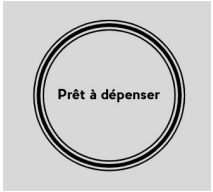


Implémentez un modèle de scoring

Guilhem Berthou - Pierre-Antoine Ganaye (mentor)

https://github.com/guilhembr/p7_scoring

Introduction

Contexte : 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.

L’entreprise souhaite mettre en œuvre un outil de “**scoring crédit**” pour calculer la **probabilité qu’un client rembourse son crédit,** puis classifie la demande en **crédit accordé** ou **refusé**.

Traiter les **variables descriptives** d’un prospect

Réaliser une **classification** des clients

Présenter les **résultats** en toute **transparence**

⇒ **Interprétation :** Il s’agit d’un problème de **classification supervisée :**

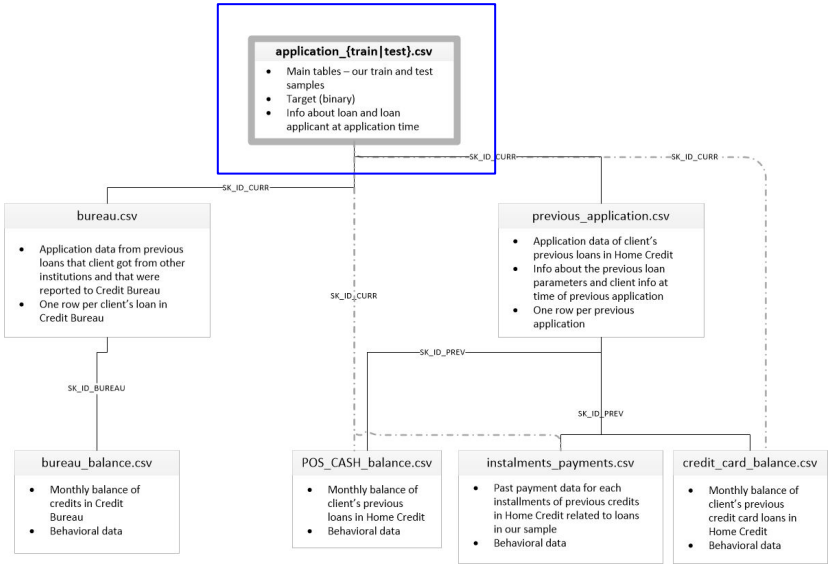
- *Supervisée* : Les données sont étiquetées dans le jeu de données d'entraînement `(application_train)` qui servira de base à l'entraînement d’un modèle afin de prédire la capacité de remboursement (le label) en fonction des features client.
- *Classification* : Le label est une variable binaire, 0 (le client rembourse son crédit), 1 (le client est insolvable).

Introduction

Jeu de données :

Le jeu de données contient 7 sources de données. Pour réaliser un premier prototype nous nous concentrons sur les 2 jeux principaux :

- *application_train/application_test* :
  - Contiennent les informations de chaque demande de prêt.
  - Chaque ligne des datasets correspond à une demande identifiée par `SK_ID_CURR.`
  - Le jeu d'entraînement contient les `TARGET`. A l'inverse le jeu de test n'est pas étiqueté.



