Scoring de crédits

OpenClassrooms Data Science - Projet 7

Cyril Baudrillart

Rappel des objectifs

Mettre au point un outil de scoring crédit calculant la probabilité de remboursement et classifiant la demande de crédit en “accordé” ou “refusé”

- preprocessing et analyse des données
- sélection d’un algorithme de classification
- être capable d’expliquer la décision ou le refus d’octroi de crédit
- mise au point d’une API
- développement d’un dashboard intéractif
- hébergement du code sur github

Présentation du jeu de données

Jeu de données disponible ici: https://www.kaggle.com/c/home-credit-default-risk/data

- 8 tables disponibles avec volume de données important
- Données quantitatives & catégorielles
- Identifiants permettant de faire des liens entre les tables

Nom fichier csv	identifiants	Taille (li x col)	Description
application_train	SK_ID_CURR	307,511 x 122	Train set (crédits x features + target)
application_test	SK_ID_CURR	48,744 x 121	Test set sans target
bureau	SK_ID_CURR / SK_ID_BUREAU	1,716,428 x 17	Crédits dans d'autres établissements
bureau_balance	SK_ID_BUREAU	27,299,925 x 3	Données mensuelles anciens crédits

Présentation du jeu de données (suite)

Nom fichier csv	identifiants	Taille (li x col)	Description
credit card balance	SK_ID_PREV / SK_ID_CURR	3,840,312 x 23	Relevés anciennes cartes de crédits
HomeCredit_columns_description	-	219 x 4	Description des features et de leur localisation dans les tables
installments_payments	SK_ID_PREV / SK_ID_CURR	13,605,401 x 8	Historique des remboursements des emprunts précédents
POS_CASH_balance	SK_ID_PREV / SK_ID_CURR	10,001,358 x 8	Données mensuelles anciens prêts à la consommation clients.

Retraitement des données

Il était recommandé de s'inspirer d'un kernel Kaggle afin d'accélérer les opérations de preprocessing.

https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script

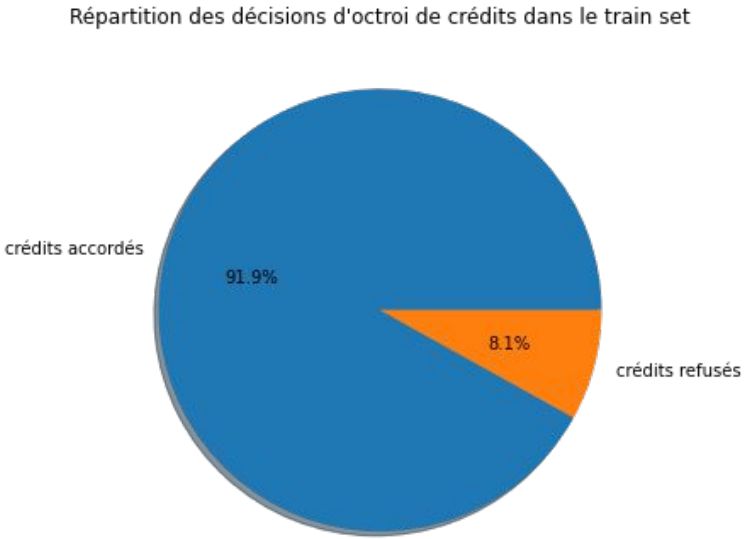
- gestion des problèmes de mémoire dans un environnement Jupyter Notebook
- création d'un script python inspiré du kernel et exécution dans une console
- enregistrement des résultats sous forme de fichier csv de taille supérieure à 1 GO (fichier data/final_credit.csv)
- création d'un train set (data_models/credit_train.csv)
- et d'un test set (/data_models/credit_test.csv)

Nettoyage & sélection des features

- On utilise le **train set** créé qui contient 307,511 lignes (crédits) et 797 colonnes (features)
- suppression des variables avec plus de 50% de valeurs manquantes
- utilisation de la méthode des filtres:
  - suppression des features à variance faible (<2%)
  - suppression des features très corrélées (>0.5)
  - suppression des lignes ayant plus de 30% de valeurs manquantes (5% des données)

Répartition des classes

- Les classes sont fortement déséquilibrées
- La classe 0 (crédit accordé) est largement majoritaire
- La classe 1 (crédit refusé est minoritaire)
- Création de jeux d'entraînement et de validation avec stratification afin d’obtenir des échantillons ayant le même pourcentage de données des classes 0 et 1 que dans le dataset d’origine.



