Projet N°7 Openclassroom

Implémentez un modèle de Scoring

BOUAZIZ Gérard 27/10/2022

Sommaire

- Problématique
- Présentation des Données
- Analyse Exploratoire et Feature Engineering
- Modélisation
- Interprétation
- Mise en oeuvre Dashboard
- Conclusion

Problématique

Présentation

La société financière "Prêt à dépenser" propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

Objectifs :

- **Mettre en œuvre un outil de «crédit scoring»** qui calcule la probabilité de remboursement d'un crédit-client et classifie la demande en crédit accordé ou refusé.
- **Expliquer de façon la plus transparente possible les décisions d'octroi de crédit**, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

Missions

- **Développer un algorithme de classification** qui s'appuiera sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc…).
- D**évelopper un dashboard interactif** pour les chargés de relation client permettant d'interpréter les prédictions faites par le modèle et d'améliorer la connaissance client des chargés de relation client.

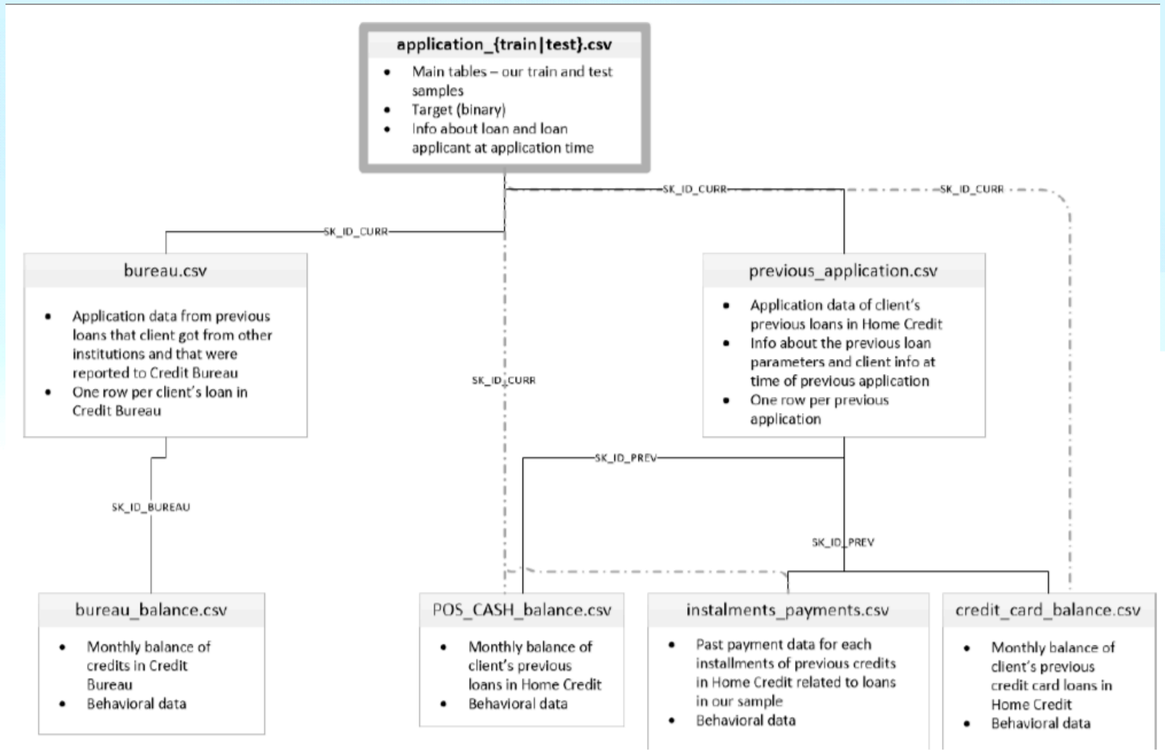
Présentation des Données

Un jeu de données quasi-relationnel issu de Kaggle

Il est composé de **8 tables** :
- **1 table cliente** principale de 307511 clients
- **7 tables historiques** de détail du crédit.