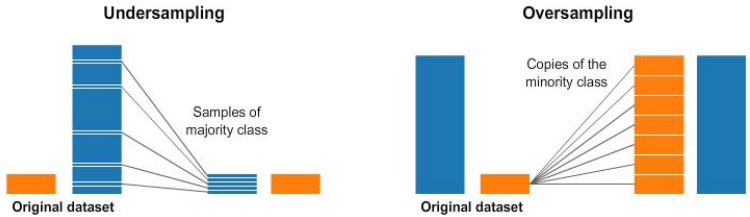
Sommaire

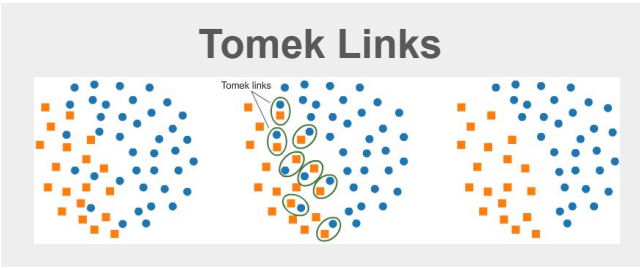
Sommaire

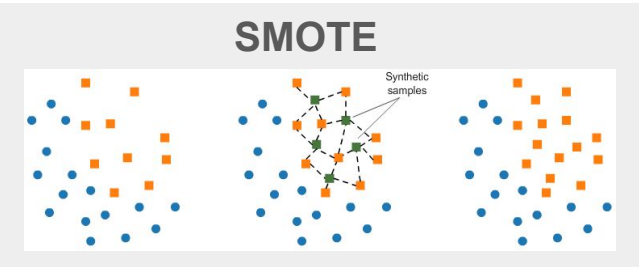
Présentation des enjeux

➢ Déséquilibre des classes

- **Enjeu du sujet :** Nous constatons tout d'abord qu'il s'agit d'un dataset dont les classes de la variable à prédire (`TARGET`) sont déséquilibrées. Il y a ainsi beaucoup plus de lignes correspondant à des remboursements qu'à des défauts.
- **Réponse apportée :**
  - *Option 1 :* **Undersampling / Oversampling**







Présentation des enjeux

➢ Déséquilibre des classes

- **Enjeu du sujet :** Nous constatons tout d'abord qu'il s'agit d'un dataset dont les **classes de la variable à prédire** (`TARGET`) sont ***déséquilibrées***. Il y a ainsi beaucoup **plus de lignes correspondant à des remboursements** qu'à des défauts.
- **Réponse apportée :**
  - *Option 2 :* **Hyperparamètre** `class_weights = balanced`
    - le modèle **ajuste automatiquement** le **nombre d'items** de chaque classe en les **pondérant** de manière inversement proportionnelle à la fréquence de leur classe :

```
n_samples / (n_classes * np.bincount(y))
```

⇒ *Nous avons obtenu les **meilleures performances** en adoptant cette seconde approche.*

Présentation des enjeux

➢ Feature engineering

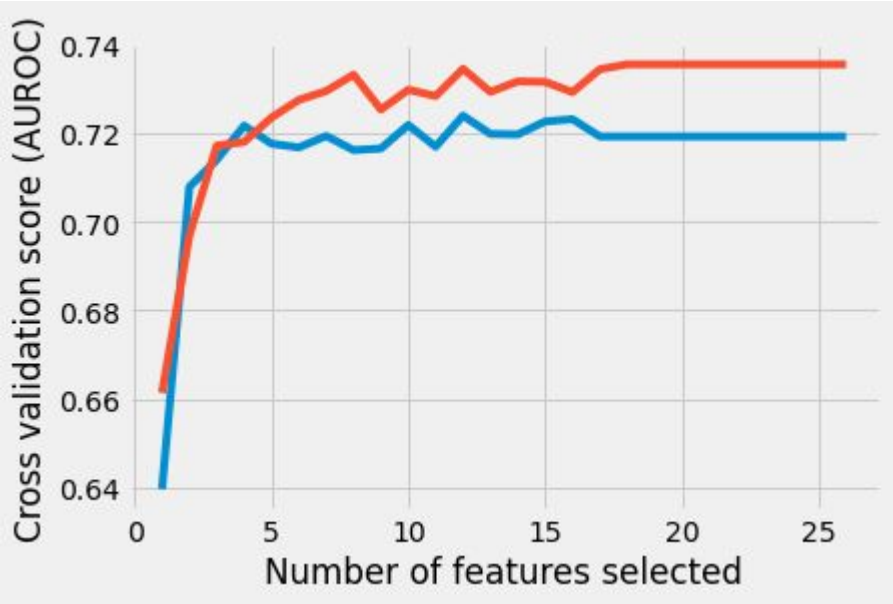
- Création de features sur la base de notre **compréhension des enjeux métiers** :
  - **Ratio d'endettement** (façon *DTI - Debt Income Ratio)*
    - `AMT_INCOME_TOTAL / AMT_ANNUITY`: Revenus client / Annuité du prêt
  - **Durée du prêt**
    - `AMT_CREDIT / AMT_ANNUITY :` Montant total du prêt / Annuité ⇔ **Durée du prêt**
  - **Profil d'endettement :**
    - **% du bien acquis financé par prêt :** `credit_goods_price_ratio= AMT_CREDIT / AMT_GOODS_PRICE`
    - **Apport :** `credit_downpayment= AMT_GOOD_PRICE - AMT_CREDIT`

⇒ La création de ces features a permis une **amélioration de la performance** des modèles *(AUROC, recall, f1 score)*

