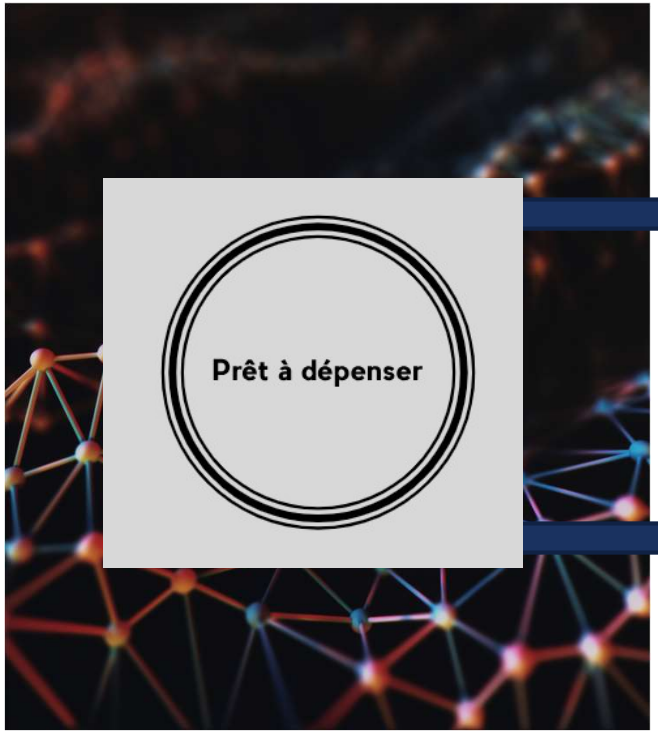
PROJET 7 : IMPLÉMENTEZ UN MODÈLE DE SCORING

LAURENT CAGNIART - OPENCLASSROOMS

PROBLÉMATIQUE

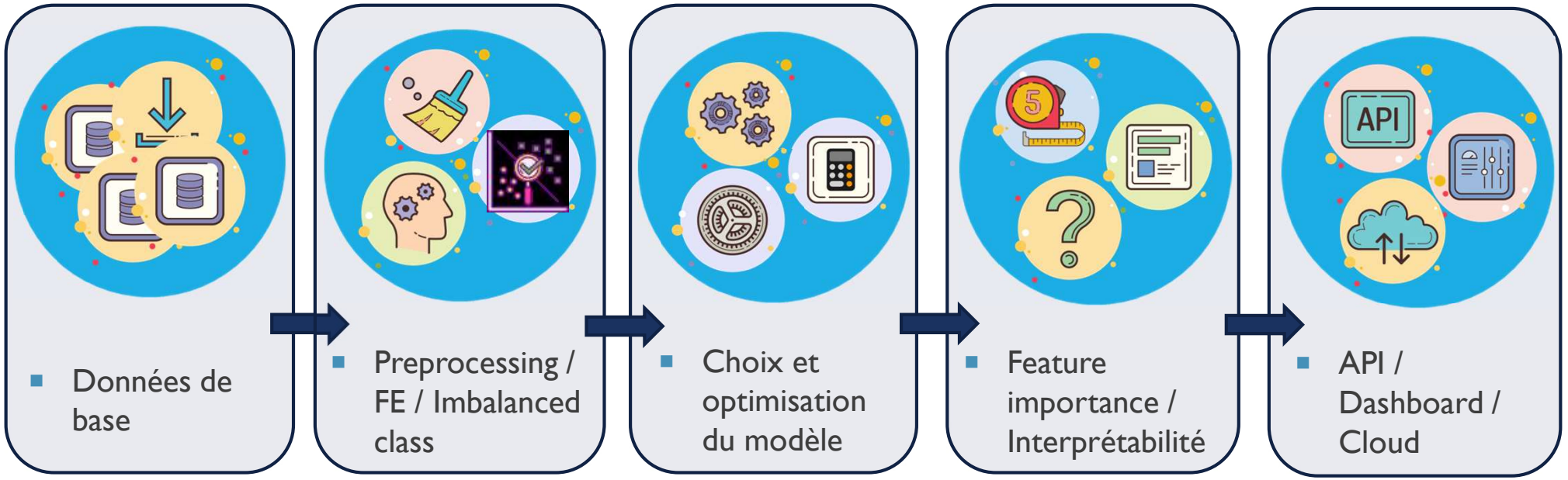


- API
- Dashboard

- Construction d'un **modèle de scoring** pour **prédire la probabilité de défaut** d'un client de façon **automatique**

- Construction d'un **dashboard interactif** utilisable par les **gestionnaires de relation clients => interprétation des prédictions** et amélioration de la connaissance client

