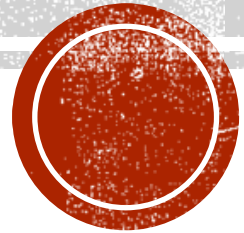
SOUTENANCE DU PROJET 7 – DATA SCIENTIST

IMPLEMENTER UN MODELE DE SCORING

Henrique DA COSTA - Openclassrooms - Octobre 2021



SOMMAIRE

1. PRESENTATION DU PROJET

1. PRESENTATION DU PROJET

Etude d'un modèle de Scoring

Prêt à dépenser souhaite ***développer un modèle de Scoring de la probabilité de défaut de paiement du client*** pour étayer la décision d'accorder ou non un prêt à un client potentiel.

Développement d'un dashboard

Développement d'un Dashboard interactif pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d'octroi de crédit.

Le dashboard doit permettre de :

1. Visualiser le score pour chaque client
2. Visualiser des informations descriptives relatives à un client
3. Comparer les informations descriptives relatives à un client à l'ensemble des clients ou à un groupe de clients similaires

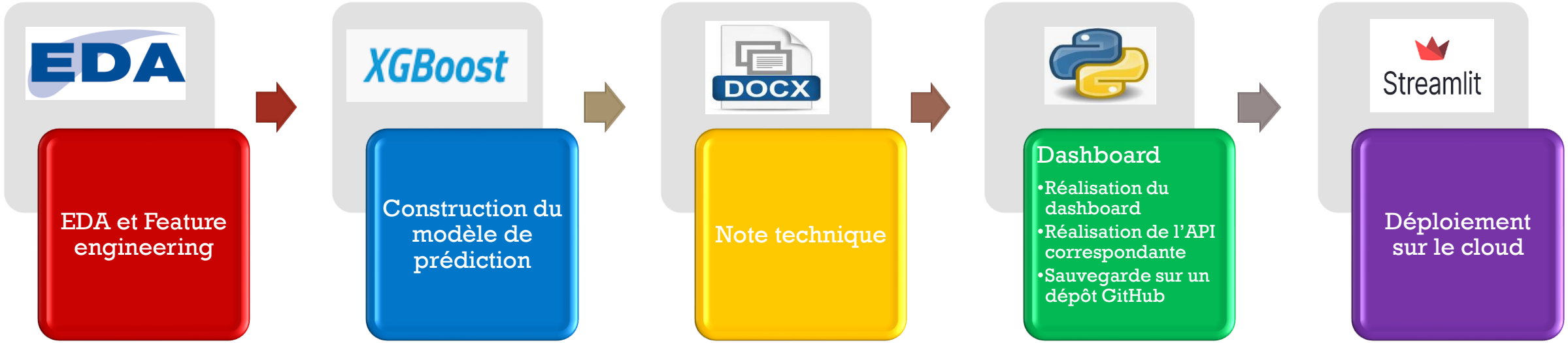
Demandes du manager

- Réaliser une note méthodologique expliquant en détails la construction du modèle.
- Déploiement du dashboard sur le Cloud.

Lien données :