Sélection des features

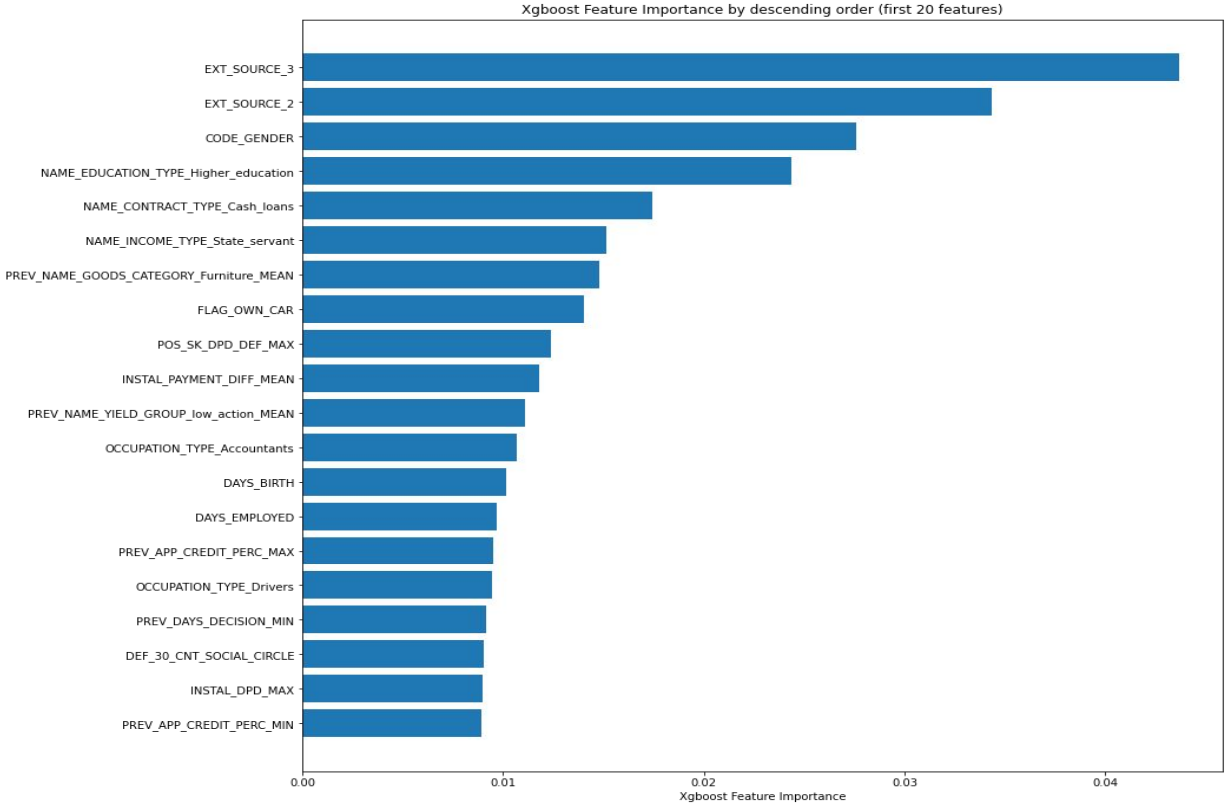
Problématique: sélectionner les features les plus pertinentes en présence de nombreuses valeurs manquantes

- solution 1: imputer les valeurs manquantes puis effectuer la sélection. C'est prendre le risque de modifier les relations entre features et target.
- solution 2: utiliser une méthode tolérant les valeurs manquantes. C'est le cas de certaines méthodes d'arbres comme XGBoost.

