Note méthodologique :

- ## La méthodologie d'entraînement du modèle

Dans le fichier, nous avons deux tableaux, un tableau d'entraînement et un tableau de test. La variable a entraîné est la colonne nommée (« TARGET »). La colonne cible qui indique si un client est à risque ou pas.

Cette colonne est présente dans le fichier entraînement et il n'est pas dans le fichier à tester. Il faudra donc créer cette colonne.

Comme nous avons des données déséquilibrées : (90 % des clients ne sont pas à risque et les 10 % restants le sont.) Pour avoir un modèle qui permet de détecter les personnes à risque, il faudra rééquilibrer nos données et pour cela, on utilise la méthode SMOTE.

Ensuite, on va effectuer deux méthodes et utiliser plusieurs méthodes de régression.

Le premier :

On utilise un « Pipeline » et la méthode du « GridSearch ». On vérifie nos résultats avec la méthode ROC, notre résultat est de 0.76. On fait ce calcul deux fois, un avec plusieurs paramètres et un avec les paramètres précis.

Puis on teste les modèles de régression DummyClassifier et RandomForest.

Le second :

On utilise la méthode de « Cross Validation » et on teste plusieurs modèles de régression : LGBM Classifier, XB Boost et Ada.

On affiche le résultat des scores ROC-Accuracy sur un graphique, à part Dummy Classifier, les résultats sont proches de 1, et le RandomForest est de 1, cette méthode sera écartée par précaution. On va donc utiliser LGBM Classifier, car c'est le meilleur résultat qu'on est, la note est de 0.97.

- ## La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation

Après, avoir appliqué notre modèle sur notre jeu de données, « fit » sur les valeurs à entraîner et « predict » sur les valeurs à tester, on va se pencher sur les matrices de confusion.

Comme nous n'avons pas de valeurs de TARGET pour les valeurs de test, on ne peut pas comparer nos résultats et nous allons le faire sur les valeurs à entraîner. On construit notre

valeur de prédiction et on le compare avec nos valeurs de test et on obtient notre matrice de confusion.

Avec ce tableau, on peut calculer notre coût métier.

Les faux négatifs coûtent 10 fois plus cher que les faux positifs, donc on rajoute un poids de 10. On obtient un résultat de 0.70. Ce qui reste un bon résultat, car il reste proche de 1. Le calcul revient à $\frac{TP+TN}{TP+TN+10*FN+FP}$

On calcule d'autres résultats :

ROC est de 0.98

F1_score est de 0.95.

- L'interprétabilité globale et locale du modèle

On peut remarquer que les informations les plus importantes sont le crédit, la rente, l'argent, des notes internes et aussi l'horaire du client.

Et pour le local, les informations sont proches et d'autres informations supplémentaires, le genre, le nombre de jour de travail et les retards de paiement.

- Les limites et les améliorations possibles

La prédiction de l'API est longue, il faudrait reconstruire des fonctions, non seulement pour rendre lisibles et aussi pour éviter d'avoir le code qui fait tout le temps à même chose.

Les deux derniers graphiques prennent beaucoup de temps à charger, il recalcule à chaque changement de client. L'idéal est de retravailler les fonctions pour qu'il n'y ait pas autant de calcul à chaque fois.

Il faudrait construire des fonctions qui seraient dans des fichiers différents, il y aurait un fichier sur la prédiction, un fichier sur les informations importantes, un fichier sur les informations personnelles du client.

Il faut donc se concentrer sur l'optimisation et la lisibilité.

- Github :

https://github.com/CecileLecoublet/Projet7_LFS