Présentation des enjeux

➢ Feature selection

- **Enjeu du sujet :** Afin d'améliorer la performance de notre modèle il est **nécessaire de sélectionner** les ***features*** les plus importantes en éliminant les variables non nécessaires
- **Réponse apportée :**
  - RFECV *(Recursive Feature Elimination with Cross-Validation)*
  - sélectionner les features par récursivité (*Cross-Validation* en maximisant *l'AUROC*) en considérant un nombre de plus en plus faible de variables selon leur `feature_importance.`
  - Nombre optimal de features obtenu via : `rfecv.n_features_`



Optimal number of features : 108

⇒ *La sélection de features a permis* ***l'amélioration*** *de la* ***performance*** *: notamment en* ***réduisant le temps d'exécution.***

Présentation des enjeux

➢ Description des métriques d'évaluation

○ Enjeu du sujet :

- Choix des métriques pour un problème de classes déséquilibrées (`accuracy_score` inadapté)

○ Réponse apportée

■ AUROC

- Aire sous la courbe ROC ⇔ **%TP = f(%FP)**

■ Recall (sensibilité)

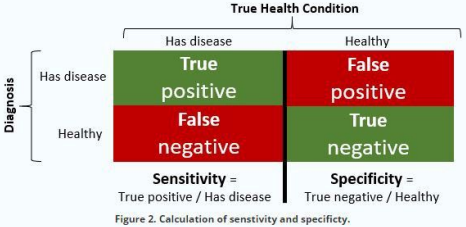
- Taux de vrais positifs ⇔ **TP / TP + FN**

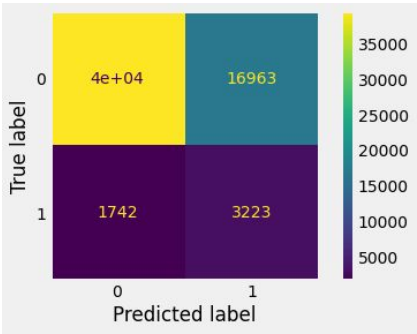
■ F1-Score

- Moyenne harmonique de la *précision* (**TP / TP + FP**) et du *recall*

■ Matrice de confusion

- Chaque ligne correspond à une classe réelle, chaque colonne correspond à une classe estimée.





Matrice de confusion - LogisticRegression

Présentation des enjeux

➢ Fonction coût métier

○ Enjeu du sujet

- L'objectif de ce projet est d'**être capable de prédire la solvabilité** d'un client ⇔ **prémunir** la banque face au **risque de défaut.**

○ Réponse apportée

- La ***matrice de confusion peut ainsi être lue*** du point de vue de la banque comme suit :
  - **TP** (*True Positives*) : Clients insolvables correctement identifiés
    - Perte = 0 ; Gain = 0
  - **FP** (*False Positives*) : Clients identifiés à tort comme insolvables
    - Perte = Intérêts du prêt (perte d'opportunité car prêt non alloué à tort)
  - **TN** *(True Negatives)* : Clients solvables correctement reconnus
    - Gain = Intérêt du prêt
  - **FN** (*False Negative)* : Clients insolvables non identifiés
    - Perte = Principal du prêt + intérêts non perçus

⇒ Si nous **traduisons cette problématique métier** dans notre processus de modélisation, nous devons **adapter la métrique** de mesure de performance des modèles.

Présentation des enjeux

➢ Algorithme d'optimisation et métrique d'évaluation pertinente

○ Enjeu du sujet

- Les **faux négatifs (FN)** correspondent clairement au pire scénario que nous devons éviter.
- Il conviendrait donc de **préférer le *recall*** (% de clients insolvables identifiés) **à la *précision*** (% de clients insolvables identifiés comme tels).

○ Réponse apportée

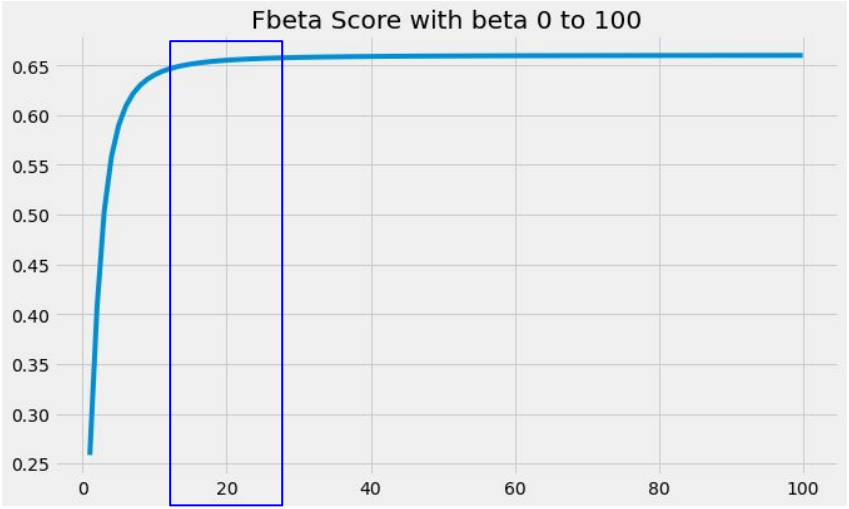
- Afin de tenir compte de cette contrainte opérationnelle nous avons utilisé le `fbeta_score`.
- Il s'agit d'un hyper-paramètre similaire au `f1_score` qui ajoute une dimension `beta` permettant de déterminer le point du *recall* dans le score combiné :
  - `beta` < 1 favorise la *précision*
  - `beta` > 1 favorise le *recall*

