OPENCLASSROOMS

FORMATION DATA SCIENTIST


# NOTE METHODOLOGOGIQUE

## Implémentez un modèle de scoring
## Projet 7

**Samir HINOJOSA DIAZGRANADOS**

Février 2022

# Table des matières

1. Introduction ..... 3
2. Présentation des données ..... 3
3. Modélisation ..... 4
   3.1. Sélection du Kernel ..... 4
   3.2. Optimisation du modèle ..... 4
      3.2.1. Traitements initiaux ..... 5
      3.2.2. Données déséquilibrées ..... 5
      3.2.3. Fonction coût ..... 6
      3.2.4. Hyperopt ..... 6
4. Interprétation du modèle ..... 7
5. Limites et améliorations possibles ..... 8

# 1. Introduction

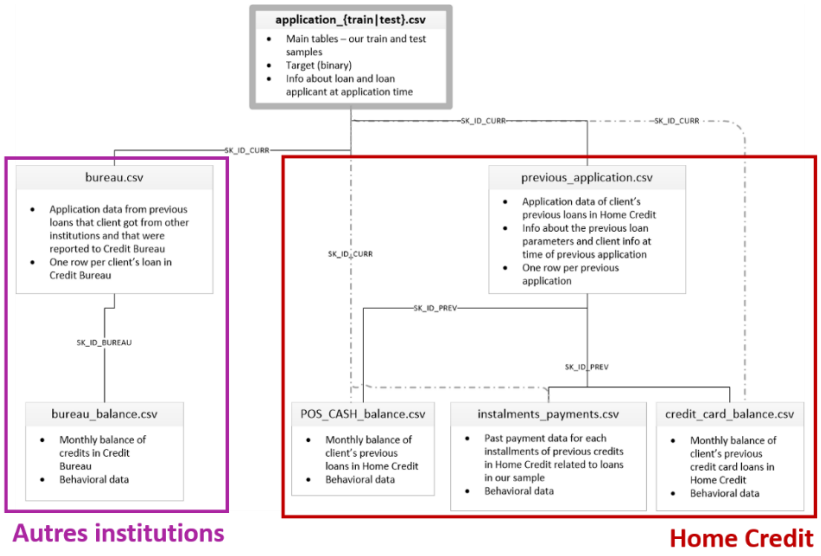
La société « **Prêt à dépenser** » souhaite mettre en œuvre un outil de « **scoring crédit** » pour calculer la probabilité qu'un client rembourse son crédit d'après diverses données.


La mission est de développer un algorithme qui classifie la demande en crédit accepté ou rejeté.

Prendre en compte :

- Sélectionner un kernel Kaggle pour faciliter la préparation des données nécessaires à l'élaboration du modèle de scoring.
- Déployer le modèle de scoring comme une API.
- Construire un tableau de bord interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle de scoring

# 2. Présentation des données




Sept (7) tables de données sont mises à disposition pour la démarche.

Il y a de tables de données qui viennent des autres institutions :

- Bureau : Antécénts de crédit du client (autres institutions financières)
- Bureau_balance : Soldes mensuelles de crédits précédents

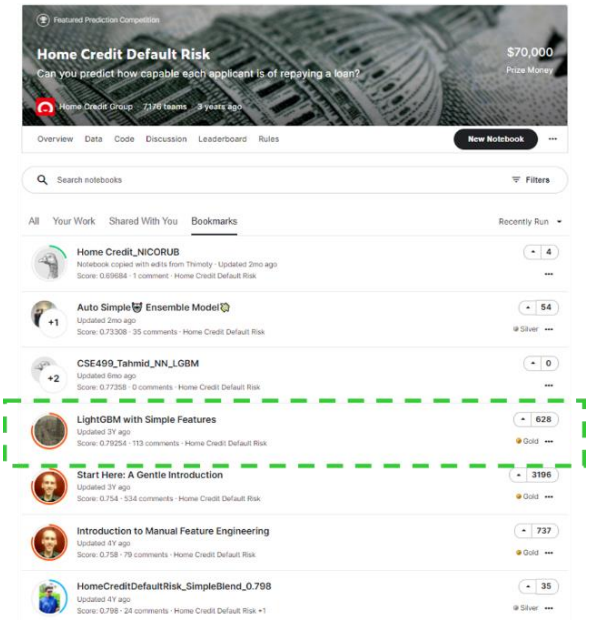
Aussi, on peut trouver de tables de données propre de « **Prêt à dépenser** »

- Previos_application: Demandes de crédit antérieures
- Pos_cash_balance: Des bilans mensuels des anciens points de vente et des prêts cash
- Installements_payments: Historique de remboursement des crédits précédents
- Credit_card_balance: Solde mensuel des cartes de crédit antérieures

# 3. Modélisation

## 3.1. Sélection du Kernel

Le kernel utilisé pour le projet est [LightGBM with Simple Features].