https://www.kaggle.com/c/home-credit-default-risk/data

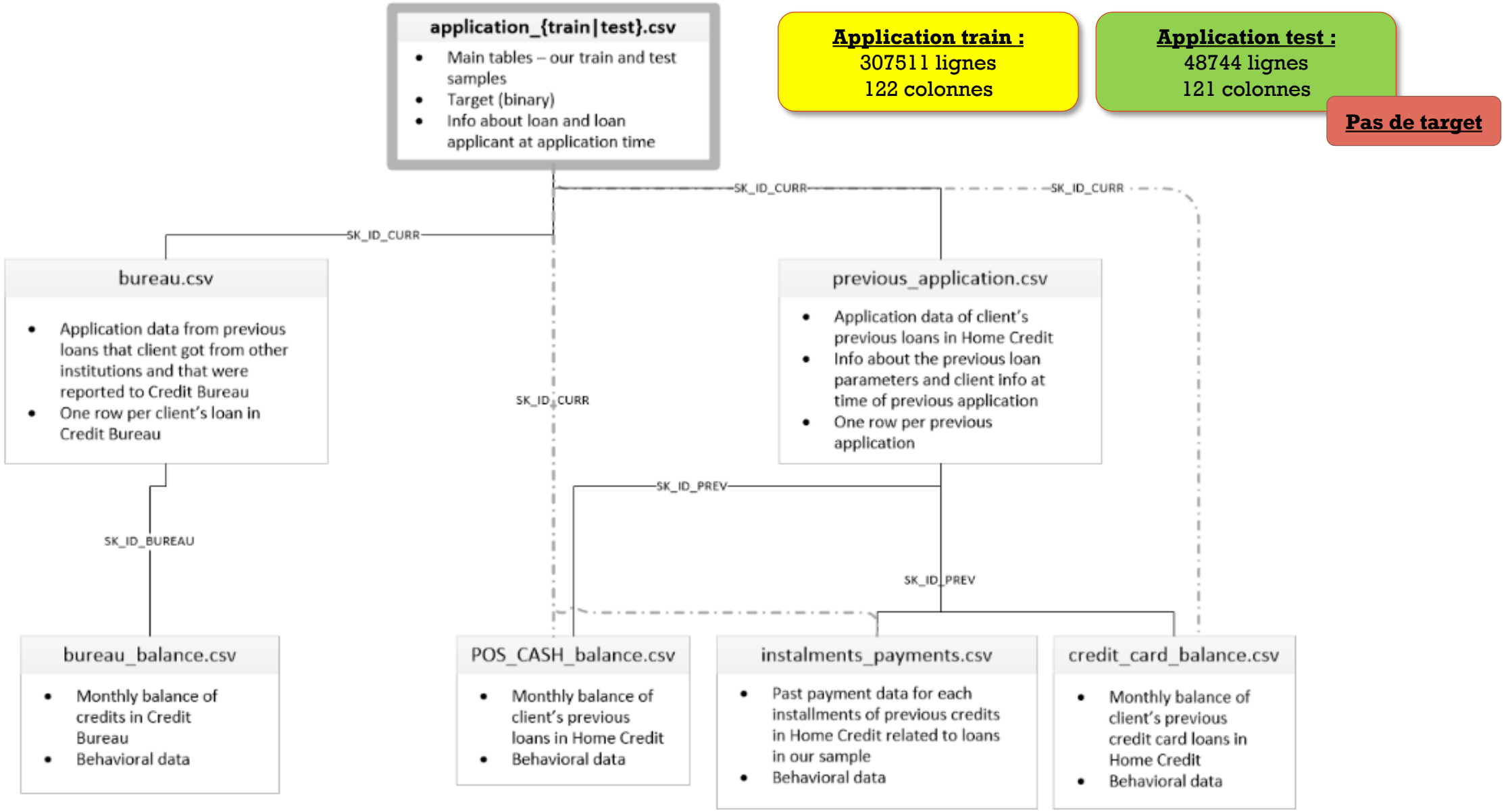
1. PRESENTATION DU PROJET



Plan d'actions

2. PRESENTATION DES DONNÉES, EDA ET FEATURE ENGINEERING

2. PRESENTATION DES DONNÉES



2. EDA ET FEATURE ENGINEERING

Data train
Data test

- Rappel : "test.csv" est le dataset que nous utilisons pour simuler un nouveau client dans la base. Toutefois il convient que ces deux datasets aient la même structure à l'issu du feature engineering.
- Fusion de l'ensemble des tables sur la base de la clé correspondant à l'identifiant client 'SK_ID_CURR',

Valeurs manquantes

- Traitement par imputation simple par 0 après avoir essayé différentes méthodes (médiane, mode, moyenne).

Encodage variables

- Label encoding pour les variables à 2 catégories.
- One Hot Encoding pour les variables à plus de deux catégories.

Alignement datasets

- Alignement des datasets "train" et "test" pour conserver des structures identiques.

Outliers

- Pour la prédiction du risque de défaut, les défaillants ont généralement tendance à avoir un comportement qui s'écarte de la normale et, par conséquent, nous ne pouvons pas supprimer les valeurs aberrantes ou les points éloignés, car ils peuvent suggérer une tendance importante au défaut.

