1. Méthodologie d'entrainement des modèles

Les modèles naïfs n'étant présents que pour créer une référence sur la fonction cout métier, l'effort s'est concentré sur les modèles XG boost et Light GBM.

A. Pre processing

Le kernel LightGBM with Simple Features a été utilisé comme base pour effecture la plupart des taches de feature engineering.

A travers un code synthétique, il permet d'obtenir un jeu de données relativement propre avec:

- One-hot encoding des colonnes de catégories du dataframe principal (application)
- Pre processing (nettoyage de certaines valeurs aberrantes) et fusion de tous les datasets en un unique dataframe niveau client en supprimant les categorical features des fichiers « satellites »
- Création de nombreux indicateurs pertinents : calculs supplémentaires sur les colonnes existantes : min max, sum, mean… Ajout de pourcentage pour « standardiser » les valeurs importantes (revenu par personne dans la famille, part de jours travaillés depuis la naissance, montant obtenu / montant demandé sur le crédit précédent,..)
- Le résultat est un dataframe de 797 features et 356 254 lignes / clients dont on réduit la mémoire utilisée en convertissant au maximum le type de valeur numérique utilisé (int8 ou float16 si possible), ce qui permet de réduire la mémoire de 65%.

À la suite du feature engineering inspiré des compétitions kaggle, nous avons imputé les valeurs manquantes (Nan) par la valeur médiane de chaque colonne concernée afin de pouvoir simuler chacun des modèles sélectionnés sans déformer la distribution des valeurs.

L'ensemble des clients présents dans le jeu de données « application train » a été splitté entre entrainement et test en utilisant la fonction train_test_split de Scikit-learn (avec une taille de l'échantillon de test de 25%).

B. Under Sampling -> déséquilibre des classes

L'échantillon de résultats « target » (y_train) était fortement déséquilibré : 92% des clients de la banque obtenant réellement leur crédit, j'ai utilisé une fonction provenant de la librairie « imbalanced-learn » (s'appuyant sur scikit-learn) afin de rééquilibrer le dataset. Pour ce la fonction RandomUnderSampler a permis de choisir alétoirement parmi les 212 061 individus de la classe « 0 » (crédit accordé), 18 569 individus (soit le même nombre que les individus de la classe 1 / crédit refusé) qui seraient retenus pour la suite de l'entrainement des modèles. Cela présentait aussi l'immense avantage de réduire la taille d'un dataset très volumineux avec 797 colonnes…

C. Définition de la fonction « métier » / métrique d'évaluation

Comme suggéré dans l'énoncé, il a été décidé de prendre un coût 10 fois plus important pour les faux négatifs (clients ayant obtenu le prêt faisant défaut) que pour les faux positifs. Les vrais

positifs et vrais négatifs ne sont pris en compte qu'implicitement dans la formule : dans le dénominateur de la fonction est le nombre total d'éléments de l'échantillon (« len(y_true) ») afin de ramener un « ratio » comparable sur plusieurs datasets différents plutôt qu'un nombre entier.

Pour ce faire, la fonction prend en compte 2 paramètres : la valeur réelle de la cible (y_train) et la valeur prédite (y_pred). Cette dernière est transformée en classification binaire (0 ou 1) avec un seuil à 0.5. Une matrice de confusion est utilisée pour classifier en FN/FP/TN/TP. La fonction « cost » = (10 * FN + FP) / len(y_train) ».

Cette fonction est ensuite intégrée avec « make_scorer » (paramètre « greater_is_better=False » pour minimiser la valeur obtenue) pour utilisation dans la recherche d'hyperparamètres de nos modèles

D. Définition des autres métriques d'évaluation

Afin de s'assurer de la pertinence de notre métrique métier, d'autres outils sont mis en place pour le suivi des indicateurs de performance du modèle :

- matrice de confusion : pour évaluer visuellement très vite les résultats
- AUC/ ROC : pour évaluer le seul de décision et la performance du modèle sur base de l'AUC
- Accuracy score : valeur indicative même si peu utilisée car dans un contexte de classe déséquilibrée (notamment sur le jeu de test)
- F1 score : Le F1 score donne une mesure combinée de la capacité du modèle à fournir des prédictions précises (precision) et à capturer toutes les instances positives (recall). Il prend donc en compte les FP et les FN ce qui est important dans notre cas (même s'il n'a pas le même degré de pénalisation que notre score métier)

Afin d'estimer la performance des modèles et le potentiel overfitting, les métriques ont été également définies sur le jeu de test et d'entrainement.

