

Projet 7 : Implémentation d'un modèle de scoring

présentation du 21 décembre 2021



OPENCCLASSROOMS

Plan de la présentation

- 1) Problématique métier
- 2) Description des données
- 3) Nettoyage et feature engineering
- 4) Développement d'un modèle prédictif :
 - choix du modèle et prise en main
 - rééquilibrage de la classe cible
- 5) Optimisation du modèle en tenant compte d'une fonction de coût métier
- 6) Feature importances globale et locale
- 7) Conception d'une API de prédiction et d'un dashboard interactif
- 8) Conclusion



Problématique de la société "Prêt à dépenser"

Contexte :

- société qui propose des crédits à la consommation.
- besoin d'un outil de scoring pour déterminer si un client remboursera son crédit.
- volonté de transparence sur le scoring, pour les clients.

Missions confiées :

- **développer un modèle de scoring** qui donnera une prédiction sur la probabilité de défaut d'un client.
- **construire un dashboard interactif** pour les chargés de relation client avec explications transparentes.

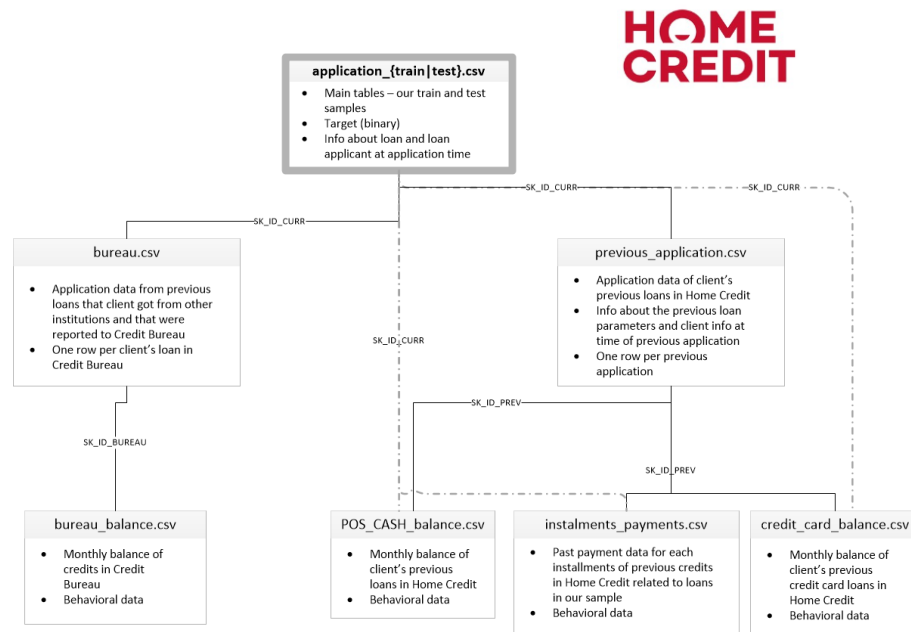
Présentation des données

Source : <https://www.kaggle.com/c/home-credit-default-risk>

8 fichiers de données tabulaires
provenant d'une compétition Kaggle

Description des données source:

- **train dataset** : 307000 demandes de crédits et leur issue (variable binaire "TARGET").
- **test dataset** : 49000 demandes de crédits sans connaissance de l'issue.
- **218 variables** : données détaillées sur le client : emploi, cadre de vie, historique de crédit, de tenue de compte bancaire, ...



Nettoyage et feature engineering

Basé sur le kernel Kaggle de jsaguiar :

- disponible à : <https://www.kaggle.com/jsaguiar/lightgbm-with-simple-features/script>
- **kernel exploitant toutes les tables de données.**
- nettoyage de valeurs aberrantes.
- ajout de **226 variables par feature engineering** (transformations des principales features métier) :
vitesse de remboursement du crédit, taux d'endettement, etc...

