Projet 7 : NOTE METHODOLOGIQUE

Introduction :

"Prêt à dépenser", est une société financière qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

L'entreprise souhaite **mettre en œuvre un outil de "scoring crédit" pour calculer la probabilité** qu'un client rembourse son crédit, puis classifie la demande en crédit accordé ou refusé. 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.

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.

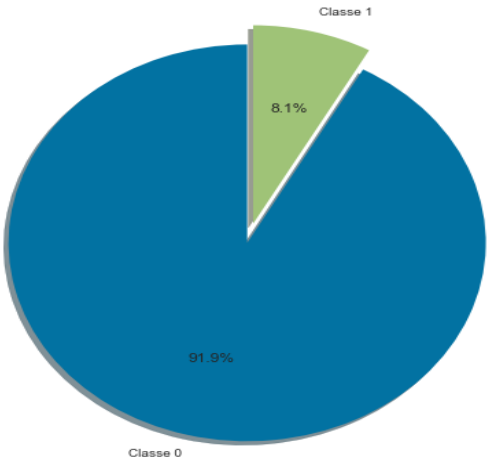
1. La méthodologie d'entraînement du modèle

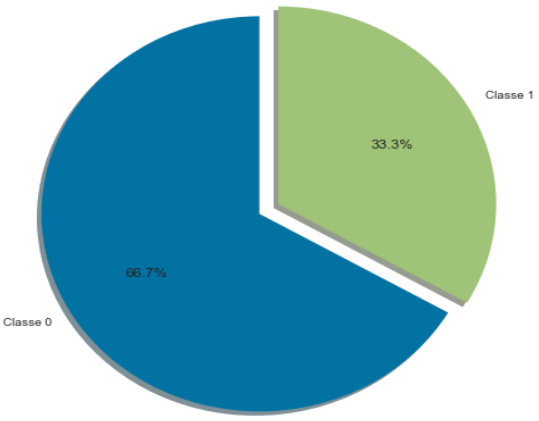
▪ Traitement des classes déséquilibrées

Il s'agit d'un problème de classification avec deux classes déséquilibrées(8% de clients en défaut de paiement contre 92% de clients sans défaut de paiement). Le déséquilibre des classes détériore la qualité de prédiction du modèle puisqu'à l'étape d'entraînement, l'apprentissage se fera au détriment de la classe minoritaire (client en défaut) qui sera mal détectée par le modèle au moment de la prédiction.

Trois méthodes de traitement des classes déséquilibrées ont été testées : **SMOTE** , **RandomUnderSimple** et la **combinaison des deux**.

Dans ce projet, nous avons utilisé la méthode **SMOTE** qui nous donne les meilleures métriques.





- **Modèles étudiés**

Il existe différents modèles de Machine Learning plus ou moins performants et plus ou moins complexes. Nous avons testé les quatre modèles suivants avec leurs paramètres par défauts : **Régression Logistique, Décision Trees Classifier, Random Forest Classifier, Light Gradient Boosting Classifier.**

Il en est sorti que le **Light Gradient Boosting Classifier** a les meilleures performances. Les modèles ont été évalués selon les métriques suivantes: **Accuracy**, **Precision, Recall**, **Auc** et **Fbeta_score.**

2. La fonction coût métier, l'algorithme d'optimisation et les métriques d'évaluation

Accuracy : mesure combien d'observations, positives et négatives, ont été correctement classées.

Precision : décrit la fraction des vrais positifs et de tous les positifs.

Precision= TruePositives/(TruePositives+FalsePositives)

Recall : décrit la fraction des vrais positifs et de tous les positifs (c.-à-d. y compris les faux négatifs)**.**

Recall=TruePositives/(TruePositives+FalseNegatives)

Auc : La courbe ROC représente graphiquement le taux de positif réel par rapport au taux de faux positifs. L'Auc est simplement l'air sous cette courbe.

Nous avons aussi affiché la matrice de confusion pour le modèle final retenu

Fbeta_score : Le score F-beta est la moyenne harmonique pondérée de précision et de rappel, atteignant sa valeur optimale à 1 et sa pire valeur à 0. Le paramètre bêta détermine le poids du rappel dans le score combiné.