Un client est décrit par **122 features** :
- **1 Target** (1: client en défaut de paiement, 0: client sans défaut)
- **121 autres features**
  - âge, sexe, situation maritale
  - emploi, logement, revenus,
  - informations relatives au crédit,
  - notation externe, …

PS : un sous-ensemble de 48744 clients n'a pas de Target disponible. Il pourra servir comme groupe de clients Tests pour la prédiction



Data Model

Analyse Exploratoire et Feature Engineering (1)

Première Sélection des **Features pertinentes** avec un **Noyau** Kaggle «LightGBM with Simple Features»

Objectifs :

- **Nettoyer** : valeurs incohérentes mise à nulles ou supprimées.
- **Réduire la dimensionnalité** du jeu de données :
  - **Agréger par client** son historique du crédit pour compresser l'information utile avec les 5 opérations suivantes : moyenne, somme, variance, min et max.
  - **Synthétiser de nouvelles features** à partir de features existantes :

    ex : taux d'employabilité = nombre de jours en emploi / nombre de jours depuis la naissance

    pourcentage de revenu (par rapport au crédit) = revenu total / montant du crédit

    revenu par personne = revenu total / nombre de membre dans le ménage

    pourcentage d'annuité (par rapport au revenu) = annuité / revenu total

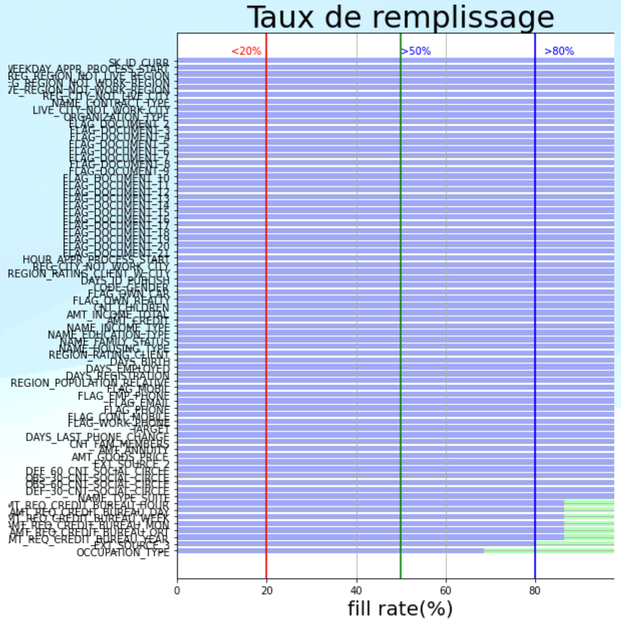
    taux de remboursement annuel = annuité / montant du crédit

    … …
- **Encoder les features catégoriques** (réprésentation des variables sous forme de vecteurs binaires) avec One-Hot Encoding.

► **Total Features après selection** par le Noyau : **751**

Analyse Exploratoire et Feature Engineering (2)

Seconde Sélection des Features pertinentes



- **Traitement des Valeurs Manquantes**

  On supprime toutes les features ayant plus de 45% de valeurs manquantes

  ► **237 Missing Features**

- **Imputation des Valeurs Manquantes**
  - Traitement des Features Numériques :

    imputation par la médiane (plus robuste que l'imputation par la moyenne car elle mitige l'effet des outliers)
  - Traitement des Features Catégoriques :

    imputation par le mode (la valeur la plus fréquente)

