PARCOURS DATA Scientist
Projet 7

Implémentez un modèle de scoring

Sommaire:

- Contexte du projet
- Présentation du jeu des données
- Pre - processing
- Déséquilibre des données
- Modélisation
- API
- Dashboard
- Conclusion

Contexte du projet

En tant qu'un Data Scientist dans une société financière « Prêt à dépenser »

- mettre en œuvre un outil de "scoring crédit" pour calculer la probabilité en développant un algorithme de classification afin de classifier une demande d'un client ayant peu ou pas de crédit historique, sa demande accordée ou refusée

- Mise en place d'industrialisation d'un API via *Flask* en appliquant le modèle entraîné en amont

- Création d'un Dashboard interactif sous *Steamlit* afin de disposer un maximum de transparence de la décision de la société et offrir un service illustrative par le service client en face des clients

- Un déploiement par *Heroku* sur chacun du API et du dashboard afin de disposer aux conseillers bancaires

*source de données:fr.wikipedia.org

Présentation du jeux des données

8 fichiers disposé:
application_test.csv
bureau.csv
bureau_balance.csv
credit_card_balance.csv
installments_payments.csv
POS_CASH_balance.csv
previous_application.csv

HomeCredit_columns_description.csv

Avant le traitement:

- Taille des données: 30,700 lignes (clients); 522 variables
- Type d'information:
  - Information générale: revenue/ âge/ genre etc…
  - Information spécifique: nb de prêt précédant / durée / solde bancaire etc…

Pre – Processing & Feature Engineering

- **Un kernel de notebook Kaggle a été utilisé**
- **Chargement des données**
- **Fusion des fichiers**
- **Encodage des variables catégorielles** (OneHotEncoding)
- **Feature engineering**:
  - Montant remboursé par le client/ montant du crédit précédent (%)
  - Durée expérience professionnelle/ âge du client (%)
  - Revenu du client / montant total du crédit précédent (%)
  - Revenu du client / membre de la famille (revenu par personne)
  - Annuité du prêt/ revenu client (%)
  - Annuité du prêt/ montant du crédit (%)

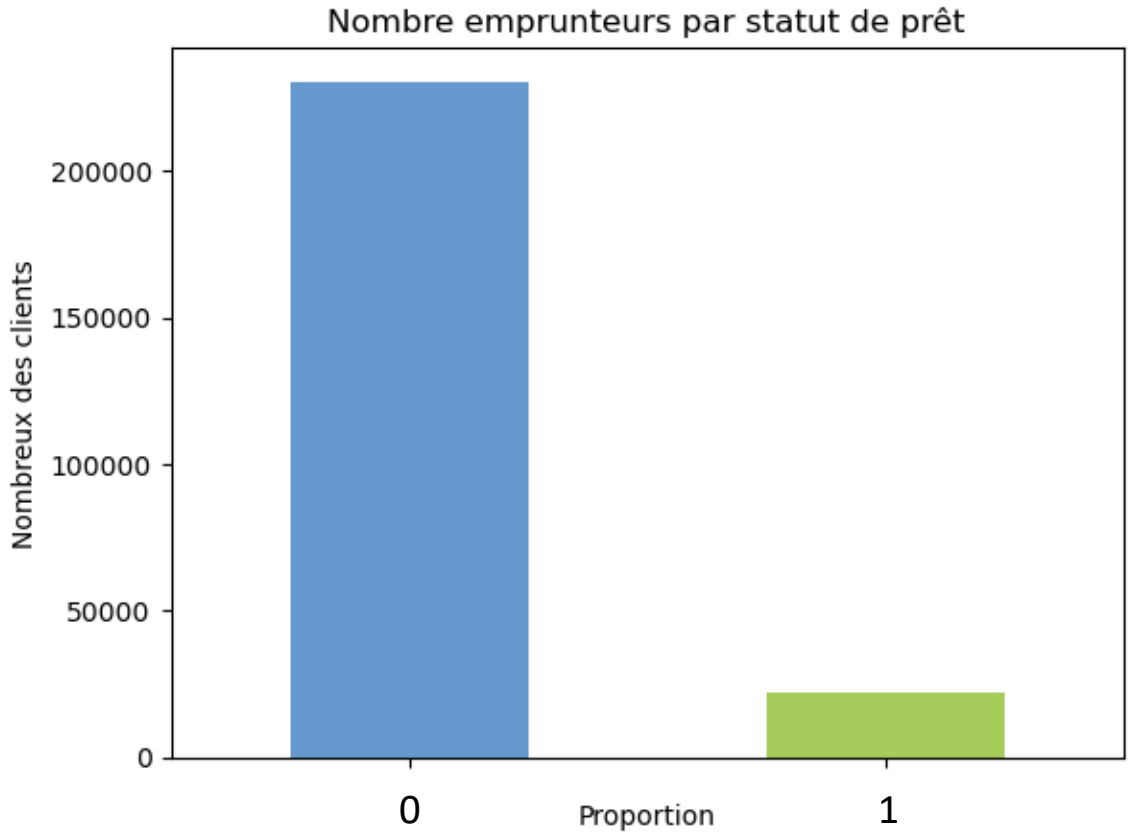
- **Sélection des données**
  (retirer des valeurs manquantes > 50% complétion)
- **Séparation des données** (80% apprentissage , 20% test)
- **Imputation des données**
  - variable numérique -> médiane
  - variable catégorielle -> le plus fréquent
- **Normalisation des données**

