Implémentation d’un modèle de scoring

OPENCLASSROOMS

Etudiant : Raphaël PUIG

Evaluateur : Mously DIAW

Mise en contexte

- L’entreprise *Prêt à dépenser* propose des crédit à la consommation à des clients ayant peu ou pas du tout d’historique de prêt.

- Nécessité d’estimer le risque d’échec de remboursement.

- **Objectifs :**
  1 - Implémenter un modèle prédictif de la probabilité de non remboursement
  2 - Rendre les résultats accessibles *via* une application interactive

Plan

- ❏ Présentation des données
- ❏ Réduction de la base de données
- ❏ Sélection du modèle
- ❏ Choix des variables d’intérêt
- ❏ Optimisation du modèle
- ❏ Présentation de l’application web
- ❏ Conclusion et perspectives

Présentation des données

- Données initiales exploitables :
  - 8 tableaux de données au format CSV (Home Credit Default Risk | Kaggle)
  - Informations strictement financières : revenus, montant du prêt, mensualités, etc.
  - Informations “étendues” : âge, secteur professionnel, statut marital, type de logement, etc.
  - Variable cible TARGET binaire : 0 = crédit remboursé, 1 = échec de remboursement.

- **Des centaines de variables** qu’il convient d’agréger en une seule base.
  - Certaines base comptent plusieurs lignes par client → à résumer en une seule ligne
  - Limiter la perte d’information lors du processus (occurrence, minimum, maximum, moyenne)

Après agrégation : ~ 1900 variables ⇒ nécessité de réduire la base de données

Réduction de la base de données

- Suppression des variables vides à +75%
  - Seulement 90 variables supprimées (peu efficace)
  - Choix arbitraire
- Suppression des variables corrélées à +90%
  - Variables faisant en quelque sorte “double emploi”
  - Beaucoup plus de variables supprimées (~700)

Base de données réduite à ~1100 variables → encore trop large ⇒ faire un choix

Sélection du modèle

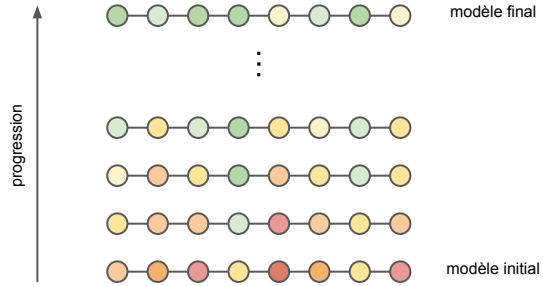
- Plusieurs modèles essayés a priori sur la base application_train.csv
- Modèles de classification basés sur des arbres de décisions de faible profondeur (apprenants dits “faibles”).

Modèles séquentiels	Modèles parallèles
AdaBoost	Extra Trees
GradientBoosting	Random Forest
Light Gradient Boosted Machine	Autre : DecisionTree

Sélection du modèle

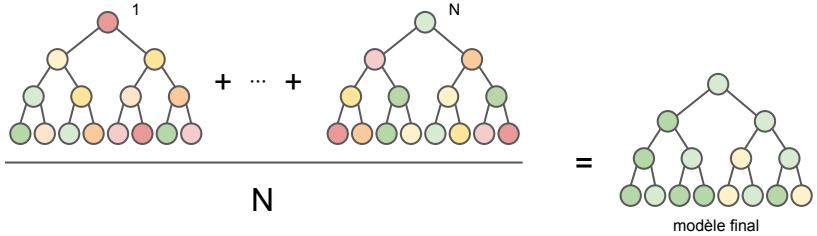
Modèles séquentiels (boosting)

Un seul modèle est créé puis successivement amélioré en pénalisant les coefficients qui réduisent ses performances.



Modèles parallèles (random forest)

Un très grand nombre de sous modèles est créé simultanément puis moyenné pour créer un modèle final.