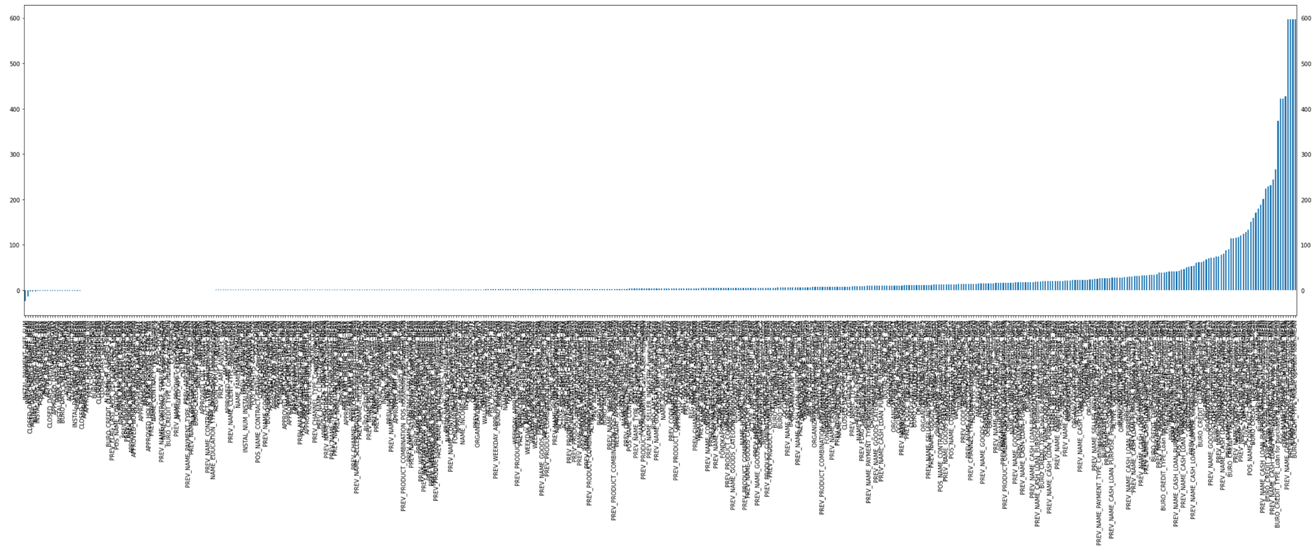
- **Suppression des features de variance nulle ou de grande variance par le Coefficient de Variation**

  CV = (Écart-type / Moyenne) * 100

  Si CV < 0.01 alors feature rejetée

  Si CV > = 0.01 et CV <= 30 alors feature acceptée

  => 74 Features Supprimées



► **Total Features après sélection** : 751 - 237 - 74 = **440**

Modélisation (1)

Jeu de données splitté en 3 parties

- **clients sans Target** : utilisés comme évaluation predictive finale de Test
- **clients avec Target :** utilisés pour l'entraînement du modèle
  - un jeu d'entraînement **Train avec 80% d'individus**
  - un jeu de **Test avec 20 % d'individus** pour l'évaluation finale du modèle

Problème d' « Imbalance Classification »

- **Classification binaire** avec 2 Targets :
  - **91%** clients **en défaut** de paiement
  - **9 %** clients **sans défaut** de paiement
- Les clients **en défaut** de paiement représentent la **classe minoritaire** en sous représentation
  - 2 Techniques pour **rééquilibrer** le jeu de données :
    - Les méthodes d'**Undersampling** fonctionnent en supprimant aléatoirement des points de la classe majoritaire
    - Les méthodes d'**Oversampling** consistent à créer des points artificiels de la classe **classe minoritaire** à partir des points existants

