

Implémentez un modèle de scoring

Projet 7 du parcours « Data Scientist » d'OpenClassrooms

Mark Creasey

Sommaire

Implémentez un modèle de scoring

- 01 La problématique
- 02 Les données
- 03 Modélisation
- 04 Dashboard
- 05 Conclusion

01 Présentation de la problématique

Mission – Implémenter un modele de scoring

La société financière « Prêt à dépenser » propose:

- de crédits à la consommation
- pour les personnes ayant peu ou pas de tout d'historique de prêt

Basée sur les données financières

- Données internes
- Données externes

Critères de succès

- Transparence de la décision sur l'octroi du crédit
- Deploiement d'un dashboard permettant de visualiser les informations clients pour
- Permettre l'interprétation la décision fait par la modèle



02 Les données

Nettoyage, exploration

Données financières sous 7 tables

bureau.csv

Application data from previous

loans that client got from other

institutions and that were

reported to Credit Bureau

SK ID BUREAU

bureau balance.csv

Monthly balance of

credits in Credit

Behavioral data

Bureau

Credit Bureau

One row per client's loan in

Feature engineering script:

 https://www.kaggle.com/jsaguiar/lightgbmwith-simple-features/script

application_{train|test}.csv Main tables – our train and test samples Target (binary) Info about loan and loan applicant at application time SK_ID_CURR -SK_ID_CURR SK_ID_CURR previous_application.csv Application data of client's previous loans in Home Credit Info about the previous loan parameters and client info at SK ID CURR time of previous application One row per previous application -SK_ID_PREV SK_ID_PREV credit_card_balance.csv POS_CASH_balance.csv instalments_payments.csv Past payment data for each · Monthly balance of · Monthly balance of installments of previous credits client's previous client's previous in Home Credit related to loans loans in Home Credit credit card loans in

in our sample

Behavioral data

Home Credit

Behavioral data

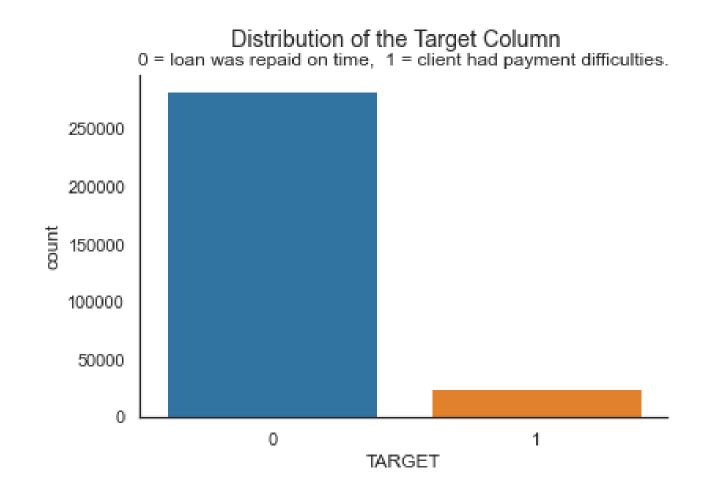
Behavioral data

Distribution de la variable cible très déséquilibré

Prédiction que tous les clients sont bons

- → précision de 93%
- → on aura identifié aucun client défaillant.

- 0 clients non-défaillant
- 1 clients défaillants

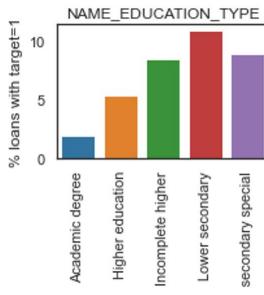


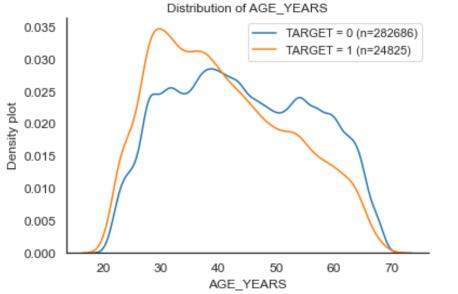
Exploration des features importants

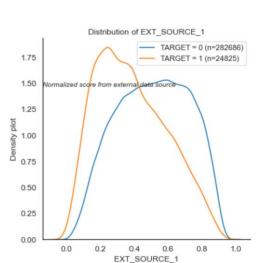
Facteurs de risque (entre autres)

