OPENCLASSROOMS

FORMATION DATA SCIENTIST

NOTE METHODOLOGOGIQUE

Implémentez un modèle de scoring
Projet 7

Samir HINOJOSA DIAZGRANADOS

Février 2022

Table des matières

1. Introduction

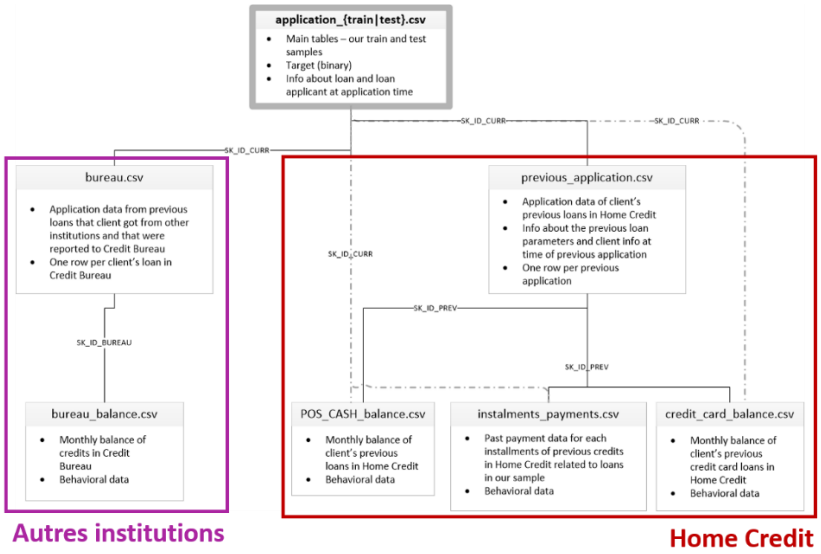
La société « **Prêt à dépenser** » souhaite mettre en œuvre un outil de « **scoring crédit** » pour calculer la probabilité qu'un client rembourse son crédit d'après diverses données.

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

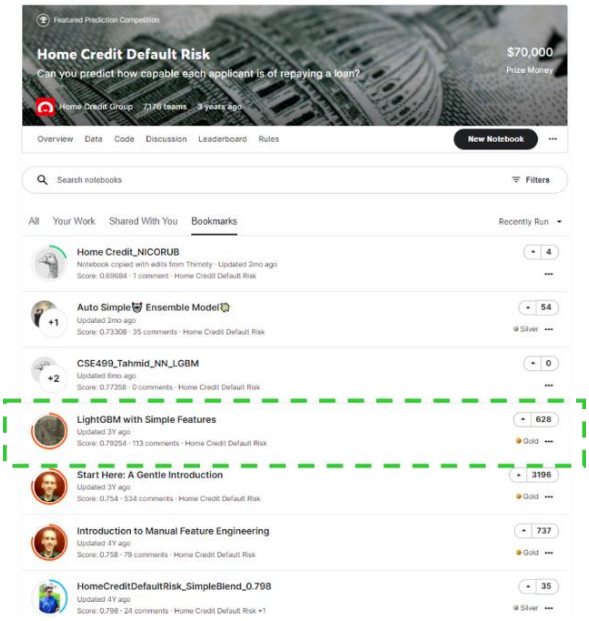
Aussi, on peut trouver de tables de données propre de « **Prêt à dépenser** »

- 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 pour la Feature Engineering

Le kernel utilisé pour le preprocessing est [LightGBM with Simple Features](#).



La sélection est basée sur

- Un bon score dans la compétition
- Un Feature Engineering performant
- Le kernel recommandé dans le projet

