

Note méthodologique

Cette note technique présente l'ensemble du processus d'élaboration du modèle.

Sommaire

- I- Processus d'entraînement des modèles
 - Traitement des données
- II- Traitement du déséquilibre des classes
- III- La fonction coût métier et les métriques d'évaluations
- IV- Application de différents modèles et le choix du modèle final
- V- L'analyse du Data Drift

Références :

Base de données : <https://www.kaggle.com/c/home-credit-default-risk/data>

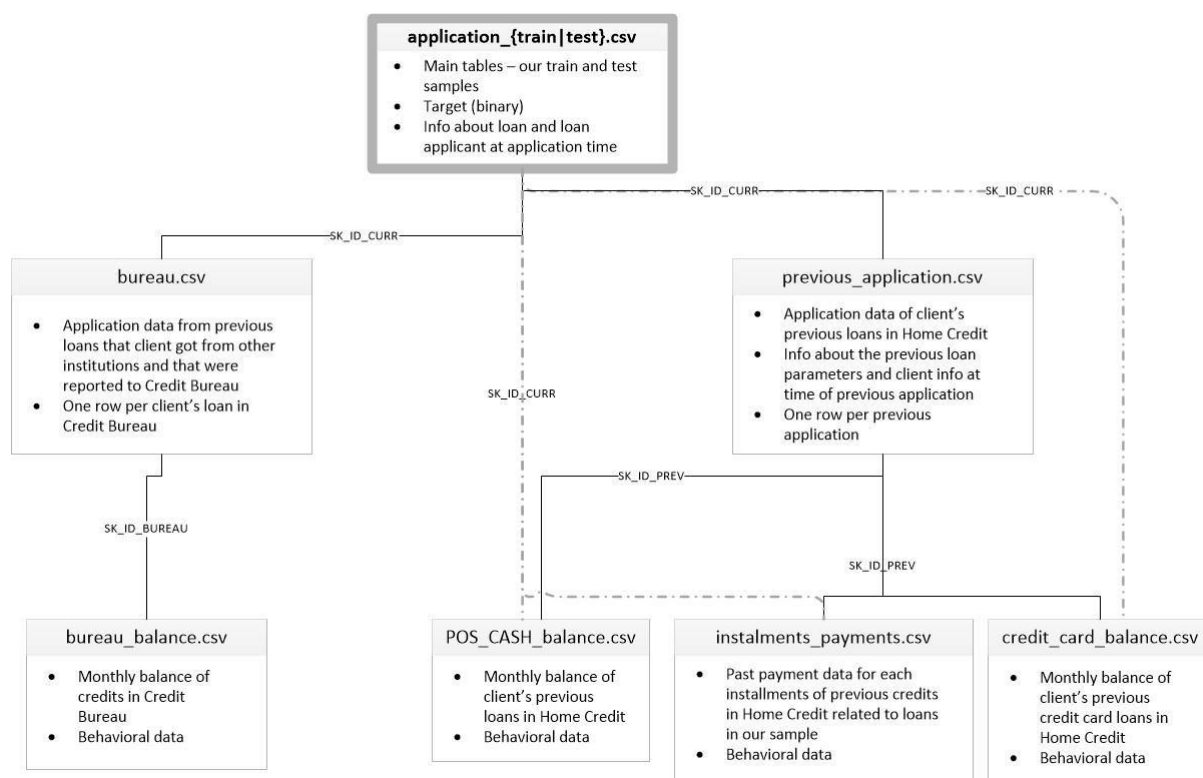
GitHub: <https://github.com/sorayadata?tab=repositories>

Le projet consiste à développer un outil de « scoring crédit » pour l'entreprise **Prêt à dépenser**. Cet outil a pour objectif de calculer la probabilité qu'un client rembourse son crédit, puis de classer la demande en crédit accordé ou refusé.

I- Processus d'entraînement des modèles

-Traitement des données

Les données mises à disposition sont issues d'une base de données Kaggle de la compétition *Home Credit Default Risk*. Elle comporte 10 fichiers.



Pour faciliter l'analyse exploratoire et le feature engineering, nous utilisons ce [kernel](#) Kaggle. À la suite de ce traitement, nous avons 307 507 lignes et 561 variables pour le dataset d'entraînement et 48 744 lignes et 561 variables pour le dataset de test.