Sélection du modèle

- Critères de décision :
  - Performance sur la base de validation (valeur moyenne du *ROC AUC score*)
  - Écart de performance entre la phase d'entraînement et de validation (*train-valid gap*)
  - ⚠️ Temps de calcul ⚠️

	Ada Boost	Gradient Boosting	LGBM	Decision Tree	Extra Trees	Random Forest
Validation ROC AUC score	0.740	0.749	0.752	0.537	0.702	0.713
Train-valid gap	0.006	0.009	0.049	0.463	0.298	0.287
Computation time	3’ 20”	13’ 36”	0’ 19”	1’ 00”	4’ 43”	3’ 53”

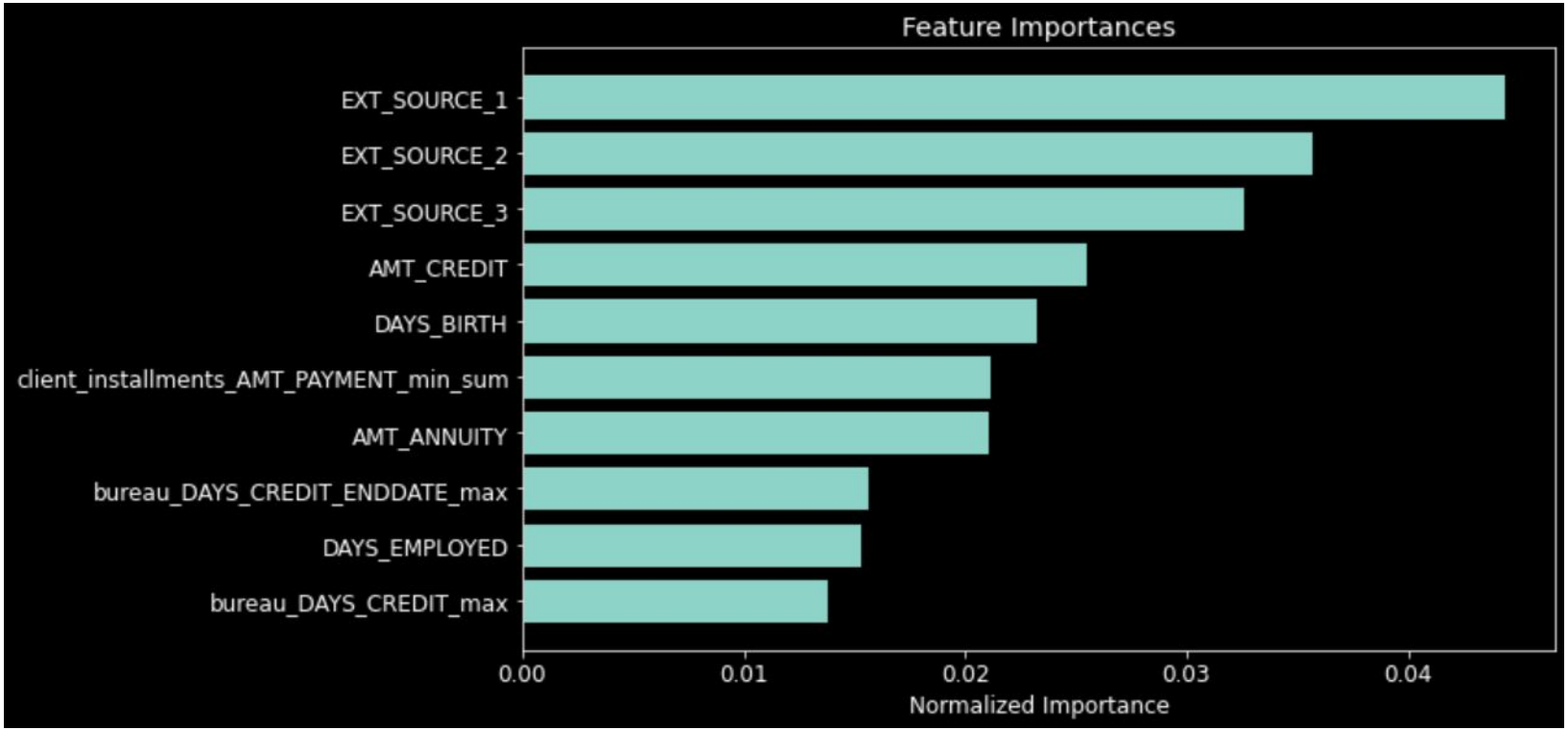
Temps de calcul très avantageux pour Light Gradient Boosted Machine Classifier.

Choix des variables d'intérêt

- ❏ Utilisation de la fonctionnalité *Features importance*
- ❏ Valeurs par défaut des hyperparamètres du modèle
- ❏ Succession entraînement + validation → identification des variables d'importance nulle → suppression des variables concernées
- ❏ Réduction progressive de la base jusqu'à stabilisation (i.e. aucune variable identifiée comme d'importance nulle)

Base réduite à 498 variables → la base est à présent exploitable.