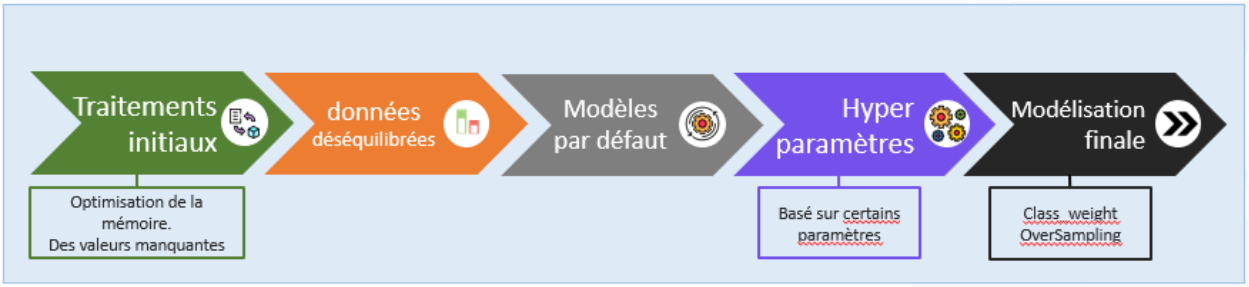
Tout au long du kernel, des traitements sont faits pour obtenir des nouvelles données

- 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

3.2. Optimisation du modèle



Pour faire la modélisation, 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")
```

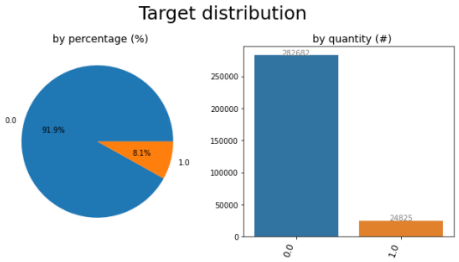
```
if df[col].dtype == "float64" and df[col].min() >= -2147483648 and
   df[col].max() <= 2147483648:
        df[col] = df[col].astype("float32")
```

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 :

- Suppression des colonnes avec 20 % ou plus de valeurs manquantes
- 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).

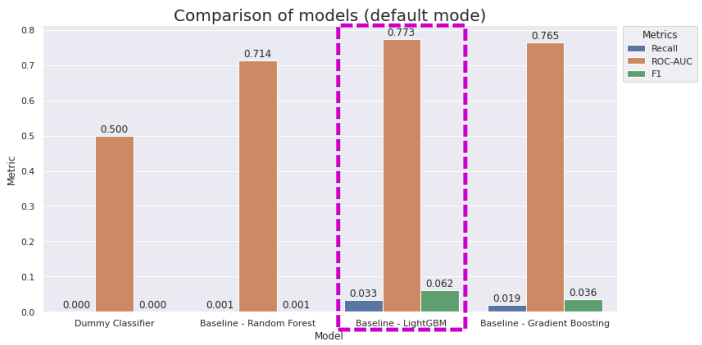
(0) Ce sont des prêts qui ont été remboursés et (1) Ce sont des prêts qui n'ont pas été remboursés

Dans ce cas, on va utiliser deux stratégiques :

- Une stratégie appelée rééchantillonnage (re-sampling), en se concentrant sur la méthode du *sur-échantillonnage (Oversampling) avec SMOTE*[1] *(Synthetic Minority Oversampling Technique).*

- Utilisation de class_weight comme paramètres pour faire une pondération inversement proportionnellement à la fréquence des classes

3.2.3. Modèles par défaut



Au moment de réaliser la modélisation en prenant en compte les valeurs par défaut, le LightGMB a obtenu le meilleur résultat.

[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.

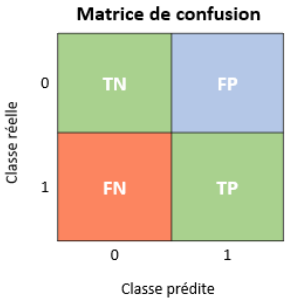
3.2.4. Modélisation finale

On va utiliser trois algorithmes (Random Forest, LightGBM et Gradient Boosting).

3.2.4.1. 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) :** 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)	-1
FN (faux négatifs)	-10

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

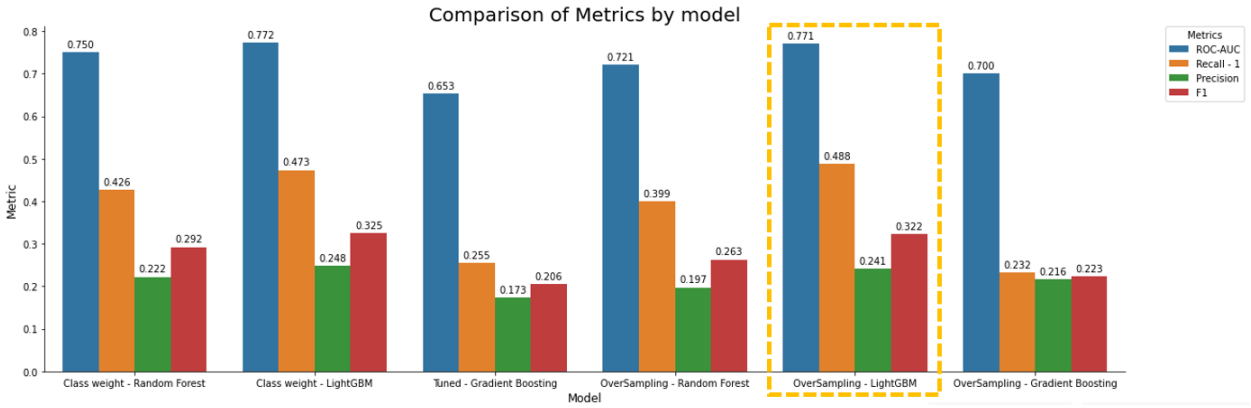
Ces poids ont été fixés arbitrairement et il est tout à fait envisageable de les modifier en fonction de l'optique métier.