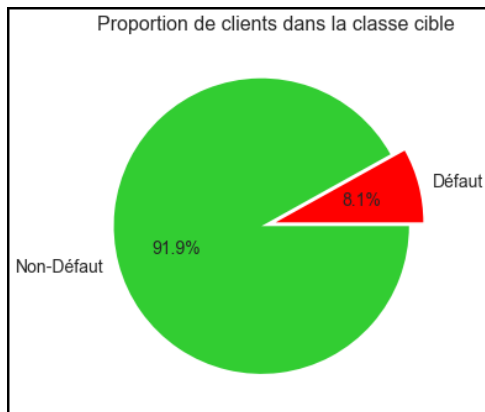
Principaux changements réalisés dans le kernel de jsaguiar :

- remplacement du one-hot-encoding par un **encodage ordinal** des variables catégorielles (pertinence LGBM) :
on passe de 795 à 434 variables, post encodage.
- adaptations spécifiques pour tests de balanced datasets (imputations).

Brève discussion autour des données

Dimension du dataframe utilisé pour la modélisation : 307511 lignes * 431 colonnes
Valeurs manquantes : 37%

Problème majeur : déséquilibre du dataset pour la variable binaire prédite



Analyse exploratoire complète disponible via le dashboard :

[Lien vers le dashboard](#)

Choix des métriques

Importance de **choisir les bonnes métriques** pour l'évaluation du modèle :

- exactitude (accuracy) facilement élevée lorsque la spécificité augmente → mauvais choix.

- **AUC-ROC :**

- intègre la courbe ROC sur toutes les valeurs de spécificité.
- utilisé pour le pré-développement du modèle.
- point de comparaison : compétition Kaggle.

- **fonction coût métier :**

- utilisée pour l'optimisation finale du modèle.

Développement d'un modèle prédictif – choix de l'algorithme

Modèle choisi : Light Gradient Boosting Machine



Algorithme de type « boosting de gradient » sur des forêts aléatoires. Particularités :

- variables continues regroupées en classes (**binning**) → « Light ».
- la croissance des arbres se fait **par feuille** plutôt que par profondeur → convergence rapide.
- **ignore les NaN** durant le split, et les alloue au mieux post-split.

Remarques sur la préparation des données :

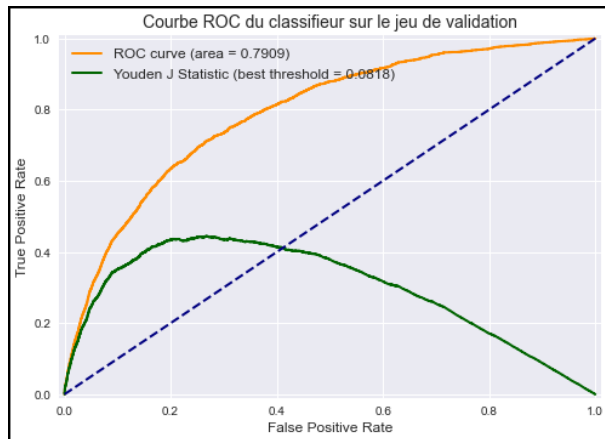
- données catégorielles : ordinal encoding (vs one-hot : gros gain sur le nb de variables à entraîner).
- données manquantes : non imputées (vs autres modèles de ML : gros avantage en terme de prédiction).

Note : pas de test comparatif possible avec des méthode de ML classiques de sklearn car les données entrantes contiennent des variables catégorielles encodées ordinalement et des valeurs NaN.

Développement d'un modèle prédictif – prise en main

Prise en main du couple { données, modèle LGBM } :

- optimisation a minima des paramètres LGBM.
- entraînement sur les données du « train set ».
- validation sur le « test set ».
- score ROC-AUC obtenu dans la compétition Kaggle : **0.78** (vs 0.81 pour le modèle gagnant).



⇒ prise en main des données et du modèle : validée

Gestion de l'imbalanced data



Approches testées pour rééquilibrer la classe cible (modules imbalanced et lgbm) :

Méthode de rééquilibrage	Variation AUC-ROC	Commentaire
Aucune (référence)	0 (réf)	-
Random over-sampling	+0.001	Temps de calcul en hausse
Random under-sampling	-0.005	Temps de calcul en baisse
SMOTE	-0.070	Imputation des NaN obligatoire fait chuter le score
Paramètre LGBM : is_unbalance=True	+0.002	Doc LGBM : "while enabling this should increase the overall performance metric of your model, it will also result in <u>poor estimates of the individual class probabilities</u> "
Paramètre LGBM : scale_pos_weight=3 (varié de 1 à 10 (optimum pour 3))	+0.002	

Tests réalisés comparativement avec LGBM (paramètres par défaut). Les datasets de référence peuvent varier d'un test à l'autre. Scoring effectué par une soumission du dataset « test » à Kaggle (pour garantir l'absence de data leak).

