Note Méthodologique

Application de détection de risque de faillite bancaire

Mission de Data Science pour la société Prêt à dépenser qui propose des crédits à la consommation pour des personnes ayant peu ou pas d'historique de prêt

Naomi Guarneri

14/07/2023

Parcours Data Scientist - OpenClassrooms

Liens GITHUB :

Notebooks https://github.com/naomigua/OC7-Notebooks

API https://github.com/naomigua/OC7-Api

Dashboard https://github.com/naomigua/OC7-Dashboard

INTRODUCTION DE LA MISSION DE DATA SCIENCE

Mon rôle : Je suis Data Scientist au sein de la société Prêt à dépenser.

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

Le contexte : L'entreprise souhaite mettre en œuvre un outil de "scoring crédit" pour calculer la probabilité qu'un client rembourse son crédit, puis classifier la demande en accord ou refus. Elle souhaite donc développer un **algorithme de classification** en s'appuyant sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.). De plus, les chargés de relation client ont fait remonter le fait que les clients sont de plus en plus demandeurs de **transparence** vis-à-vis des décisions d'octroi de crédit. Cette demande de transparence des clients va tout à fait dans le sens des valeurs que l'entreprise veut incarner.

La solution : **Prêt à dépenser** décide donc de **développer un dashboard interactif** pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d'octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

Les données à disposition : des données anonymisées sur des clients avec :

- TARGET : 1 difficultés de paiement, ou 0 absence de difficultés (ce que l'on cherche à prédire)
- plein de données sur les clients : genre, si le client a une voiture, s'il possède de l'immobilier, son nombre d'enfants, ses revenus, sa source de revenus, son type d'habitation, ses études, qui l'a accompagné pour la demande d'emprunt...

La mission :

- Construire un modèle de scoring qui donnera une prédiction sur la probabilité de faillite d'un client de façon automatique.
- Construire un dashboard interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d'améliorer la connaissance client des chargés de relation client.
- Mettre en production le modèle de scoring de prédiction à l'aide d'une API, ainsi que le dashboard interactif qui appelle l'API pour les prédictions.

MÉTHODOLOGIE D'ENTRAÎNEMENT DU MODÈLE

Il s'agira de :

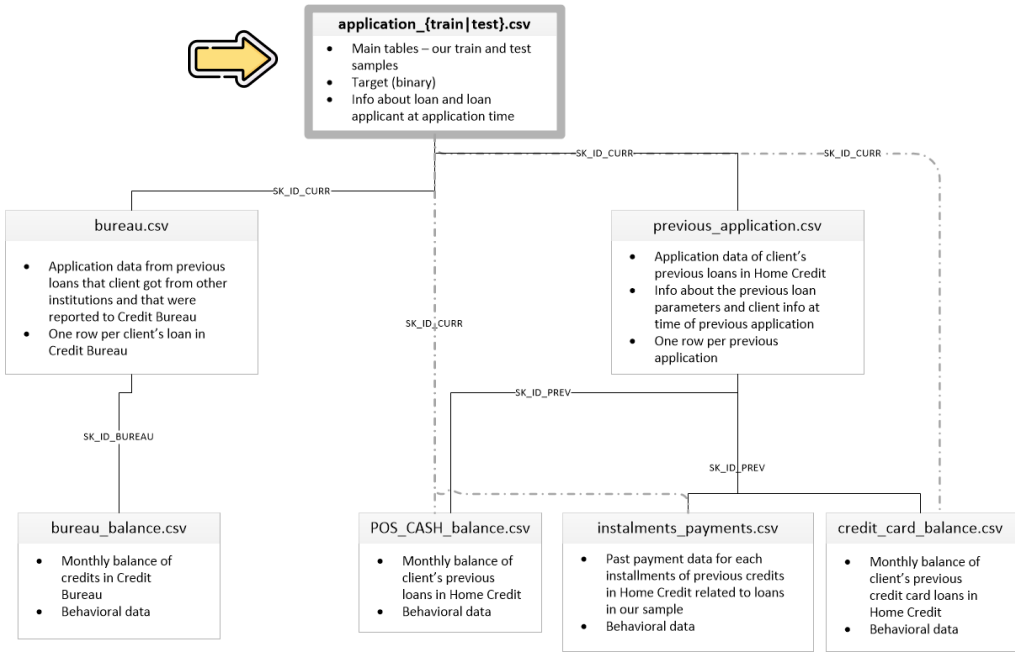
- importer les librairies ainsi que les données
- explorer les données
- créer différents features engineering
- Séparation en Train / Test et prétraitement additionnels
- tester différents modèles de ML pour modéliser, rechercher les hyper params et les comparer