```
# 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)
```

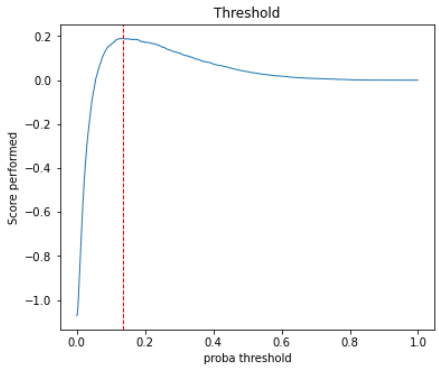
La fonction coût sera utilisé au moment de calculer le seuil

```
thresholds = np.arange(0, 1, 0.001)
scores = []

for threshold in thresholds:
    y_pred = (y_prob >= threshold).astype("int")
    score = custom_score(y_test, y_pred)
    scores.append(score)
```

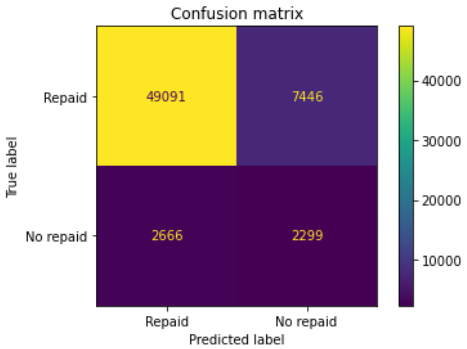
3.2.4.2. Résultats de modélisation

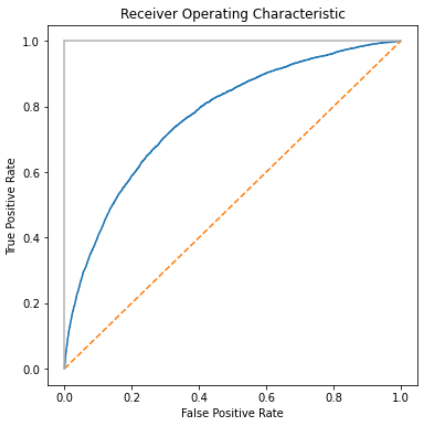


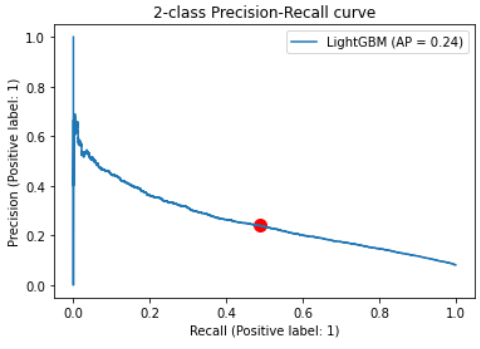
Voici, on peut noter que les résultats sont très similaires. À gauche ceux sont de modèles avec Class weight. À droite ce sont de modèles avec OverSampling. Il faut dire que le Gradient Boosting n'a pas le paramètres Class weight. Basé sur la métrique recall - 1, on a choisi le modèle OverSampling – LightGBM pour continuer.



Le seuil est bas, ça veut juste dire que le modèle a tendance à prédire trop de faux négatifs et positifs







Le modèle est acceptable avec un ROC-AUC de 0,771 et un Recall pour 1 de 0,488.

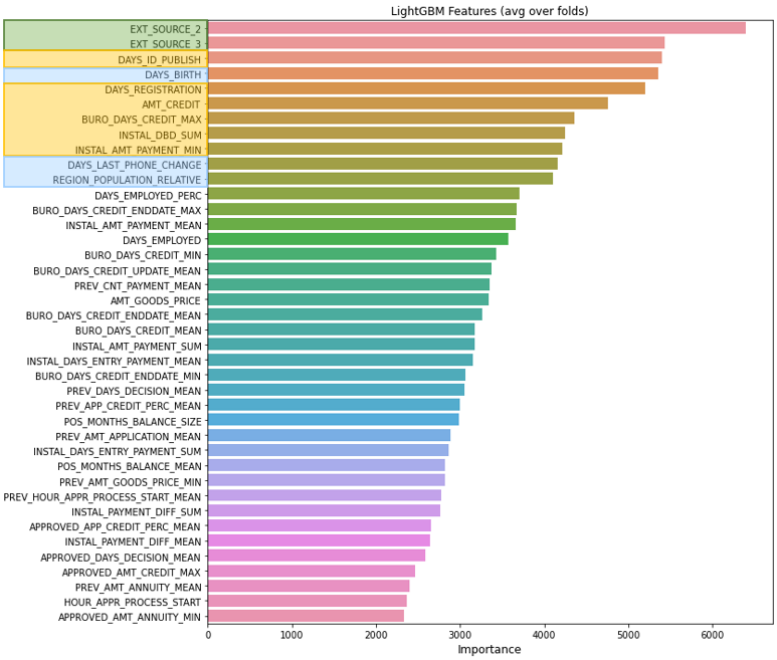