Choix de la solution 2 avec mise au point d'un modèle XGBoost utilisant la fonction RandomUnderSampler de la librairie imblearn pour rééquilibrer les classes.

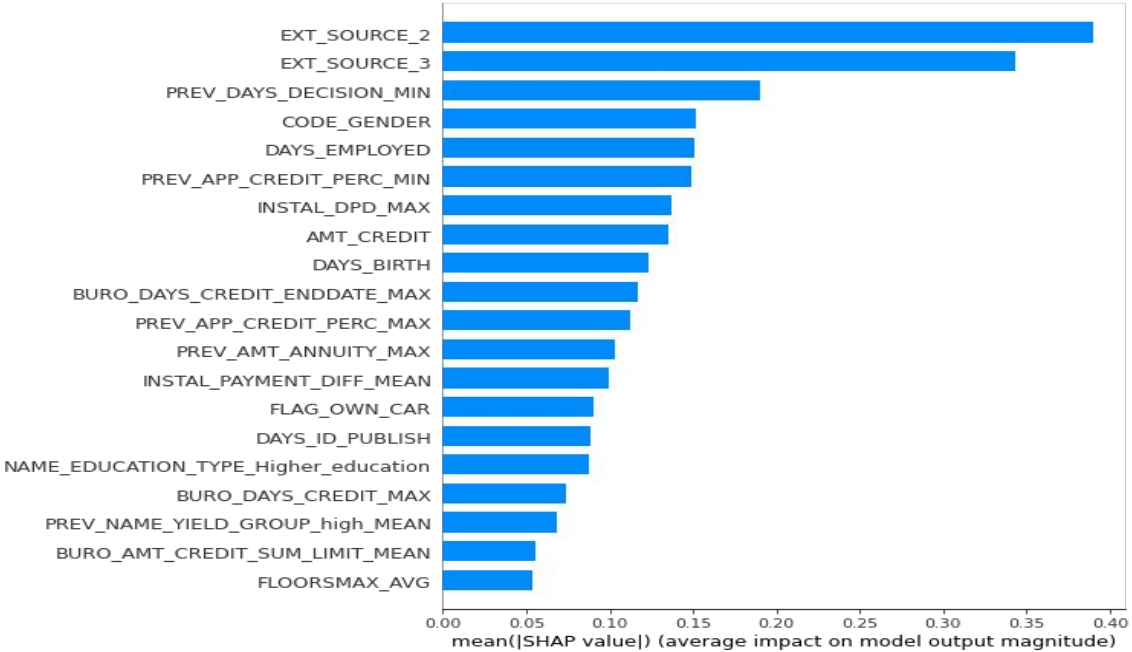
Performance obtenue: Recall sur test set à 0.68, score F1 0.26 et ROC AUC Score à 0.68.

Features importantes modèle XGBoost



Analyse Shapley modèle XGBoost

Identification des variables importantes avec les valeurs de Shapley



Sélection finale des features

Les différentes méthodes de sélection des features donnent des résultats hétérogènes même si certaines features sont communes aux 2 méthodes.

Une sélection finale doit également être effectuée en fonction des besoins métier une fois la présélection des variables effectuée.

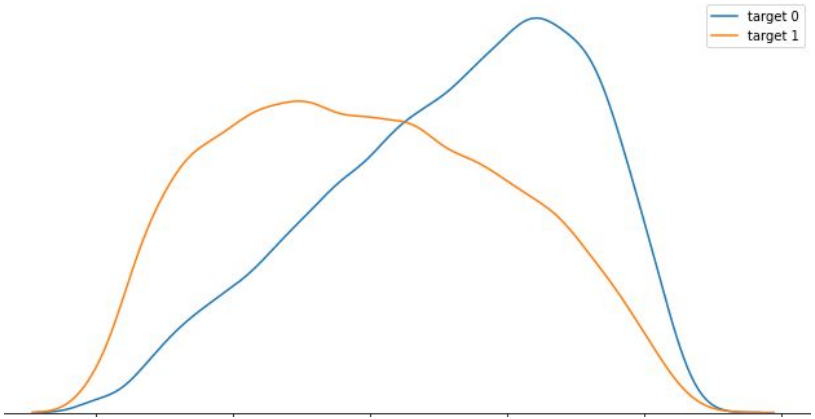
La liste finale des variables des variables sélectionnées est disponible à la page suivante