- Les hommes
- < 40 ans
- Bas niveau d'éducation
- Un score <0.5 dans des sources externes
 - o EXT_SOURCE_1
 - EXT_SOURCE_2
 - o EXT_SOURCE_3









03. Modélisation

Sampling: Re-équilibrage des classes

Aucun

SMOTE

Cost- sensitive

• Random under/over sample

SMOTE Tomek links

Métriques d'evaluation

Précision

 Quelle portion des clients prédit comme défaillant sont du vrai classe défaillant?

$$\frac{TP}{TP + FP}$$

Recall

• Quelle portion du vrai classe sont présent dans le cluster prédit ? $\frac{TP}{TP + FN}$

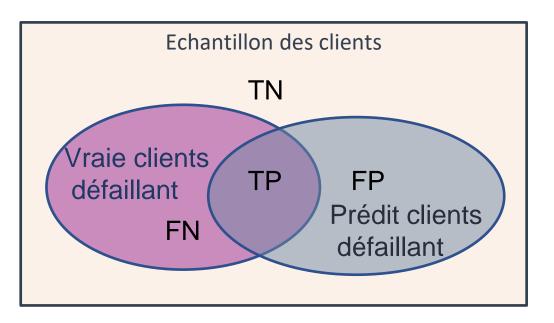
F1 Score

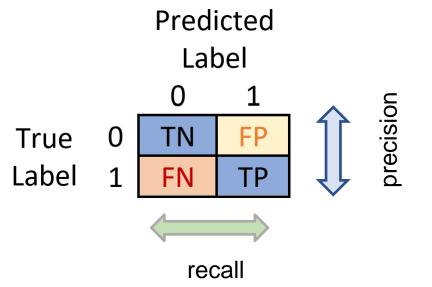
accuracy « equilibré » :