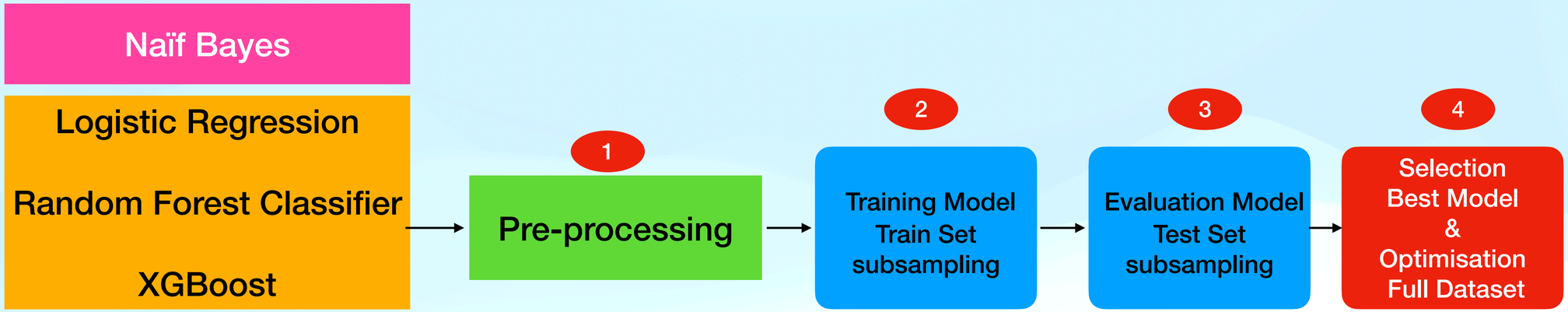
      SMOTE => répartition 50/50 (classe majoritaire / classe minoritaire) au final.

PS : Algorithme SMOTE :

- un individu A de la classe minoritaire tiré au hasard
- calcul de son k voisinage de la même classe 2/
- tirage aléatoire d'un voisin B
- un nouvel exemple synthétique est créé en sommant une distance aléatoire issu de la distance des 2 points au point A

Modélisation (2)

Méthodologie d'entraînement en 4 étapes



- Naïf Bayes utilisé comme modèle de référence
- Etapes :
  1. Pre-processing avec Standardisation des données (z = (x - u) / s)
  2. Subsampling de Taille 0.05% nécessaire à cause des temps d'exécution très longs sur Jeu Complet
     Recherche et optimisation des hyper-paramètres par Grille (GridSearchCV) et K-Fold Cross-validation (K=3)
     - ► Input Pipeline : données oversamplées SMOTE avec voisinage k=5
  3. Evaluation du Modèle sur l'ensemble de Test avec les métriques F2 Score (F1 Score, Precision, Recall, AUC Score)
  4. Apres sélection du Best Model de l'étape précédente sur le F2 Score on va l'optimiser sur 70% du Full Dataset par affinage des meilleurs hyper-paramètres obtenus précédemment

Modélisation (3)

Résultats

Train Size=0.05	Precision	Recall	Score AUC	Accuracy	F1 Score	F2 score
Naïf Bayes	Nan	0	0.5	0.92	0	0
Logistic Regression	0.16	0.62	0.73	0.71	0.26	0.39
Random Forest Classifier	0.08	1.0	0.5	0.08	0.15	0.30
XGBoost	0.12	0.56	0.63	0.64	0.20	0.32
Best Model Train Size=0.7						
Logistic Regression	0.17	0.65	0.74	0.71	0.26	0.40

2 types d'erreur :

- Un Faux Positif (FP) est une erreur de **type 1** : un client est **prédit en défaut de paiement** alors qu'il **ne l'est pas réellement**.
- Un Faux Négatif (FN) est une erreur de **type 2** : un client **n'est pas prédit en défaut de paiement** alors qu'il **l'est réellement**.

2 métriques : **Precision** = **TP / (TP + FP)** (ratio of correctly classified positive samples),
Recall = **TP / (TP+FN)** (detection positive samples)

Hypothèse métier : Une **erreur de type 2 est plus coûteuse financièrement** qu'une erreur de type 1
=> il est plus important de minimiser le nombre de FN que celui de FP : **On maximise le Recall**

=> Optimisation de la métrique du **F2 Score** : **F2 Score qui accorde un poids plus important au Recall**

