NOTE METHODOLOGIQUE

1- Méthodologie d'entraînement du modèle

L'entraînement d'un modèle nécessite une étape transitoire entre le preprocessing et la modélisation. Ce modèle de prédiction peut être représenté par une fonction qui prend des données en entrée et une décision en sortie. Dans le cas d'apprentissage supervisé où nous souhaitons expliquer une variable de sortie, l'échantillon est classiquement subdivisé en 2 parties :

(i) L'échantillon d'apprentissage correspond à l'échantillon principal où sont appliquées les méthodes, sur lesquelles les algorithmes apprennent. Il sert à ajuster le modèle, et représente dans notre cas 80% des données.

(ii) L'échantillon de test est utilisé pour évaluer le modèle optimal (au sens du résultat de la validation croisée). Il n'a donc pas été utilisé pour l'apprentissage, ce qui fait que le modèle sélectionné est indépendant de cet échantillon test. L'idée est de simuler la réception de nouvelles données (entrée) afin de prédire la variable à expliquer à partir du modèle final et de les comparer aux « vraies » valeurs de la variable à expliquer. Cet échantillon permet d'évaluer objectivement l'erreur réelle. L'échantillon test représente donc 20% des données.

Ce découpage de données dans un projet de Machine Learning est une étape très importante qu'il ne faut pas négliger, il existe en effet un risque de surévaluer le modèle (over-fitting) ou tout simplement le contraire (under-fitting). En effet, par nature un modèle va coller (mais pas trop) à ses données d'entraînement. train_test_split() de Scikit-Learn permet d'obtenir facilement une répartition aléatoire des individus en base d'apprentissage et de test.

Le problème de Machine Learning à résoudre est un problème de classification binaire, il faut s'assurer que chaque côté contient une proportion raisonnable des classes 0 et 1. L'Analyse Exploratoire des Données a permis d'identifier un fort déséquilibre entre la précision trouvée entre la classe 0 et la classe 1. L'échantillon de travail est déséquilibré, 92% en classe 0 vs 8% en classe 1. Le traitement Oversampling (ou suréchantillonnage en français) permet d'ajuster la distribution de classe de manière à avoir une répartition plus égalitaire.

L'existe plusieurs techniques pour résoudre le problème de déséquilibre de classe dans les données. Le ré échantillonnage des données est l'une des techniques les plus utilisées. La technique de suréchantillonnage des minorités synthétiques (Synthetic Minority Oversampling Technique ou **SMOTE** en anglais) est une technique sophistiquée qui ne se contente pas de juste dupliquer des cas de défaut mais utilise les caractéristiques des plus proches voisins des cas de défaut de paiement pour créer de nouveaux cas de défaut synthétiques. Cette méthode bien

qu'utilisant des algorithmes hypersophistiqués présente le risque que les voisins les plus proches des cas de défaut ne soient pas en réalité des cas de défaut ce qui peut entraîner des erreurs de modélisation.

Oversampling Data Using SMOTE

```
print("Label 1, Before using SMOTE: {} ".format(sum(y_train==1)))
print("Label 0, Before using SMOTE: {} ".format(sum(y_train==0)))
```

```
Label 1, Before using SMOTE: 17412
Label 0, Before using SMOTE: 197845
```

```
%%time
sm = SMOTE(random_state=2)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train)
```

```
Wall time: 1min 22s
```

```
print("Label 1, After using SMOTE: {}".format(sum(y_train_res==1)))
print("Label 0, After using SMOTE: {}".format(sum(y_train_res==0)))
```

```
Label 1, After using SMOTE: 197845
Label 0, After using SMOTE: 197845
```

2- Fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation

Toute la force du Machine Learning réside dans la diversité des approches utilisées. Plus le nombre de méthodes testées est élevé, plus il sera possible de trouver le meilleur algorithme permettant de répondre à la problématique, ici le modèle de scoring. Contrairement aux modèles statistiques classiques qui doivent vérifier certaines hypothèses sur la distribution des données, le Machine Learning a de fortes capacités prédictives compte tenu de son approche non paramétrique et de la faculté de ses algorithmes à apprendre sans a priori, à partir des données. Au vu du large panel de possibilités, trois algorithmes de Gradient Boosting ont été testés. Une baseline a également été fixée par Régression Logistique, un modèle statistique classique.