Choix des variables d’intérêt



Optimisation du modèle

- LGBMClassifier (hyperparamètres par défaut) + 498 variables
  - ROC AUC score moyen (validation) : 0.776
  - ROC AUC score *gap* (train vs valid) : 0.051
  - Temps de calcul : 2’30”
- Objectif : identifier la combinaison des hyperparamètres permettant
  - Réduire autant que possible le *gap* entre la phase d’entraînement et de validation.
  - Augmenter (si possible) le ROC AUC score de validation.
- Le nombre d’hyperparamètres à ajuster est trop important pour une recherche systématique (temps de calcul vraiment prohibitif).

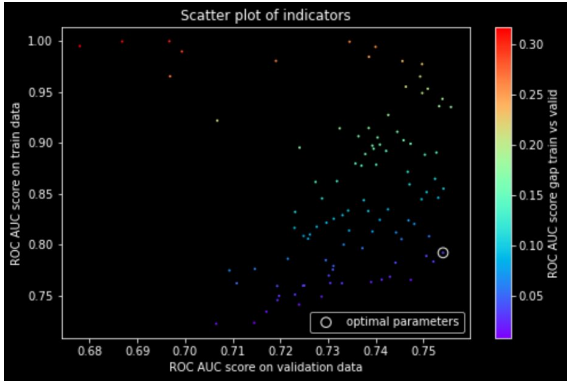
Optimisation du modèle

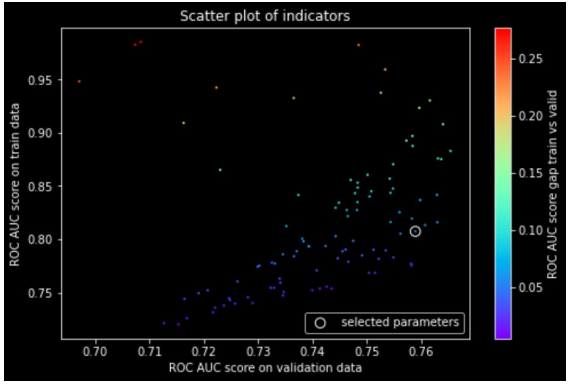
- Recherche en deux temps :
  - Une recherche sur un nombre limité de combinaison composées aléatoirement.
  - Une recherche systématique autour de la combinaison la plus prometteuse.
- Nécessité de passer par un échantillon de la base de données.
  - Le résultat de la *random search* dépend de la taille de l'échantillon.
  - Plusieurs échantillons sont considérés : (3,25%), 10% et 20% de la base.
- Des compromis sont nécessaires :
  - Taille de l'échantillon d'entraînement
  - Nombre de combinaisons
  - ⚠️ Temps de calcul ⚠️