⇒ Pas d'amélioration significatives du score AUC-ROC.

⇒ Optimisation du modèle en conservant le déséquilibre dans le dataset.

Plan de la présentation

- 1) Problématique métier
- 2) Description des données
- 3) Nettoyage et feature engineering
- 4) Développement d'un modèle prédictif :
 - choix du modèle et prise en main
 - rééquilibrage de la classe cible
- 5) Optimisation du modèle en tenant compte d'une fonction de coût métier
- 6) Feature importances (globale et locale)
- 7) Conception d'une API de prédiction et d'un dashboard interactif
- 8) Conclusion

Définition d'une fonction de coût métier

Hypothèses pour les candidats au crédit :

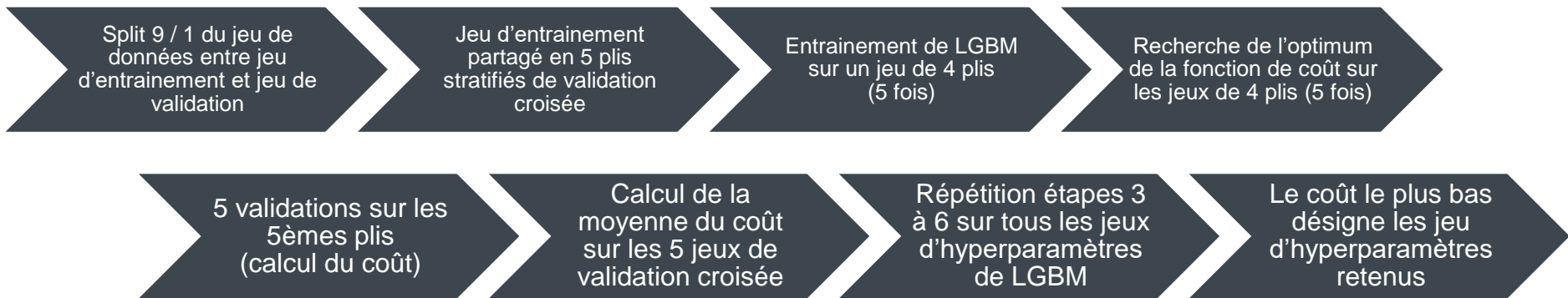
Objet	Coût par client	Classe
octroi de crédit à un client qui fait défaut	100	Faux négatif (FN)
octroi de crédit à un client qui ne fait pas défaut	-10	Vrai négatif (TN)
refus de crédit à un client qui aurait fait défaut	0	Vrai positif (TP)
refus de crédit à un client qui n'aurait pas fait défaut	0	Faux positif (FP)
frais généraux pour chaque client	1	-

Fonction de coût (rapportée à un client) :

$$Coût = \frac{100*FN - 10*TN + 1*(TP + TN + FP + FN)}{TP + TN + FP + FN}$$

Modèle LGBM optimisé sur fonction de coût métier - entraînement

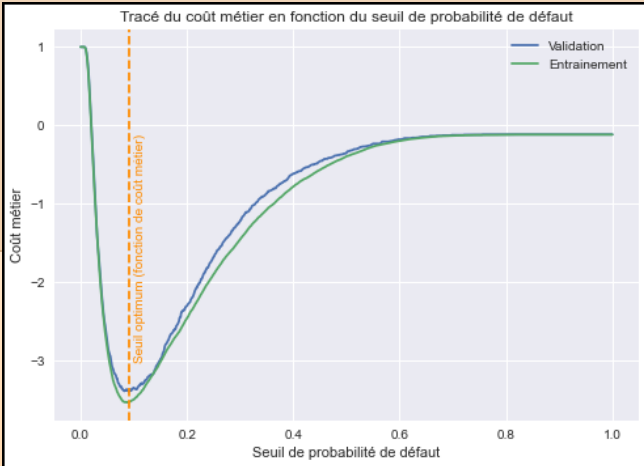
Schéma synoptique simplifié :



