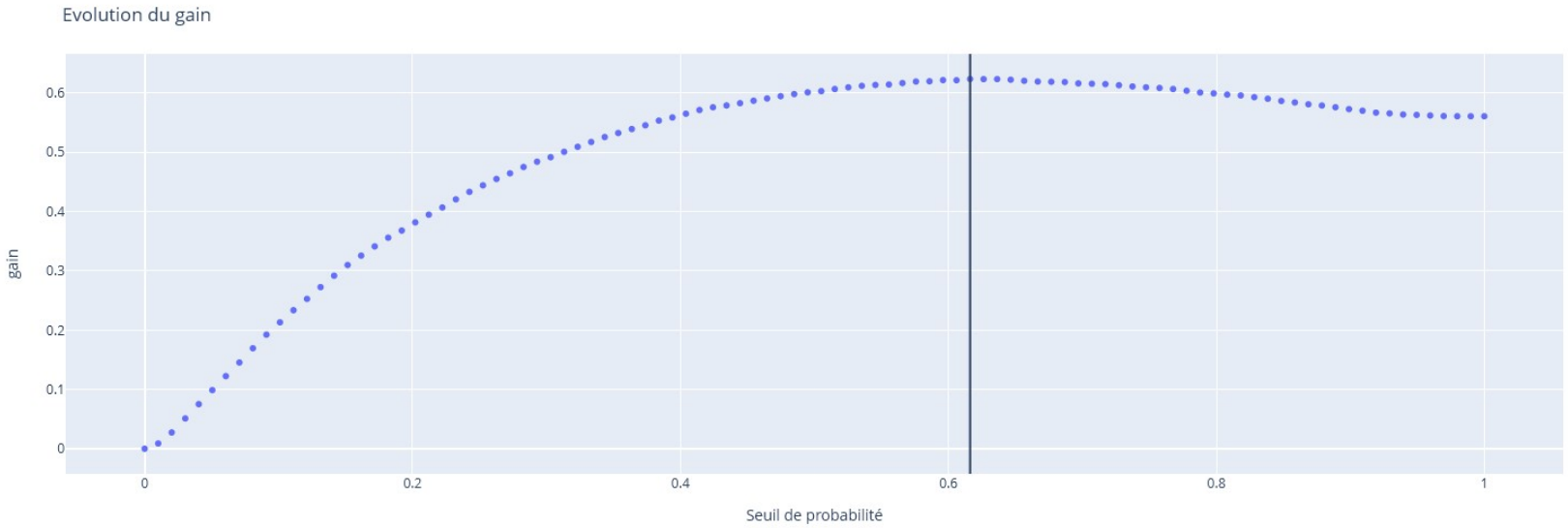
Recherche par grille avec validation croisée

Modèle : LGBMClassifier, scorer : gain

- Séparation en deux ensembles : entraînement du modèle et validation (20% des données)
- Tests des combinaisons de valeurs des hyperparamètres
- 5 folds stratifiés (stratifiedKfold) : l'ensemble d'entraînement est divisé entre 5 plis testés alternativement. 4 sont utilisés pour l'entraînement et 1 pour le test. Le résultat est la moyenne des scores de tests.
- Evaluation du meilleur modèle obtenu sur l'ensemble de validation
- Observation générale : l'amélioration importante des scores d'entraînement fait baisser les scores de tests (overfitting)