Optimisation du modèle

10 %

100 points
23 min

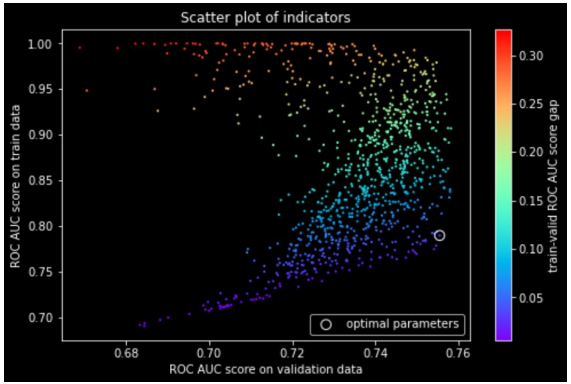


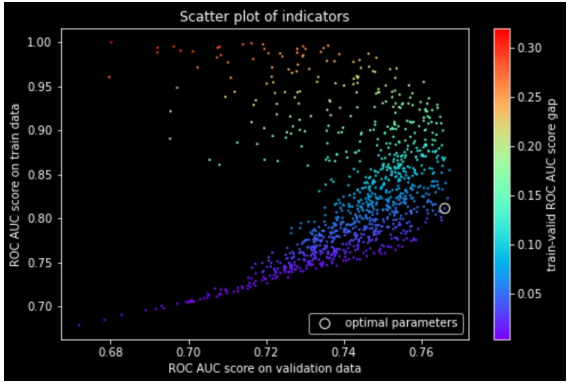


20 %

100 points
40 min

1000 points
3 h 50 min



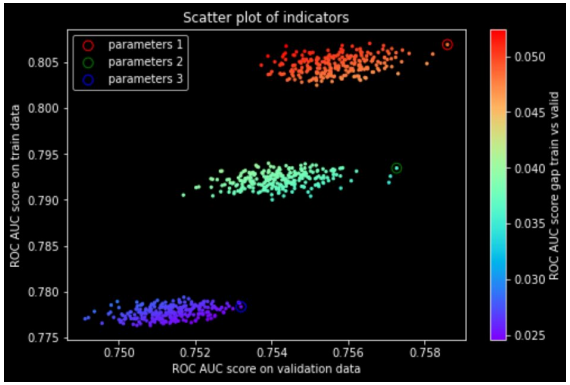


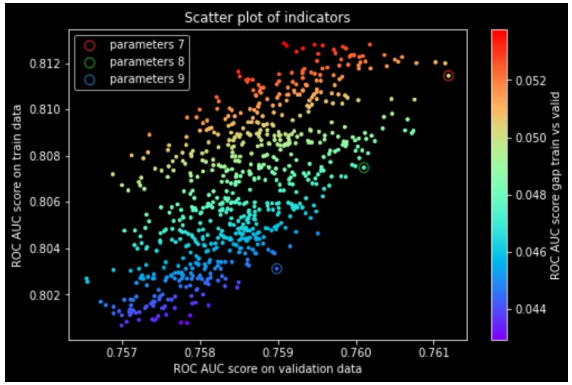
1000 points
6 h 30 min

Optimisation du modèle

10 %

100 points
2 h 13 min

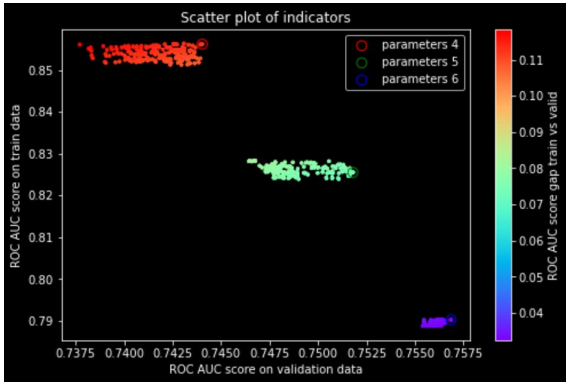