Configuration de l'environnement

Les bibliothèques nécessaires ont été importées et l'environnement a été configuré

pour l'analyse des données. Cela comprenait l'importation de bibliothèques pour la manipulation des données, le traçage, le prétraitement, la gestion des avertissements et le suivi des résultats.

Chargement des données

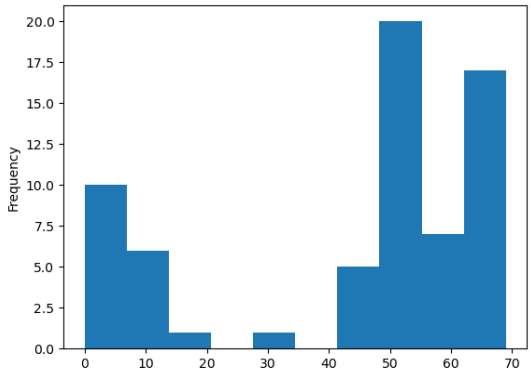


Dans ce projet nous allons utiliser seulement le fichier application_train_test.csv. Nous allons par la suite le diviser en train et test. Mais dans un premier temps, place à l'exploration des données.

Exploration des données & début du Feature Engineering

Répartition des classes pour la faillite bancaire : 91.93% sans problème de remboursement 🙂 et 8.07% avec des soucis de remboursement 😡.
Données manquantes : 122 colonnes dont 67 avec des données manquantes.

Tableau de répartition des fréquences du pourcentage de données manquantes par colonne