$$2*\frac{Precision*Recall}{Precision+Recall}$$

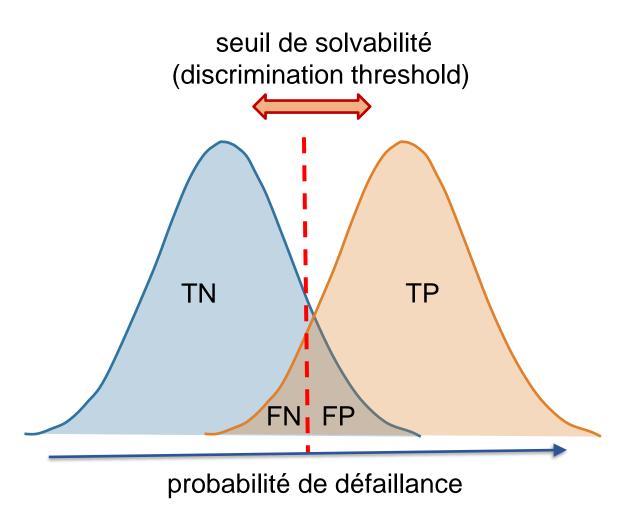
F(beta) score

Pèse plus sur recall

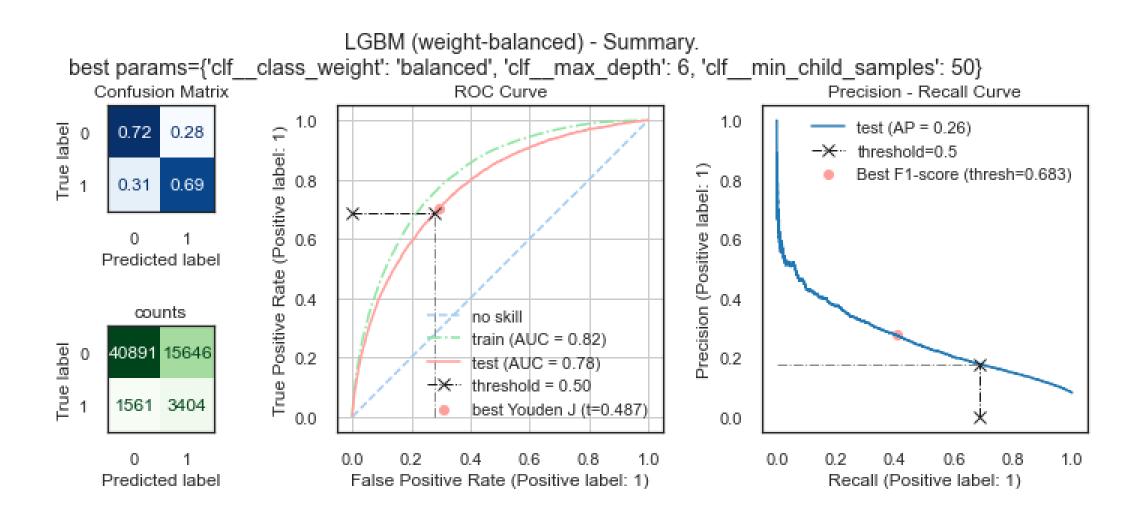




La fonction cout metier



Evaluation des modèles



Les modèles (classifiers)

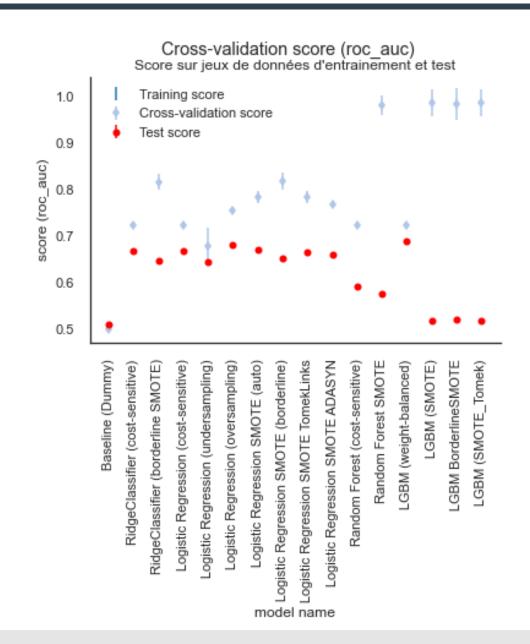
Pipelines imblearn

- Preprocess
- Sampling
- Feature Selection
- Classification

Classifiers

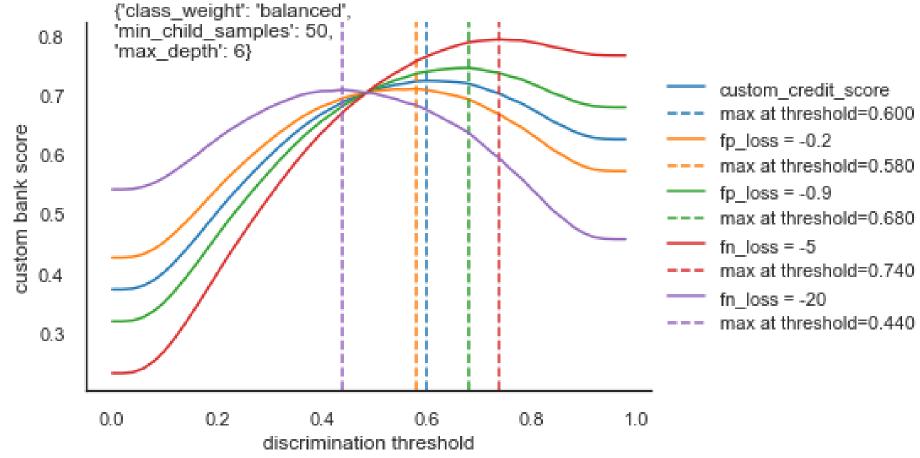
- Dummy
- Ridge
- LogisticRegression
- RandomForest
- LightGBM