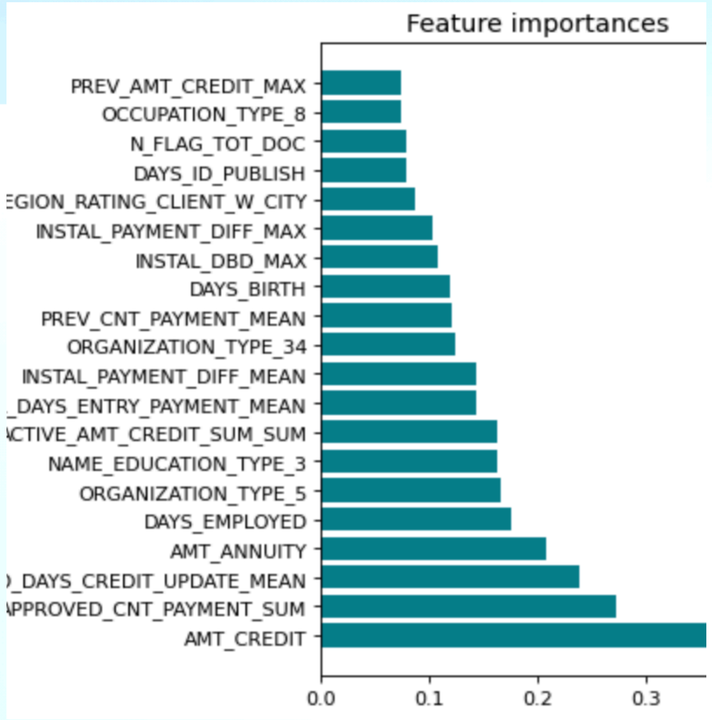
Le **Best Model** est la **Régression Logistique**

Interprétation

Feature Importance Globale

Quelles features sont importantes et quelles types d'interactions entre elles prennent place ? L'interprétabilité du modèle global doit nous permettre de comprendre la distribution des résultats à un niveau global grâce aux features apprises

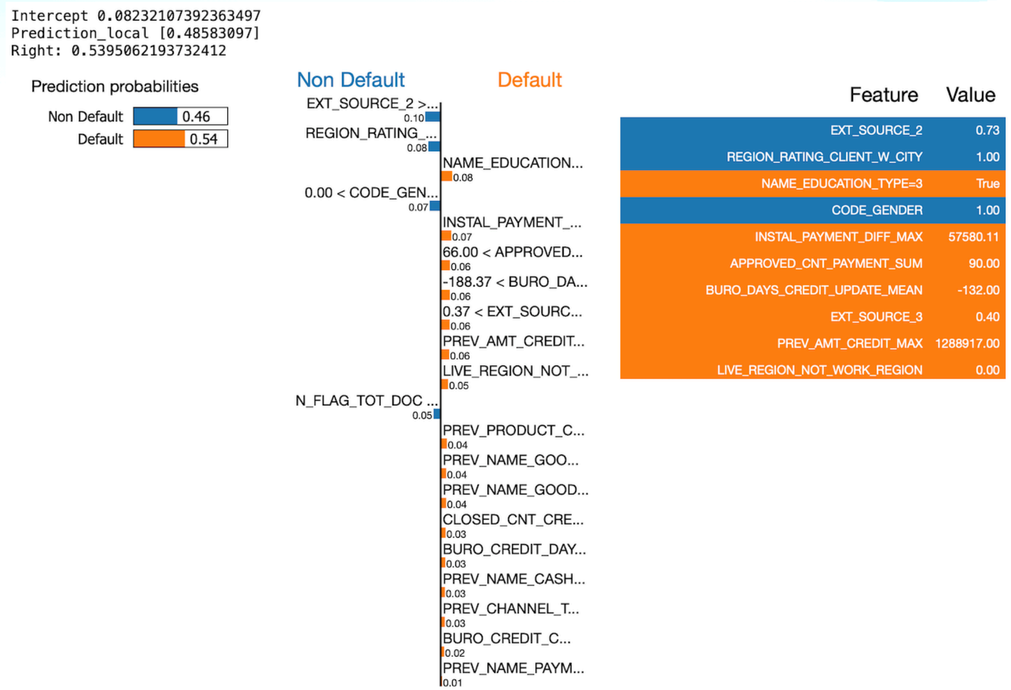
Les coefficients de la Regression Logistique nous donne l’importance des Features



Feature Importance Locale

Avec l'interprétabilité locale on ne cherche plus à expliquer le modèle globalement mais à un niveau local pour lequel on cherche l'explication de chaque prédiction individuellement.