Maximisation du gain

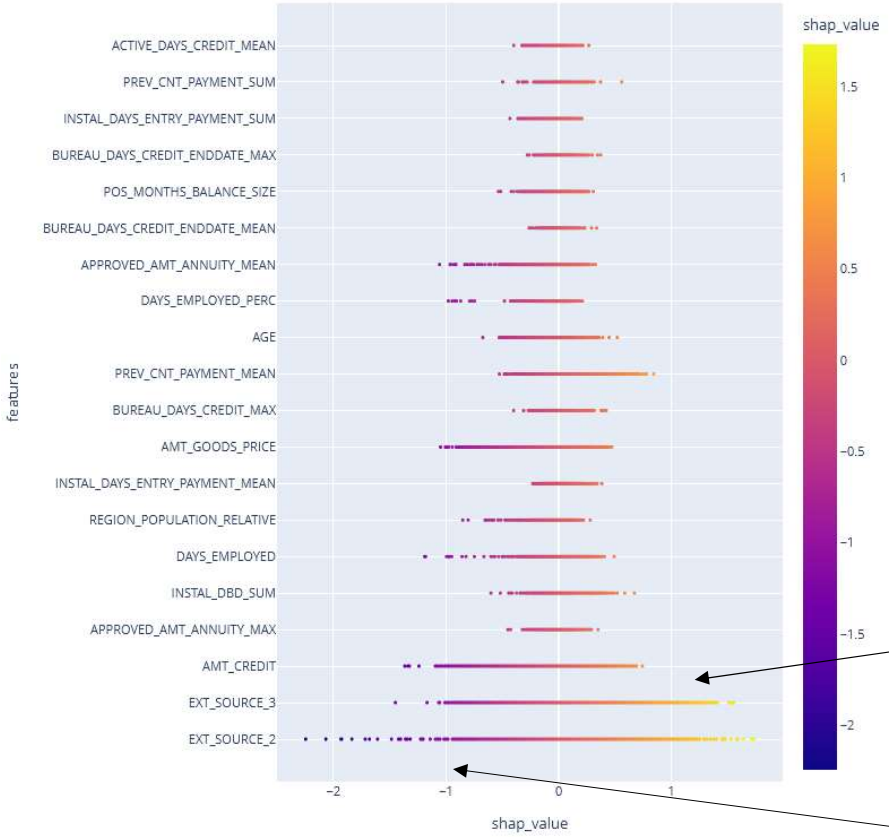
Détermination du seuil de probabilité



Threshold	Matrice de confusion	auc	**f2_score**	accuracy	precision	recall	gain
0.5	[[45673 10865] [2317 2648]]	0.7435	0.3967	0.7857	0.1960	0.5333	0.6029
0.61616	[[50279 6259] [3003 1962]]	0.7435	0.3493	0.8494	0.2387	0.3952	0.6237

Interprétabilité locale du modèle

Ou comment expliquer la décision au client



Contrairement à l'importance des variables qui ne donne qu'une vision globale, l'utilisation des valeurs SHAP des variables lors d'une décision pour un cas individuel permet d'obtenir des informations précises sur le sens de l'influence des variables.

Les influences respectives pour chaque individu d'un sous-ensemble des clients ont été représentées sur le graphique. Il est facile de constater qu'une même feature peut influencer la décision de façon positive ou bien négative.

valeur shap positive : influence en faveur du refus de prêt

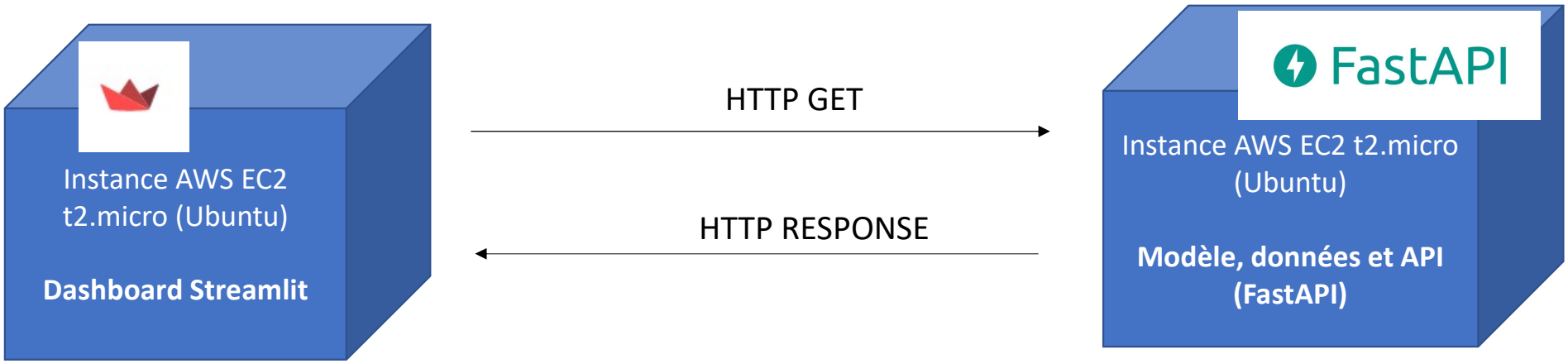
valeur shap négative : Influences en faveur de l'acceptation