Présentation des enjeux

➢ Algorithme d'optimisation et métrique d' évaluation pertinente

○ Réponse apportée

- Nous avons ainsi défini un **score à optimiser** basé sur ce `fbeta_score` lors de notre entraînement
  - via la fonction `make_scorer` de *scikit learn.*
- Nous avons fait varier la valeur de `beta` en fonction du `fbeta_score` afin d'identifier la valeur à retenir.



⇒ Le `beta` optimal identifié (`beta = 20`) permet la **prise en compte des objectifs métiers** dans l'évaluation de la performance des modèles.

Présentation des enjeux

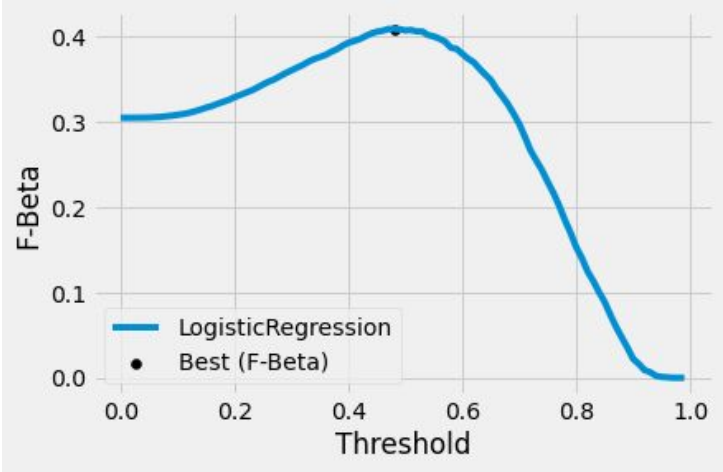
➢ Optimisation du seuil de décision

○ Enjeu du sujet

- La définition du **seuil de décision optimal** ⇔ *seuil à partir duquel une prédiction doit être considérée comme égale à 1* - est une **étape clé du processus de modélisation**

○ Réponse apportée

- Une fois le `beta` optimal identifié, nous avons fait varier le seuil de décision afin de maximiser le `fbeta_score.`



⇒ Le seuil optimal a été identifié `(Best_threshold=0.48)`
Il permet de maximiser le `fbeta_score (Fbeta=0.409)`

Sommaire

Modélisation

➢ Méthodologie d'entraînement

- Dans la version finale de notre *notebook* de pre-processing (Voir Summary Notebook) nous avons pris les arbitrages suivants :
  - Variables catégorielles : *One Hot Encoding* (OHE), imputation des valeurs manquantes par la valeur la plus fréquente (`most frequent`)
  - Variables numériques : Imputation des valeurs manquantes par la médiane (`median`), Standardisation via `StandardScaling.`

*⇒ L'encodage et le traitement des valeurs manquantes permet de **maximiser la performance** des modèles*

Modélisation

➢ Méthodologie d'entraînement

○ Entrainement de modèles de classification

- Nous avons entraîné plusieurs modèles afin d'identifier les plus performants pour répondre à notre problématique :
  - *DummyClassifier (baseline), LogisticRegression, Decision Tree, Random Forest, LGBM*