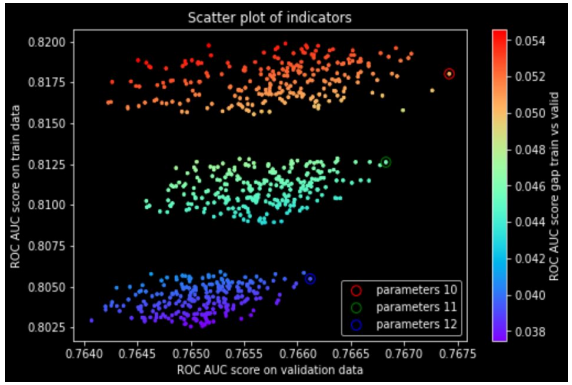


20 %

100 points
12 h 57 min

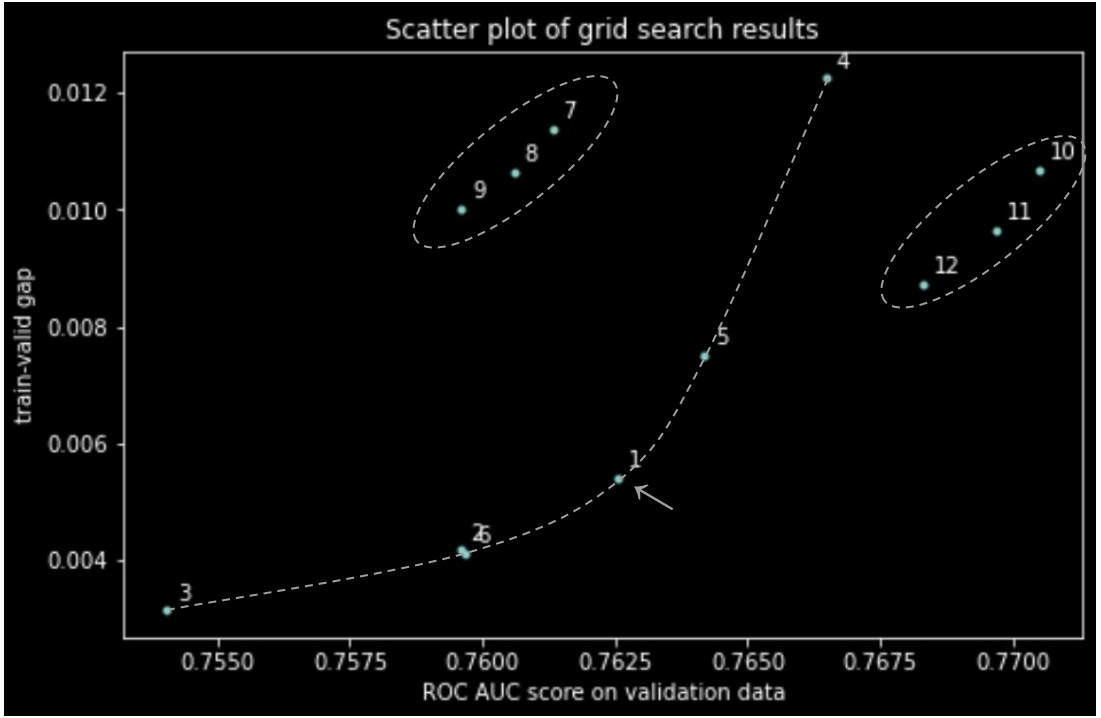
1000 points
5 h 52 min





1000 points
14 h 19 min

Optimisation du modèle



Jeu de paramètres sélectionné : 1	
boosting_type	goss
objective	binary
num_leaves	5
n_estimators	82
learning_rate	0.092
reg_alpha	0.82
reg_lambda	0.4
subsample	1.0
colsample_bytree	0.57
is_unbalance	False