Hyperparamètre optimisé	Valeur optimum
num_leaves : max number of leaves in one tree	15
num_iterations : number of boosting iterations	200
min_data_in_leaf : minimal number of data in one leaf (can be used to deal with over-fitting)	40
learning_rate	0.05
seuil de probabilité de défaut	0.087

Modèle LGBM optimisé sur fonction de coût métier - résultat

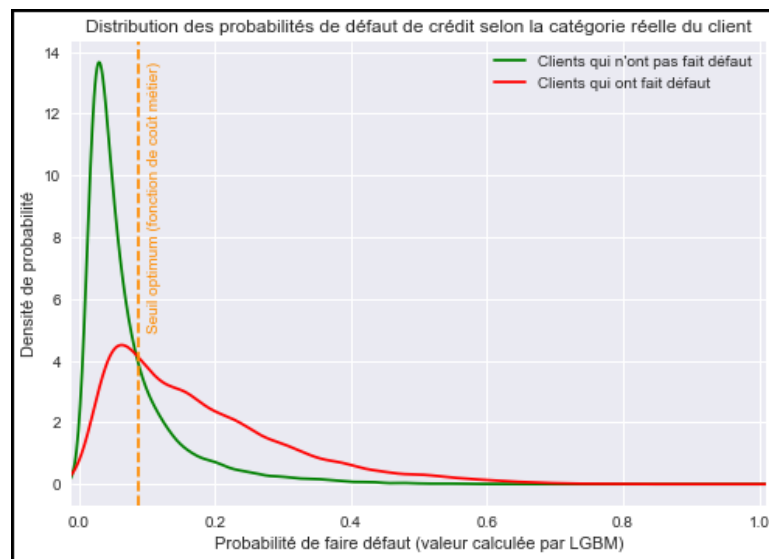
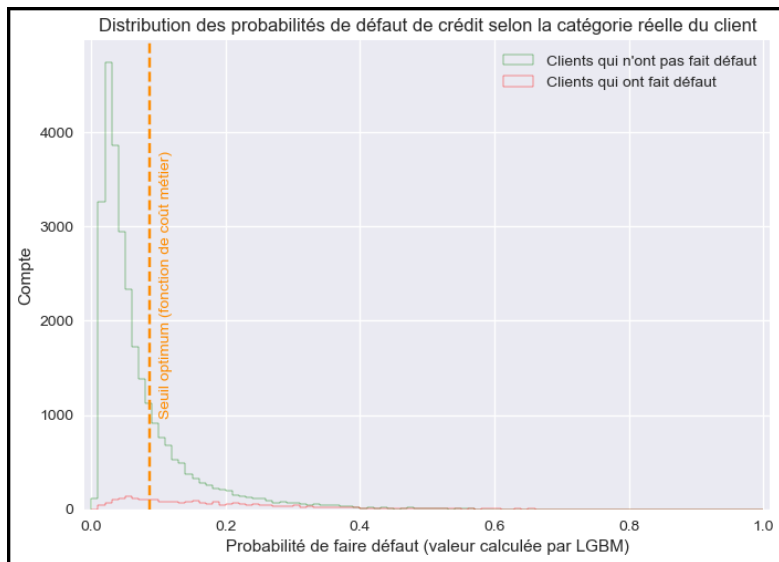
Coût métier (moyenne par client) en fonction du seuil de probabilité de défaut :