Création de variables

- WEIGHTED_EXT_SOURCE = Pondération (2,3 et 4) des 3 variables EXT_SOURCE fortement corrélées avec la cible et discriminantes entre les défaillants et les non défaillants => Scores normalisés (entre 0 et 1) de différentes sources externes,
- TARGET_NEIGHBORS_500_MEAN = Moyenne des valeurs cibles des 500 voisins les plus proches de chaque point de données => caractéristiques utilisées [CREDIT_ANNUITY_RATIO et les 3 scores EXT_SOURCE],
- Nombre total de features au terme du process d'engineering = 1 241,

3. MODELISATION

3. MODELISATION

Utilisation de la librairie Pycaret

- PyCaret est une ***bibliothèque d'apprentissage automatique open source en Python qui automatise les flux de travail d'apprentissage automatique***. Il s'agit d'un outil d'apprentissage automatique et de gestion de modèles de bout en bout qui accélère le cycle d'expérimentation de manière exponentielle et le rend plus productif.
- En comparaison avec les autres bibliothèques d'apprentissage automatique open source, PyCaret est une bibliothèque alternative qui peut être utilisée pour ***remplacer des centaines de lignes de code par quelques lignes de code seulement.*** Cela rend les expériences exponentiellement rapides et efficaces.
- ***PyCaret est essentiellement un wrapper Python autour de plusieurs bibliothèques et frameworks*** d'apprentissage automatique tels que scikit-learn, XGBoost, LightGBM, CatBoost, spaCy, Optuna, Hyperopt, Ray et bien d'autres.

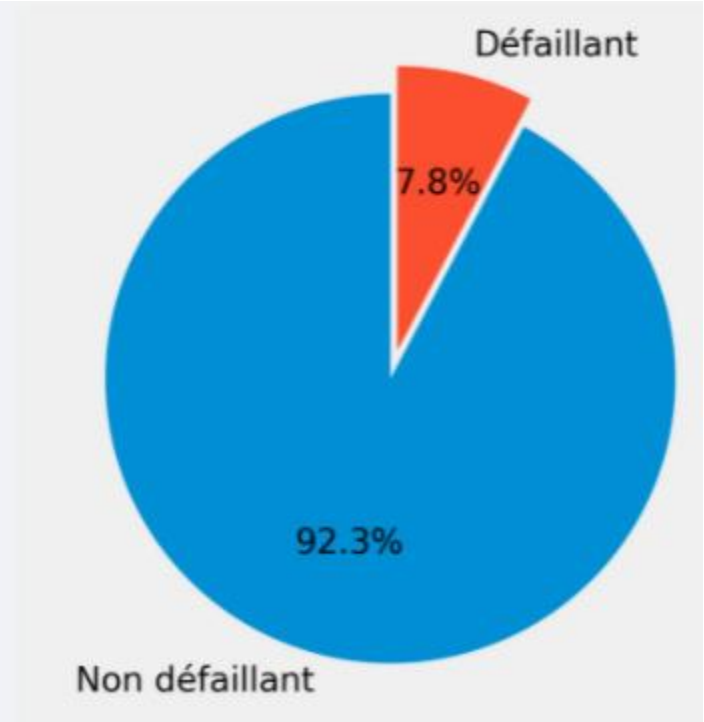
3. MODELISATION

Etape 1 **Choix des modèles**	Entraînement tous modèles disponibles avec notation validation croisée Kfold (10 folds) pour l'évaluation des métriques (AUC, F1-Score, Log-Loss) et du temps d'entraînement.
Etape 2 **Création et notation du modèle**	A l'aide de la validation croisée K-fold (10 plis).
Etape 3 **Optimisation – Réglage des hyperparamètres**	Recherche par grille aléatoire sur un espace de recherche prédéfini. Utilisation possible de fonctions de recherche optimisées avec Hyperopt et Optuna.
Etape 4 **Prédiction sur le jeu de données entraîné avec une validation croisée**	Prédire l'ensemble test/hold-out (30 %) du jeu d'entraînement initial.
Etape 5 **Finaliser le modèle pour le déploiement en production**	Ajustement du modèle sur l'ensemble de données d'entraînement complet, y compris l'échantillon test/hold-out (30%).
Etape 6 **Prédictions sur le 'unseen df'**	Jeu de test non traité et représentant 20 % du jeu de données complet.