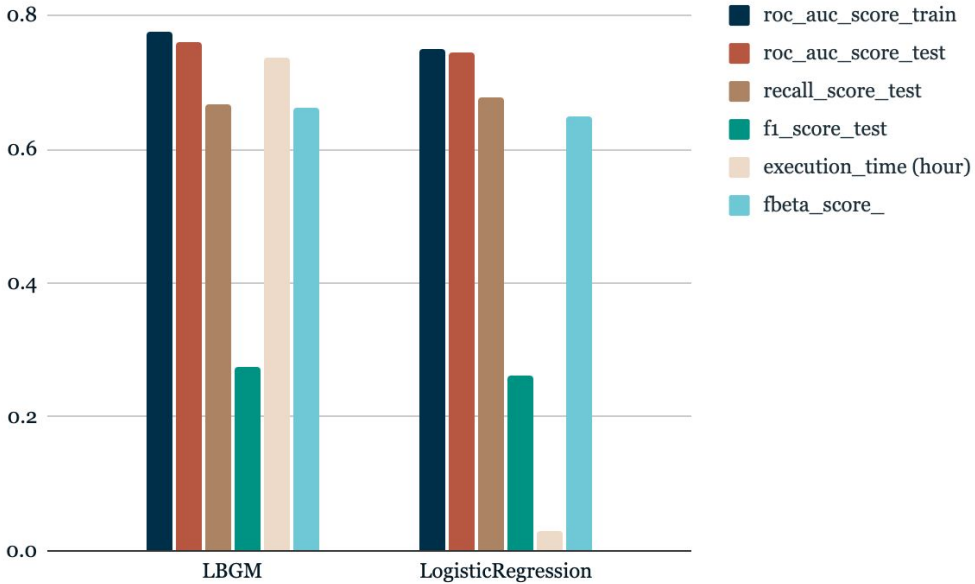
○ Recherche d'hyperparamètres via GridSearchCV

- Les hyperparamètres optimisés lors de l'entraînement sont détaillés dans le notebook
- Ils sont recherchés via ***GridSearchCV*** afin de se **prémunir du surapprentissage**.
- La recherche est paramétrée de sorte à **maximiser** : `Scoring = fbeta_score`

*⇒ La recherche d'hyperparamètres permet d'obtenir des **résultats comparables** et **généralisables** pour chacun des modèles entraînés.*

Modélisation

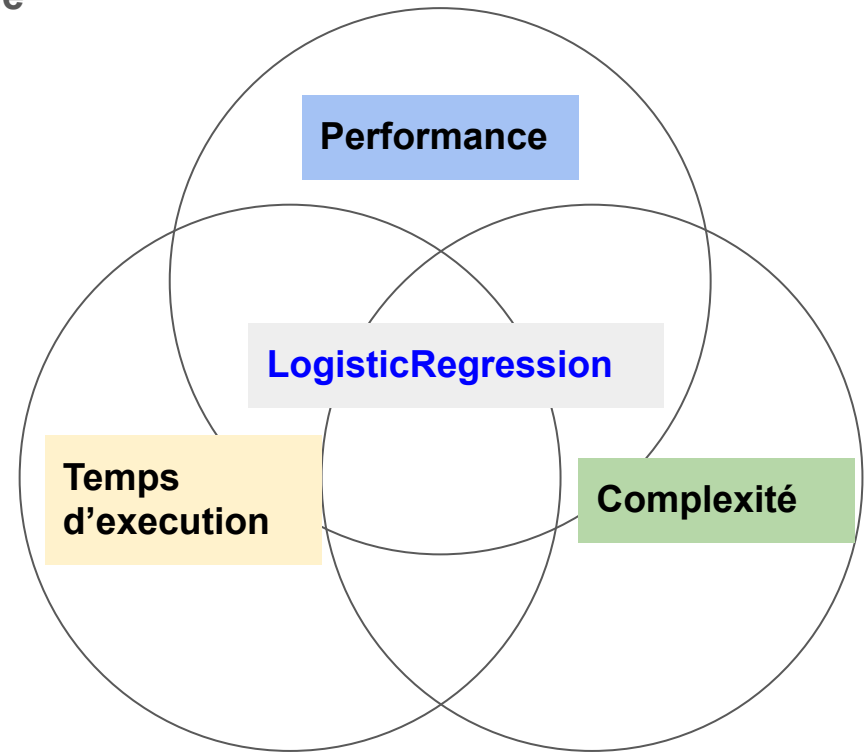
➢ Comparaison des résultats des modèles



Model	roc_auc_score_train	roc_auc_score_test	recall_score_test	f1_score_test	execution_time (hour)	fbeta_score_
LBGM	0.7759	0.7604	0.6673	0.2759	0.7358	0.6626
LogisticRegression	0.7498	0.744	0.6765	0.2611	0.0308	0.648