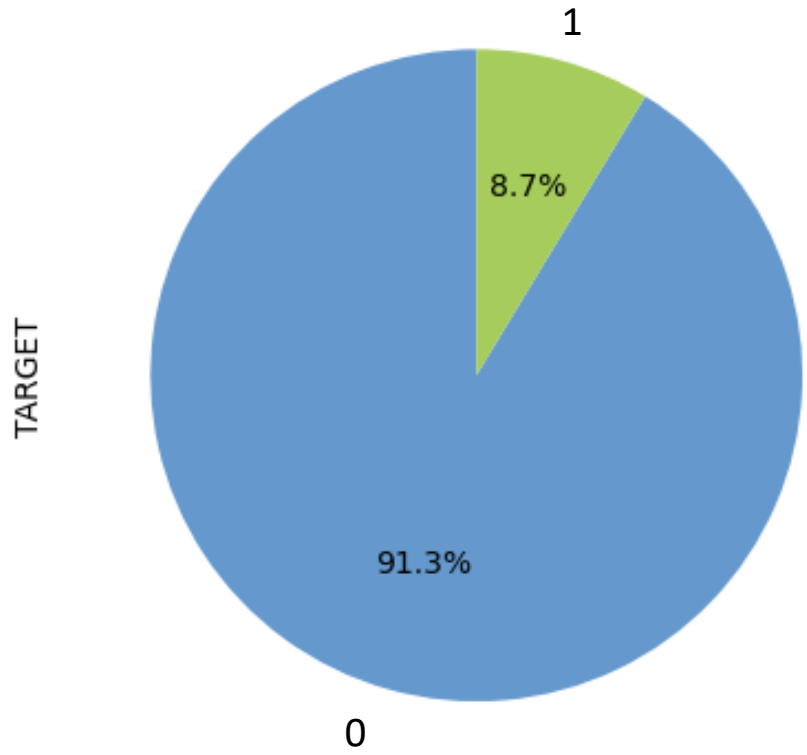
Après le traitement des données, nous avons obtenu un fichier en taille :

252133, 522

Déséquilibre des données

Répartition des clients des différents classes





un problème de classification binaire d'une base de données avec des classes déséquilibrées

« 0 » ---→91.3 % des individus ne seront pas capables de rembourser ses crédits contre ---→crédit non accordé

« 1 » ---→ 8.7 % des individus seront capables de rembourser ses crédits ----→crédit accordé

Modélisation

Recherche d'hyperparamètres (GridSearchCV):

- GridSearchCV simple
- GridSearch + SMOTE
- GridSearch + SMOTE + RandomUnderSampler

* SMOTE: sur échantillonnage aléatoire de la classe minoritaire
* RandomUnderSampler: sous échantillonnage aléatoire de la classe majoritaire

Algorithme:

- Dummy baseline
- Random Forest classifier
- LGBM classifier
- XGBoost
- Logistic Regression