'EXT_SOURCE_3' & 'EXT_SOURCE_2'	scores normalisé provenant de sources extérieures
'PREV_DAYS_DECISION_MIN'	durée minimum séparant le crédit actuel du précédent
'CODE_GENDER'	genre (0 pour femme, 1 pour homme)
'DAYS_EMPLOYED'	ancienneté de l'emploi en jours
'PREV_APP_CREDIT_PERC_MIN'	ratio minimum montant du crédit demandé / montant final du crédit
'INSTAL_DPD_MAX'	plus gros retard de paiements sur les précédents crédits
'AMT_CREDIT'	montant du crédit précédent
'DAYS_BIRTH'	age de l'emprunteur en nombre de jours
'FLAG_OWN_CAR'	possession d'une voiture (0 non / 1 oui)
'NAME_EDUCATION_TYPE_Higher education'	niveau d'éducation supérieur (0 non / 1 oui)

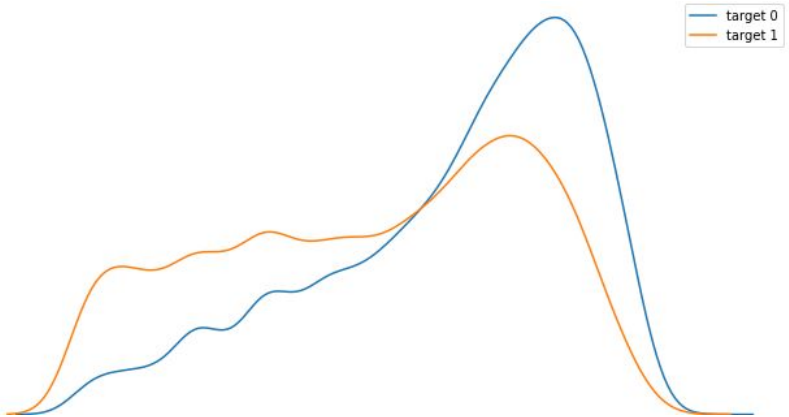
Imputation des variables manquantes

La distribution des variables est assez différente entre les classes. Voir Notebook 6. L'imputation des variables manquantes se fait en prenant la valeur la plus fréquente observée dans chaque classe.

Distribution of EXT_SOURCE_3 by Target Value



Distribution of EXT_SOURCE_2 by Target Value



Sélection du modèle de scoring

Problème de classification binaire supervisée avec des classes déséquilibrées:

- classe 0: crédit accordé
- classe 1: crédit refusé

Un rééquilibrage préalable des classes a été effectué. Compte tenu de la grande taille du jeu de données et des objectifs, les modèles suivants ont été retenus.

- Gaussian Bayes
- Régression logistique Ridge
- Arbres de décision
- XGBoost
- Random Forest
- LightGBM Classifier

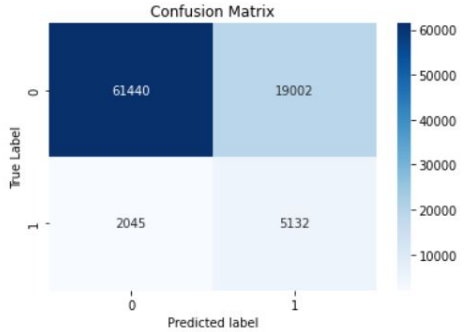
Les 2 premières méthodes ont des résultats corrects (Test recall 0.72 & 0.69; Test AUC 0.78 & 0.80; score F1 0.71 & 0.72).

XGBoost améliore la performance (test recall 0.77, test AUC 0.85, test F1 0.76) avec des temps de calcul plus longs mais corrects comparés à l'algorithme Random Forest.