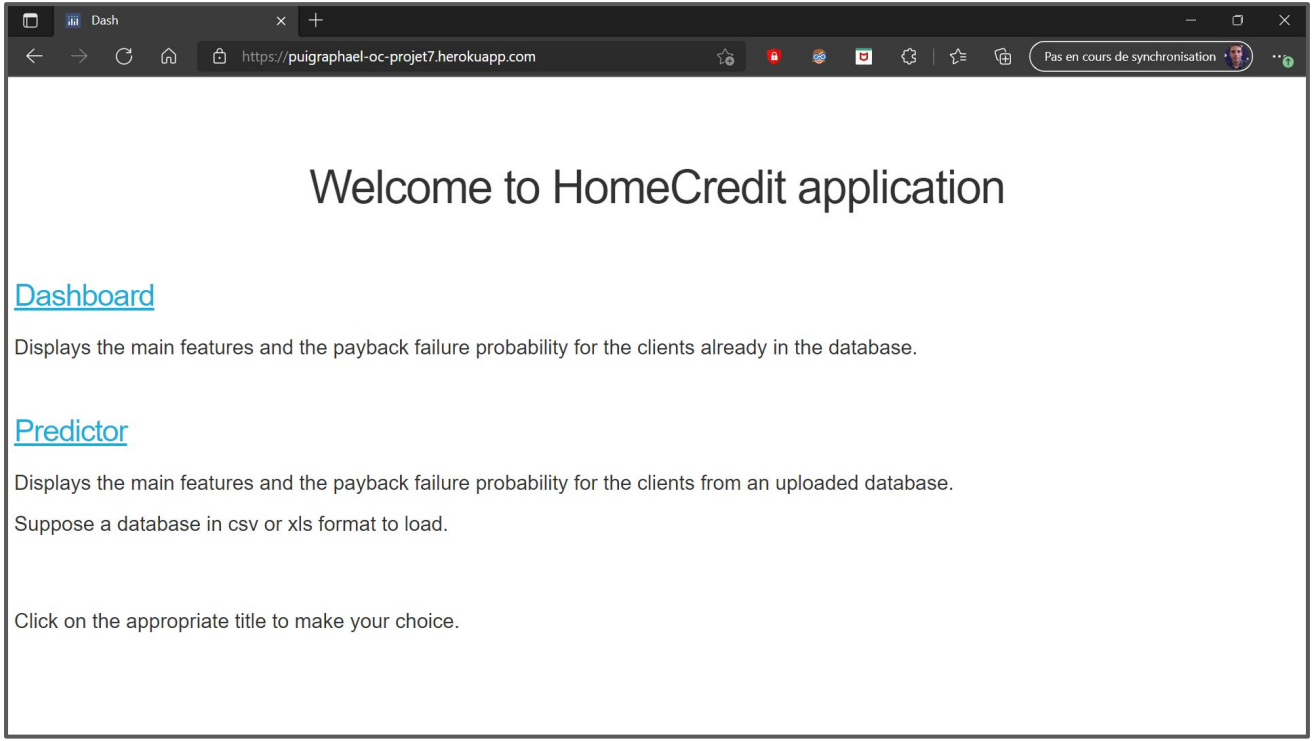
Application web

- Objectifs :
  - Afficher les prédictions et les variables les plus importantes des clients de toute la base.
  - Permettre le calcul et l'affichage de prédictions effectuées sur des nouveaux clients.
  - Déployer l'application sur un serveur de manière à la rendre accessible depuis n'importe où.

- Préalables :
  - Constituer une base de donnée des clients et leur prédiction.
  - Exporter le modèle et les utilitaires (standardiseur et imputeur de données manquantes).
  - Constituer des échantillons de clients (test du prédicteur).

- Interface de programmation : plotly | Dash Déploiement : HEROKU

Application web / homepage

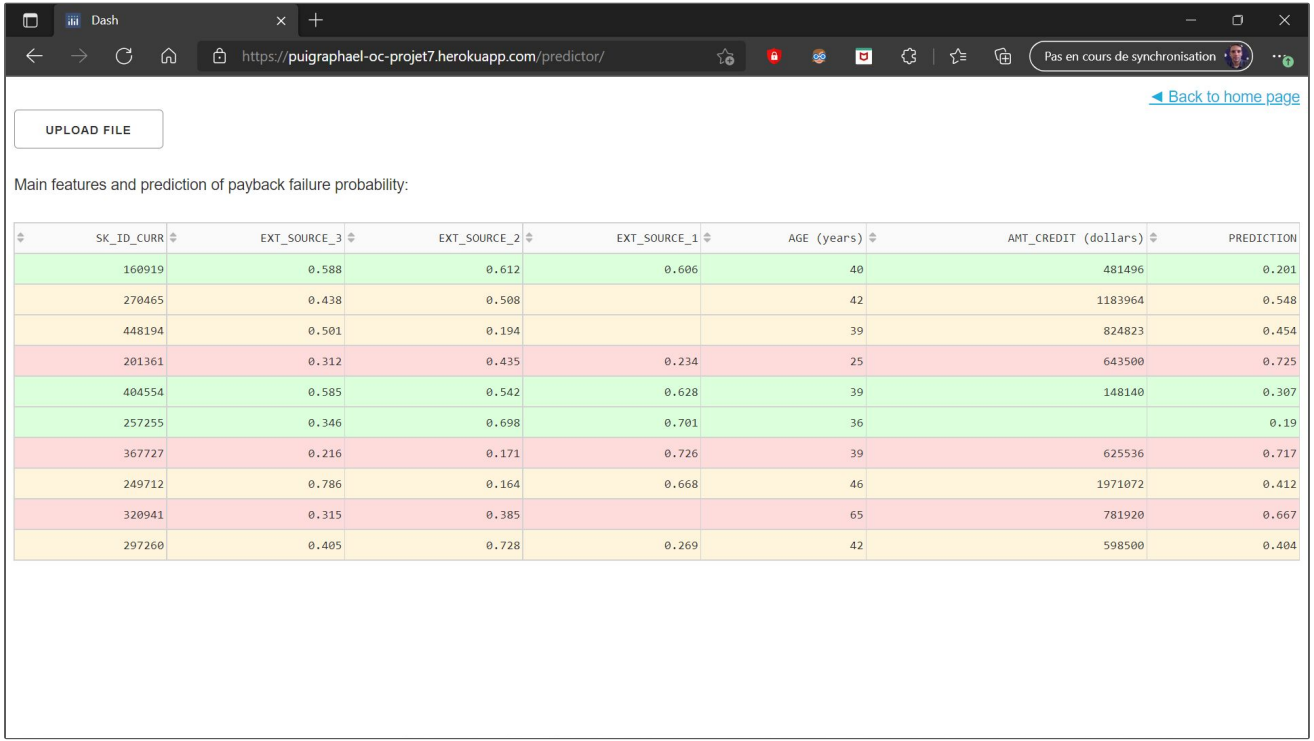


- Page d'accueil à partir de laquelle l'utilisateur peut sélectionner la fonctionnalité désirée.
- Changement de page par clic sur le titre.