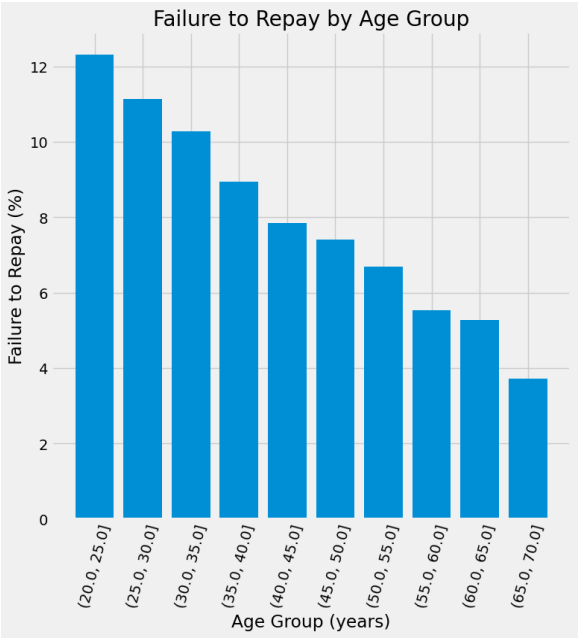


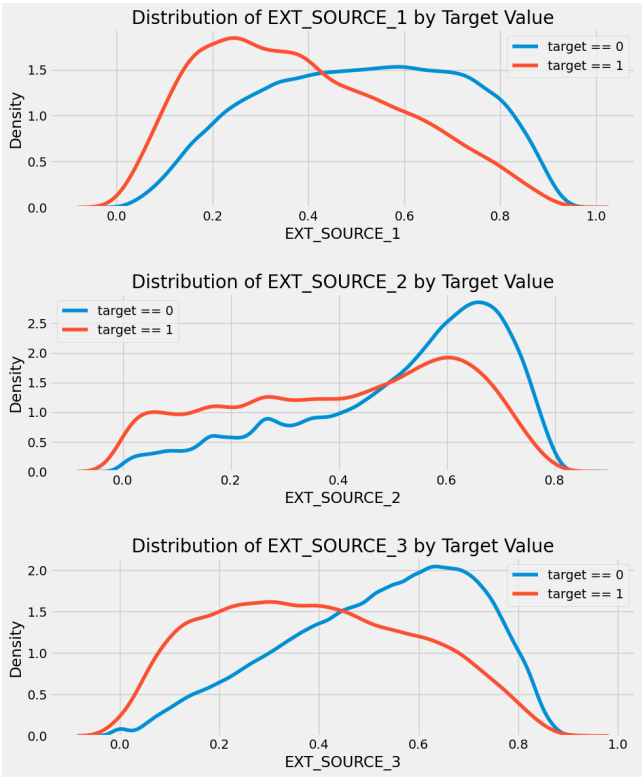
Il y a beaucoup de données manquantes, nous ne pouvons ni ignorer ces données ni les lignes concernées nous allons donc les conserver.
Les différents types de variables : 65 float64, 41 int64 et 16 object. Pour pouvoir entraîner un modèle il faut pouvoir transformer les variables existantes nous allons donc passer à l'encodage.
Encodage des variables catégorielles : 2 méthodes différentes selon le nombre de catégories. Effectivement il y a des variables avec 2 catégories et on va même jusqu'à 58 catégories pour 'ORGANIZATION_TYPE'

- Label Encoder pour les variables catégorielles à 2 catégories
- One Hot Encoding pour les variables catégorielles à plus de 2 catégories pour que le modèle ne mésinterprète pas les poids associés par un label encoder arbitraire.

Traitement des anomalies : 18% des individus comportent un nombre très étrange de 'DAYS_EMPLOYED' correspondant à près de 1000 ans. Nous allons donc remplacer ce nombre par NaN et créer une colonne pour identifier les individus avec l'anomalie 'DAYS_EMPLOYED_ANOM'.

Corrélations avec la TARGET :





La variable EXT_SOURCE_3 est celle qui représente le plus une différence entre les target 0 et 1, elle sera donc utile pour un modèle de machine learning.

Suite du Feature Engineering

Nous allons tester plusieurs feature engineering que nous allons tester par la suite avec les différents types de modélisation de Machine Learning.

Données de Base : les données telles que transformées jusqu'à maintenant

Polynomial features : Données de Base + Polynomial Features avec ['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DAYS_BIRTH']

Les caractéristiques polynomiales dans un modèle de machine learning permettent de capturer des relations non linéaires entre les caractéristiques et la cible, offrant ainsi une meilleure précision du modèle.

Domain Knowledge Features : Données de Base +

`CREDIT_INCOME_PERCENT`: the percentage of the credit amount relative to a client's income

`ANNUITY_INCOME_PERCENT`: the percentage of the loan annuity relative to a client's income

`CREDIT_TERM`: the length of the payment in months (since the annuity is the monthly amount due

`DAYS_EMPLOYED_PERCENT`: the percentage of the days employed relative to the client's age

Modélisation

Séparation des données en train et test via train_test_split de sklearn avec test_size=20%.
Prétraitement additionnels avec SimpleImputer (strategy='median') + MinMaxScaler(0,1)
Création d'une fonction pour rechercher les hyperparamètres et afficher les résultats (**recall, F1–Score, AUC-ROC**)

- *Recall : proportion de vrais positifs correctement identifiés parmis tous les cas réels positifs*
- *F1-Score : moyenne harmonique de la précision et du recall, offrant un équilibre entre les deux*
- *AUC-ROC : Aire sous la courbe ROC, qui mesure la capacité d'un modèle à distinguer entre les classes à tous les seuils de classification.*

Les différents modèles testés : tous les modèles sont testés en utilisant une Cross-Validation et GridSearchCV afin d'utiliser l'ensemble des données d'entraînement pour trouver les meilleurs hyperparamètres.