Short list des modèles de scoring

Régression logistique

- test de différentes méthodes d'équilibrage des classes: RandomUnderSampler, RandomOverSampler, SMOTE.
- optimisation des paramètres avec GridSearchCV
  - l1 ratio: 0.01
  - penalty: l2
  - solver: newton-cg
  - Standardscaler



	precision	recall	f1-score	support
0.0	0.97	0.76	0.85	80442
1.0	0.21	0.72	0.33	7177
accuracy			0.76	87619
macro avg	0.59	0.74	0.59	87619
weighted avg	0.91	0.76	0.81	87619

ROC AUC Score: 0.74

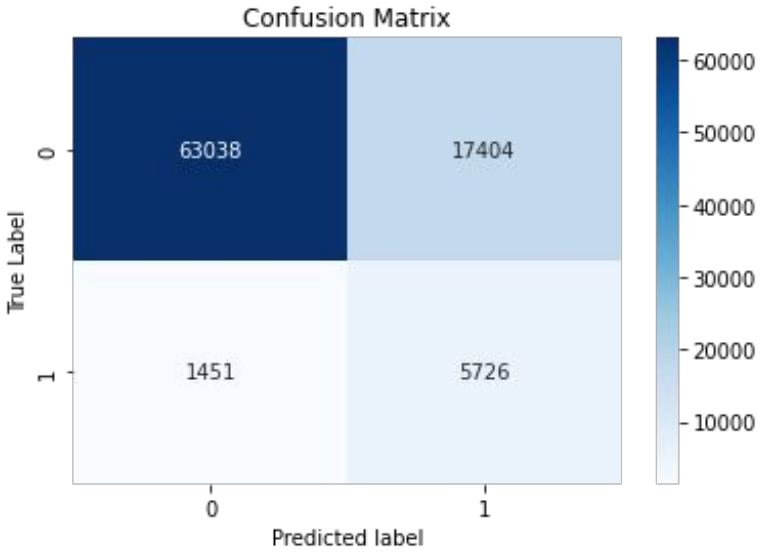
Short list des modèles de scoring

XGBoost

- Rééquilibrage des classes avec RandomUnderSampler.
- Début d'optimisation des paramètres avec RandomizedGridSearchCV
  - model__max_depth: 2
  - model__min_child_weight: 3

	precision	recall	f1-score	support
0.0	0.98	0.78	0.87	80442
1.0	0.25	0.80	0.38	7177
accuracy			0.78	87619
macro avg	0.61	0.79	0.62	87619
weighted avg	0.92	0.78	0.83	87619

ROC AUC Score: 0.79
CPU times: user 744 ms, sys: 188 ms, total: 933 ms
Wall time: 381 ms