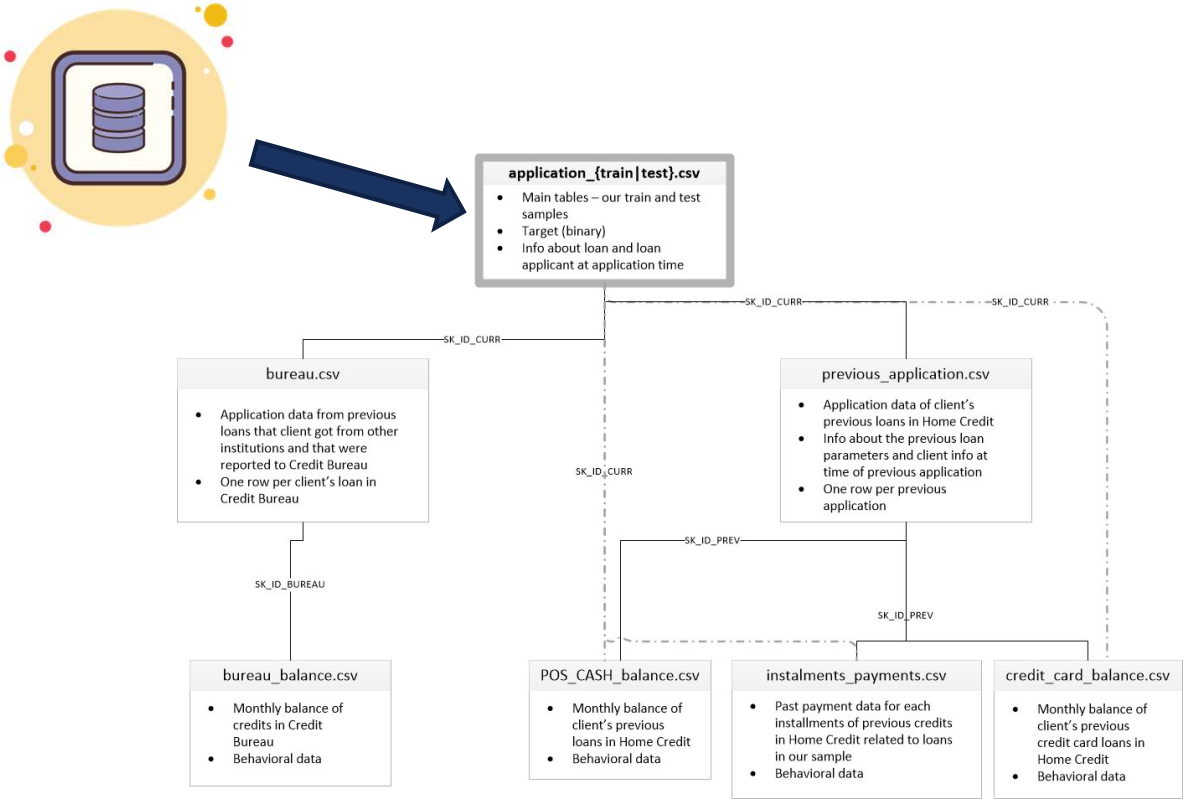
MÉTHODOLOGIE



- Données de base
- Preprocessing / FE / Imbalanced class
- Choix et optimisation du modèle
- Feature importance / Interprétabilité
- API / Dashboard / Cloud

- **10 Fichiers source** (dont 1 descriptif des features

- Le jeu d'entraînement contient :
  - 307,511 clients,
  - 121 features (caractéristiques démographiques, du foyer, des revenus et emploi, type de prêt demandé…)
  - La feature « target » : le défaut ou non de crédit (constaté historiquement)



DONNÉES DE BASE

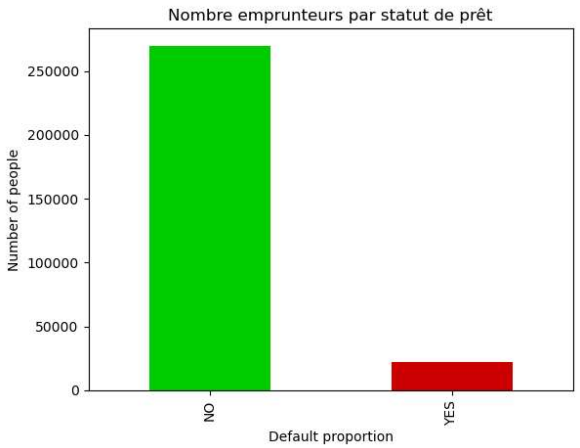
- Chargement et Fusion des fichiers (agrégation des features)
- Encodage des variables catégorielles (OHE, binary encoder)
- Imputation des valeurs manquantes (médiane ou le plus fréquent) au-dessus d'un seuil (50%)
- Détection et traitement des outliers