- **DummyClassifier**: approche naïve qui va servir de référence, ici elle va prédire la most frequent class donc 0 soit l'absence de faillite bancaire.
- **Régression Logistique** : Un algorithme de classification qui utilise une fonction logistique pour prédire la probabilité qu'une observation appartienne à une catégorie particulière.
- **Arbre de décision** : prédit en divisant les données en sous-ensembles selon les caractéristiques qui maximisent l'information gagnée à chaque division.
- **Random Forests** : algorithme d'apprentissage ensembliste qui crée un ensemble de modèles d'arbres de décision formés sur des sous-ensembles aléatoires des données, puis fait des prédictions en agrégeant les prédictions de chaque arbre
- **LightGBM& CatBoost** : algorithmes de boosting de gradient qui construisent des modèles en ajoutant de manière itérative des prédicteurs pour corriger les erreurs des modèles précédents. Ils sont conçus pour être rapides et précis, même sur de grands ensembles de données.

LE TRAITEMENT DU DÉSÉQUILIBRE DES CLASSES

Comme dit plus haut lors de l'exploration, il existe un déséquilibre des classes puisque seulement 8% des individus ont eu des problèmes pour le remboursement de leur crédit. J'ai donc dans un premier temps entraîné tous mes modèles et récupérer mes scores sur mes données post Feature engineering sans traiter le déséquilibre et j'obtiens les résultats suivants :

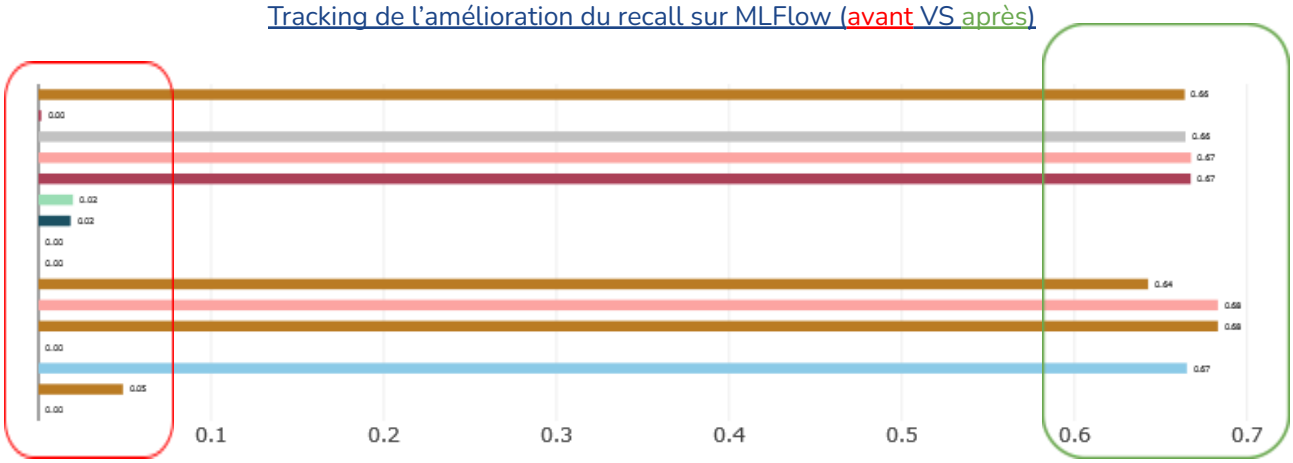
	Données	Modèle	recall	F1-score	AUC ROC
0	data de base	DummyClassifier(strategy='most_frequent')	0.0	0.0	0.5
1	data de base	LogisticRegression(C=0.01, solver='saga')	0.00162	0.00322	0.7407
2	data de base	DecisionTreeClassifier(max_depth=30, max_featu...	0.14387	0.14406	0.52321
3	data de base	RandomForestClassifier(max_depth=5, max_featur...	0.0	0.0	0.72387
4	data de base	LGBMClassifier(max_depth=5, reg_alpha=0.316227...	0.01859	0.03566	0.73289
5	data de base	<catboost.core.CatBoostClassifier object at 0x...	0.02	0.03825	0.73465
6	poly_features	LogisticRegression(C=0.01, solver='saga')	0.0	0.0	0.71961
7	data_domain	LogisticRegression(C=0.01, solver='saga')	0.0	0.0	0.72837

A ce stade je décide d'abandonner le modèle catboost car même sur un très petit échantillon il est beaucoup trop lent comparé aux autres.
Ensuite dans un second temps j'ai ajouté pour chacun de mes modèles le paramètre 'class_weight' = 'balanced' pour pallier le problème et voici les nouveaux résultats obtenus :