E. Définition du seuil optimal de probabilité du modèle

La fonction permettant de définir le seuil de probabilité optimal du modèle s'appuie sur un processus itératif permettant de déterminer la valeur optimale entre 0 et 1 de ce seuil en partant de 0.01 et en effectuant des pas de 0.001 jusqu'à 1 pour trouver le « meilleur score métier » (le plus faible).

F. DummyClassifier

Il s'agit d'un classifieur dit "naïf" dans le sens où il ne tient pas compte des données d'entrée pour faire des prédictions. Il n'est utilisé que pour avoir une première marque concernant notre score métier.

G. Logistic Regression (Régression logistique)

Classifieur linéaire utilisant une fonction logistique (ou fonction sigmoïde) pour serrer les prédictions dans un intervalle compris entre 0 et 1. Si cette probabilité est supérieure à un certain seuil, l'échantillon est classé dans la classe positive, sinon il est classé dans la classe

négative. Ils'agit d'une première itération permettant de comparer avec notre modèle naïf (aucune investigation supplémentaire au vu du cout métier, plus élevé que le modèle naïf).

H. LightGBM (Light Gradient Boosting Machine) classifier

Classifieur non-linéaire utilisant des arbres de décision. Il se sert d'une approche basée sur une croissance verticale de l'arbre plutôt que sur la profondeur pour faire croître ses arbres. Cette technique lui permet d'être plus rapide et de consommer moins de mémoire comparé à d'autres modèles utilisant des arbres de décision.

Plusieurs hyperparamètres ont été considérés :

1. learning_rate : vitesse d'apprentissage
2. n_estimators : nombre d'arbres à entraîner
3. max_depth : profondeur des arbres
4. subsample / colsample_bytree / colsample_bylevel / colsample_bynode: suppression de certains éléments (lignes ou colonnes) en utilisant différentes méthodes de sélection.

Une validation croisée avec 5 échantillons a été utilisée dans un GridSearch pour la sélection des hyperparamètres avec pour objectif d'optimiser notre fonction métier créée avec « make_scorer ».

I. XG Boost classifier

XGBoost est une technique d'ensemble qui combine les prédictions de plusieurs modèles faibles (généralement des arbres de décision) pour améliorer la performance globale. Il utilise une approche de boosting, où chaque modèle successif corrige les erreurs du modèle précédent. XGBoost utilise un processus itératif, ajoutant des modèles un à un, en se concentrant sur les erreurs résiduelles des prédictions précédentes. Il intègre des termes de régularisation pour éviter l'overfitting et améliorer la généralisation du modèle et est efficace en termes de calcul, en exploitant le parallélisme pour accélérer le processus d'entraînement.

Les hyperparamètres testés sont sensiblement les mêmes que pour LightGBM ainsi que la méthodologie de validation des hyperparamètres.

2. Synthèse des résultats

Les résultats pour les modèles XG Boost et Light GBM ont été loggés dans ML Flow à l'adresse suivante : https://7bfe-35-204-243-226.ngrok-free.app

Les scores métier pour le modèle naïf et la régression logistique ont été directement calculés à la volée dans le notebook jupyter.

Le tableau-ci-dessous permet de visualiser l'ensemble de ces résultats :

Métrique	DummyClassifier	LogisticRegression	LightGBM	XGBoost
Score métier	0.81	0.87	0.49	0.49
AUC test /train			0.78 / 0.86	0.78 /0.85
Accuracy test /train			0.71 / 0.78	0.73 / 0.77
F1 score			0.77/0.77	0.78/0.77

Duration fit			16.51	50.3
Duration predict			1.59	2.46
Optimal threshold			0.56	0.524

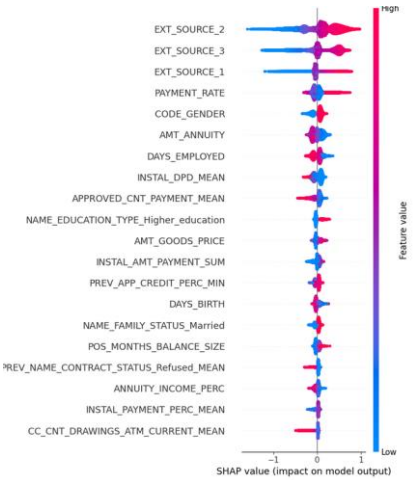
En résumé, les 2 modèles ont sensiblement les mêmes résultats dans l'absolu. Le choix se porte donc sur base du temps utilisé pour entrainer et prédire le modèle qui est largement en faveur de Light GBM.