3. MODELISATION

Problème de classification supervisée binaire portant sur un ensemble de données déséquilibré

- L'ensemble de données fourni est un ***ensemble de données déséquilibré*** (Environ 8 % de clients en défaut contre 92 % de clients sans défaut).
- Ainsi, nous avons besoin de corriger ce déséquilibre car certains algorithmes de ML sont sensibles au déséquilibre des données. Ce ***déséquilibre*** sera ***corrigé*** à l'aide ***du paramètre « fix_imbalance » de la fonction setup de Pycaret***. ***L'approche SMOTE*** (Synthetic Minority Oversampling Technique) ***est appliquée par défaut*** pour créer des points de données synthétiques pour la classe minoritaire.



3. MODELISATION

<u>Choix des métriques pour évaluer la performance du modèle</u> :

- **Objectif** = ***minimiser les faux négatifs*** (personnes prédites comme non-défaillants par le modèle mais qui étaient en fait des défaillants). Nous ne voulons pas passer à côté d'un ***défaillant*** comme étant ***classé comme non défaillant*** car le ***coût des erreurs*** pourrait être ***très élevé***.

- ***Moindre mal*** : même si certains des ***non-défaillants*** sont **<u>classés comme défaillants</u>**, ils ***peuvent postuler à nouveau*** et demander une vérification de profil spéciale par des experts. De cette façon, ***nous réduisons le coût des erreurs commises par le modèle***.

- On privilégiera le ***F1-Score*** qui ***combine subtilement la précision et le rappel.***

$$\text{F1-Score} = 2\,\frac{\text{precision} * \text{recall}}{(\text{precision} + \text{recall})}$$

- *On choisira en seconde position l'****AUC****,* insensible au déséquilibre des classes. Il fonctionne en classant les probabilités de prédiction de l'étiquette de classe positive et en calculant l'aire sous la courbe ROC qui est tracée entre **True Positive Rates** et **False Positive Rates** pour chaque valeur de seuil.

3. MODELISATION – FONCTION DE COÛT

- S'agissant d'un problème de classification binaire, nous utiliserons la ***fonction de Cross-Entropy Loss*** (Log Loss) :
  - mesure la divergence entre deux distributions de probabilité,
  - dans le cas d'une classification binaire, chaque probabilité prédite est comparée à la valeur de sortie de classe réelle (0 ou 1) et un score est calculé qui pénalise la probabilité en fonction de la distance par rapport à la valeur attendue.

$$CE = -(y \log(p) + (1 - y) \log(1 - p))$$

- ***Fonction de coût personnalisée*** : On peut définir une fonction de coût en accordant des poids différents aux éléments de la matrice de confusion. Nous avons ainsi affecté un ***poids de - 10*** à chaque dossier prédit négatif mais réellement positif (***FN***) et un ***poids de +100*** aux dossiers négatifs identifiés comme tels (***TN***). Les poids des dossiers FP et TP sont nuls.