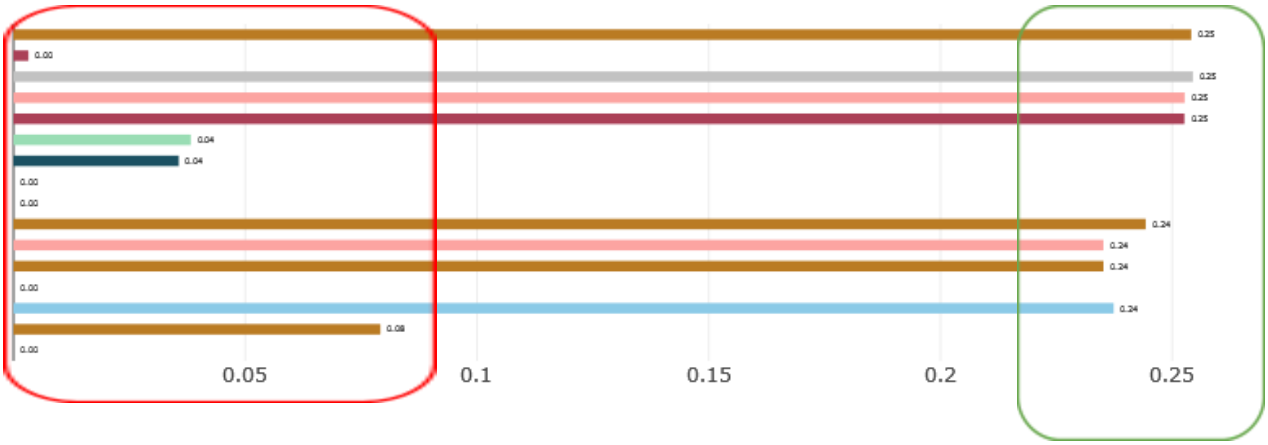
	Données	Modèle	recall	F1-score	AUC ROC
0	data de base	DummyClassifier(strategy='most_frequent')	0.0	0.0	0.5
1	data de base	LogisticRegression(C=0.01, class_weight='balan...	0.66397	0.25409	0.74246
2	data de base	DecisionTreeClassifier(class_weight='balanced'...	0.62114	0.2148	0.65153
3	data de base	RandomForestClassifier(class_weight='balanced'...	0.64276	0.24428	0.72128
4	data de base	LGBMClassifier(class_weight='balanced', max_de...	0.66438	0.25446	0.73779
5	poly_features	LogisticRegression(C=0.01, class_weight='balan...	0.68317	0.23515	0.72018
6	data_domain	LogisticRegression(C=0.01, class_weight='balan...	0.66741	0.25263	0.73689

On observe bien que toutes les métriques ont augmenté, cela est flagrant avec le recall et le F1-Score.

Tracking de l'amélioration du recall sur MLFlow (avant VS après)



Tracking de l'amélioration du F1-Score sur MLFlow (avant VS après)