La sélection est basée sur

- Un bon score dans la compétition
- Un Feature Engineering performant
- L'opportunité d'utiliser un Framework basé sur le Gradient Boosting
- Le kernel recommandé dans le projet

Tout au long du kernel, des traitements sont faits pour obtenir des nouvelles données

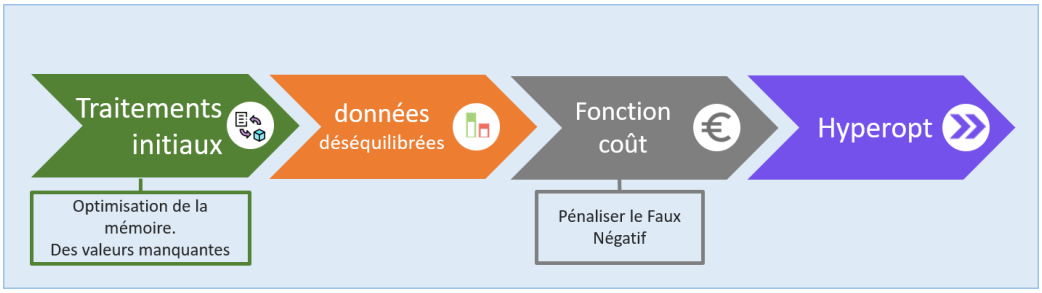
- Identification et traitement des variables catégorielles

  Création de nouvelles variables
  DAYS_EMP_% = DAYS_EMP / DAYS_BIRTH
  Min, Max, Mean, Sum, Var

- Unification de jeux des données
  +700 variables en total

- Modélisation avec LGBMClassifier


## 3.2. Optimisation du modèle




Pour faire l'optimisation du modèle, on va prendre en compte le Feature Engineering déjà fait dans le kernel choisi.

### 3.2.1. Traitements initiaux

Tout d'abord, il faut faire une optimisation d'usage de la mémoire parce que l'utilisation actuelle est 2,1 GB. Alors, on va transformer les types de colonnes

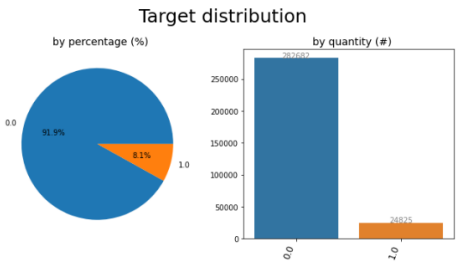
- ```
  if df[col].dtype == "int64" and df[col].nunique() == 2:
          df[col] = df[col].astype("int8")
  ```

- ```
  if df[col].dtype == "float64" and df[col].min() >= -2147483648 and
     df[col].max() <= 2147483648:
          df[col] = df[col].astype("float32")
  ```

À travers cette transformation, l'usage de la mémoire est passé de 2.1Gb à 941 Mb.

D'ailleurs, on a fait aussi, ***le traitement des valeurs manquantes*** à travers l'imputation de la moyenne basée sur la colonne.

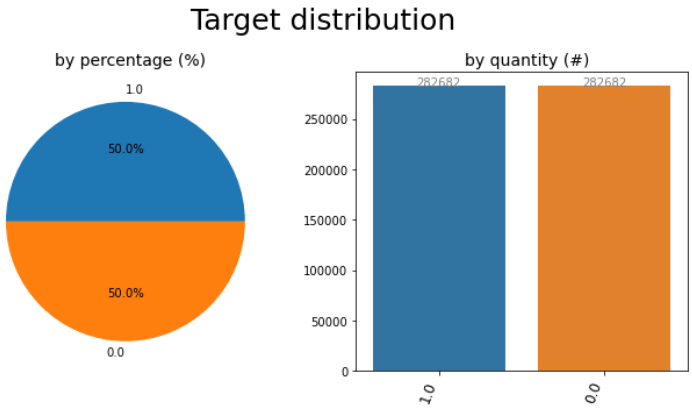
### 3.2.2. Données déséquilibrées




L'analyse exploratoire a montré que la mission a un problème de classification déséquilibrée.