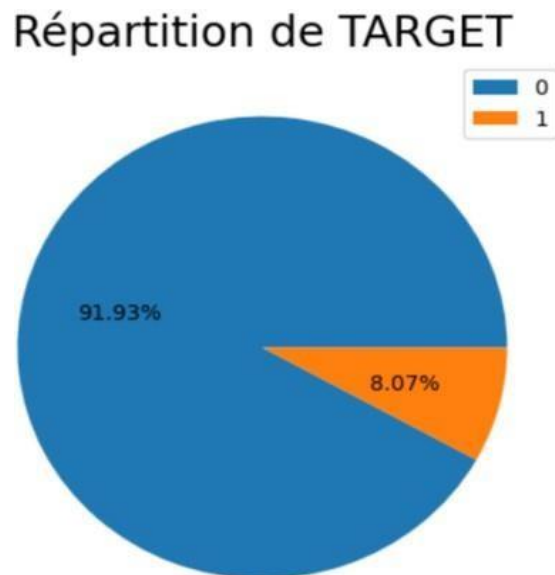
Nous traitons les données manquantes qui représente 25,93% de nos données. Nous commençons par supprimons les variables ayant plus de 50% de données manquantes. Ensuite, nous procédons à l'imputation. Nous appliquons différentes méthodes d'imputation simples (0, médiane et mode). Nous avons également appliqué un preprocessing de Standard scaler.

Nous n'utilisons que le dataset d'entraînement qui sera séparé en jeu d'entraînement, de validation et de test, pour entraîner chaque modèle.

Le dataset de test ne contient pas de colonne **Target** et sera utilisé lors du déploiement du modèle final.

II- Traitement du déséquilibre des classes

Nous avons un problème de déséquilibre de classe représentée par 8.07 % de clients en défaut contre 91.93 % de clients sans défaut.



Ce déséquilibre des classes doit être pris en compte lors de l'entraînement des modèles. En effet, un modèle « naïf » prédisant systématiquement que les clients sont sans défaut pourrait être considéré à tort comme performant, alors qu'il ne permettrait pas de détecter les clients à risque.

L'approche utilisée pour rééquilibrer les deux classes est le sous-échantillonnage **undersampling**, une méthode pour équilibrer les classes dans un jeu de données déséquilibré. Il réduit le nombre d'échantillons de la classe majoritaire.

III- La fonction cout métier et les métriques d'évaluations

La fonction de coût métier

Une fonction de coût métier sur mesure a été créée pour la société *Prêt à dépenser*. Nous créons donc un score métier qui a pour objectif de pénaliser plus sévèrement les **Faux Négatifs** qui sont donc des clients ayant fait défaut mais qui ont été prédit comme ne faisant pas défaut. En effet, ce type d'erreur engendre une perte en capital plus importante pour l'entreprise.

Nous devons donc pénaliser plus fortement les **Faux Négatifs** que les **Faux positifs**.

		Classe réelle	
		-	+
Classe prédite	-	True Negatives (vrais négatifs)	False Negatives (faux négatifs)
	+	False Positives (faux positifs)	True Positives (vrais positifs)

TP_coeff	= 0	# Vrais positifs
FP_coeff	= 0	# Faux positifs
FN_coeff	= -10	# Faux négatifs
TN_coeff	= 1	# Vrais négatifs

Nous obtenons la métrique suivante :

$$gain = \frac{TP * TP_{coeff} + TN * TN_{coeff} + FP * FP_{coeff} + FN * FN_{coeff}}{TN + FP + FN + TP}$$

En plus du score métier, d'autres métriques d'évaluation ont également été calculées et observées telles que :

- **Accuracy** : somme de tous les vrais positifs et vrais négatifs divisés par le nombre total d'instances. Des valeurs élevées de ce paramètre sont souvent souhaitables.
- **Precision** : indique le rapport entre les prévisions positives correctes et le nombre total de prévisions positives.
- **Recall/Rappel** : paramètre qui permet de mesurer le nombre de prévisions positives correctes sur le nombre total de données positives.
- **F1 score** : moyenne harmonique de la précision et du rappel. Sa valeur est maximale lorsque le rappel et la précision sont équivalents.
- **Fbeta score** : généralisation de la F-mesure qui ajoute un paramètre de configuration appelé beta. Une valeur bêta < 1, donne plus de poids à la précision et moins au rappel, tandis qu'une valeur bêta > 1 donne moins de poids à la précision et plus de poids au rappel. Ici nous donnons plus de poids au rappel qui minimise les faux négatifs.
- **ROC AUC score** : mesure de façon globale la performance d'un modèle de classification. Il indique à quel point le modèle est capable de faire la distinction entre les classes.