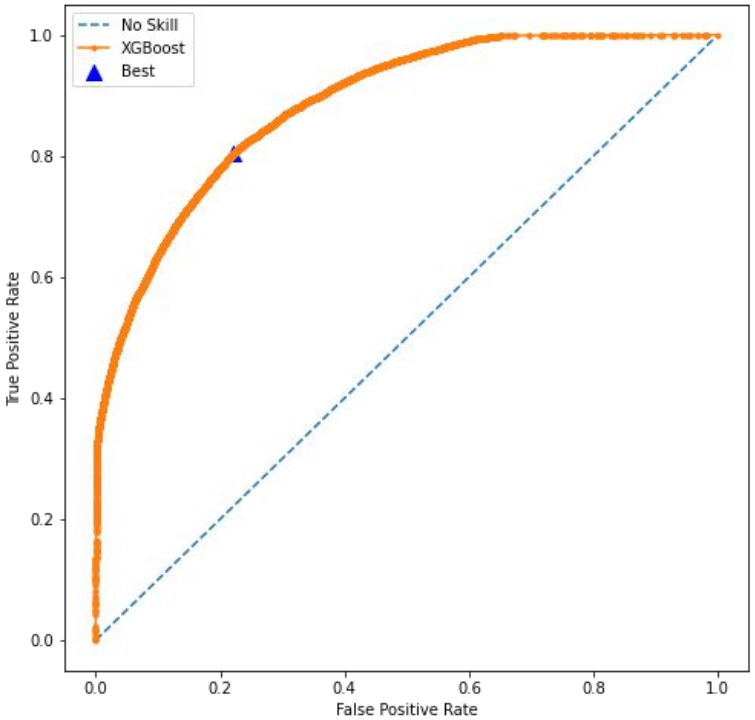


Recherche du seuil optimal XGBoost

Optimisation du seuil de probabilité d'appartenance à la classe 1

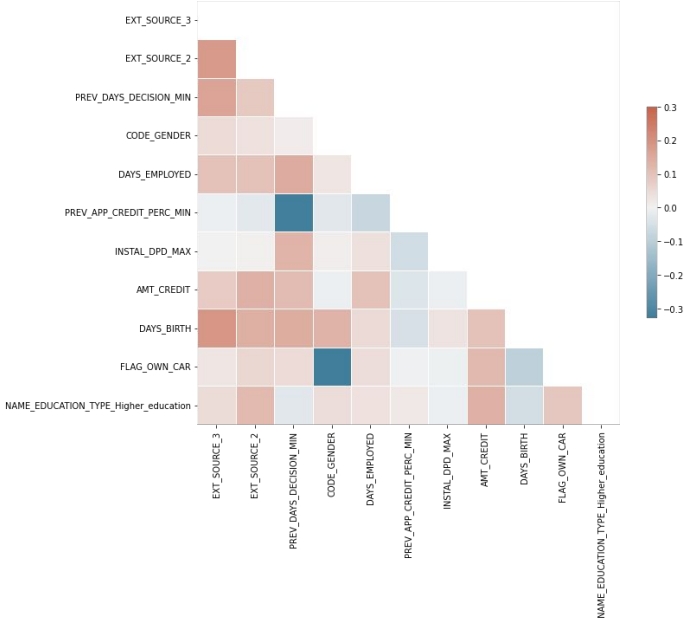
Permet d'obtenir le meilleur équilibre entre le taux de vrai positifs et de faux positifs.

Seuil optimal à 49.55% soit très proche du seuil standard à 50%.