Le modèle LIME est utilisé pour calculer la Prédiction de défaut et les % d’attribution des features à cette probabilité.



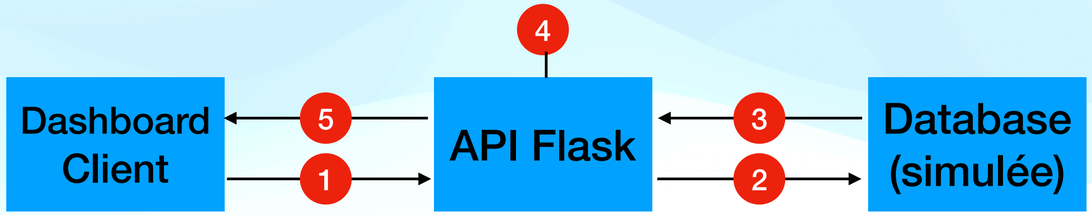
Mise en oeuvre Dashboard

Mise en oeuvre de l’API de prédiction

Une API Flask est une interface HTTP créée pour connecter le Dashboard (client) au modèle de calcul de la prédiction et à l’accés aux données

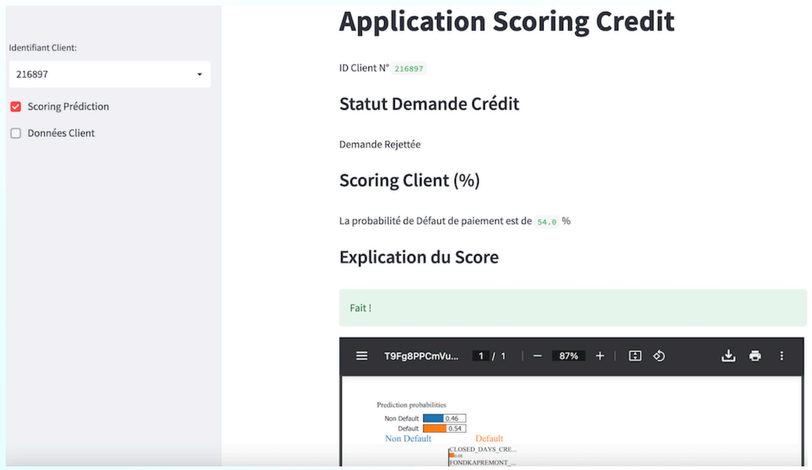
Principe de fonctionnement :

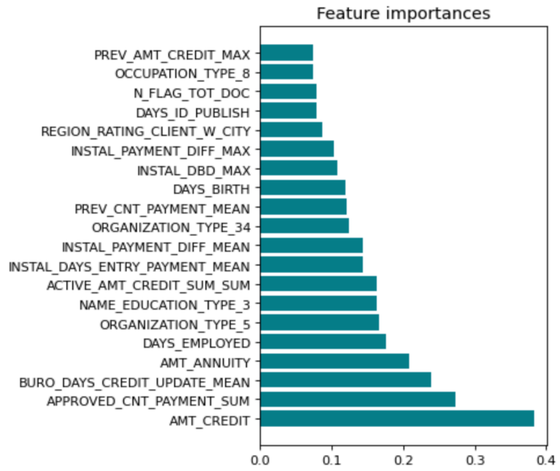
1. Envoi d’une requête Get avec l’ID Client en paramètre
2. Requête à la base de données pour récupérer les données clients
3. Renvoi des données clients
4. Calcul de la Prédiction avec le modele chargé
5. Renvoi des données au format JSON