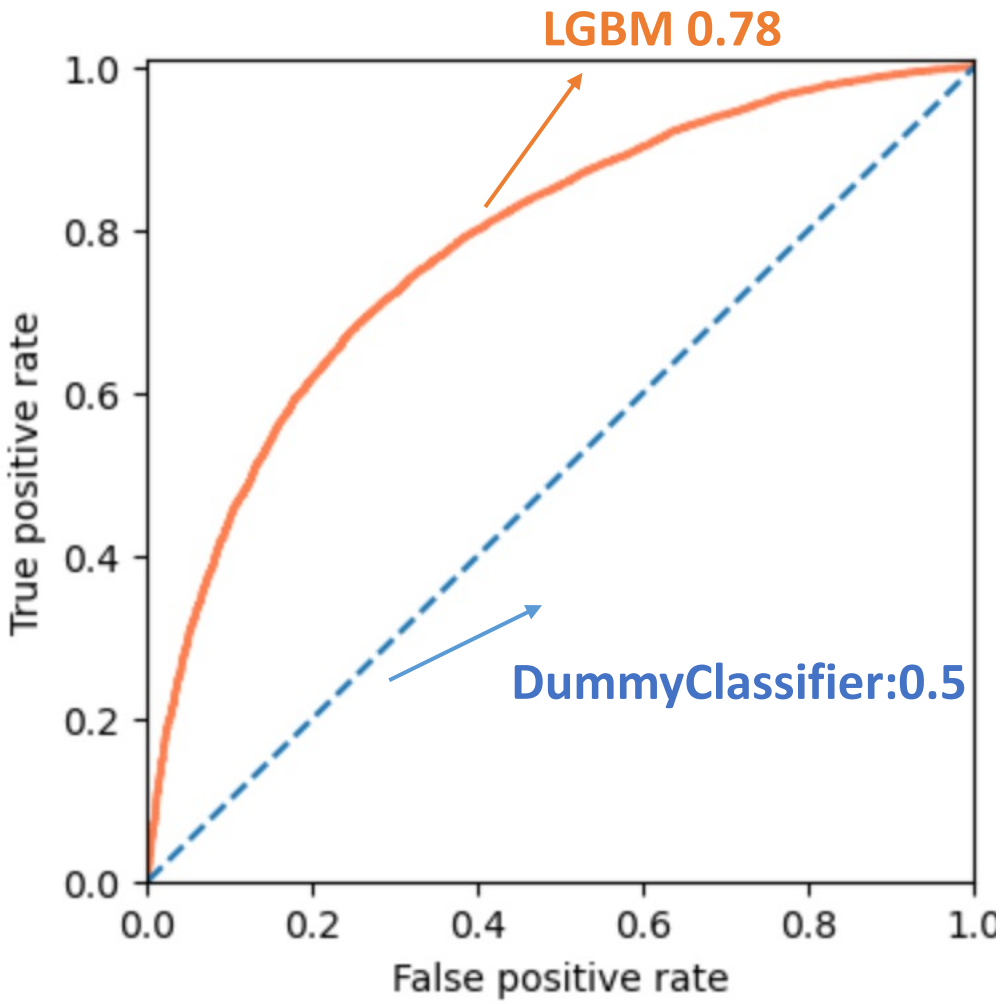
Modélisation

Modèle	Hypèparamètre
DummyClassifier	baseline
Random Forest Classifier	bootstrap': False, 'max_depth': None, 'min_samples_leaf': 4, 'n_estimators': 200
Light GBM Classifier	max_depth': 5, 'n_estimators': 300, 'num_leaves': 20
Extreme Gradient Boosting	learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 300
Logistic Regression	C': 29.763514416313132, 'penalty': 'l2'

Modélisation – Métrique d'évaluation

Métrique d'évaluation:

- RMSE:
- MSE:
- RSE:
- **ROC_AUC**:
- Recall:
- Precision :
- Score F1 :



Modélisation – GridSearch

Modele	Dummy Classifier	Random Forest	LGBM	XGBoost	Logistic Regression
Score de Cross-Validation		0.744457	0.777414	0.778557	0.732256
Score de test	0.913677	0.745732	0.782245	0.783034	0.734438
RMSE	0.293807	0.269664	0.265215	0.265058	0.271892
MSE	0.086323	0.072719	0.070339	0.070256	0.073925
RSE	1.094478	0.921996	0.891818	0.890763	0.937288
AUC	0.5	0.745732	0.782245	0.783034	0.734438
Rappel	1.0	0.999978	0.996527	0.996397	0.997439
Precision	0.913677	0.913748	0.916941	0.916931	0.915039
F1	0.954892	0.95492	0.955079	0.955014	0.954464