Déploiement de l’application

Streamlit et FastAPI sur deux instances AWS EC2 distinctes

Architecture de déploiement

Streamlit et FastAPI



http://13.39.10.188:8501/

http://15.236.4.1/

Place à la démonstration !


Merci de votre attention ☺

BACK-UP - DEMO

```
Last login: Sat Dec 17 18:51:07 2022 from 35.180.112.83
ubuntu@ip-172-31-34-64:~$ cd frontend/
ubuntu@ip-172-31-34-64:~/frontend$ streamlit run dashboard.py

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.


  You can now view your Streamlit app in your browser.

  Network URL: http://172.31.34.64:8501
  External URL: http://13.39.10.188:8501
```

i-085fbb3f653163059 (dashboard)

PublicIPs: 13.39.10.188 PrivateIPs: 172.31.34.64

Lancement du dashboard Streamlit

Lancement du serveur uvicorn pour fastAPI

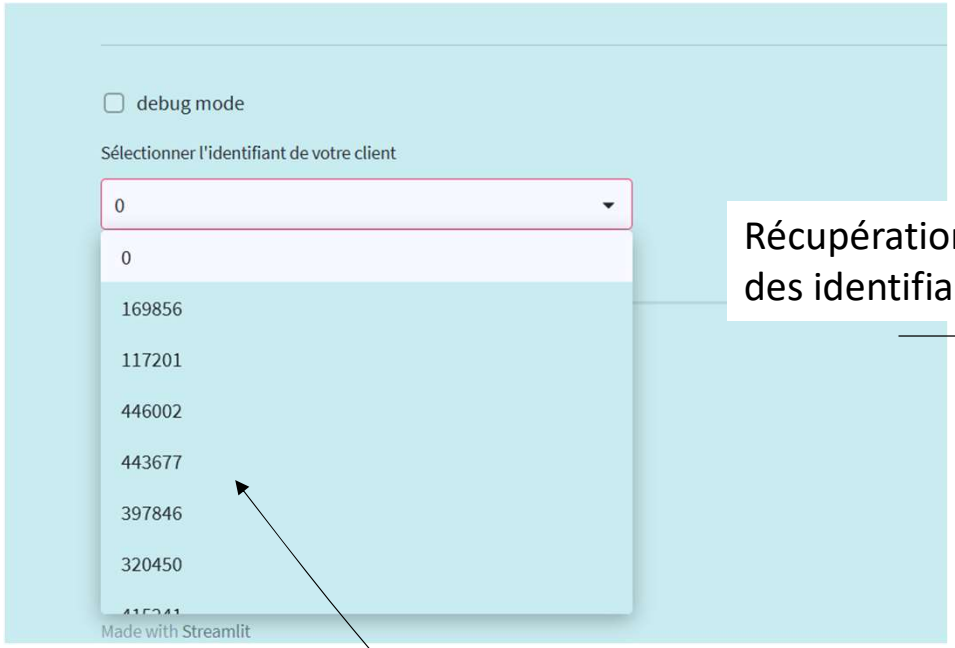
```
Last login: Sat Dec 17 18:14:24 2022 from 35.180.112.84
ubuntu@ip-172-31-39-188:~$ cd backend
ubuntu@ip-172-31-39-188:~/backend$ uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['/home/ubuntu/backend']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [1289] using StatReload