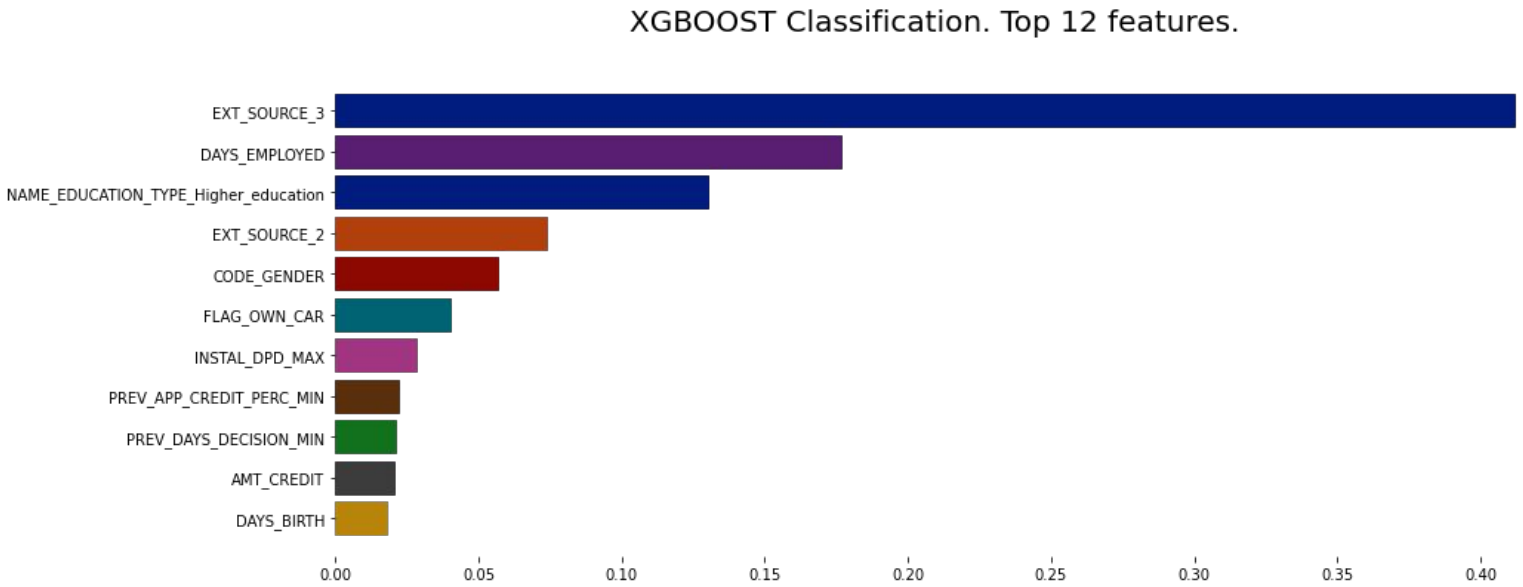
Corrélation des variables sélectionnées

Les variables sélectionnées sont peu corrélées entre elles



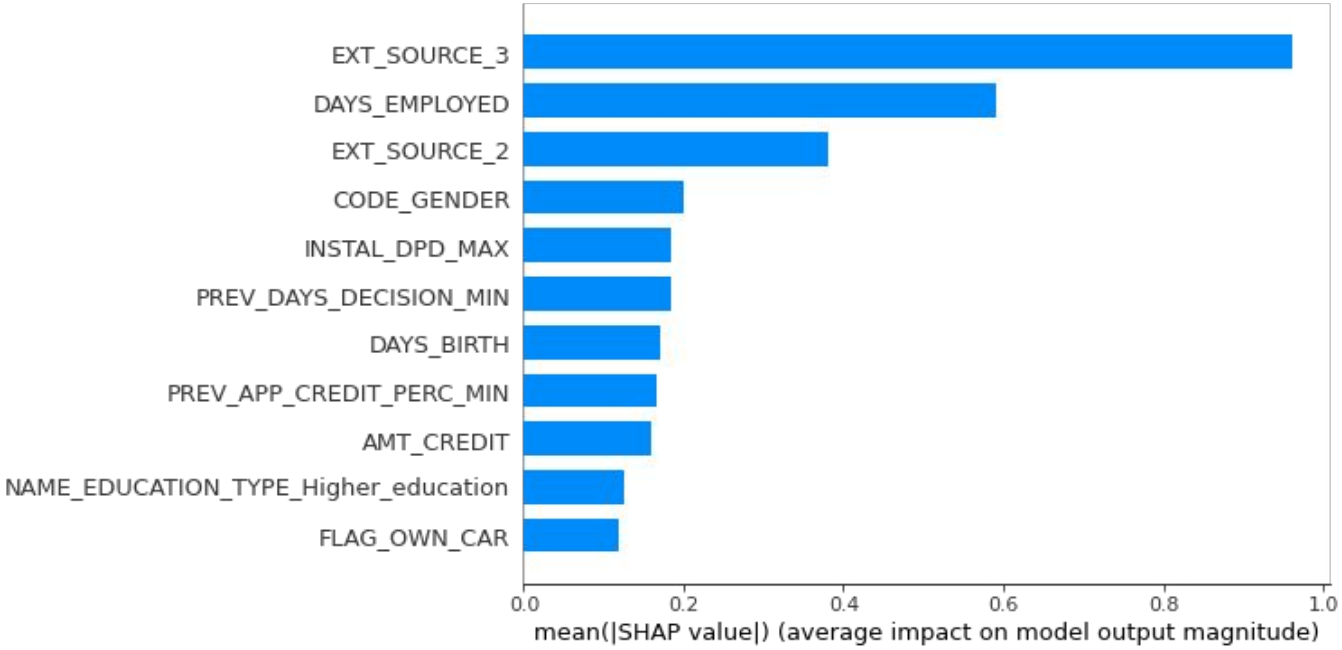
Interprétabilité XGBoost

Importance des features dans le modèle XGBoost sélectionné .