Fonction de coût métier

- Accorder un crédit à un client ne pouvant pas le rembourser (FP) ----> perte
- Accorder un crédit à un client pouvant rembourser (TP) ----> gain
- Ne pas accorder un crédit à un client ne pouvant pas rembourser (TN) ----> neutre
- Ne pas accorder un crédit à un client pouvant rembourser (FN) ----> perte

Un client étiqueté comme **faux positif** (fp) coûte 10 fois plus cher qu'un **faux négatif** (fn).

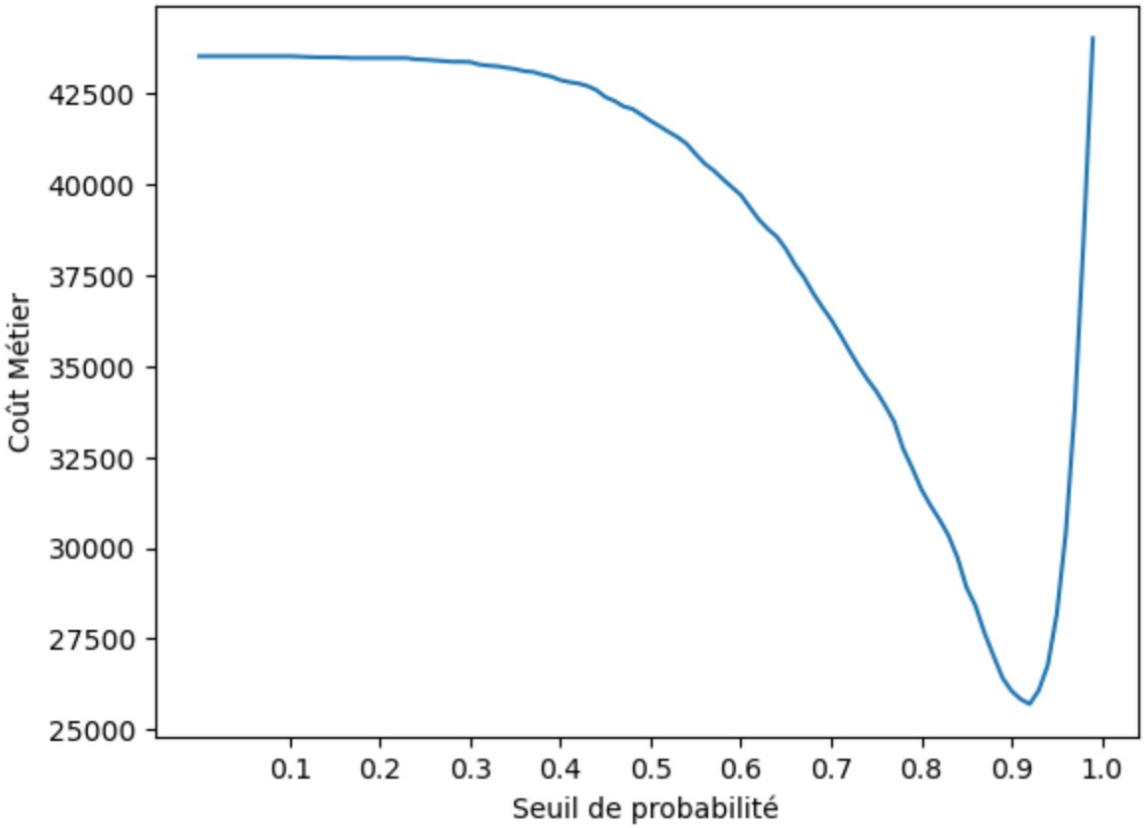
La fonction de coût métier doit pénaliser 10 fois plus les **faux positifs**.