Données	Coût métier pour les hyperparamètres retenus	Matrice de confusion (1 = défaut)	Coût métier									
Jeu d'entrainement (277000 clients)		<table><tr><th></th><th>Predicted 0</th><th>Predicted 1</th></tr><tr><th>Actual 0</th><td>190725</td><td>63692</td></tr><tr><th>Actual 1</th><td>6499</td><td>15843</td></tr></table>		Predicted 0	Predicted 1	Actual 0	190725	63692	Actual 1	6499	15843	-3.54
	Predicted 0	Predicted 1										
Actual 0	190725	63692										
Actual 1	6499	15843										
Jeu de validation (31000 clients)		<table><tr><th></th><th>Predicted 0</th><th>Predicted 1</th></tr><tr><th>Actual 0</th><td>21174</td><td>7095</td></tr><tr><th>Actual 1</th><td>772</td><td>1711</td></tr></table>		Predicted 0	Predicted 1	Actual 0	21174	7095	Actual 1	772	1711	-3.38
	Predicted 0	Predicted 1										
Actual 0	21174	7095										
Actual 1	772	1711										

Le couple (modèle entraîné, seuil optimum), issu de l'optimisation, donne un coût métier proche en passant du jeu d'entraînement au jeu de validation.

Distributions de probabilité de défaut des clients

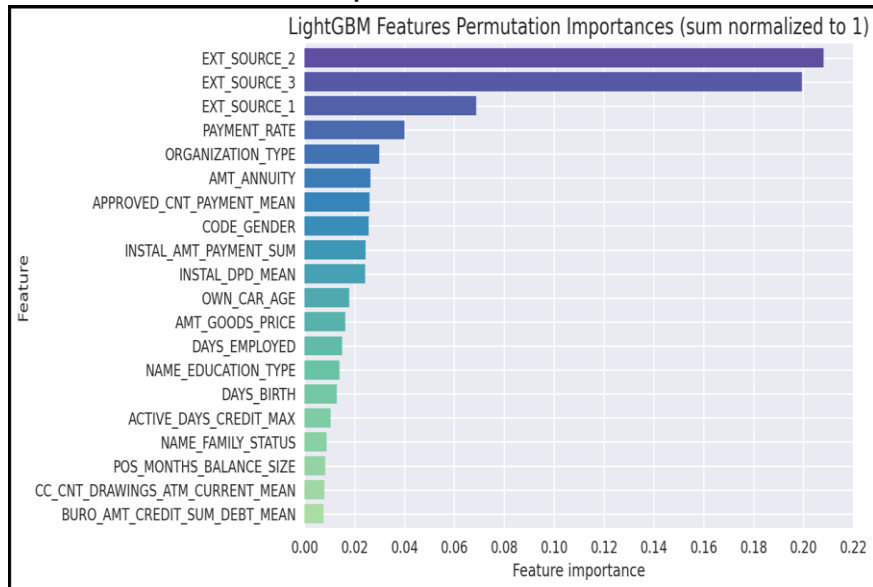
Calculées avec le modèle LGBM optimisé sur la fonction de coût métier et un jeu de validation :



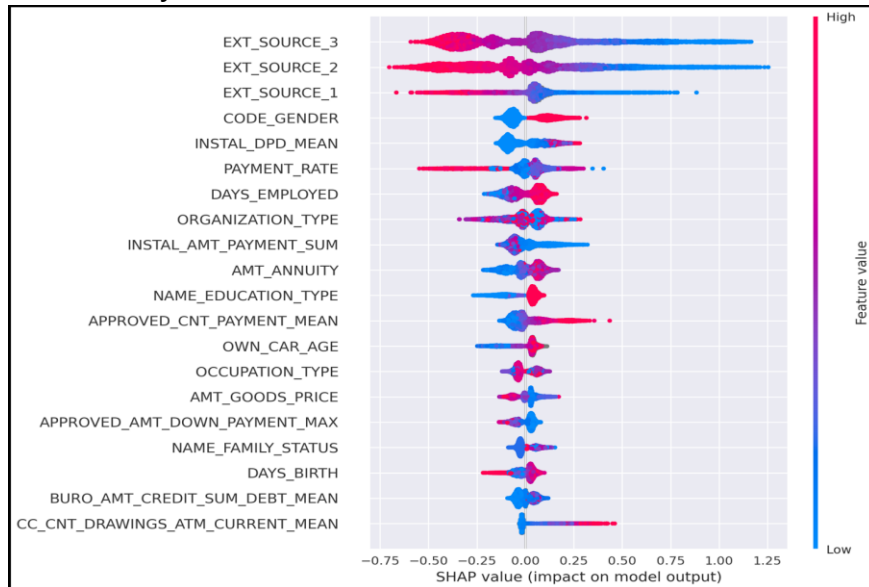
→ bonne séparation des clients selon la variable cible (défaut de crédit).