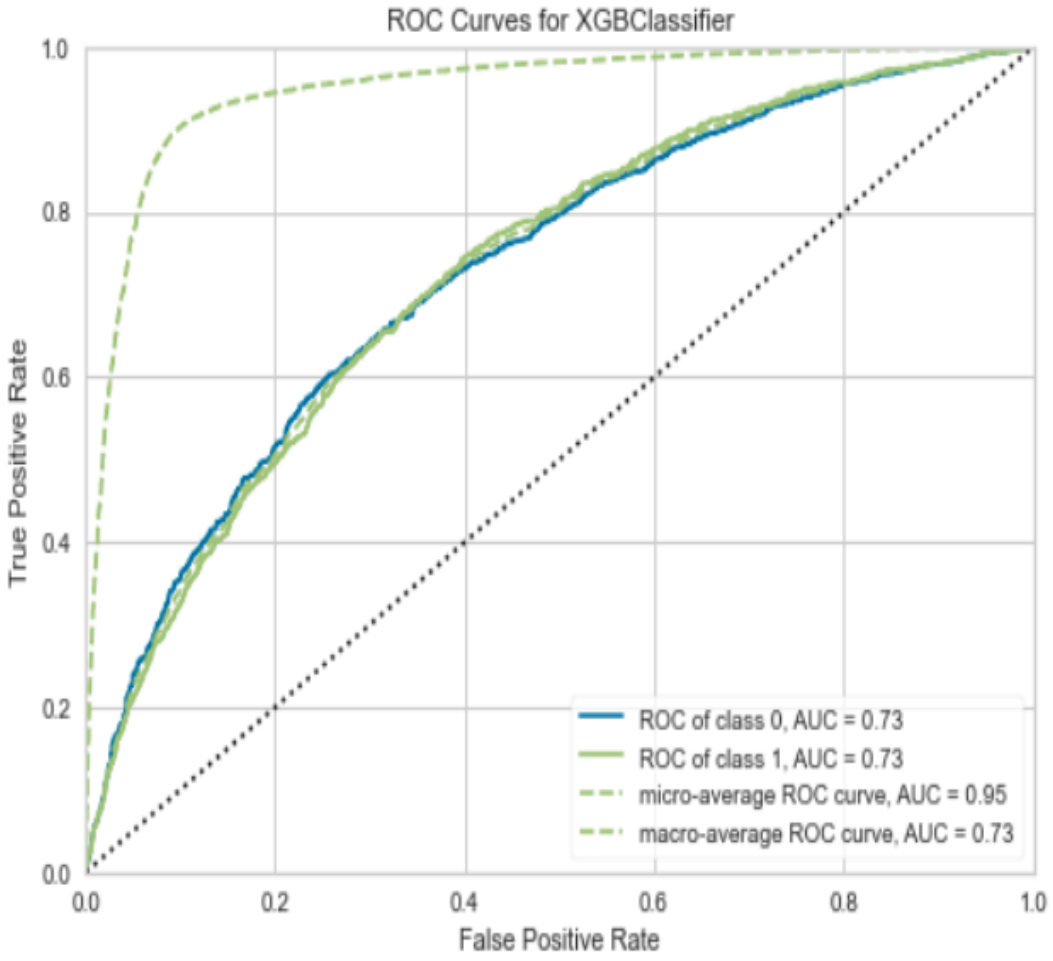
3. MODELISATION

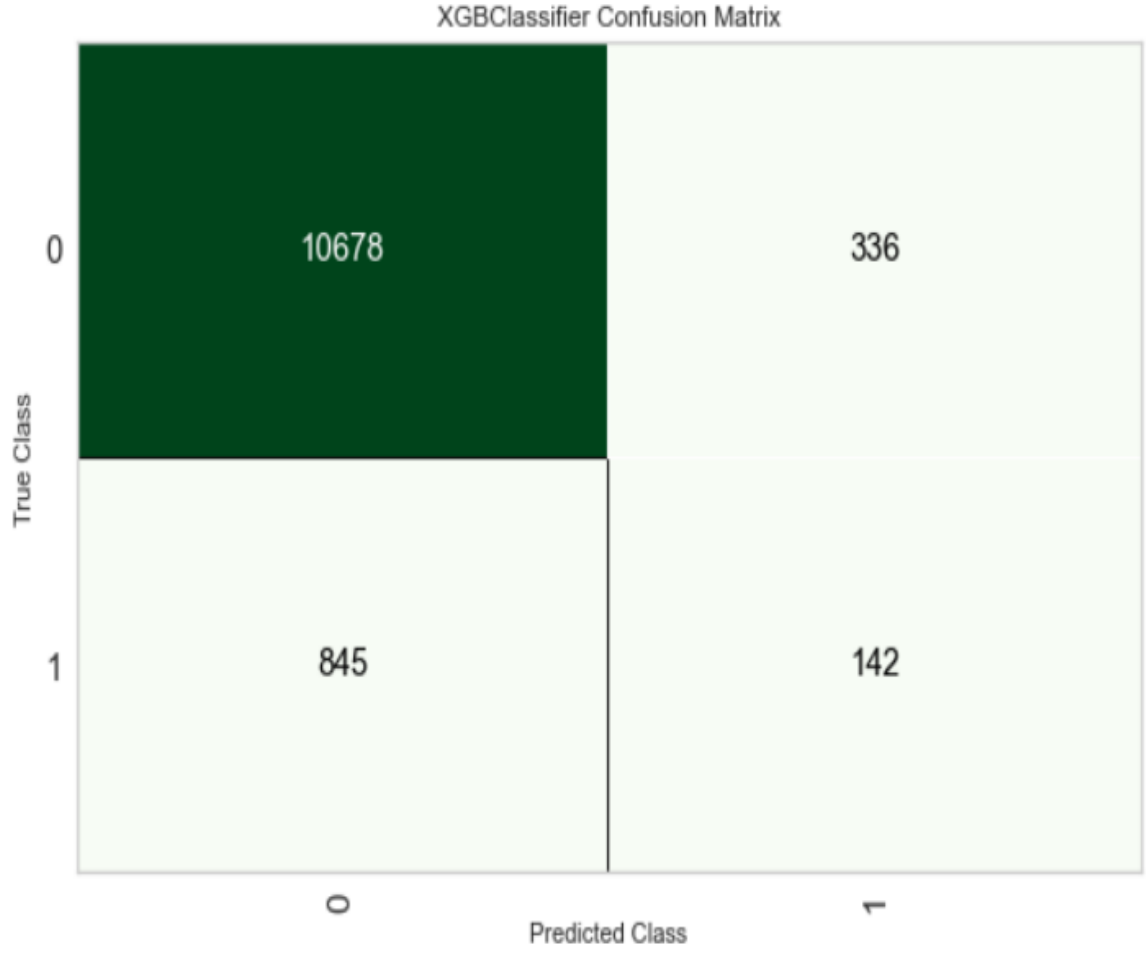
- Nous avons testé nos modélisations sur plusieurs configurations de données (Cf, Notebooks) avec les modèles Lightgbm, LDA et XGBOOST :