Nous avons donc la fonction de coût suivante :

$$coût\ métier = 10 * nombre\ de\ fp + 1 * nombre\ de\ fn$$

Fonction de coût métier

Calcul d'un seuil de probabilité



Définition de la fonction :

•Faux positifs (fp) : Le crédit est accordé à un client non solvable

► perte d'argent

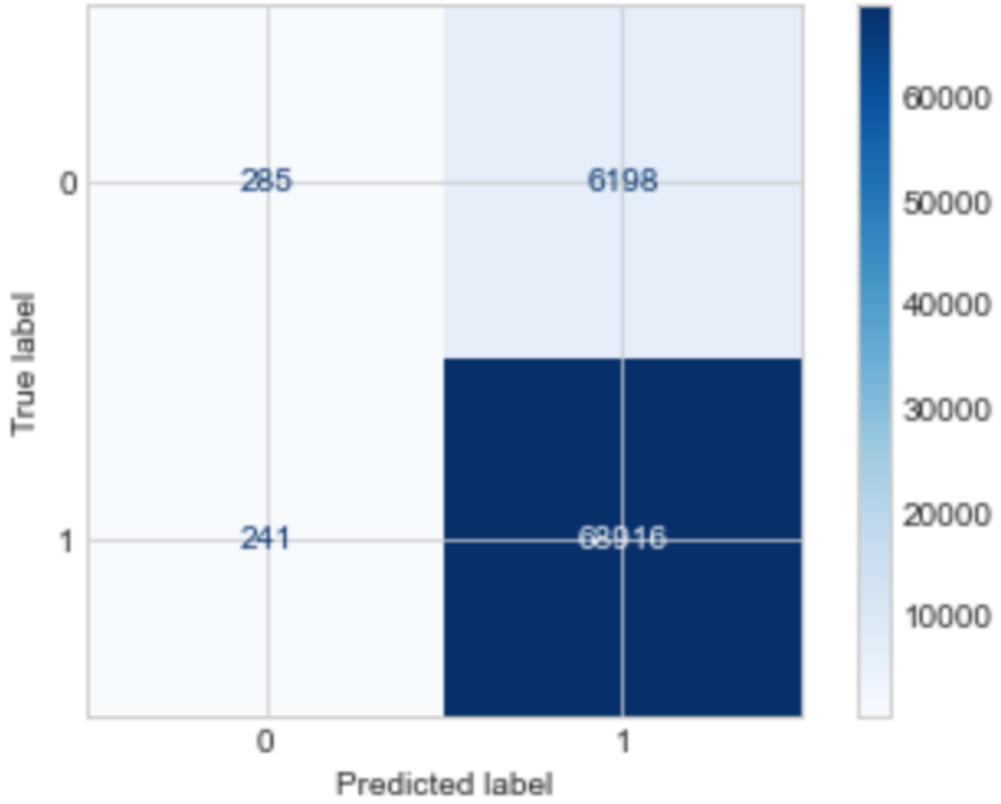
•Faux négatifs (fn) : Le crédit n'est pas accordé à un client solvable

