NOTICE EXPLICATIVE

1. Entraînement du modèle
   1. Prepocessing des données

Les données ont d’abord été traitées par un kernel Kaggle qui ressort avec un dataframe avec 38000 lignes et 154 colonnes. Chaque ligne correspond à un client, nous restons avec certaines colonnes avec des valeurs NA importantes et de part le procédé de créations de certaines features nous avons aussi des corrélations importantes susceptibles d’amener un biais dans notre modèle.

Une des premières étapes a été de traiter les corrélations entre features via une matrice de corrélation en éliminant toutes les corrélations supérieures à 0,95 et inférieures à -0,95, ce qui a permis de sortir 34 features du pour la construction du modèle.

La deuxième était de régler le problème d’équilibre dans nos classes, le rapport étant de 92% d’un côté et seulement 8% de l’autre. En se basant sur la classe la plus faible on diminue la classe la plus importante pour qu’elle est la même taille via une fonction aléatoire.

La troisième étape a été de séparer le jeu de donnée en un jeu d’entraînement représentant 77% des données et un jeu de test représentant les 33% restant.

Ensuite il a fallu standardiser les données quantitatives, les données catégorielles avaient déjà été préalablement encodées.

* 1. Modélisation

Pour la modélisation, nous étions sur une problématique de classification supervisé, l’un des modèles les plus performants et rapide en ce moment est le LightGBMclassifier, j’ai donc décidé de l’utiliser sur cette problématique, il est intéressant car il une grande variété d’hyperparamètres.

Les premiers tests sont acceptables, un score ROC-AUC approchant les 70%, cependant il arrive plus facilement à trouver la classe des clients solvable que ceux non solvables. Même si les résultats restent corrects, le contexte de ce projet donne un risque plus important à mal classifier un mauvais payeur, qu’un bon payeur. Donc ici il va falloir trouver un moyen pour que la métrique utiliser donne une part plus importante à la classification des mauvais payeurs même si le score côté bon payeur en pâtit.

1. Optimisation du modèle
   1. La problématique projet

Comme expliqué dans la partie précédent le but ici va être de construire une métrique donnant une importance plus importante à la classification des mauvais payeurs. Pour cela j’ai sélectionné deux métriques : ROC-AUC en introduisant une notion de coefficient pour donner un poids plus important à la partie Recall, FBeta ou F1score où on joue sur Beta pour donner un poids plus important au Recall que la précision.

Dans les faits après différents test les deux métriques donnent les mêmes résultats étant très proche dans leur construction, je suis donc resté sur le premier ROC-AUC.

* 1. L’optimisation des hyperparamètres

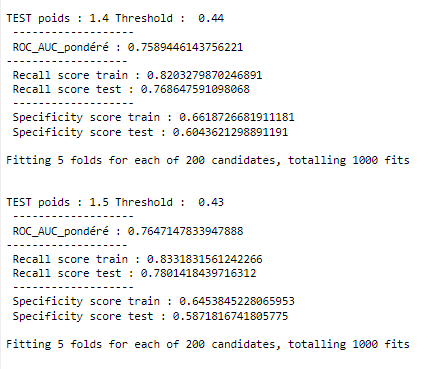
Pour optimiser les hyperparamètre j’ai décidé d’utiliser la méthode de l’optimisation Bayésienne qui permet de définir des zones de recherche restreintes pour les hyperparamètres afin d’effectuer une futur cross-validation via une grille de recherche.

Exemple des zones de recherches définies par mon optimisation :



* 1. La cross-validation

Suite à la définition d’espace de recherche pour les hyperparamètres, on entre les zones dans une gridsearch qui va finir d’optimiser le modèle et permettre les prédictions. Elle permit après plusieurs utilisations d’optimiser le poids choisi pour favoriser la classification des clients non solvables et le seul de prédiction optimal.



1. Interprétabilité du modèle

Ayant choisi un seuil de 0,44, si le client a une probabilité supérieure à 0,44 d’être dans la classe 1 il est considéré comme non solvable.

1. Limites et améliorations

Comme tout modèle il est sujet à l’obsolescence sur la durée, plus on avancera dans le temps et plus ses prédictions risquent de se dégrader. Le modèle n’est pas parfait, on ne prédit que 60% de personnes solvable, c’est plus une aide à la décision, qu’un outil dans lequel il faut avoir une foi aveugle. On peut aussi citer le fait que nous sommes dans un contexte financier où l’incertitude règne, un client peut perdre sont emploi du jour au lendemain sans signes précurseurs et ça le modèle pourra difficilement le prédire.

Pour les améliorations, on pourrait chercher de nouvelles variables, tester d’autres modèle comme le XGBoost, avoir des explications sur la variable ayant le plus de poids NEW\_EXT\_MEAN qui est la moyenne des trois features : EXT\_1, EXT\_2, EXT\_3. Sur lesquelles on ne sait rien sur les composantes de sa création.