INFO:     Started server process [1291]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

i-036bc086ffd640775 (api_prediction)

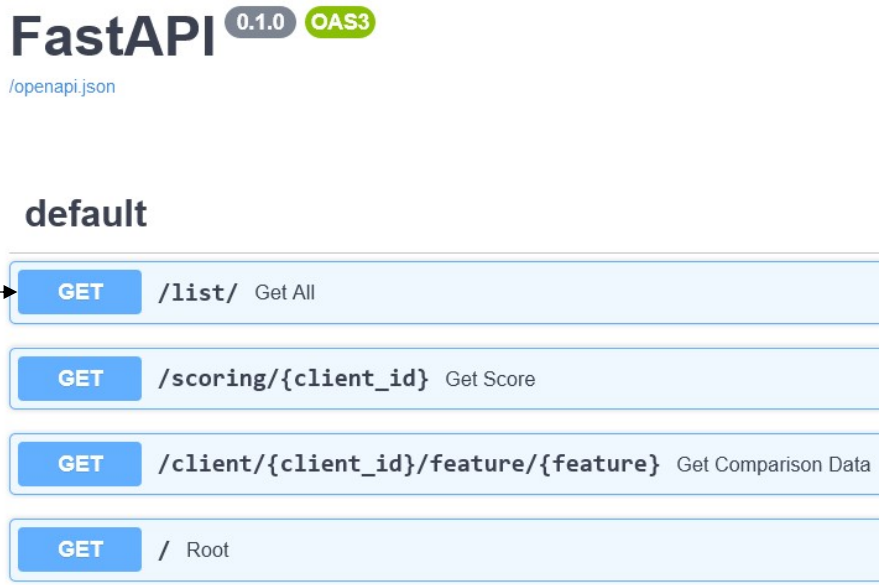
PublicIPs: 15.236.4.1 PrivateIPs: 172.31.39.188

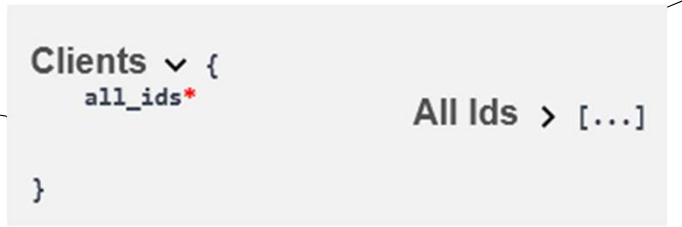
FastAPI

Connexion au dashboard



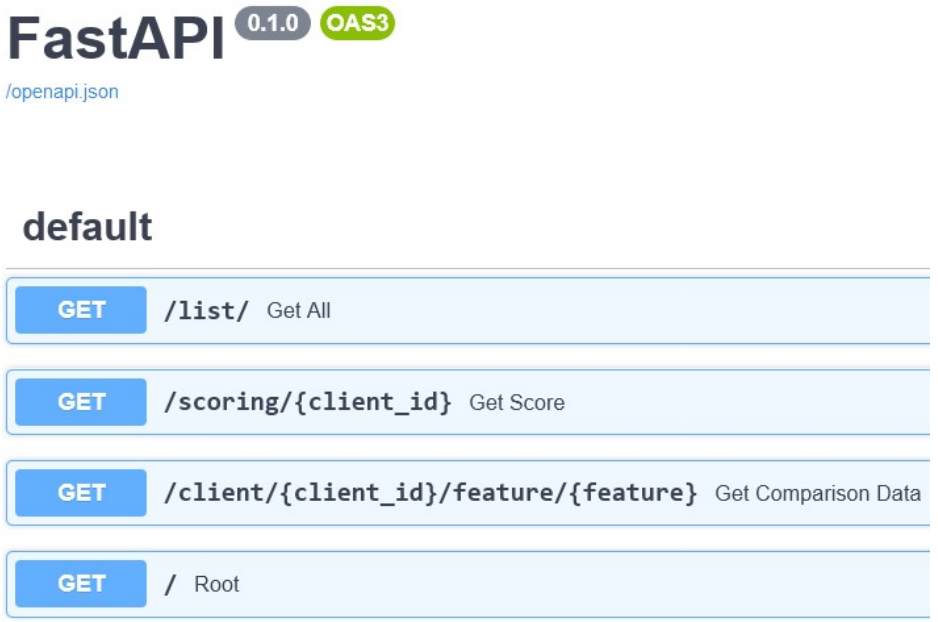
Récupération de la liste des identifiants des clients

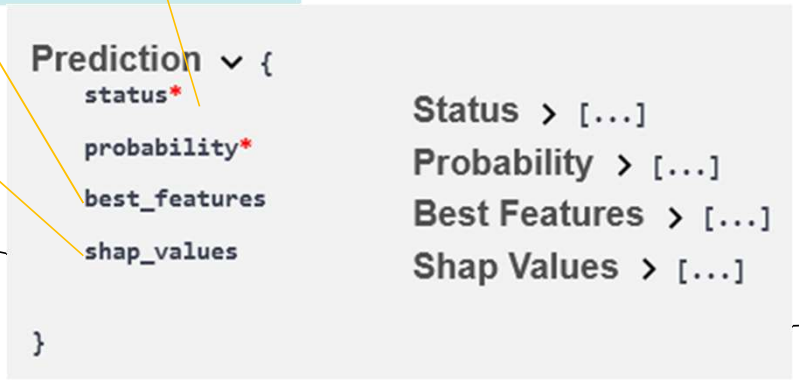