Modélisation

➢ Choix du modèle



Sommaire

Interprétation des prédictions

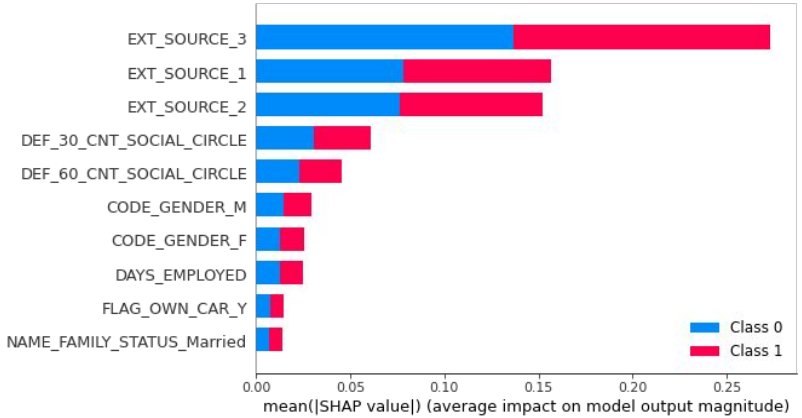
➢ Interprétation globale

○ Objectif

- obtenir une **compréhension générale** de l'importance des features utilisées par notre modèle pour calculer sa prédiction.
- Les coefficients d'une régression sont des **associations conditionnelles** permettant de quantifier la variation de *l'output* du modèle quand une feature donnée varie, en conservant les autres constantes.

○ Résultat

- Conformément à nos attente, nous constatons que les variables `EXT_SOURCES` sont les plus contributrices
- Suivies par `DEF_30_CNT_SOCIAL_CIRCLE` et `DEF_60_CNT_SOCIAL_CIRCLE`
  - ⇔ Nombre de **personnes en défaut dans l'entourage** des prospects.



Interprétation des prédictions

➢ Interprétation locale (SHAP)

- Afin d'expliquer chaque sortie du modèle, nous avons calculé les **valeurs de Shapley.** Les valeurs de Shapley proviennent de la *théorie des jeux* et permettent de **moyenner l'impact qu'une variable a** pour **toutes les combinaisons de variables possibles**.

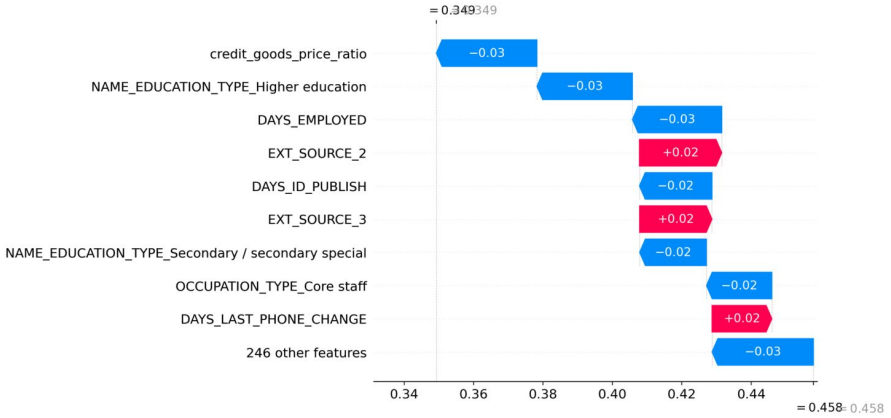
- La méthode SHAP (***SHapley Additive exPlanation***) permet ainsi de **décrire une prédiction** comme la **sommes des différents effets des variables** (appréciés par les valeurs de shapley) que l'on ajoute à la valeur de base du modèle (valeur moyenne prédite sur le dataset d'entraînement).

Interprétation des prédictions

3. **Interprétation**
   a. Interprétation globale des coefficients du modèle
   b. Interprétation locale des items (calcul des valeurs de Shapley)