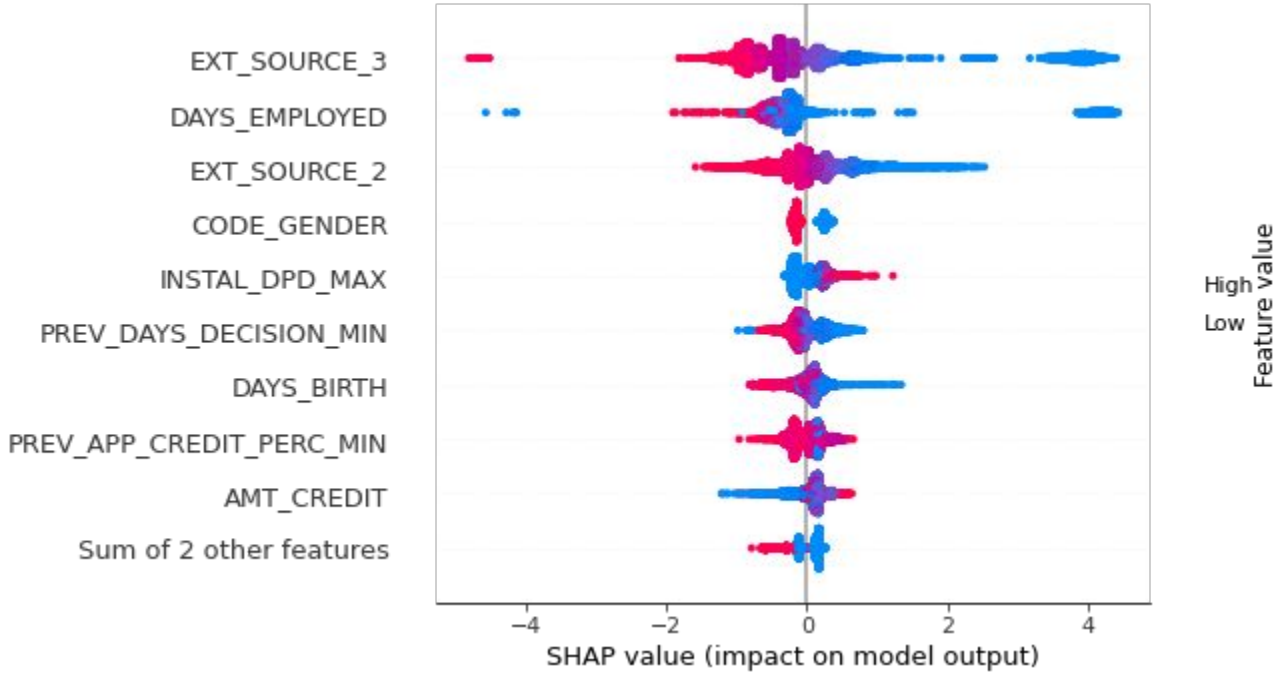
Corrélation des variables sélectionnées

On voit que les valeurs de Shapley diffèrent des pondérations du modèle XGBoost



Impact globale des valeurs de Shapley

rouge -> augmente la probabilité d'un refus de crédit; **bleu** -> hausse probabilité d'acceptation



Mise au point d’une API

Nous avons sélectionné FastAPI pour cette tâche.

L’application a été déployée sur Heroku. Elle est disponible à l’adresse suivante:

https://test-cyril-fastapi.herokuapp.com

Le code de l’API est visible sur Github à l’adresse suivante:

https://github.com/cyrbaufr/fastapi

Mise au point d’un dashboard

Nous avons sélectionné Streamlit pour cette tâche.

L’application a également été déployée sur Heroku.

La version déployée est disponible à l’adresse suivante:

https://cyril-credit-scoring.herokuapp.com/

Le code du dashboard est disponible à l’adresse suivante:

https://github.com/cyrbaufr/credit_heroku

Mise à disposition du code du projet

L’intégralité du code de ce projet est disponible sur un repo Github à l’adresse suivante:

https://github.com/cyrbaufr/projet7_oc