4. Interprétation du modèle

4.1. Interprétation global

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

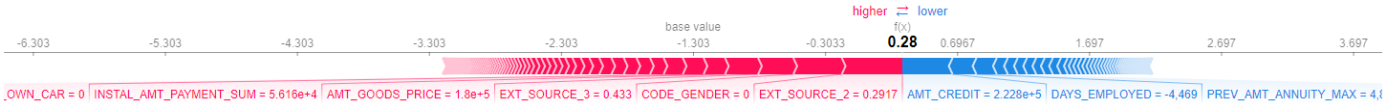
- Le Feature Engineering a ajouté la valeur au moment de la modélisation. On peut le remarquer au travers des terminaisons de noms de Features (mean, max, etc.).

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



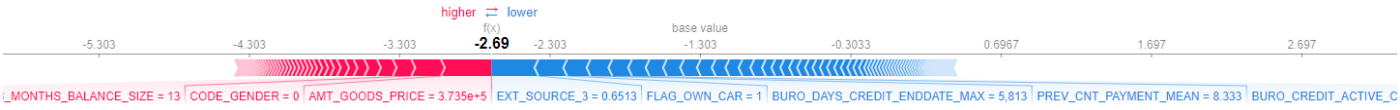
4.2. Interprétation locale

- Une observation qui est en défaut



Nous pouvons remarquer que ext_source_2, ext_source_3 et amt_goods_price ont une forte influence dans le résultat pour cette observation poussant la prédiction à droite.

- Une observation qui est en défaut



Dans ce cas, ext_source_3 appuie la prédiction à gauche

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 plusieurs méthodes de Feature Selection par Scikit-Learn, PCA, T-SNE, etc.
- Aller plus loin dans l'hyperparametrisation du modèle
- Essayer d'autres algorithmes comme OneClassSVM, etc.
- Essayer plusieurs coefficients dans le custom score pour avoir un modèle plus ou moins strict