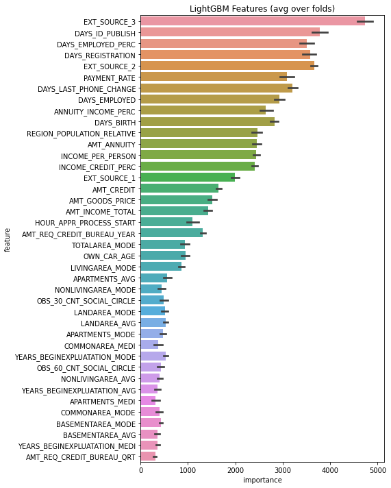
Note méthodologique

Méthodologie de choix et d’entrainement du modèle

Réduction dimentionnelle :

Entrainer un algorithme de machine learning avec plus de 700 features étant compliqué, tant sur l’aspect de la forte présence de features qui peut ralentir fortement l’algorithme mais aussi sur l’aspect du risque de surapprentissage du modèle, il était donc nécessaire de réduire le nombre de features et ne garder que l’essentiel. Ainsi, par le biais d’une LightGBM, le nombre de features a pu descendre de 768 à 40. Le travail effectué par la LightGBM était de faire un premier essai de prédiction et de déterminer quelles étaient les features les plus importantes dans la prédiction. On a ainsi pu obtenir le graphique ci-dessous :



Après cela, il a fallu vérifier que les variables/features sont indépendantes les unes des autres. Pour cela une analyse par tableau de corrélation a été réalisée. Cela a ainsi permis de réduire de nouveau le nombre de features, le faisant ainsi passer de 40 à 30.

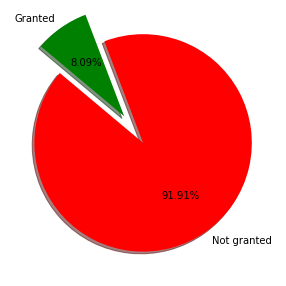
Préparation de la donnée avant sélection du modèle :

Après cela, une exploration des données était importante afin de déterminer la « fréquence » d’absence de données. Le résultat de cette petite observation à prouvé que l’absence de données pour certains clients était importante et donc, afin de faire fonctionner les méthodes de machine learning par la suite, il a fallu combler les vides. La méthode retenue pour cela était le KNNInputer de la librairie sklearn. Cette fonction permettait de combler les vides en recherchant les « k » valeurs les plus « proches » du client à qui des données sont absentes et remplir la donnée par une valeur représentant la moyenne des k voisins.

Sélection du modèle :

Resampling :

Lors de l’observation des données, une large disparité a pu être observée entre le nombre de clients dont le prêt leur a été accordé (valeur 1) et le nombre de client dont le prêt ne leur a pas été accordé (valeur 2) comme sur l’image ci-dessous :



Ainsi, un resampling est nécessaire pour que les algorithmes aient le plus de données fiables. 3 méthodes sont possibles pour cela : undersampling, oversampling et SMOTE.

Ces 3 méthodes équilibrent le nombre de clients ayant des valeurs égales à 1 et égales à 2. Ainsi, si 5000 clients avaient 0 en valeur « target » et 1000 avaient 1 en valeur « target » alors après application de chacune de ces 3 méthodes, il y aura autant de clients avec une valeur « target » à 0 et autant avec une valeur à 1. Cependant, lors de l’application de l’undersampling, il y a une perte d’information du côté des valeurs « target » égales à 0 car l’algorithme va équilibrer en se basant sur les 1000 valeurs à 1. Inversement, lors de l’application de l’oversampling, des données seront dupliquées du côté des clients avec des valeurs « target » égales à 1. Résultat, après oversampling, il y aura 5000 clients à 0 et 5000 à 1.

Finalement, la dernière méthode de resampling étudiée est le SMOTE qui consiste, à l’image de l’oversampling, à équilibrer les clients à 1 avec les clients à 0. Cependant, à la différence de l’oversampling, l’algorithme ne duplique pas simplement les données mais en créé de nouvelles en se basant sur les données des clients à 1. Pour donner une image de ce à quoi ça correspond, on peut considérer un segment dont chacune des extrémités correspond à un client ayant pour valeur « target » 1, un point sera créé au milieu de ce segment, cette nouvelle donnée sera donc ajoutée à l’ensemble des données. Ce processus est donc répété jusqu’à ce qu’il y ait autant de 0 que de 1.

Méthodologie d’entrainement du modèle :

Sélection du modèle :

Après avoir préparé les données, plusieurs modèles ont été préselectionnés. Ces modèles sont le light gradient boosting (LGBM), le K-nearest neighbour (KNN), le random forest classifier (RFC), SVM classifier (SVC), SGD classifier (SGD) et la régression logistique (LReg pour Logistic Regression).