► manque à gagner

•$coût\ métier = 10 * nombre\ de\ fp + 1 * nombre\ de\ fn$

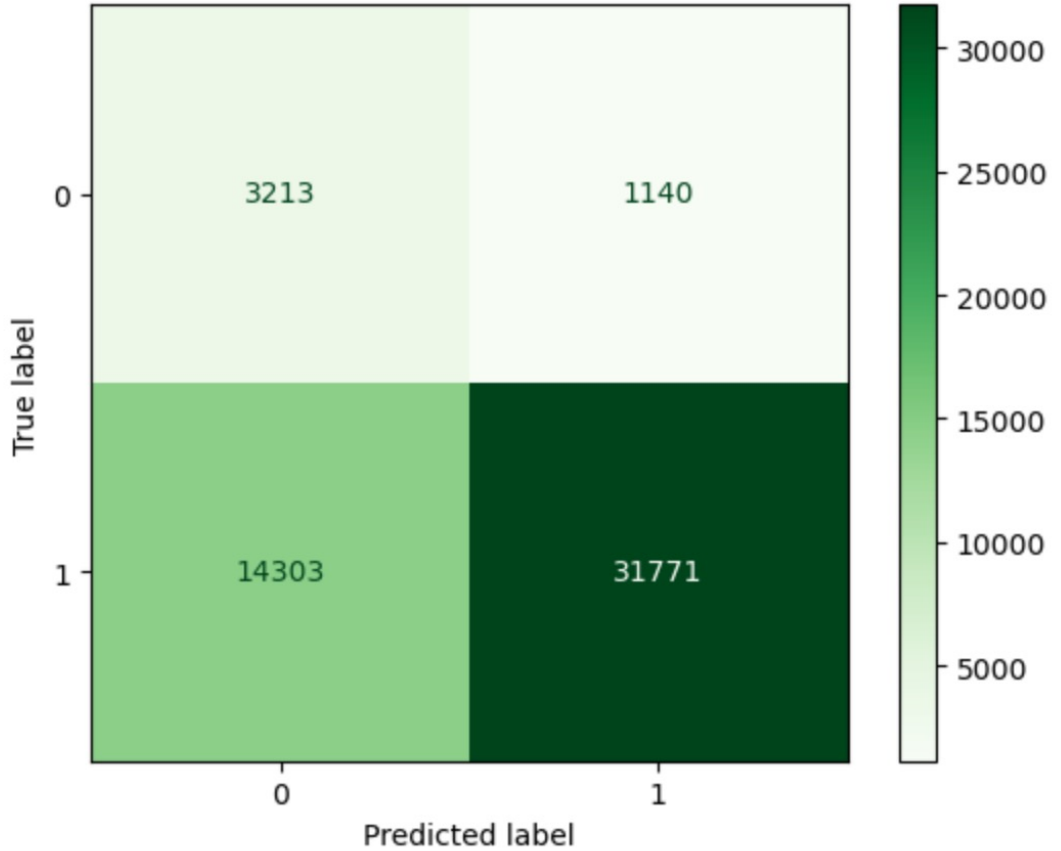
Optimisation du coût métier

Matrice de confusion du meilleur modèle opérationnel:



Coût métier :41750

Matrice de confusion avec l'optimisation du coût métier :

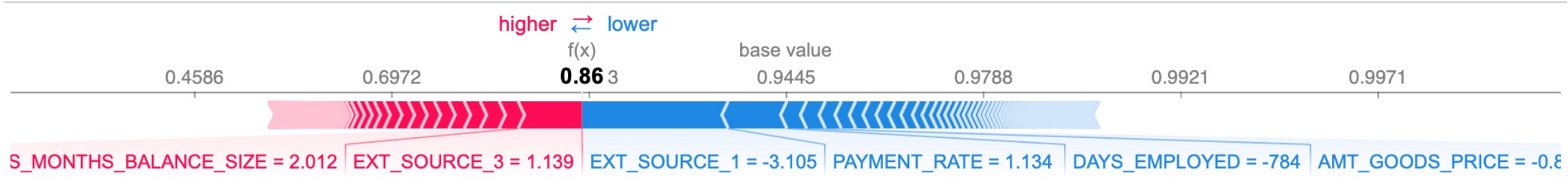


Coût métier :25703

Interprétation du modèle

Variable locale

Ex: ID 60



Ex: ID 60

Les variables en rose ont contribué à accorder le crédit (donc à augmenter le score). – e.g:EXT_SOURCE_3
Les variables en bleu ont contribué à refuser le crédit (donc à diminuer le score)- e.g: EXT_SOURCE_1