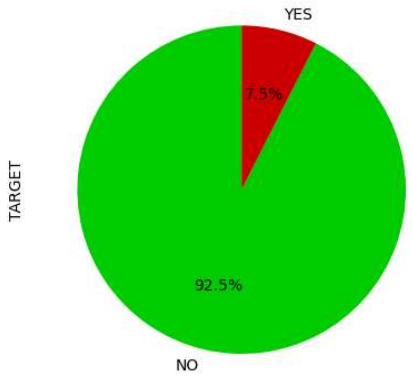
Feature engineering :

- Taux d'endettement : Mensualité du prêt / revenu mensuel client (%)
- Taux de remboursement : Montant remboursé par le client / montant du crédit précédent (%)
- Taux durée d'emploi : Durée expérience professionnelle / âge du client (%)
- Revenu du foyer / nb membres du foyer (revenu par personne)

PREPROCESSING- FEATURE ENGINEERING

PREPROCESSING – IMBALANCED CLASS



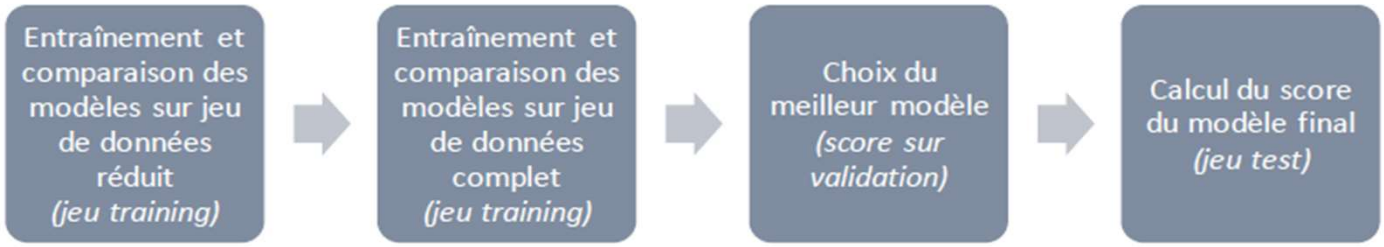


- **92,5%** de clients sans défaut de remboursement
- **7,5%** des clients avec défaut de remboursement

- Dans ce cas de classification binaire avec des classes déséquilibrées, nous ne pouvons pas nous appuyer sur la métrique habituelle « accuracy » (cf. « dummy classifer »)

- 2 méthodes pour prendre en compte ce déséquilibre :
  - La méthode SMOTE : création d'exemples synthétiques pour un oversampling de la classe minoritaire
  - Les arguments « class-weights » des modèles : modification des poids associés aux deux classes

- **Modélisation et optimisation des hyperparamètres**
  - 1ère étape : échantillonnage réduit pour comparer l'application des 2 méthodes SMOTE et « arguments – class_weights » => méthode SMOTE non retenue (temps de calculs élevés et tendance à l'overfitting)



  - 2ème étape : application du processus d'optimisation à l'ensemble du jeu de données
    - Train/test/split (70-30%)
    - Cross-validation : Folds stratifiés et Randomized search, optimisation des hyperparamètres « scale_pos_weight » et « class_weight »
  - Modèles testés : DummyClassifier, Logistic Regression, RandomForest Classifier et LightGBM

CHOIX ET OPTIMISATION DU MODÈLE

- **Modélisation et optimisation des hyperparamètres**

  - 3ème étape : définition d'une métrique répondant aux problématiques métier :

    - Risques financiers possibles :

      1/ mal catégoriser un client avec un risque élevé de défaut (Faux Négatif = erreur de type II)

      2/ mal catégoriser un client avec un risque faible de défaut (Faux Positif = erreur de type I)