IV- Application de différents modèles et le choix du modèle final

Nous avons sélectionné quelques algorithmes de classification :

- **Dummy Classifier (baseline)** : ne prend pas en compte les caractéristiques du jeu de données et se contente de faire des prédictions en utilisant des règles simples. Ici, on renvoie l'étiquette de classe la plus fréquente.
- **Logistic Regression** : le but est de trouver une relation mathématique entre les variables d'entrée (features) et la variable de sortie (classe prédite). Cette relation est généralement exprimée sous la forme d'une fonction logistique qui transforme la sortie en une probabilité.
- **Random Forest** : algorithme d'apprentissage automatique qui combine plusieurs arbres de décision pour effectuer des prédictions. Chaque arbre de la forêt donne une prédiction et la classe prédite est déterminée par un vote majoritaire.
- **Light GBM** : algorithme d'apprentissage automatique basé sur le gradient boosting qui est conçu pour offrir une exécution rapide et des performances élevées. Il utilise une technique d'échantillonnage basée sur le gradient pour sélectionner les échantillons les plus informatifs pendant le processus d'apprentissage.

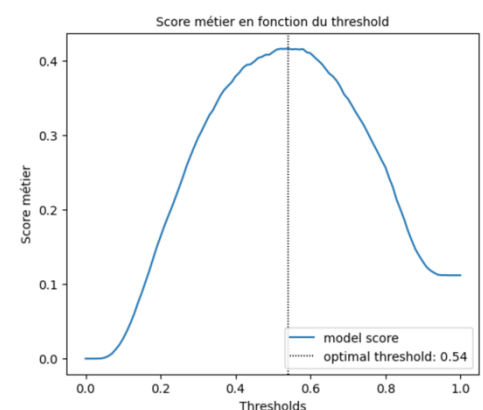
Les algorithmes testés utilisent un pipeline afin de rassembler les étapes de traitement du déséquilibre des données et entraînement du modèle, avec une recherche d'hyperparamètres et cross validation (via une *GridSearchCV*) en passant par les étapes suivantes :

- Définition de la grille d'hyperparamètres
- GridSearchCV : trouve les paramètres optimaux minimisant la fonction de coût
- Choix du meilleur modèle sur le score de validation de la fonction métier

L'algorithme final et optimisé :

-Notre modèle le plus performant est Light GBM avec les paramètres (`learning_rate=0.05`, `max_depth=-1`, `n_estimators=500`, `num_leaves=31`, `random_state=0`)] et un seuil de 0,09

-Un tracking a été effectué pour le modèle via MLflow.



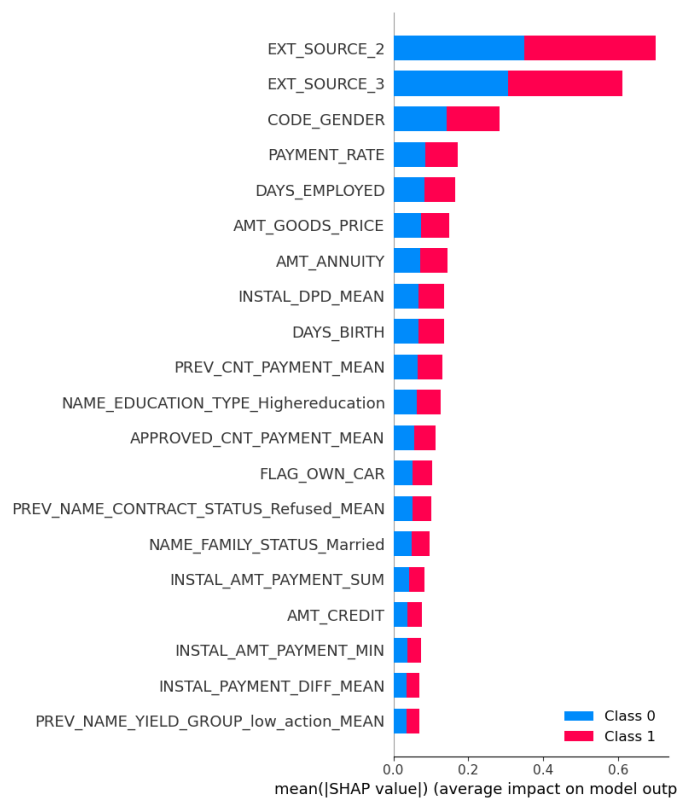
Score métier : 0.41
Accuracy score : 0.71
Precision score : 0.18
Recall score : 0.70
F1 score : 0.28
Fbeta score : 0.44
ROC AUC score : 0.71