LA FONCTION COÛT MÉTIER, OPTIMISATION

La fonction de coût métier

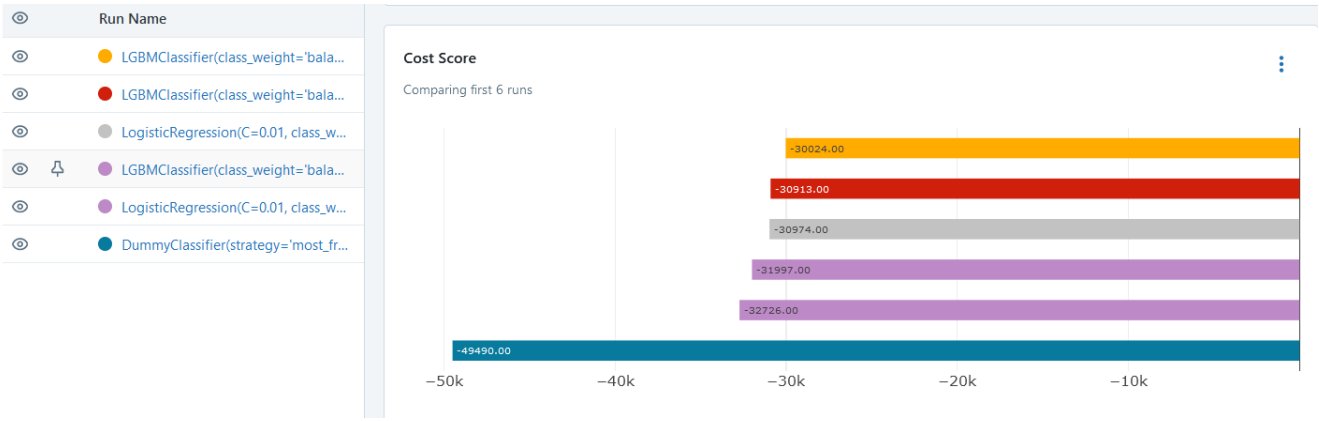
Afin d'obtenir un score au plus proche possible de la problématique métier j'ai mis en place mon propre scorer pour évaluer mes différents modèles. Je me suis basée sur la consigne suivante : le coût d'un faux négatif (mauvais client prédit bon client, donc crédit accordé et perte en capital) est 10 fois supérieur au coût d'un faux positif (bon client prédit mauvais donc refus crédit et manque à gagner en marge).

```python
def cost_function(y_true, y_pred):
    tp = np.sum((y_true == 0) & (y_pred == 0))
    tn = np.sum((y_true == 1) & (y_pred == 1))
    fp = np.sum((y_true == 0) & (y_pred == 1))
    fn = np.sum((y_true == 1) & (y_pred == 0))

    total_cost = tn * 1 - fp * 1 - fn * 10
    return total_cost


from sklearn.metrics import make_scorer

cost_scorer = make_scorer(cost_function, greater_is_better=True)
```

Vrai Positif = Neutre
Vrai Négatif = Gain de 1 (perte évitée)
Faux Positif = Perte de 1 (manque à gagner)
Faux Négatif = Perte de 10 (perte en capital non remboursé)

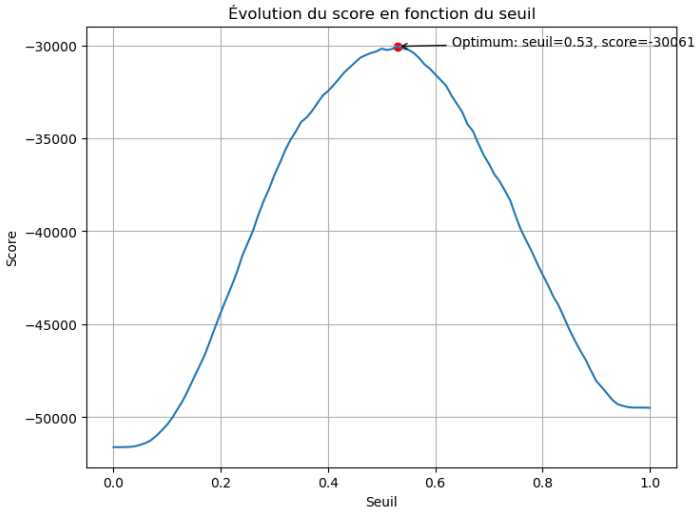
Le but est donc de maximiser le score de cette fonction (minimiser la perte donc).
Je choisis d'utiliser mes meilleurs modèles : Logistic Regression et LGBM pour les comparer avec les différents feature engineering présentés plus haut.

Tracking du cost score sur MLFlow



	Données	Modèle	recall	F1-score	AUC ROC	cost_function	normalized_cost
0	poly_features	LogisticRegression(C=0.01, class_weight='balan...	0.68317	0.23514	0.72018	-32726	0.338735
1	data_domain	LogisticRegression(C=0.01, class_weight='balan...	0.66761	0.25268	0.73689	-31045	0.372702
2	poly_features	LGBMClassifier(class_weight='balanced', max_de...	0.66013	0.24509	0.72533	-31997	0.353465
3	data_domain	LGBMClassifier(class_weight='balanced', max_de...	0.62679	0.26293	0.74326	-30913	0.375369
4	data de base	LogisticRegression(C=0.01, class_weight='balan...	0.66397	0.25409	0.74246	-30974	0.374136
5	data de base	LGBMClassifier(class_weight='balanced', max_de...	0.67286	0.26176	0.74997	-30024	0.393332
6	data de base	DummyClassifier(strategy='most_frequent')	0.0	0.0	0.5	-49490	0.000000