Application web / dashboard

- *Kernel density estimators* des principales variables du modèle et probabilité d'échec de remboursement.
- Choix du client par menu déroulant ou directement en tapant son identifiant.
- Le repère en pointillé se déplace en fonction du client.

Application web / predictor



SK_ID_CURR	EXT_SOURCE_3	EXT_SOURCE_2	EXT_SOURCE_1	AGE (years)	AMT_CREDIT (dollars)	PREDICTION
160919	0.588	0.612	0.606	40	481496	0.201
270465	0.438	0.508		42	1183964	0.548
448194	0.501	0.194		39	824823	0.454
201361	0.312	0.435	0.234	25	643500	0.725
404554	0.585	0.542	0.628	39	148140	0.307
257255	0.346	0.698	0.701	36		0.19
367727	0.216	0.171	0.726	39	625536	0.717
249712	0.786	0.164	0.668	46	1971072	0.412
320941	0.315	0.385		65	781920	0.667
297260	0.405	0.728	0.269	42	598500	0.404

- Chargement d'un fichier client au format csv ou xls en cliquant sur UPLOAD FILE.
- Affichage des principales variables et de la prédiction associé à chaque client.
- Possibilité de classer par ordre croissant ou décroissant chaque variable.
- Code couleur pour faciliter la lecture.

Conclusions et perspectives

- Travail réalisé à ce stade :
  - Bases de données éparses agrégées en une seule.
  - Choix de l'algorithme de classification LGBM Classifier.
  - Réduction de la base (abandon des variables corrélées et identification des variables d'intérêt).
  - Optimisation du modèle par *random search* puis *grid search*.
  - Mise à disposition des résultats et du modèle prédictif *via* une application web.

- Améliorations possibles :
  - Implémentation d'un *early stopping* pour réduire le temps de calcul des *random* et *grid search*.
  - Changement de l'algorithme d'optimisation pour une optimisation Bayésienne.

Merci de votre attention.

Annexe 1 : ROC-AUC score

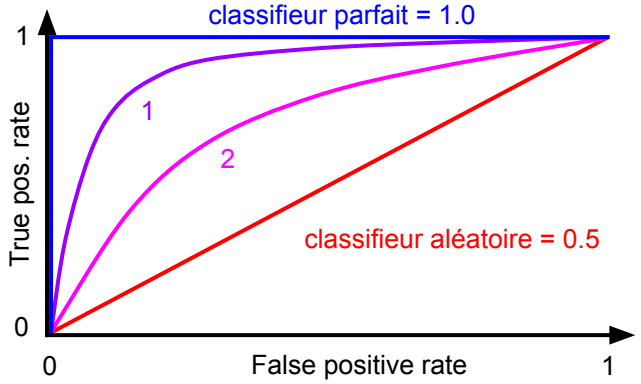
- Métrique adaptée au classification binaire.
- Receiver Operating Characteristic - Area Under (the) Curve.

	Prédit 0	Prédit 1
Réel 0	Vrai nég. (TP)	Faux pos. (FP)
Réel 1	Faux nég. (FN)	Vrai pos. (TP)

Matrice de confusion

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

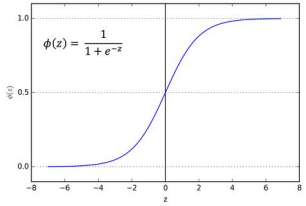


ROC-AUC score 1 > ROC-AUC score 2

Annexe 2 : Fonction de coût

- Fonction coût de LGBM Classifier = valeur moyenne de la fonction de perte.
- Fonction de perte dans une classification binaire : *logistic loss function*

fct logistique : $\hat{y}_i = g(\boldsymbol{w}.\boldsymbol{x}_i) = \dfrac{1}{1 + e^{-\boldsymbol{w}.\boldsymbol{x}_i}}$



fct de perte : $L_{\log}(y, \hat{y}) = -\,(y \,.\, \log(\hat{y}) + (1 - y) \,.\, \log(1 - \hat{y}))$ ← à minimiser (ou min. local)

fct de coût : $J(w) = \dfrac{-1}{N} \sum_{i=1}^{N} [\, y_i \,.\, \log(\hat{y}_i) + (1 - y_i) \,.\, \log(1 - \hat{y}_i)]$