Choix du modèle

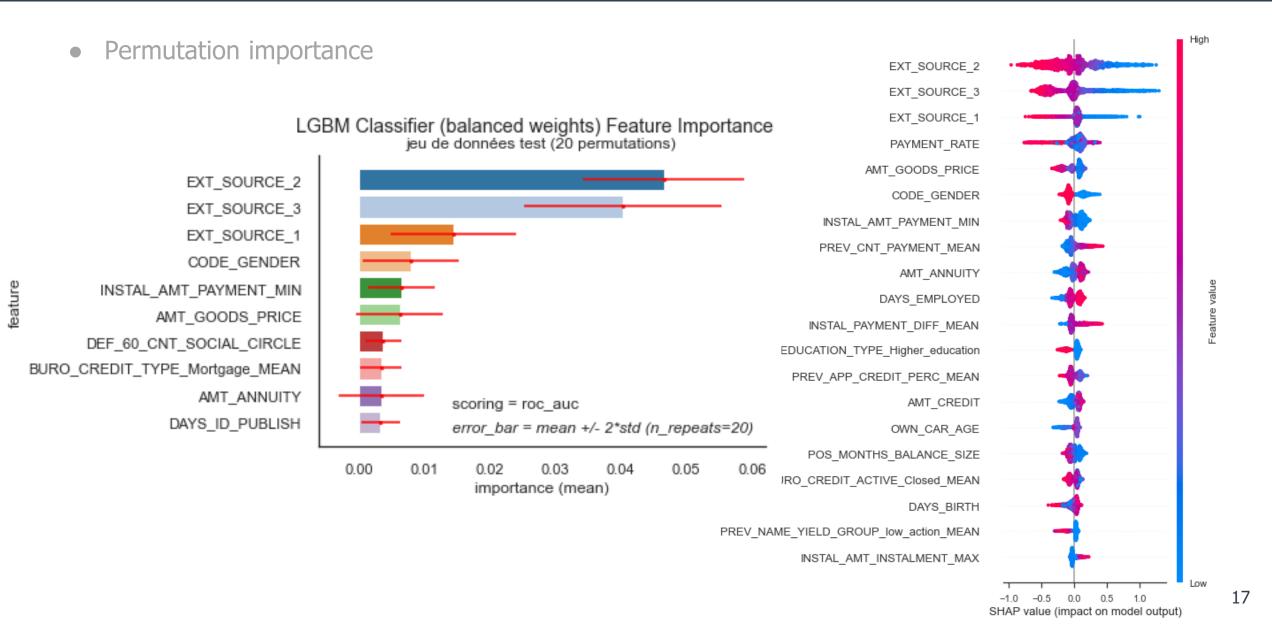


Optimisation du seuil (discrimination threshold)

GBM Classifier (balanced weights) : Effet des couts des faux negatifs et faux positifs sur le seuil de probabilité optimal.