Feature importance globale

Par la méthode des permutations :



Par analyse de la distribution des valeurs SHAP :

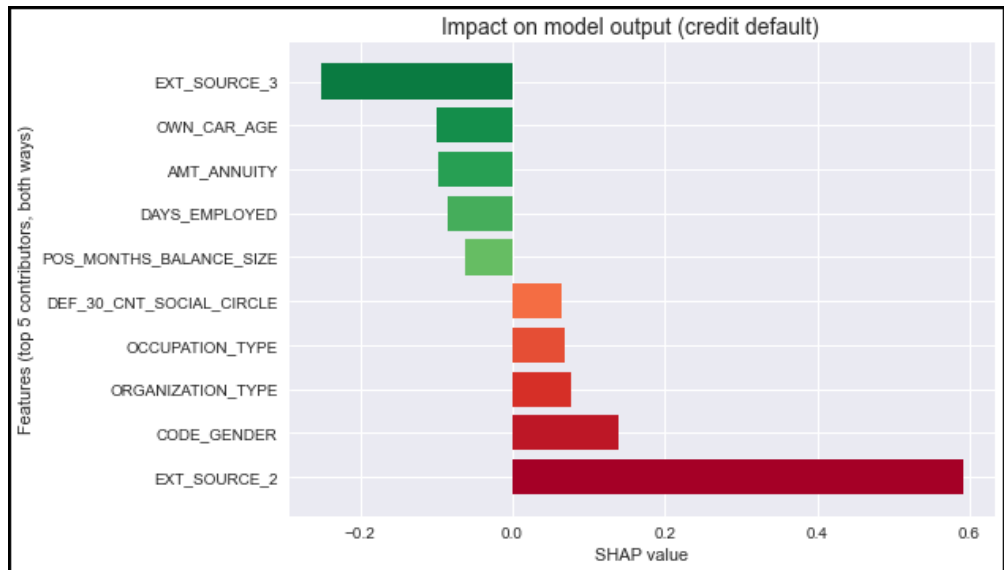


→ Cohérence des résultats entre les deux méthodes

→ Features les plus importantes : EXT_SOURCE_{1|2|3}

Feature importance locale par SHAP





Exemple (client 324806) :



- Indique aux chargés de clientèle quelles données du client ont un **impact fort pour l'attribution d'un crédit**.
- Les valeurs SHAP sont des logarithmes d'odds ratios ; elles sont donc **additives**.
- Analyse disponible pour tous les clients dans le **dashboard (calcul en temps réel)**.