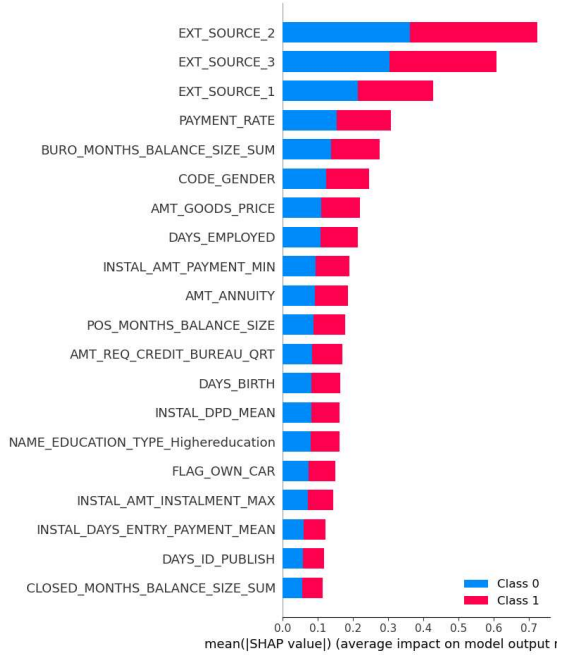
    - Nous souhaitons donc minimiser majoritairement les faux négatifs et ensuite les faux positifs. Autrement dit, on cherche à maximiser le recall et la précision en donnant plus d'importance au recall car le FN a un coût plus élevé que le FP => Perdre un client potentiel coûte moins cher que d'attribuer un crédit à un client qui fera défaut de remboursement.

    - C'est le rôle de la métrique « f beta score » où, en prenant des hypothèses métier, on fixera le coefficient beta à 2,775

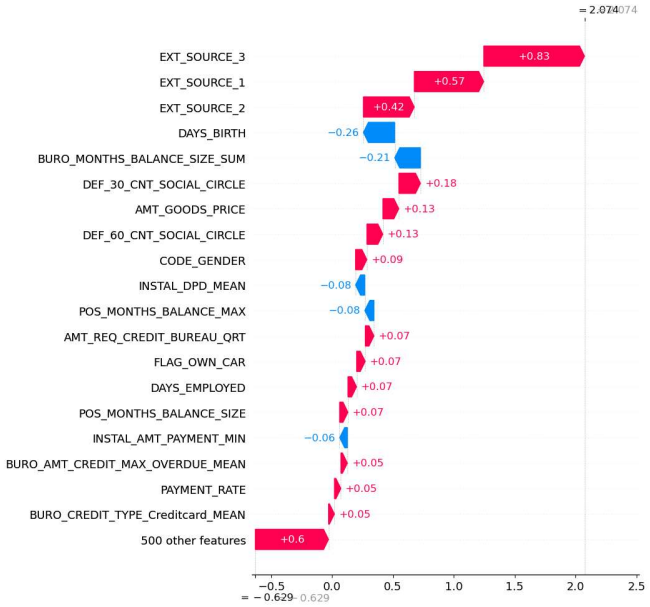
  → Sur cette base, le modèle le plus performant est lightGBM 

CHOIX ET OPTIMISATION DU MODÈLE - MÉTRIQUES

- **Utilisation de SHAP**
  - **Global**



  - **Local**



FEATURE IMPORTANCE / INTERPRÉTABILITÉ

- **API et dashboard**

Flask
web development, one drop at a time

- API (prédiction probabilité de défaut) ID client

Streamlit

- Outil de conception de dashboard

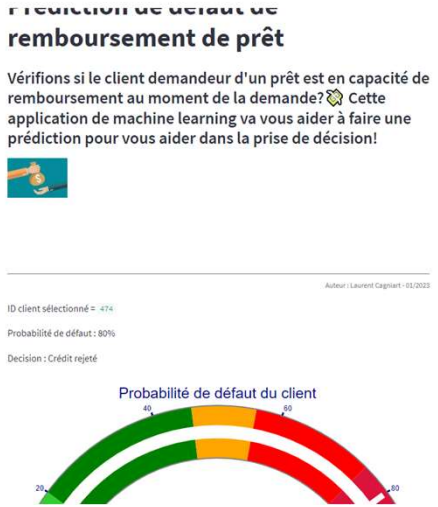
GitHub

- Outil d'hébergement et de versioning du code

HEROKU

- Outil de déploiement de l'application sur le cloud

- Livrable :



https://credit-risk-dashboard-pretadep.herokuapp.com/

API/ DASHBOARD / CLOUD

AXES D'AMÉLIORATION

- S'assurer avec les personnes du métier que les hypothèses prises pour la détermination du fbetascore sont valides
- travailler davantage sur le feature engineering pour diminuer le nombre de features à considérer dans le modèle pour des soucis de simplification du modèle pour le rendre plus intelligible et plus léger pour le déploiement
- Améliorer l'approche du business du dashboard en permettant au collaborateur de modifier avec le client certaines features (comme le montant ou la durée du prêt donc le montant des mensualités) pour voir si le client rentre cette fois-ci dans les critères d'octroi du prêt

MERCI DE VOTRE ATTENTION

LAURENT CAGNIART

MARQUEUR ICON BY ICONS8