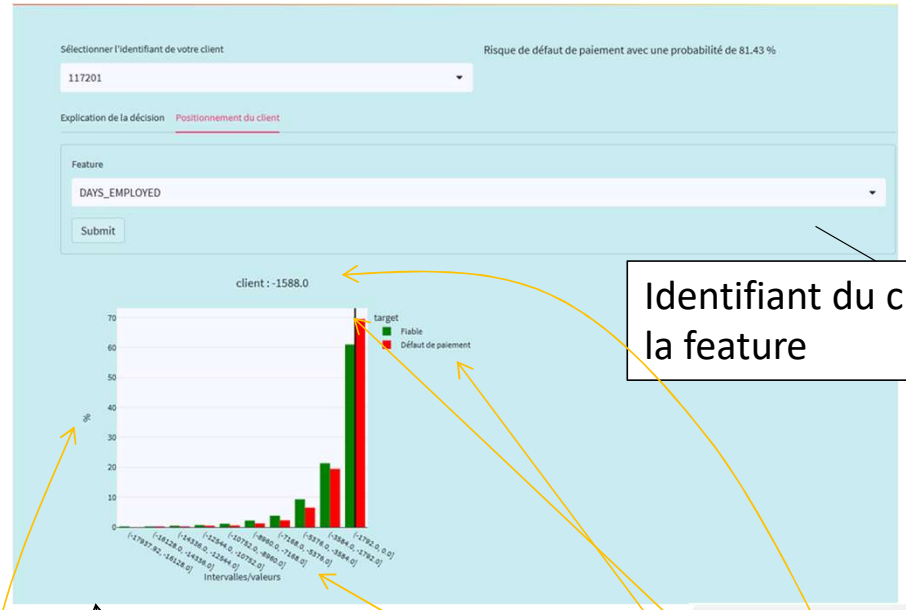
Sélection d'un client

Identifiant du client

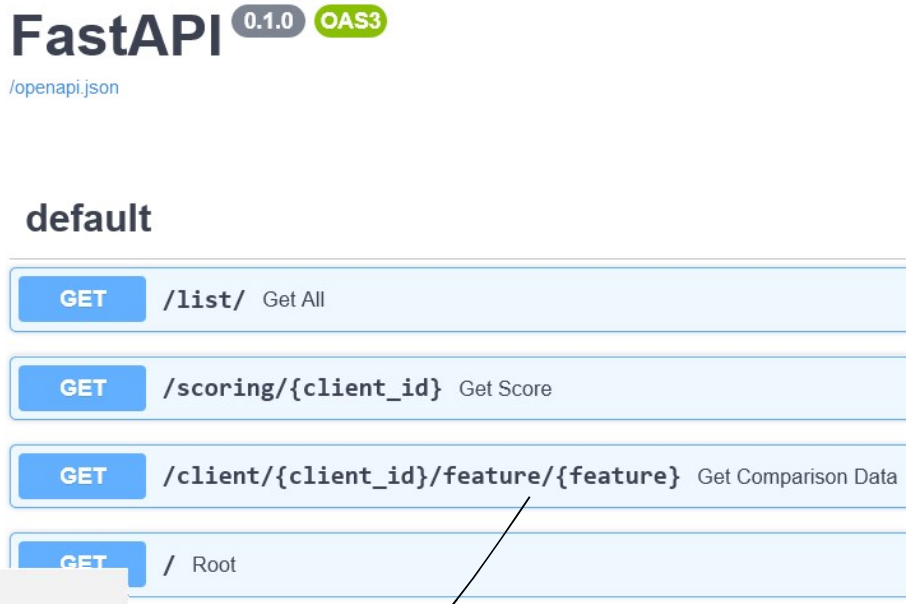


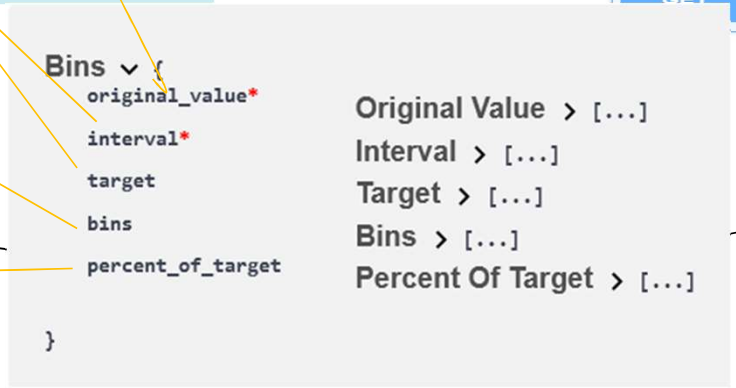


Positionnement du client parmi les autres clients pour une feature donnée, avec détails sur les décisions

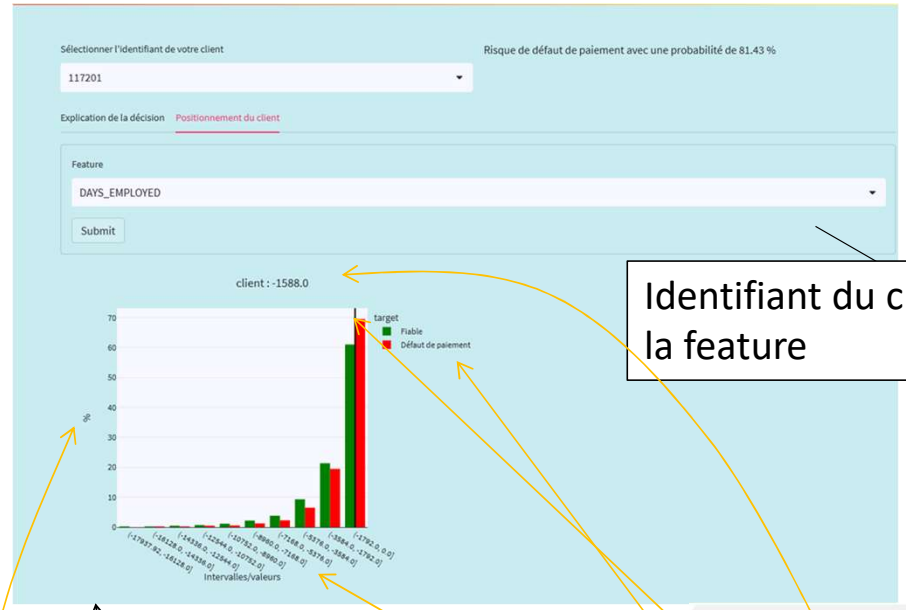


Identifiant du client et nom de la feature





Positionnement du client parmi les autres clients pour une feature donnée, avec détails sur les décisions



Identifiant du client et nom de la feature