L'entreprise nommée "Prêt à dépenser" lutte contre les défauts de paiement des clients, les pertes financières ne sont en effet pas souhaitables (frais de recouvrement, pertes,..). Le modèle ne permettra pas d'éviter totalement ce risque, à titre d'exemple une erreur de prédiction aura pour conséquence soit un défaut de paiement du client, soit un refus de crédit à un client qui pourrait rembourser sa dette sans aucune défaillance. Les erreurs de prédiction doivent être minimisées, dans cette logique une fonction coût ayant pour objectif de pénaliser les Faux Positifs et les Faux Négatifs a été implémentée. Terminologie :

FP (Faux Positifs) : les cas où la prédiction est positive, mais où la valeur réelle est négative. Perte d'opportunité si le crédit client est refusé à tort, alors qu'il aurait été en mesure d'être remboursé. FN (Faux Négatifs) : les cas où la prédiction est négative, mais où la valeur réelle est positive. Perte réelle si le crédit client accepté se transforme en défaut de paiement.

TP (Vrais Positifs) : les cas d'acceptation, le crédit client sera remboursé.

TN (Vrais Négatifs) : les cas de refus, le crédit client ne pourra pas être remboursé. Ainsi, les pertes d'un crédit en raison d'une mauvaise classification dépendent des probabilités Faux Positifs et Faux Négatifs. L'idée est d'éviter les clients avec un fort risque de défaut. Il est donc nécessaire de pénaliser les FP et FN cités précédemment. Pour réduire ce risque de perte financière, il faut maximiser deux critères Recall et Precision.

L'application de cette métrique métier passe par la quantification de l'importance relative entre Recall et Precision, à savoir Beta (β). Cela revient à estimer le coût moyen d'un défaut, et le coût d'opportunité d'un client refusé par erreur. Cette connaissance métier n'est pas évoquée à ce stade du projet, nous allons donc l'estimer. Cette hypothèse pourra bien entendu être modifiée avec un interlocuteur métier

3- Interopérabilité globale et locale du modèle

Une alternative pour l'optimisation des fonctions ainsi que des hyperparamètres des pipelines Machine Learning. Le réglage des hyperparamètres est basé sur l'optimisation bayésienne. from hyperopt import fmin, tpe, hp, STATUS_OK, Trials La configuration optimale des hyperparamètres pour une fonction donnée ne doit pas être entièrement basée sur l'intuition ou l'expérience de certains. La recherche d'une telle configuration optimale doit s'appuyer sur des approches garantissant l'optimalité nécessaire. Parmi les approches les plus utilisées, on retrouve les méthodes basées sur des recherche exhaustives (par exemple Grid Search et Random Search), ou par optimisation bayésienne. L'approche de l'optimisation bayésienne se concentre sur un modèle de probabilité P (score | configuration), qui est mis à jour par un processus itératif d'interrogation d'un historique (score, configuration) dont l'objectif est la maximisation du score donné pour une configuration. HyperOpt prend l'optimisation bayésienne comme prémisse en faisant quelques variations dans le processus d'échantillonnage, la définition et la réduction de l'espace de recherche et les algorithmes pour maximiser le modèle de probabilité HyperOpt nécessite 4 paramètres pour une implémentation de base qui sont : la fonction à optimiser, l'espace de recherche, l'algorithme d'optimisation et le nombre d'itérations. La fonction StratifiedKFold(5) a été utilisée pour trouver la meilleure itération.

La réponse au besoin d'interopérabilité est prépondérante, le contexte de prédiction n'est pas uniquement appliqué à des experts de la data science mais au contraire à des experts du crédit. Un chargé de clientèle doit pouvoir utiliser le modèle via l'application mise à disposition, en face à face avec son client, dans le but de lui expliquer le plus simplement possible la décision envisagée dans l'étude de son dossier. En d'autres termes, « l'interprétation » désigne l'évaluation globale du processus de prise de décision. Elle vise à représenter l'importance relative de chaque variable. L'idée est donc d'expliciter au mieux le score renvoyé par le modèle. La classe lightgbm. LGBMClassifier permet de mieux comprendre le choix et l'importance de des caractéristiques via un attribut feature_importances_

4- Limites et les améliorations possibles

Les résultats obtenus sont basés sur une hypothèse non confirmée par les équipes métier. Le coefficient Beta pourrait être affiné dans ce sens pour éventuellement améliorer la métrique d'évaluation.

L'espace de recherche HyperOpt permet de larges possibilités, le choix des hyperparamètres est évolutif, la question d'élargissement vers d'autres hyperparamètres peut également permettre d'augmenter les performances actuelles.