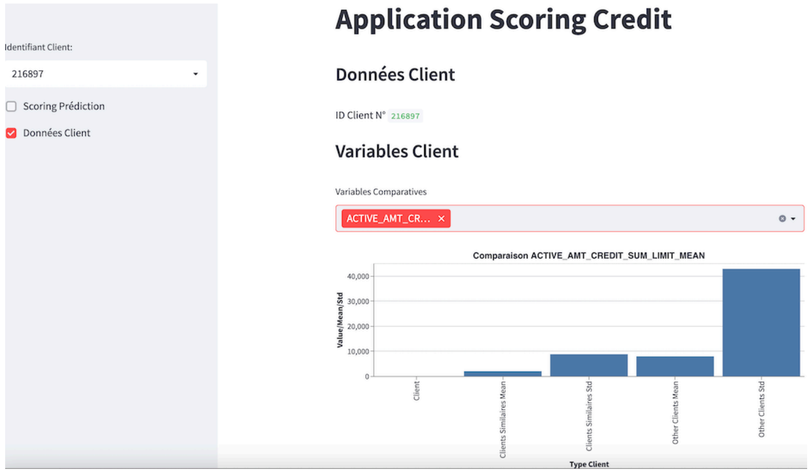
Mise en oeuvre du Dashboard

Calcul Scoring Client & Model Explication





Données Clients Comparaison



Conclusion

Feature Engineering

- Le nombre de 440 Features est élevé et devrait être raffiné par le métier par la suite pour éliminer celles totalement inutiles ou rajouter celles nécessaires (et qui ont été éliminés statistiquement).
- Le seuil de 30 pris pour le Coefficient de variation pourrait être réajusté à la hausse car il peut éliminer certaines Features qui pourrait être interessantes d'un point de vue métier.
- D'autres méthodes de sélection de variables pourraient être utilisées comme 'Select Kbest' (qui permet de sélectionner des variables dont le score de dépendance est plus élevé par rapport à la target en utilisant un test de corrélation statistique) ou 'SelectFromModel' à partir d'un autre classifier par ex RandomForest Classifier pour évaluer l'importance des Features.
- La synthétisation de nouvelles variables dans le noyau a pu amener une perte d'information.

Optimisation du Modéle

- Le modèle avec un F2 Score assez moyen de 0.41 mais un Recall de 0.65 est donc plutôt bon et il faudrait optimiser le Recall.
- Le choix de beta=2 pour le F2 Score pourrait être ajusté pour tenir compte de la vraie pondération relative entre la perte financière et la perte cliente.
- L'optimisation des modèles faite sur une plage restreinte d'hyper-paramètres pourrait être améliorée avec de meilleures capacités de traitements.
  - Algorithme SMOTE :
  - Le nombre de voisins pour l'Oversampling à été fixé à 5 sans aucune optimisation : si la taille du voisinage minoritaire < 5 alors l'individu synthétique est créé dans une zone non représentative, si la taille du voisinage minoritaire > 5 on peut peut être améliorer l'individu synthétique avec plus d'informations à disposition.
  - Il faudrait retraiter les variables discrètes (ex nombre d'enfants) après SMOTE à l'arrondi supérieur ou inférieur car elles peuvent générer une forte variance entre le modele de training et celui de test (écart de performance entre les données Train et Validation/Test).
  - Enfin il faudrait utiliser SMOTE-NC, pour SMOTE-Nominal Continuous pour traiter les variables continues mais aussi les variables catégoriques.

DashBoard

- Le Dashboard permet d'offrir aux chargés de clientèle un outil pour accéder visuellement et simplement aux résultats du modèle avec un modèle explicatif simple fourni par Lime pour une explication locale d'un client donné ou avec la Feature Importance Globale pour un modèle global qu'on devra juxtaposer avec une connaissance métier pour en apprécier la pertinence.
- On peut améliorer la partie Information Cliente en fournissant des graphes bi-variables pour améliorer la compréhension de certaines variables.