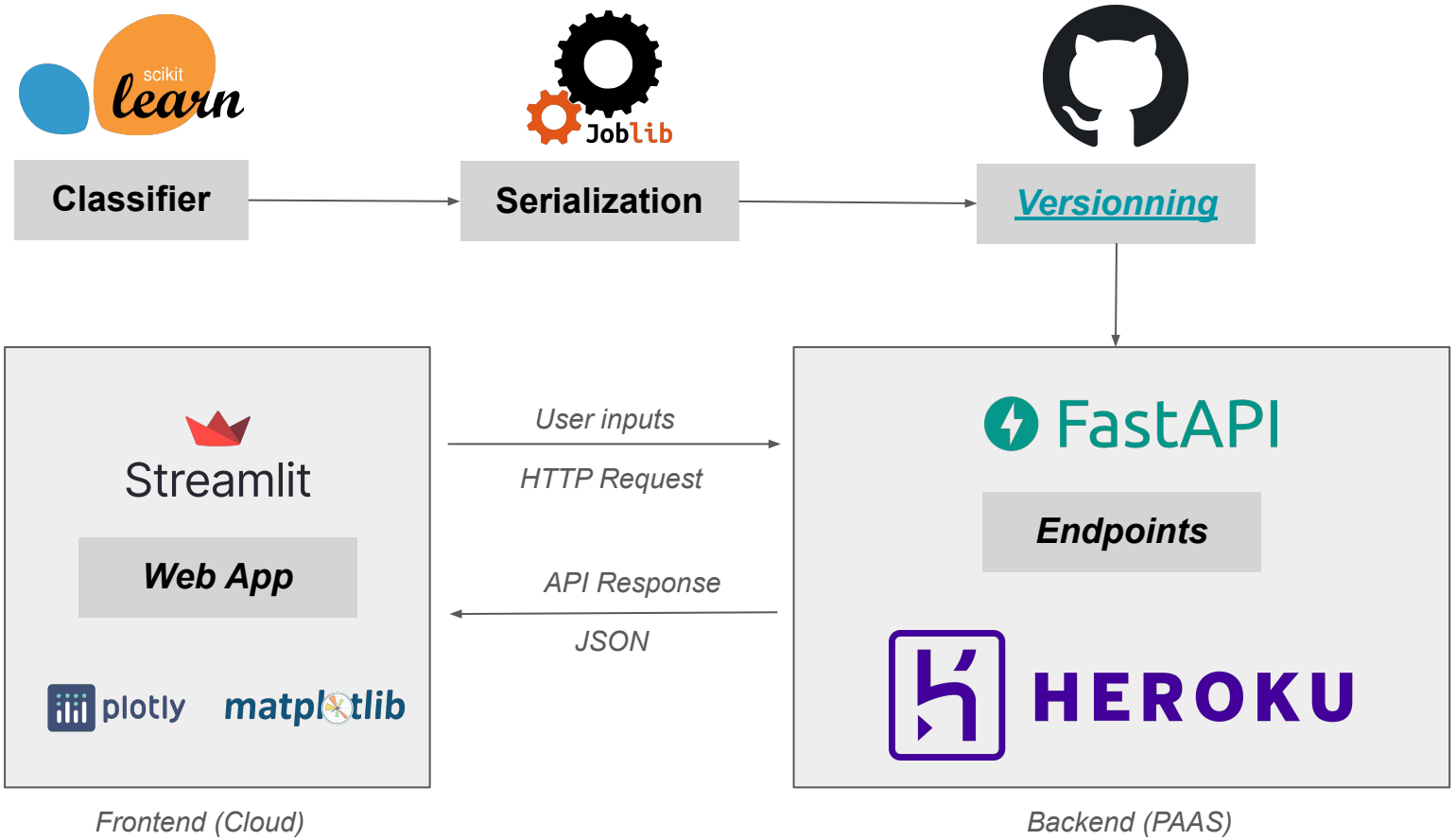
➢ Interprétation locale (SHAP)

- On peut ainsi visualiser pour un prospect donné, la décomposition de son score :
  - Valeur de base : 0.458
  - Prédiction : 0.349
- Interprétation des variables les plus contributrices :
  - `Credit_goods_price_ratio` : % du prêt servant à financer l'achat du bien. Ici le prêt correspond uniquement à une partie du financement du bien. Cela est **signe de bonne santé financière**, réduisant ainsi le score du prospect.
  - `NAME_EDUCATION_TYPE` : Le prospect a reçu une **Higher education** réduisant ainsi sa probabilité de défaut