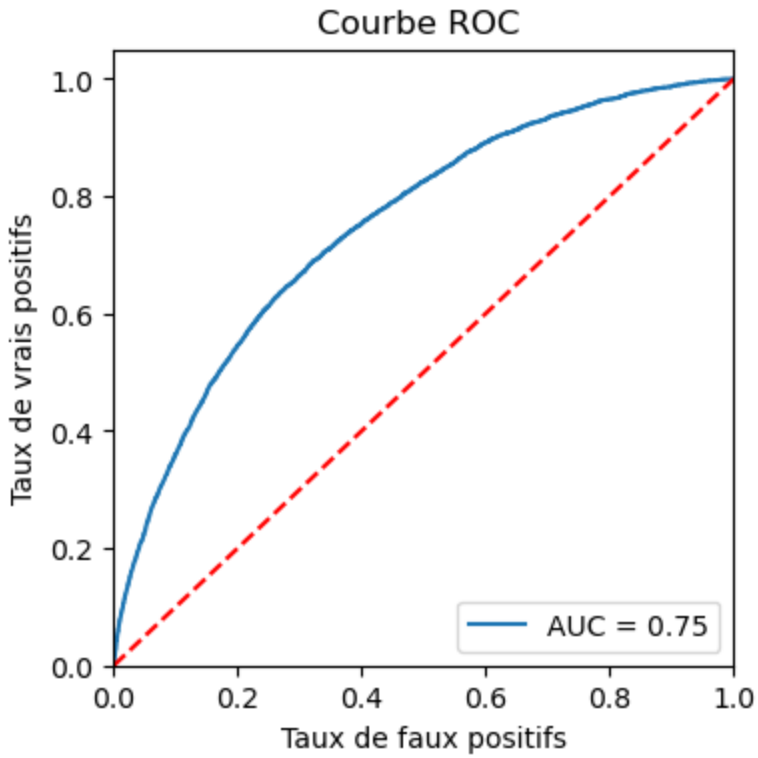
Je choisis d'utiliser pour la suite la **régression logistique** car elle est nettement plus rapide et apporte des résultats quasiment aussi bons.



Optimisation du seuil de détection de faillite bancaire

Je maximise mon score en faisant varier les seuils de détection : **J'obtiens un optimum pour le seuil de 0.53 et le score -30061** (après entraînement sur l'ensemble du dataset).