Matrice de confusion :

		Classe réelle	
		0	1
Classe prédite	0	**True Négatives (TN)**	**False Négatives (FN)**
	1	**False Positives (FP)**	**True Positives (TP)**

- **TN** : le modèle prédit 0 et la valeur réelle est 0

- **TP** : le modèle prédit 1 et la valeur réelle est bien de 1

- **FN** : le modèle prédit 0 alors que la valeur réelle est bien de 0

- **FP** : le modèle prédit 1 alors que la valeur réelle est de 0

Le modèle peut donc se tromper de deux manières :

- soit en prédisant positif alors que l'individus est négatif,
- soit en prédisant négatif alors que l'individus est positif

Dans ce projet, on cherche à minimiser les False Négatives(FN). On a utilisé les fonctions coût **AUC** et **Fbeta_score** pour tenir compte de l'importance relative de chaque erreur

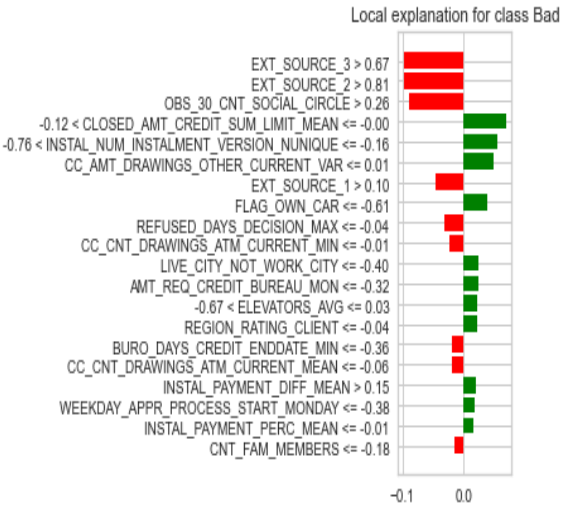
L'optimisation des hyperparamètres du modèle choisi a été réalisée via une validation croisée avec la méthode **GridSearchCV** (3 folds). Enfin, le modèle optimisé a été évalué selon les métriques définies précédemment.

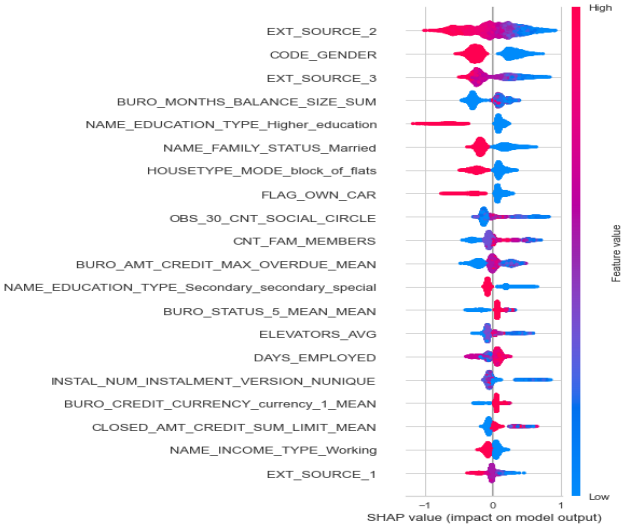
3. L'interprétabilité globale et locale du modèle

Le modèle étant destiné à des équipes opérationnelles, il est important d'être en mesure d'expliquer les décisions de l'algorithme.

Les librairies **Lime** et **Shap** ont été utilisées pour donner l'importance des variables (respectivement locale et globale).

- **Lime** permet de déterminer pour chaque observation, les « features » qui ont contribué le plus dans le résultat de la prédiction.
- **Shap** (Shapley Additive exPlanations) permet de connaître globalement les « features » les plus influentes sur les prédictions du modèle.





4. Les limites et les améliorations possibles

Notre objectif étant d'obtenir un modèle de scoring qui maximise les cas favorables au modèles économiques de l'entreprise, il serait judicieux d'améliorer les trois points suivantes :

- **<u>Choix du métrique d'évaluation</u> :**

L'axe principal d'amélioration serait de définir plus finement la métrique d'évaluation en collaboration avec les équipes métier. D'autre part, le seuil de solvabilité d'un client a été fixé arbitrairement(seuil=50%). Il conviendrait aussi de le fixer avec les équipes métier

- **<u>Sélection des variables et 'feature engineering'</u> :**

Les étapes de traitement préalable à la modélisation ont été effectuées en utilisant un Kernel Kaggle. Il serait judicieux d'effectuer une sélection des variables en utilisant d'autres méthodes comme le « **Backward** » ou le «

Rfecv ». Quant au « feature engineering », il y a probablement l'opportunité de l'améliorer en collaboration avec les équipes métier. Ces derniers sauront mieux quelles compositions de variables sont les plus pertinentes au vu de leur expérience dans leur choix d'accorder un prêt ou non à un client donné.

- **Méthode de Cross Validation** :

Il pourrait être envisageable de diviser le jeu de données en n partitions puis d'entraîner un modèle sur chacune, puis d'utiliser un « blender » sur les n modèles obtenus. Un « blender » consiste à agréger les prédictions de chacun des classificateurs et de prédire la classe qui obtient le plus grand nombre de votes. Ce classificateur par vote majoritaire obtient souvent une exactitude plus élevée que le meilleur classificateur de l'ensemble.

- **Enrichir** les informations descriptives des clients en fonction des retours des utilisateurs du Dashboard.