Une modification de l'ensemble des données est possible avant d'entraîner le modèle prédictif afin d'équilibrer les données (égaliser les clases).

Dans ce cas, on va utiliser une stratégie appelée rééchantillonnage (re-sampling), en se concentrant sur la méthode du *sur-échantillonnage (Oversampling) avec SMOTE*[1] *(Synthetic Minority Oversampling Technique).*



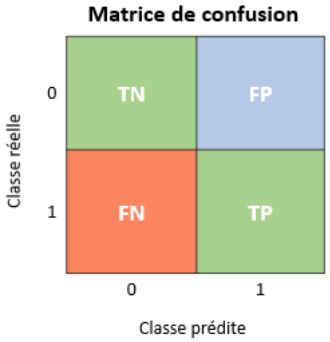

(0) Ce sont des prêts qui ont été remboursés et (1) Ce sont des prêts qui n'ont pas été remboursés.

[1] SMOTE (Synthetic Minority Oversampling Technique) : Consiste à synthétiser des éléments pour la classe minoritaire, à partir de ceux qui existent déjà en choisissant aléatoirement un point de la classe minoritaire et à calculer les k plus proches voisins de ce point.

### 3.2.3. Fonction coût

Pour ce projet, une fonction coût a été développée dans l'objectif de pénaliser des Faux Négatifs.

- **TN (Vrais Négatifs)** : des prêts qui ne sont pas en défaut et ont été prédits correctement
- **TP (Vrais Positifs) :** des prêts qui sont en défaut et ont été prédits correctement
- **FP (Faux Positif)** : des prêts qui ne sont pas en défaut et ont été prédits de manière incorrecte
- **FN (Faux Négatif)** : des prêts qui sont en défaut et ont été prédits de manière incorrecte




Un Faux Positif (FP) constitue une perte d'opportunité pour la banque, à la différence d'un Faux Négatif (FN) qui constitue une perte pour créance irrécouvrable.

| Taux | Valeur |
|---|---|
| TN (vrais négatifs) | 1 |
| TP (vrais positifs) | 1 |
| FP (faux positifs) | 0 |
| FN (faux négatifs) | -10 |

Ces valeurs de coefficients signifient que les Faux Négatif (FN) engendrent des pertes 10 fois supérieures aux autres.

Ces poids ont été fixés arbitrairement et il est tout à fait envisageable de les modifier en fonction de l'optique métier.

```
# Total of default and not default cases
total_not_default = TN + FP      # Not default cases
total_default = TP + FN          # Default cases

# Calculating the gains based on the rate/values
gain_total = TN*TN_rate + TP*TP_rate + FP*FP_rate + FN*FN_rate
gain_maximun = total_not_default*TN_rate + total_default*TP_rate
gain_minumun = total_not_default*TN_rate + total_default*FN_rate

# Normalize to get score between 0 (baseline) and 1
score = (gain_total - gain_minumun) / (gain_maximun - gain_minumun)
```

### 3.2.4. Hyperopt

Pour réaliser l'optimisation du modèle, on a pris en compte l'outil HyperOpt, conçu pour automatiser la recherche d'une configuration optimale d'hyperparamètres basée sur une optimisation bayésienne et prise en charge par la méthodologie SMBO (optimisation globale de

modèles séquentiels). Les hyperparamètres seront basés sur les paramètres déjà existants dans le kernel choisi.

Il est nécessaire de dire que le kernel choisi a déjà un cross-validation mis en œuvre. C'est pour cela que dans cette partie, on va faire attention à l'hyperparametrisation du modèle.

```
N_ESTIMATORS = [8000, 10000, 12000]
NUM_LEAVES = [32, 34, 36]
MAX_DEPTH = [7, 8, 9]

space_params = {
    "n_estimators": hp.choice("n_estimators", N_ESTIMATORS),
    "learning_rate":  hp.uniform("learning_rate", 0.002, 0.003),
    "num_leaves":  hp.choice("num_leaves", NUM_LEAVES),
    "max_depth": hp.choice("max_depth", MAX_DEPTH),
    "reg_alpha": hp.uniform("reg_alpha", 0.041545473, 0.051),
    "reg_lambda": hp.uniform("reg_lambda", 0.0735294, 0.0835294),
    "min_split_gain": hp.uniform("min_split_gain", 0.0222415, 0.0322415),
    "min_child_weight": hp.uniform("min_child_weight", 39.3259775, 49)
}
```

