Voici un processus complet du Machine Learning supervisé désigné.

1. Prétraitement des données
2. Sélection des variables
3. Entrainement du model
4. Optimisation des hyperparamètres

Pour notre contexte, nous avons choisi de réaliser des prédictions afin de :

* Savoir si un bénéficiaire bénéficie de la prise en charge à 100 % [ FLT\_DEP\_MNT, FLT\_PAI\_MNT, FLT\_DEP\_MNT ]
* Prédire les montants remboursés et payés [ PRS\_REM\_MNT >= PRS\_PAI\_MNT ]
* Anticiper les dépassements d’honoraires [ PRS\_DEP\_MNT > 0 ]

## Prétraitement des données

**Filtrage des colonnes** : Dans les données d'origine, une proportion importante des valeurs est inutilisable, notamment les valeurs inconnues ou celles sans objet. Lorsque le pourcentage de ces valeurs inutilisables dans une colonne est trop élevé, il est nécessaire de supprimer la colonne concernée. À cette fin, j'ai appliqué un seuil de 0,2 pour identifier et retenir uniquement les colonnes exploitables.

**Imputer** : J’ai ensuite remplacé manuellement, au niveau de la base de données, les valeurs inutilisables ainsi que les valeurs manquantes par la modalité **la plus fréquente** de chaque colonne. J’ai opté pour une imputation manuelle plutôt que d’utiliser la bibliothèque « sklearn.impute.SimpleImputer » en raison de la manière dont j’ai traité les données. En effet, la mémoire de mon ordinateur ne me permet pas de charger l’ensemble des données d’un mois ; j’ai donc dû me limiter à un échantillon de 10 %. L’application du SimpleImputer sur seulement 10 % des données, puis l’utilisation du même imputer sur la totalité du jeu de données, pourrait introduire un biais. Ainsi, afin de limiter la consommation de mémoire et d’éviter ce risque de biais, j’ai préféré réaliser l’imputation manuellement.

## Sélection des variables

J'ai utilisé l'algorithme **Boruta** afin d'identifier les variables les plus importantes. Dans un souci d'efficacité, j'ai procédé à un échantillonnage aléatoire de 100 000 lignes issues de l'ensemble du jeu de données, puis j'ai répété l’opération cinq fois afin d’assurer la stabilité des résultats. Parmi les trois itérations réalisées, deux n’ont permis de sélectionner aucune variable pertinente selon le test Boruta, tandis que la troisième n’a retenu qu’une seule variable comme significative.

Étant donné que l’ensemble des variables est de nature catégorielle, il n’est pas possible d’appliquer une PCA pour réduire la dimensionnalité. J’ai donc conservé le troisième processus ainsi que la seule variable sélectionnée.

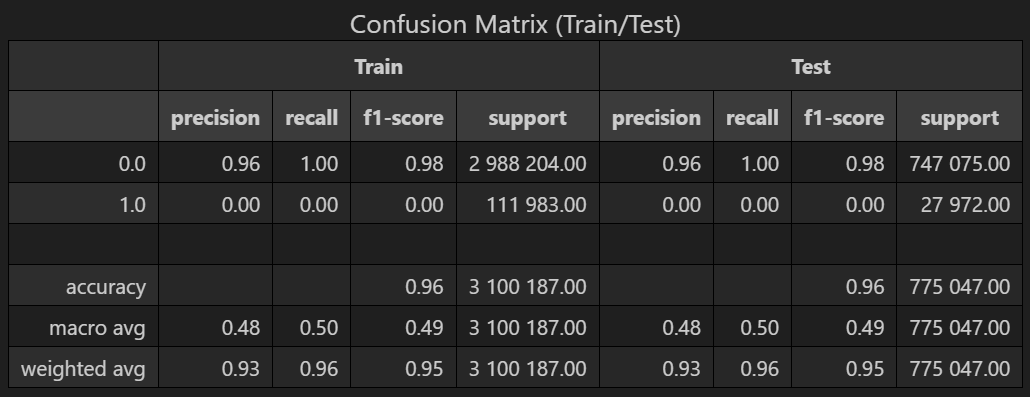
## Entrainement du model

**Choix du modèle** : J’ai utilisé le **LGBMClassifier** pour plusieurs raisons. Premièrement, le volume des données étant très important, l’algorithme LightGBM est particulièrement bien adapté au traitement de grands ensembles de données. Deuxièmement, toutes les variables étant catégorielles, LightGBM prend en charge nativement ce type de variables, sans nécessiter d’encodage supplémentaire. Troisièmement, LightGBM s’intègre parfaitement à l’API de scikit-learn, ce qui facilite son utilisation dans le cadre de mon projet.

## Optimisation des hyperparamètres

J’ai utilisé **RandomizedSearchCV** pour l’optimisation des hyperparamètres, car cette méthode est plus rapide et, dans de nombreux cas, plus performante que GridSearchCV.

## Interprétation du résultat



Une image contenant texte, capture d’écran, Police, Rectangle

Le contenu généré par l’IA peut être incorrect.

Le modèle n’effectue aucune prédiction sur la classe 1, ce qui s’explique par un déséquilibre important au niveau de la variable cible. Dans l’ensemble, les données disponibles ne sont pas idéales pour l’entraînement d’un modèle de machine learning.