3. Interprétabilité globale et locale du modèle

La librairie Shap (SHapley Additive exPlanations) fournit des explications interprétables sur les prédictions des modèles, pour comprendre comment les caractéristiques d'entrée contribuent aux résultats du modèle. Les explications SHAP sont basées sur les valeurs de Shapley de la théorie des jeux coopératifs.

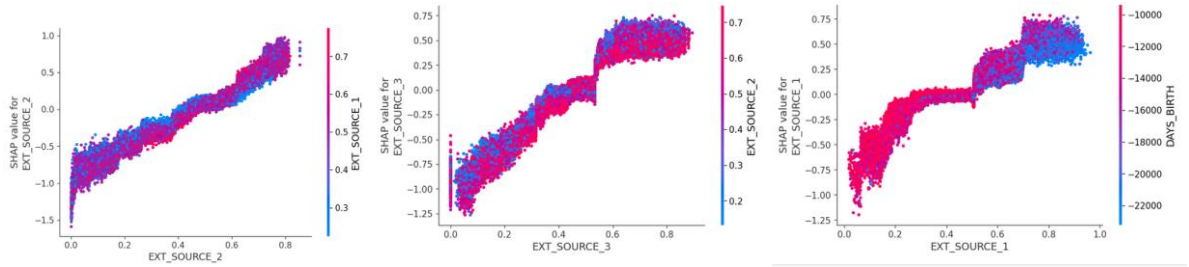
A. Interprétabilité globale

Summary plot



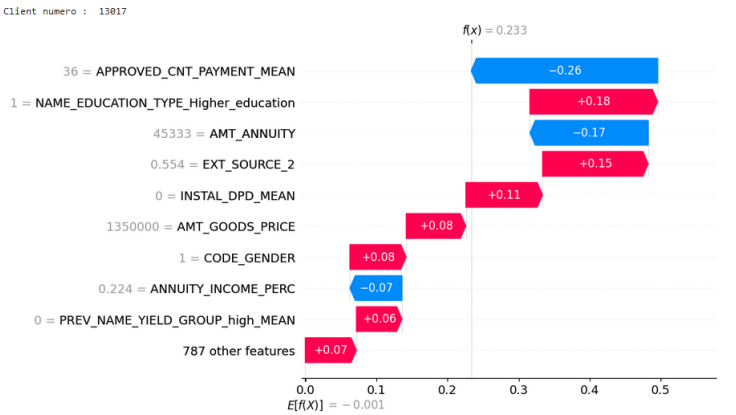
Les caractéristiques provenant de notations d'agence extérieure (EXT_SOURCE 1, 2 et 3) sont celles qui impactent le plus le modèle. Le taux de paiement des échéances des crédits précédents et le genre (0=femme / 1=homme) sont également des drivers influant sur le modèle.

Dépendance plot



Les principales caractéristiques ont un impact « positif » sur le modèle : plus la valeur des features augmente, plus leur SHAP value augmente avec elles.

B. Interprétabilité locale



Pour ce client, les features ayant le plus d'impact sur son résultat sont :

- approved_cnt_payment_mean, amt_annuity : le nombre d'échéances réglées lors de son précédent prêt semblent avoir un impact négatif sur la décision d'octroi de crédit, de même que le montant des annuités de ce prêt.
- NAME_EDUCATION_TYPE_Higher_education, EXT_SOURCE_2, INSTAL_DPD_MEAN: à contratrio, le fait que le client ait fait des études supérieures et sa notation auprès d'un organisme externe ou le fait qu'il ait payé ses échéances dans les temps ont un impact positif sur la décision.

4. Limites et améliorations

A. Stockage de la base de données

Les données client utilisées dans l'application sont chargées depuis Github dans Streamlit, ce qui alonge considérablement le temps de réponse de l'application. C'est pourquoi il a été décidé de n'utiliser qu'un échantillon de 10 000 clients pour « Proof of Concept » et démonstration dans un premier temps. Idéalement, l'ensemble des données client devrait être stocké dans une base de données dédiée à laquelle l'application web pourra faire appel. Cette fonctionnalité étant payante sur Azure, cela n'a pas pu être mis en place.