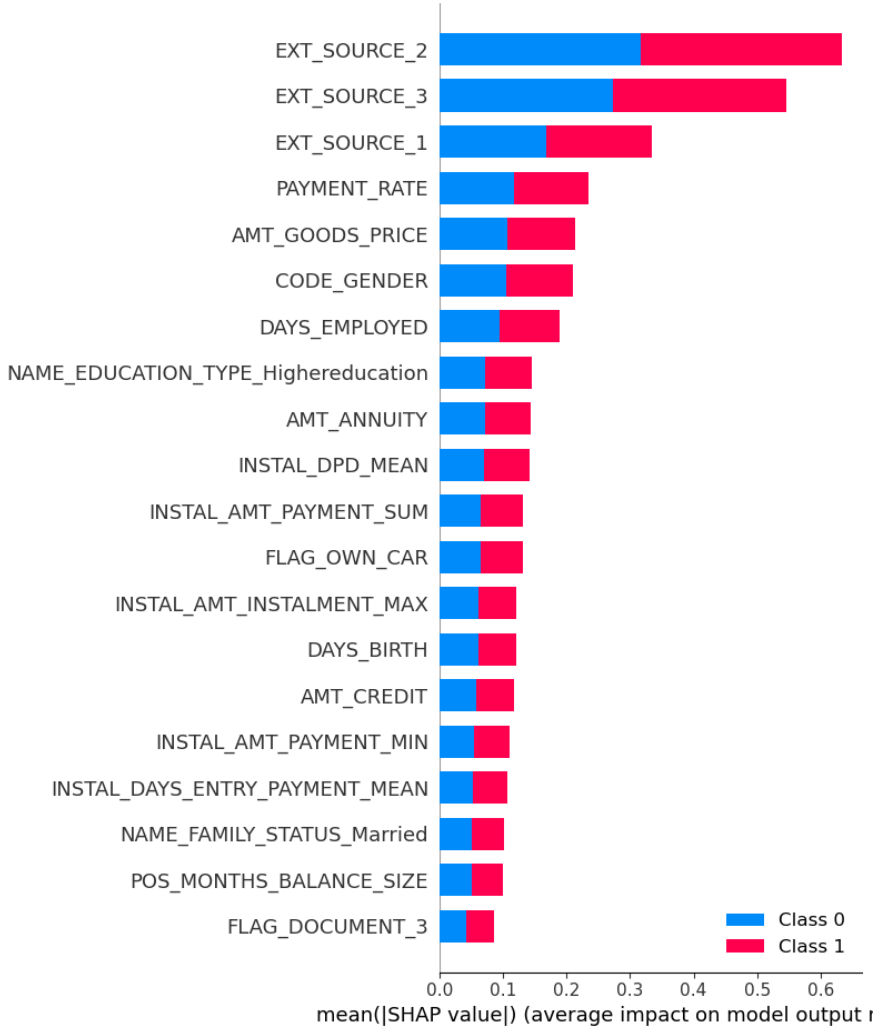
Interprétation du modèle

Variable globale

Ex: ID 60

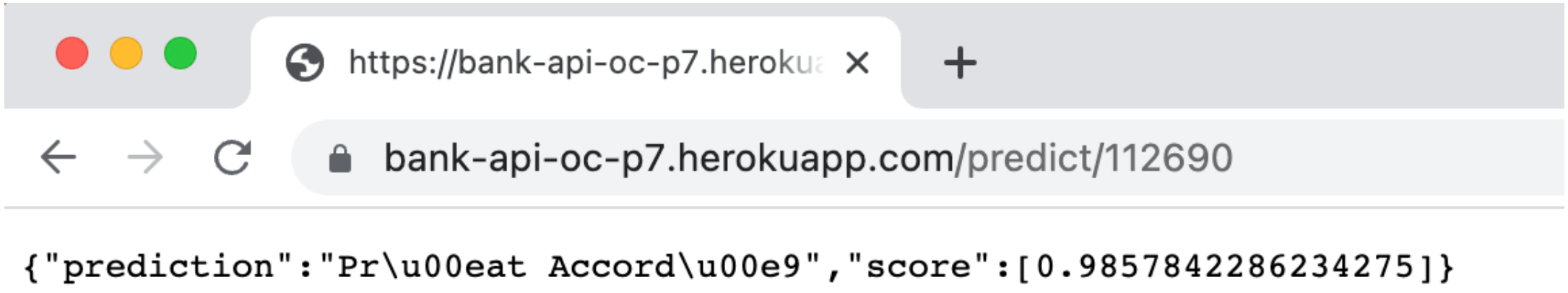
- Pour chacune des variables et chacune des classes : Calcul de la contribution à l'amélioration ou à la diminution du score