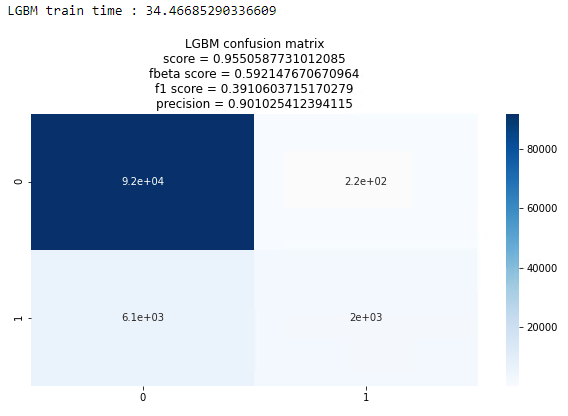
La sélection du modèle final parmi ces 6 modèles s’est faite en 3 grandes étapes :

* Préprocessing avec undersampling puis test des 6 modèles
* Préprocessing avec oversampling puis test des 6 modèles
* Préprocessing avec SMOTE puis test des 6 modèles

Dans chacune de ces étapes, le test des différents algorithmes se fait sur une partie du jeu de donnée d’entrée et se déroule comme suit :

* Séparation des données en set d’entrainement et de test.
* Utilisation de la fonction GridSearchCV pour optimiser chacun des algorithmes

On obtient donc, pour chaque algorithme de machine learning testé, une sortie comme la suivante :



Chaque algorithme de machine learning et son paramétrage ont été stockés dans des variables afin de pouvoir directement exploiter l’algorithme sélectionné avec son paramétrage optimal.

Le choix de l’algorithme final s’est donc fait en se basant sur les valeurs de « score » obtenus par chacun des algorithmes entrainés.

Les métriques d’évaluation :

Il est possible de constater 3 autres métriques en dessous du score dans l’image précédente. En effet il s’agit là des principales métriques utilisées afin de comparer les modèles entre eux ainsi que pour la validation du modèle final.

Ces 3 métriques sont le « fbeta\_score », « f1\_score » et « précision ». Elles sont calculées en comparant les résultats théoriques obtenues avec le set de test ainsi que les résultats réels déjà existants.

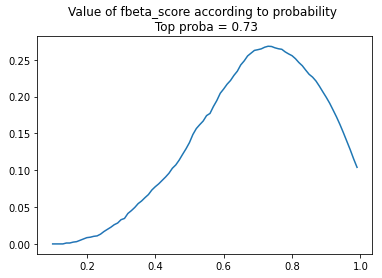
La précision permet de connaitre si le modèle à obtenu un bon ratio de vrai positifs par rapport à la somme des vrai et faux positifs. Avec cela on peut évaluer les potentielles pertes et l’objectif sera donc de maximiser cette valeur de précision pour réduire au maximum les pertes. Il ne faut pas non plus oublier les faux négatifs, représentant des clients pour lesquels le prêt aurait pu leur être accordé. Ces clients représentent ont un « non gain ». C’est donc les valeurs de f1\_score et de fbeta\_score qui permettront de les ajouter à l’analyse.

Commençons par le f1\_score ou F-measure en anglais. Le f1\_score nous permet d’évaluer la performance de notre modèle dans le cas où la précision et le rappel ont le même poids, ou, par extension, où les faux positifs et les faux négatifs auraient le même poids dans l’analyse. L’objectif sera donc, encore une fois, de maximiser cette valeur.

Finalement, le fbeta\_score fait un travail similaire au f1\_score cependant, là où le f1\_score considère que la précision et le rappel ont le même poids, le fbeta\_score va mettre plus de poids sur la précision ou le rappel en fonction de la valeur que l’on donne à « beta ». Ainsi, pour une valeur de beta égale à 0.5, valeur choisie pour l’analyse des résultats des modèles, la précision aura un poids 2 fois plus important que le rappel.

Le choix de donner plus d’importance à la précision qu’au rappel vient du fait qu’il est plus risqué, pour une banque, d’accorder un prêt à une personne ne pouvant rembourser (Faux positif) que de ne pas accorder de prêt à une personne pouvant rembourse (Faux négatif). En effet, le cas du faux positif correspond à une perte tandis que le cas du faux négatif correspond à l’absence de gains.

Un dernier point à mentionner est que l’optimisation de l’algorithme de machine learning n’est pas la seule optimisation possible. En effet, plutôt que de directement prédire les valeurs « target », il est possible de demander à l’algorithme de nous donner la probabilité d’obtenir la valeur 0 et celle d’obtenir la valeur 1. Par défaut, quand la probabilité d’obtenir 0 est supérieure à 50%, l’algorithme va nous renvoyer la valeur 0. Ainsi, l’objectif va être de tester plusieurs valeurs de probabilité (50% par défaut) afin de trouver la probabilité pour laquelle le fbeta\_score est le plus élevé. On peut ainsi obtenir le graphique ci-dessous :



En observant le graphique ci-dessus, on peut conclure que la valeur de probabilité minimale d’obtenir 0 pour laquelle on conclut qu’il ne faille pas accorder de prêt au client est d’approximativement 73%.

Interprétabilité globale et locale :