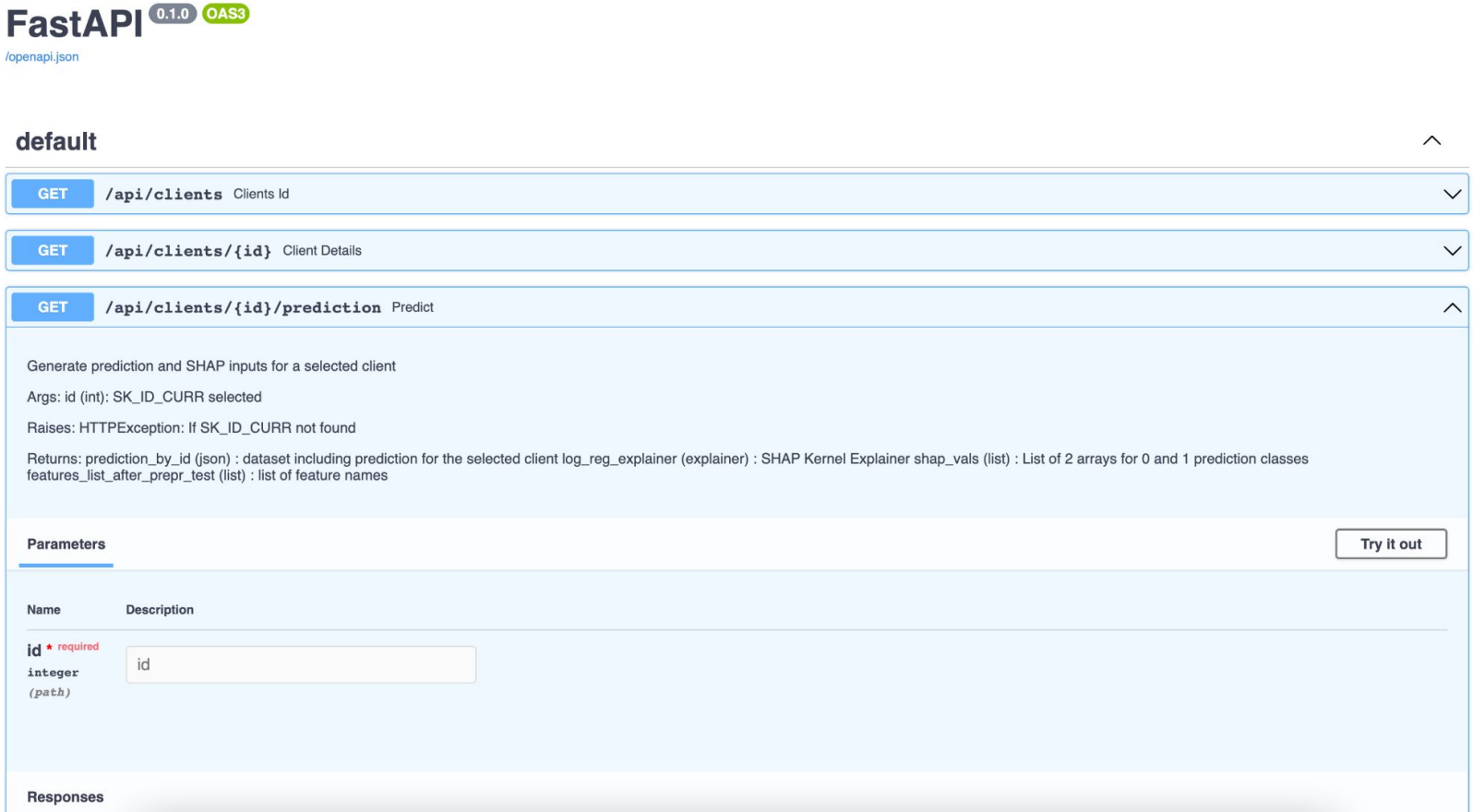


Sommaire

Architecture du Dashboard



Endpoints de l'API



Fonctionnalités du Dashboard

- Affichage des **données clients**
- **Requêtage de l'API** pour obtenir la **prédiction de score** client
- **Explication** du **score** client via SHAP
- **Comparaison** des variables clients par rapport à **la population d'entraînement**

Sommaire

Conclusion

➢ Limites et améliorations

○ Modélisation

- Exploitation de l'**ensemble de la base de données**
- Réduction de dimension (**PCA**) pour éviter la "curse of dimensionality"
- Amélioration des performances via des **modèles plus complexes** au dépend du temps de traitement *(LGBM, XGBoost)* puis faire de ***l'ensembling***

○ API

- Utilisation de **paramètres de requête** dans les URL pour une meilleure lisibilité
  - L'ID client devrait être fourni avec la syntaxe `?id=0`
- Construction d'une **base de donnée** afin de permettre à l'utilisateur de filtrer sur les champs souhaités
  - Via une requête SQL dynamique (ex : `SELECT * FROM data.db WHERE [column_to_filter] =?`)
- Création d'un **modèle Pydantic dynamique** pour encadrer le format des inputs et le documenter automatiquement

○ Dashboard

- Permettre à l'utilisateur de **filtrer sur plusieurs dimensions** pour **renforcer l'interactivité**