D'ailleurs, pour optimiser la performance et le temps, on a pris en compte, les points suivants :

- Mettre en œuvre le StandardScaler()

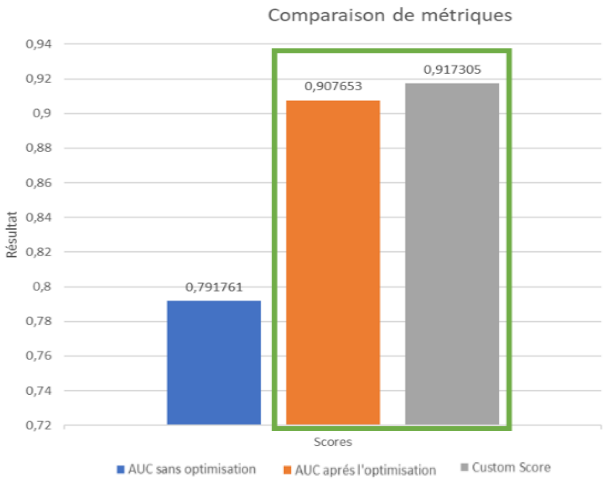
- Définir les paramètres suivants avant de lancer la modélisation (pour optimiser les temps d'exécution et pour dire au modèle que le jeu de données est déjà équilibré.)
  LGBMClassifier
    colsample_bytree=0.8
    subsample=0.8
    Is_unbalance=False

# 4. Interprétation du modèle

À travers l'optimisation du modèle mis en œuvre, on a réussi à avoir de meilleurs scores que précédemment.

Voici les meilleurs paramètres du modèle

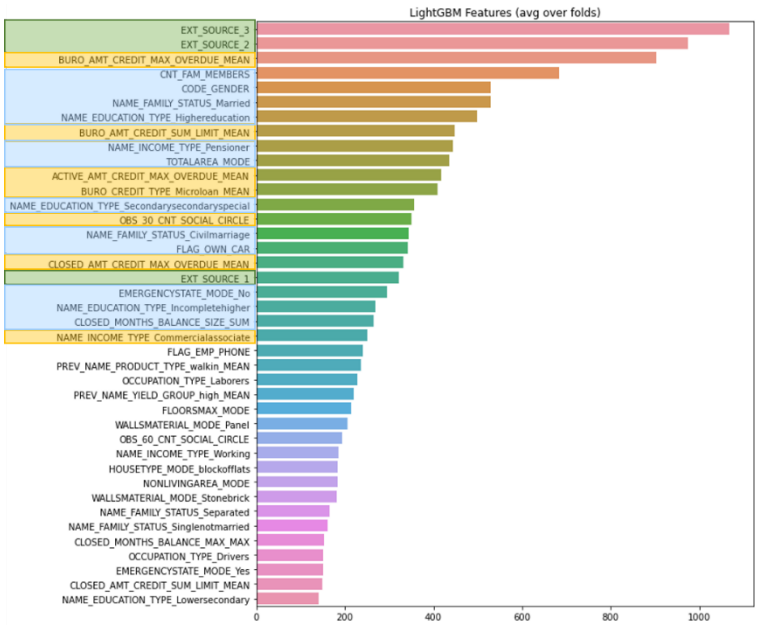
| Paramètres | Meilleur résultat |
|---|---|
| learning_rate | 0.002021947556803579 |
| max_depth | 9 |
| min_child_weight | 44.68618422455195 |
| min_split_gain | 0.030970825122649367 |
| n_estimators | 8000 |
| num_leaves | 36 |
| reg_alpha | 0.045341569610647205 |
| reg_lambda | 0.08049459639521307 |



- Les variables les plus importantes proviennent d'une source **externe.** Ce sont des scores normalisés provenant d'autres institutions.

- Le Feature Engineering a ajouté la valeur au moment de la modélisation. On peut le remarquer au travers des terminaisons de noms de Features (mean, max, etc.).

- Le modèle a pris en compte les différentes variables ***Variables personnelles***, ***Variables bancaires***, ***Variables externes.***




## 5. Limites et améliorations possibles

- Il faut faire une analyse exploratoire au début pour bien comprendre les données.

- Il est nécessaire de faire une réduction de données pour éviter le « Curse of Dimensionality ». Il faut prendre en compte une méthode comme Feature Selection par Scikit-Learn

- Une analyse du composant principal « PCA » peut apporter des avantages

- Aller plus loin dans l'hyperparametrisation du modèle