  - Sur l'ensemble des features (1241) sans SMOTE.
  - Sur les 50 principales features sélectionnées par le module _feature_importances du meilleur modèle de Pycaret (méthode RFECV) avec SMOTE.
  - Sur les 350 principales features avec SMOTE.
  - ***Sur les 300 principales features sans SMOTE avec le modèle XGBOOST finalement retenu :***
    - modèle donnant les meilleurs résultats après entraînement sur le jeu complet, y compris le test/holdout,
    - selon les métriques sélectionnées (F1-Score et AUC),
    - En fixant un seuil de classification (threshold) à 0,74.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss
0	Extreme Gradient Boosting	0.9340	0.9062	0.3495	0.6970	0.4656	0.4345	0.4641	2.2794

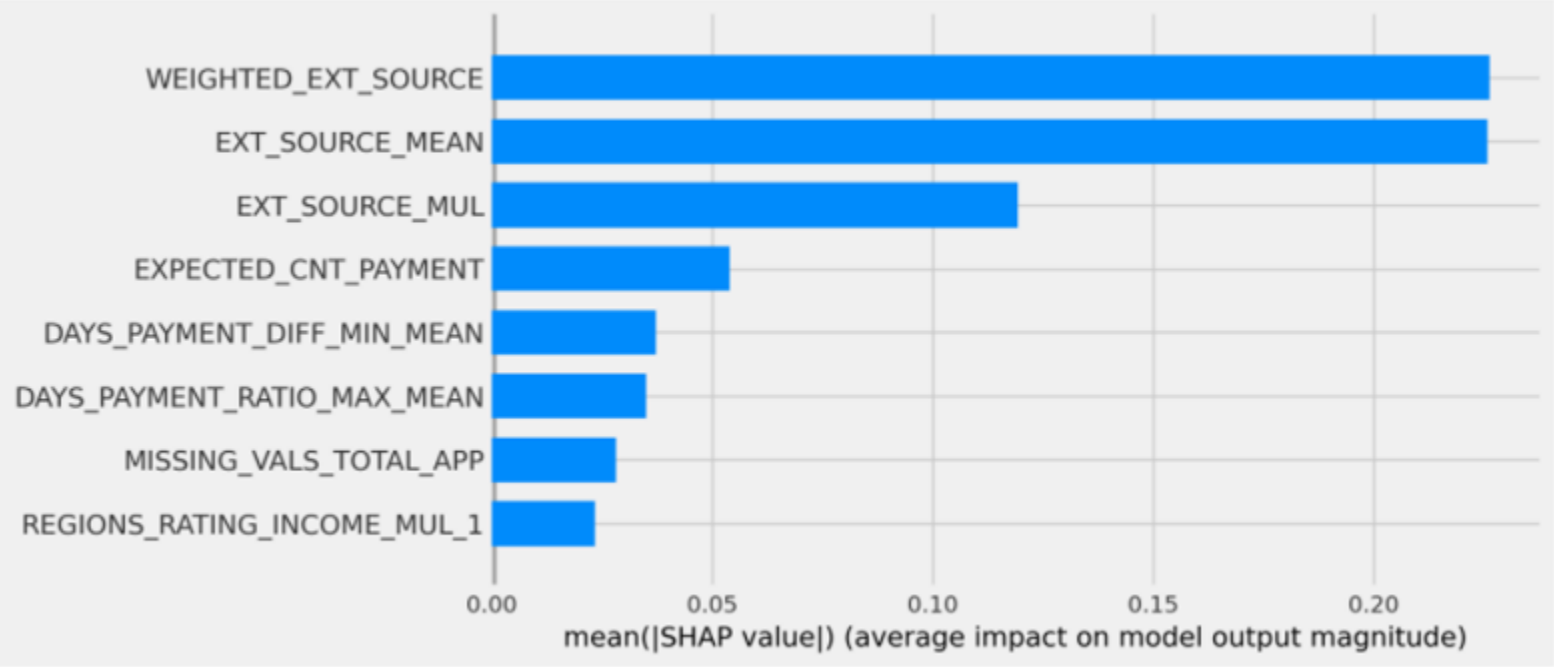
3. MODELE RETENU = XGBOOST AVEC 300 FEATURES