B. Modélisation : recherche d'hyperparamètres et essai d'autres modèles vs capacités de mémoire et taille du dataframe

Il aurait certainement été pertinent d'aller plus loin dans la démarche de modélisation en essayant d'autres modèles pour évaluer leur performance (CatBoost, CNN ou Random Forest ?). Il faudrait aussi aller plus loin dans la recherche d'hyperparamètres pour optimiser la performance des modèles.

Cependant le dataframe en sortie de l'analyse exploratoire étant très lourd, nous avons rapidement rencontré des problèmes de mémoire en local et des temps d'attente très longs sur google collab (gratuit).

Pour aller plus loin dans la modélisation, il faudrait augmenter les capacités par exemple en souscrivant à google collab pro. Une autre alternative serait d'effectuer une réduction de dimensions (via ACP par exemple) en sortie de l'analyse exploratoire pour réduire la taille du dataframe.

C. Description des features

Un fichier de description existait pour les features « originales » du fichier principal (application_train / test.csv), mais les colonnes créées à la suite de la fusion et l'ajout d'indicateurs n'ont aucune description. Pour rendre le dashboard plus explicite et compréhensible (notamment sur l'interprétation SHAP), il serait nécessaire de compléter cette description et l'intégrer dans le dashboard.

D. Validation de la fonction coût et de son impact sur la stratégie

Le fait de prendre un coût très important pour les faux négatifs a un impact direct sur la stratégie de la société si l'on suit « aveuglément » la sortie du modèle. En effet, on constate que 40% des crédits demandés sont alors refusés dans notre scénario en utilisant le jeu de données (contre 8% en « réel »).

Il s'agit donc d'un impact très important pour la société : il faudra être très prudent quant à la mise en place de cet indicateur et du rôle que l'on veut lui faire jouer avec le management.

5. Analyse du data drift

En utilisant la librairie Evidently, le data drift report a été paramétré pour obtenir les divergences entre l'échantillon de 10 000 individus du dataset application_train (référence) également utilisé pour le dashboard (en raison des problèmes de mémoire) et les individus du dataset application_test (actuel).

Les colonnes ont été dissociées entre les numériques et les catégoriques en incluant les colonnes binaires (0 / 1) dans les catégoriques en utilisant une condition sur les valeurs uniques dans une boucle, puis en les mappant à l'aide la fonction dédiée de la librairie Evidently.

Les métriques utilisées et les seuils appliqués sont :

- la divergence de Kullback-Leibler pour les colonnes numériques avec un seuil à 0.1 (mesure de dissimilarité entre deux distributions de probabilités). Le test de Kolgororov-Smirnov, utilisé par défaut dans le rapport, semblait rencontrer certains problèmes sur notre dataset (drift sur toutes les colonnes quel que soit le résultat).
- L'indice de Stabilité de la Population (PSI) pour les colonnes catégoriques qui permet de mesurer les tendances à l'évolution à travers le temps sur des colonnes numériques et catégoriques. Le seuil d'identification a été relevé à 0.2 pour cette métrique (contre 0.1 par défaut) puisque les études indiquent un risque « modéré » pour une valeur entre 0.1 et 0.2 sur cette métrique.

Sur les 798 colonnes utilisées dans notre modèle, seules 17 sont considérés comme « drifted » selon les critères établis, soit 2.13%.

Les colonnes ayant le plus de data drift proviennent des informations fournies par les autres institutions financières sur les crédits précédents du client : le nombre de mois des prêts, la valeur de chaque échéance mensuelle : minimale, moyenne, somme… Il s'agit d'informations

très spécifiques à chaque client et donc il n'est pas vraiment surprenant que ces valeurs puissent varier entre 2 populations données. De plus, en observant la distribution de chaque dataset, il se pourrait que ces différences soient aussi liées à la taille des datasets puisque notre distribution de référence n'est que de 10 000 individus contre 40 000 pour l'actuelle.

Les colonnes catégoriques n'ont pas de data drift détecté puisque le PSI maximal identifié est de 0.19 contre un seuil à 0.2. Il faudra néanmoins faire attention à celui-ci : les features concernent le type de prêt octroyé au client, avec uniquement des cash loans sur notre dataset actuel contre une quantité non négligeable de revolving loans (environ 10%) pour le dataset référence. Il pourrait être judicieux de supprimer ces derniers pour avoir un point de comparaison pertinent à l'avenir.