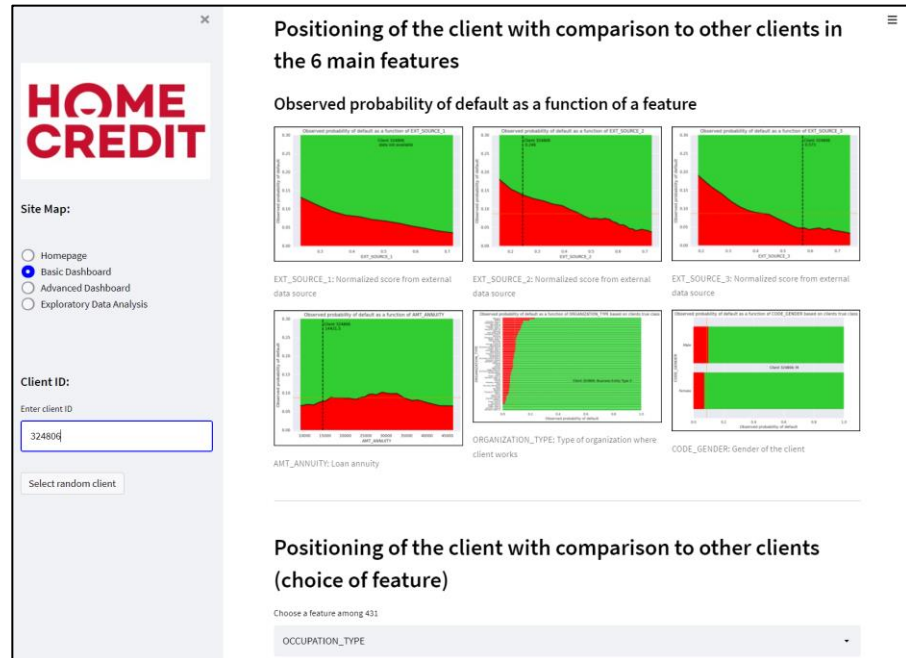
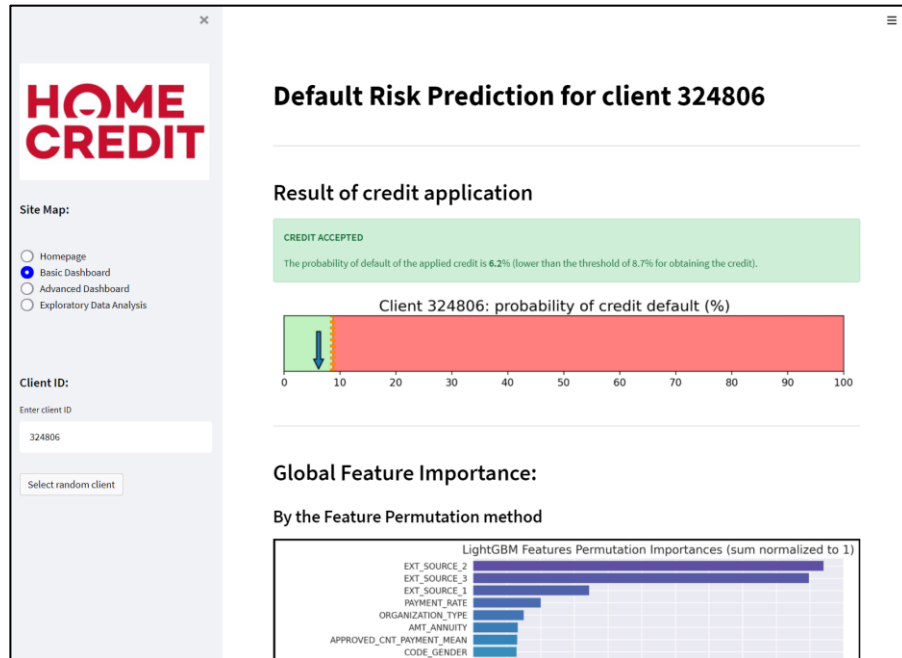
Déploiement web d'une API et d'un dashboard interactif

Resources utilisées :

Solution	Tâche	Liens
	Versionnage	https://github.com/JM-JO/Projet-7---api_ml https://github.com/JM-JO/Projet-7---dashboard-streamlit
	Framework de développement d'APIs RESTful	-
	Framework de développement de dashboards	-
	Hébergement de l'API Hébergement du dashboard	https://project7-api-ml.herokuapp.com/ https://project7-dashboard-streamlit.herokuapp.com/

Déploiement web d'une API et d'un dashboard interactif

Rendu du dashboard :