3. INTERPRETATION DU MODELE XGBOOST – METHODE SHAP

- ***La méthode SHAP consiste à calculer la valeur de Shapley pour toutes les variables de tous les individus c'est-à-dire la moyenne de l'impact d'une variable*** (sur la sortie, donc la prédiction) pour toutes les combinaisons de variables possibles. La somme des effets de chaque variable explique alors la prédiction.



18

4. DASHBOARD

4. DASHBOARD

Utilisation de Streamlit pour la création du Dashboard et son déploiement en ligne

- Streamlit est un ***framework d'application en python*** dédié à la science des données. Il est livré avec un ***serveur Web intégré***.

- Le choix a été fait de ***déployer l'application sur la plateforme Streamlit*** en mode partage.

    - https://share.streamlit.io/henrique-dacosta/p7scoring/main/app/app.py

- Cela constitue une excellente alternative à Heroku car, en accès gratuit, ce dernier limite la taille du slug à 500 Mo.

- Dés lors que la bibliothèque PyCaret possède elle-même beaucoup de dépendances, ***la taille de l'application Web dépasse rapidement la limite de taille de slug de Heroku***.

- ***Streamlit présente de ce point de vue un avantage intéressant*** puisqu'il n'est pas contraint par cette limite.

- Streamlit fonctionne ***en connexion avec Github***.

5. LIMITES ET AMELIORATIONS

5. LIMITES ET AMELIORATIONS

- ***Sélection des variables*** : Les informations disponibles relatives à l'importance des variables sont débattues avec les experts métier en vue de définir les stratégies techniques à tester dans les différents blocs concernés :
  - Valeurs manquantes
  - Corrélations entre variables
  - Seuil de variance
  - Réduction de dimensions (RFE)

- ***Fonction d'évaluation du gain :*** Les règles métier et les critères financiers relatifs aux pertes et profits doivent être communiqués en vue d'établir une fonction d'évaluation du gain adaptée (fonction de coût personnalisée optimisée).

- ***Optimisation des hyper-paramètres :*** Nous avons testé plusieurs classifieurs et retenu XGBOOST pour ses meilleures performances, bien qu'il soit plus lent à exécuter que Lightgbm. D'autres classifieurs comme Lightgbm ou Linear Discriminant Analysis peuvent potentiellement apporter de meilleures performances techniques et un gain de temps appréciable par rapport à XGBOOST. Il s'agit de les tester dans le pipeline complet si les contraintes de temps en production le permettent.

- Autres pistes d'améliorations : Cf. Note méthodologique.

MERCI

AVEZ-VOUS DES QUESTIONS ?