LE MEILLEUR MODELE DE PREDICTION / CLASSIFICATION



Le meilleur modèle est une régression logistique

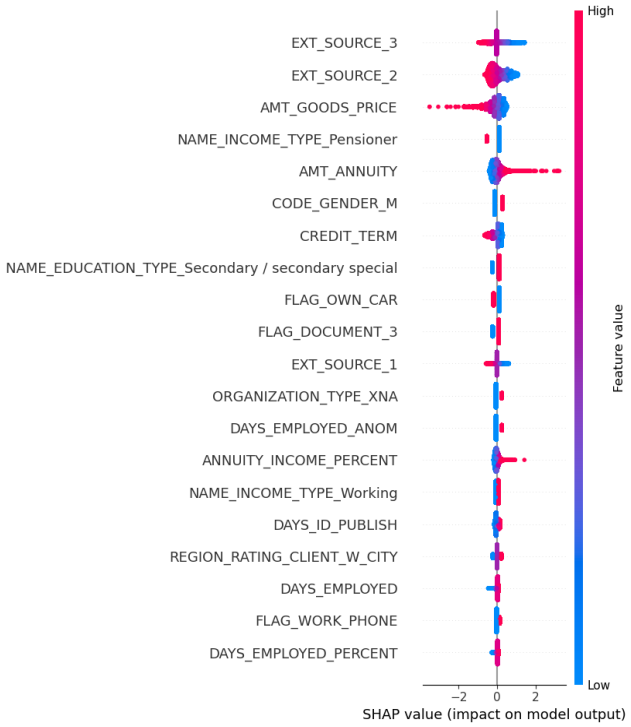
Recall : 064

F1-Score : 0.27

AUC-ROC : 0.749

Cost Function : -30061

INTERPRÉTABILITÉ GLOBALE ET LOCALE



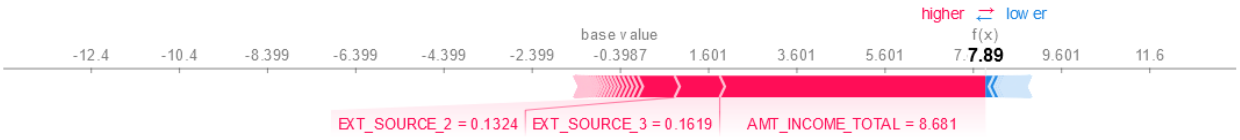
Utilisation de la librairie **SHAP**

Interprétation globale

Ce graphique permet de comprendre comment chaque variable influence le modèle. Par rapport à l'axe verticale zéro : sur la gauche tire la prédiction vers le zéro donc pas de problème de remboursement, et vers la droite la variable augmente le risque de faillite bancaire. Rouge veut dire variable élevé et bleu variable faible. Ainsi par exemple le fait d'être un homme contribue à un risque de faillite tandis que plus la valeur de la source externe 3 est faible plus le risque de faillite est élevé.

Interprétation locale

Cas du score le plus élevé (risque de faillite maximal) :



On remarquera ici que la variable qui a le plus contribué au score de faillite sont les revenus du client (trop faible).

Cas du score le plus faible (pas de risque de faillite) :

Plusieurs variables ont contribué à ce score faible : avoir fourni un document en particulier lors du dépôt de dossier (le numéro 15), le prix du produit pour lequel le crédit a été demandé, les sources externes 2 et 3.

LIMITES ET AMÉLIORATIONS POSSIBLES

La recherche des hyperparamètres aurait pu être plus aboutie, et permettre d'améliorer les métriques, cela dit les performances de ma machine ne m'ont pas permis d'aller plus loin notamment sur les algo de type gradient boosting.

Il aurait également été judicieux d'utiliser un algorithme de d'optimisation type HyperOpt.

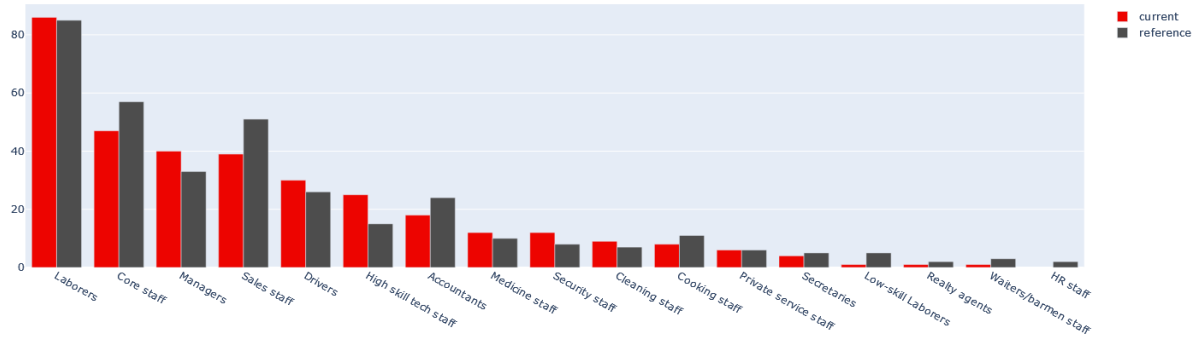
ANALYSE DU DATA DRIFT

Dataset Drift

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

120	7	0.0583
Columns	Drifted Columns	Share of Drifted Columns

Exemple de colonne qui a très faiblement drift 'OCCUPATION_TYPE'



Il n'y a pas de data drift notoire sur ce jeu de données, cela signifie que nous pouvons utiliser l'outil pendant une longue période. Cela dit, il sera forcément utile de le remettre en question de temps à autre et de vérifier avec les nouvelles données obtenues avec les nouveaux prêts.