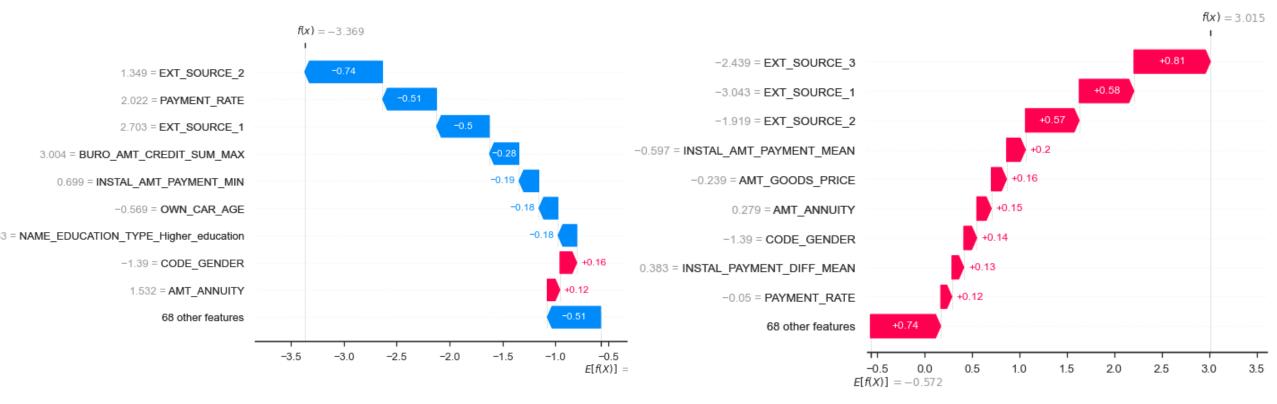


Feature Importances - globale



Interpretabilité du modèle

Bon



04 Dashboard

API du dashboard

Fonction

- Gere l'accès aux prédictions
- Gere l'accès aux données (interpretabilité)
- Fournir un interface publique pour plusieurs clients

REST API

Réponses json aux requetes GET

Code source dans dossier **api** sur dépôt : https://github.com/mrcreasey/oc-ds-p7-scoring-dashboard

Instructions pour développement

Voir README.md dans dossier api

Deploiement sur heroku

https://mc-oc-7.herokuapp.com/

Reste à Faire:

- Documentation (note: fastapi : autodocumentation)
- JWT authentication
- Cache des données en memoire

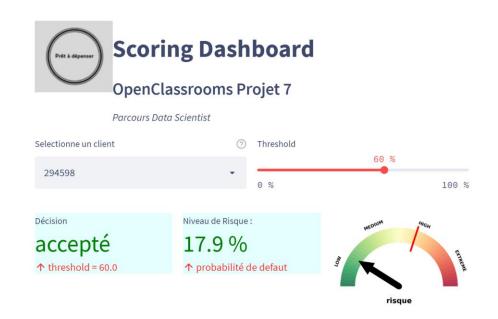
Visualisation du dashboard

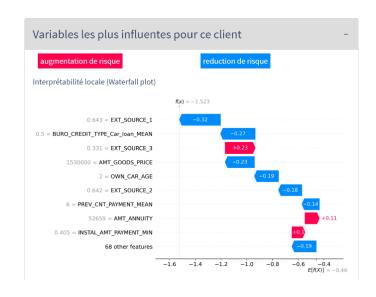
Fonction:

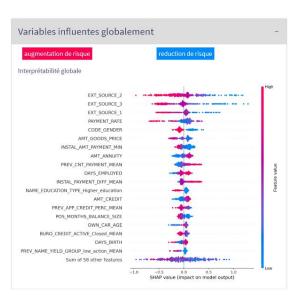
Visualisation des prédictions et données fournies par l'api

lien vers dashboard deploiement:

https://mrcreasey-oc-ds-p7-scoring-dashboard-dashboardmain-70agjx.streamlitapp.com/







05 Conclusion et améliorations à faire

Conclusions

Améliorations à faire

Modélisation

- Feature engineering avec experts du métier
- Revoire stratégie de traiter les valeurs manquantes
- Adapter la sélection de features à chaque modèle (Wrapper/Embedded)
- Optimiser la taille d'echantillons en analysant les learning curves des modeles
- Augmenter la recherche des meilleurs hyperparamètres des modèles

Deploiement

- Changer de Flask API vers fastapi (https://fastapi.tiangolo.com/)
- Ajout authentification pour accéder au dashboard
- Ajout encryptage des données client
- Stocker les données client séparément de l'API - par exemple dans un S3 bucket sur AWS
- Visualiser la position du client dans les distributions des features les plus importantes pour ce client

Questions

images: Mark Creasey

mrcreasey@gmail.com

Code source: https://github.com/mrcreasey/oc-ds-p7-scoring-dashboard

Merci !