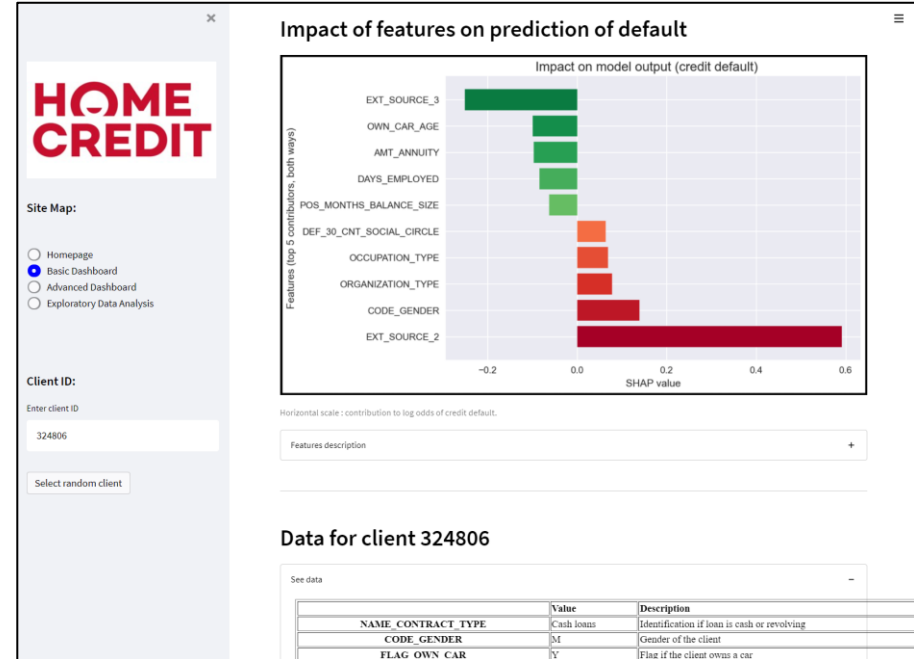
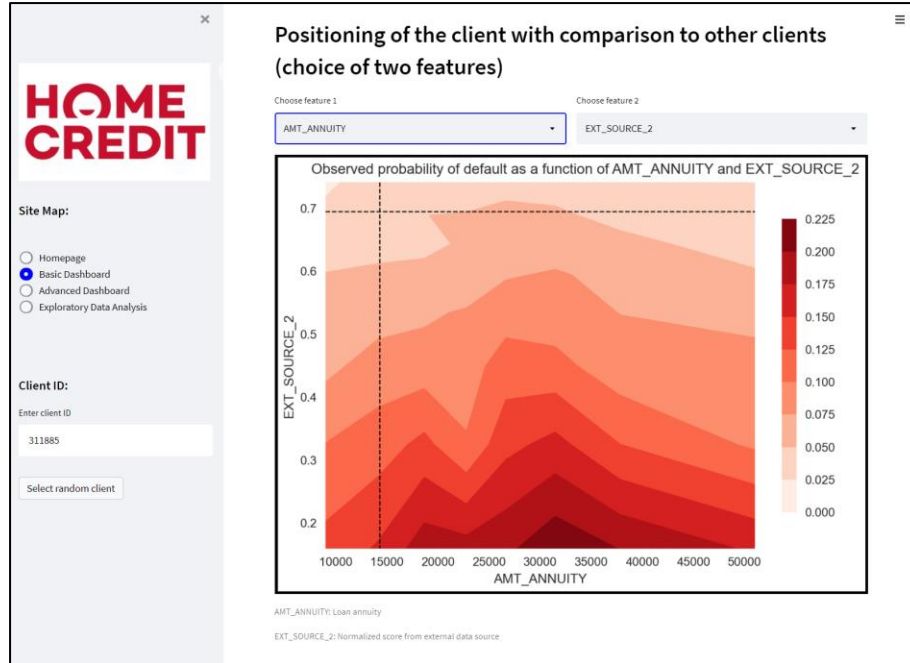


→ précalculs des graphes les plus courants (permet rendu plus rapide).

→ probabilité de défaut pour un client et ses SHAP values calculés en temps réel sur le serveur API.

Déploiement web d'une API et d'un dashboard interactif

Rendu du dashboard (suite) :



[Lien vers le dashboard](#)

Conclusion

Mise au point d'un modèle de classification :

- basé sur **toutes les tables** du dataset Kaggle.
- utilisant **LightGBM**.
- avec une bonne performance de base : **AUC-ROC = 0.78**.

Spécialisation de l'outil en créant un scoring basé sur une **fonction de coût métier**.

Analyse des prédictions par **SHAP** pour transparence / simplicité d'analyse du scoring par le client.

Déploiement dans le cloud d'un dashboard interactif, et d'une API de prédiction (score et SHAP).

Perspectives :

- Pousser l'optimisation du modèle :
 - sur un plus grand nombre d'hyperparamètres (y compris rééquilibrage de classe cible).
 - avec des rééquilibrages de dataset.
 - avec une optimisation bayésienne (hyperopt).
- Accélérer l'affichage du dashboard (utiliser un framework avec meilleure gestion du cache).