Nous obtenons les résultats suivants :

L'interprétabilité du modèle

Afin de mettre en évidence les features les plus importantes pour ce modèle, nous avons utilisé `summary_plot`. Ce type de tracé agrège les valeurs SHAP pour tous les échantillons de l'ensemble sélectionné. Ensuite, les valeurs SHAP sont triées. la première affichée est la feature la plus importante

En bleu sont représentées les caractéristiques ayant une SHAPley valeur négative et donc une contribution négative et en rouge les caractéristiques ayant une SHAPley valeur positive, donc une contribution positive.



VI- Les limites et les améliorations possibles

La modélisation est ici essentiellement basée sur la création d'une métrique d'évaluation propre au métier, permettant de pénaliser plus fortement les Faux Négatifs (FN - mauvais client prédit bon client : donc crédit accordé et perte en capital). Cependant, il pourrait être plus pertinent de collaborer avec des équipes métier afin de créer et utiliser une métrique peut être plus pertinente.

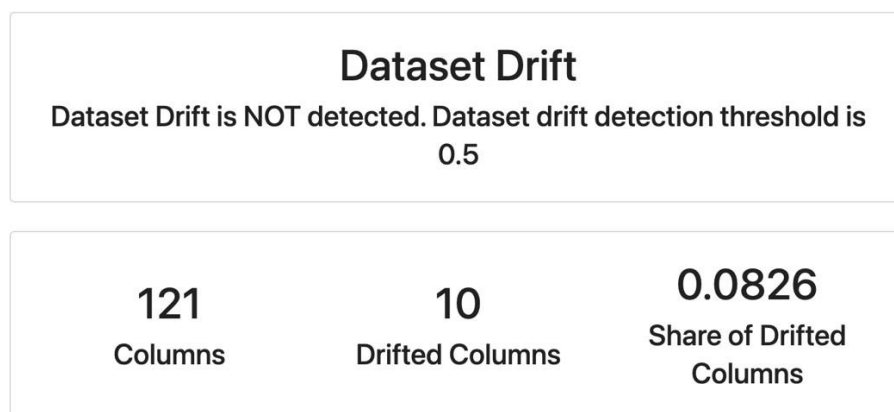
De la même manière, les équipes métier pourraient également collaborer dans le cadre du feature engineering afin de créer des variables pertinentes dans un sens métier.

Un autre axe d'amélioration serait de tester d'autres hyperparamètres ou d'autres algorithmes de classification.

V- Le Data Drift

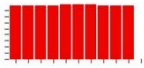
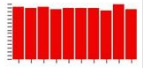
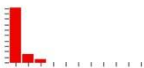
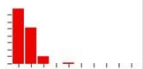




Le Data Drift est une variation des données du monde réel par rapport aux données utilisées pour entraîner, tester et valider le modèle avant de le déployer en production.

L'approche pour détecter le data drift consiste à utiliser des tests statistiques qui comparent la distribution des données de référence (data d'entraînement) et des nouvelles données (données de production). Si la différence entre les deux est significative alors un drift s'est produit.



Data Drift Summary

Drift is detected for 8.264% of columns (10 out of 121).

<div>Search</div>			
Column	Type	Reference Distribution	Current Distribution
> SK_ID_CURR	num		
> AMT_REQ_CREDIT_BUREAU_QRT	num		
> AMT_REQ_CREDIT_BUREAU_MON	num		
> AMT_GOODS_PRICE	num		

On constate donc qu'il n'y a pas de data drift entre nos données de référence (données d'entraînement) et nos données actuelles (données de test), bien qu'un léger drift ait été détecté dans 10 de nos colonnes.