- Classement de l'importance des variables par l'ordre décroissant



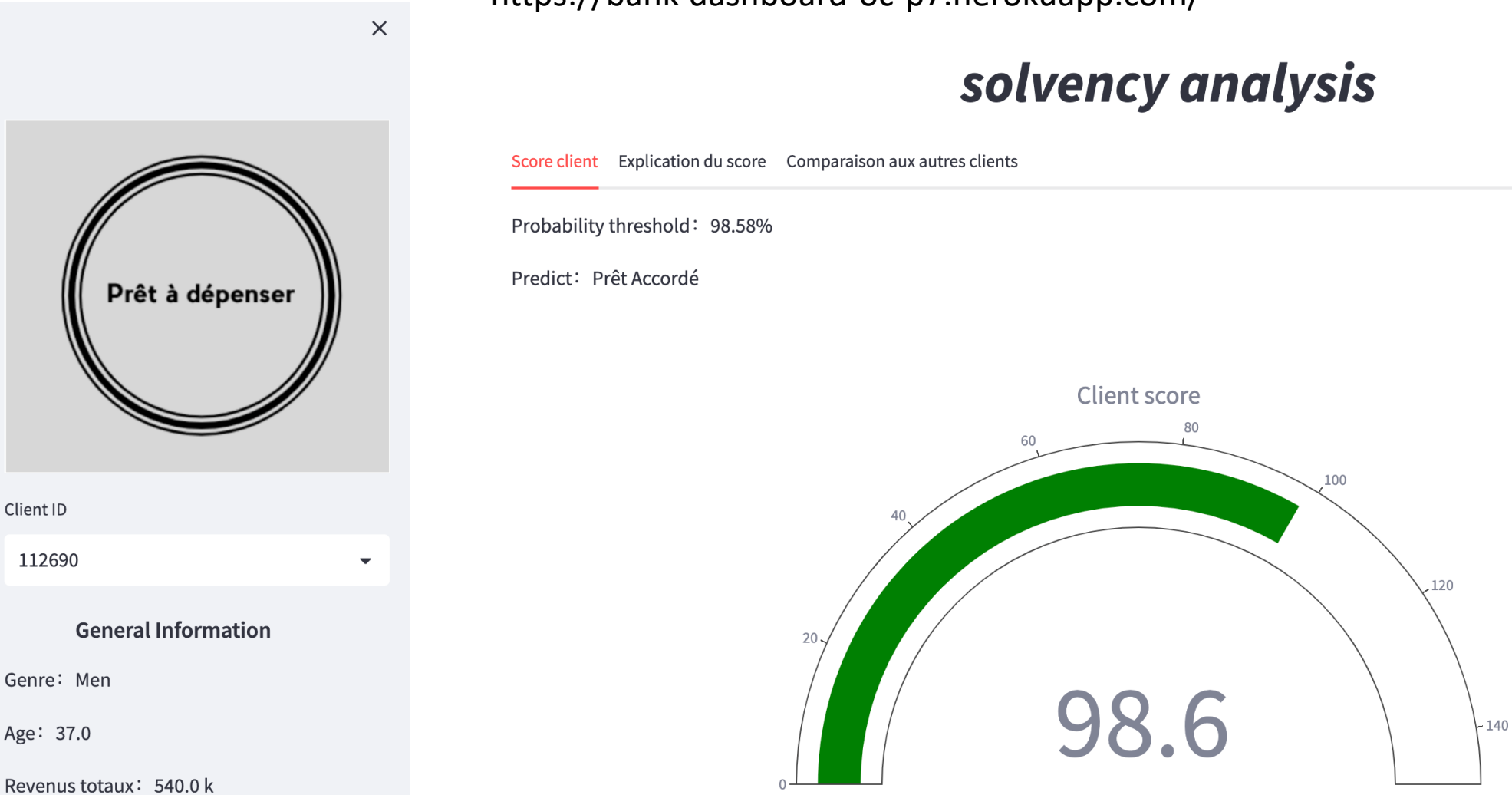
API – outil Flask

Une API créé via ***Flask*** puis déployé par ***Heroku*** afin de réaliser une prédiction de solvabilité d'un client donné de façon automatique



Dashboard – outil Steamlit

https://bank-dashboard-oc-p7.herokuapp.com/



Conclusion

- Nous avons développé un outil permettant de donner une prédiction du prêt simple et rapide.

- Cela remonte l'optimisation du coût métier au niveau de la simulation du modèle

- Voir à l'avenir un développement de son propre feature engineering

- on a remarqué dans ce projet un manque des variables pertinents (les données de conjoint / d'autre données de la personne etc) donc dans l'avenir il est possible de faire un traitement de feature enginnering plus poussé afin d'améliorer la performance de notre modèle

MERCI DE VOTRE ATTENTION