Construire un modèle de scoring

Prêt à dépenser

Emmanuel Letremble

Contexte

Prêt à dépenser propose des **crédits** à la consommation pour des personnes ayant **peu ou pas d'historique de prêt**.

Son intérêt est de **prêter aux clients qui seront en capacité de rembourser** pour faire du chiffre, mais aussi d'éviter de prêter à ceux qui ne le pourront pas pour **éviter de les mettre en difficulté financière** ou de perdre de l'argent.

Contexte

Prêt à dépenser

Cependant en cas de refus, le RGPD stipule qu'il faut pouvoir expliquer simplement la décision prise par un système automatisé.

C'est important pour que le client sache ce qu'il doit éventuellement améliorer, mais c'est également important pour motiver l'accord d'un prêt auprès de la direction de la banque.

Objectif du projet

Prêt à dépenser

L'entreprise souhaite un outil de scoring crédit qui calcule la probabilité qu'un client le rembourse, puis classifie la demande : accordé ou refusé.

Il faut donc sélectionner l'algorithme le plus adapté et l'optimiser pour minimiser les erreurs.

Les utilisateurs principaux seront les chargés de relation client qui s'adressent aux clients, et ont donc besoin que le modèle soit facilement interprétable localement et globalement.

Présentation du jeu de données

Le jeu de données contient 307.511 dossiers de prêts fournies par Home Credit, un service dédié à la fourniture de lignes de crédit à des populations non bancarisée.

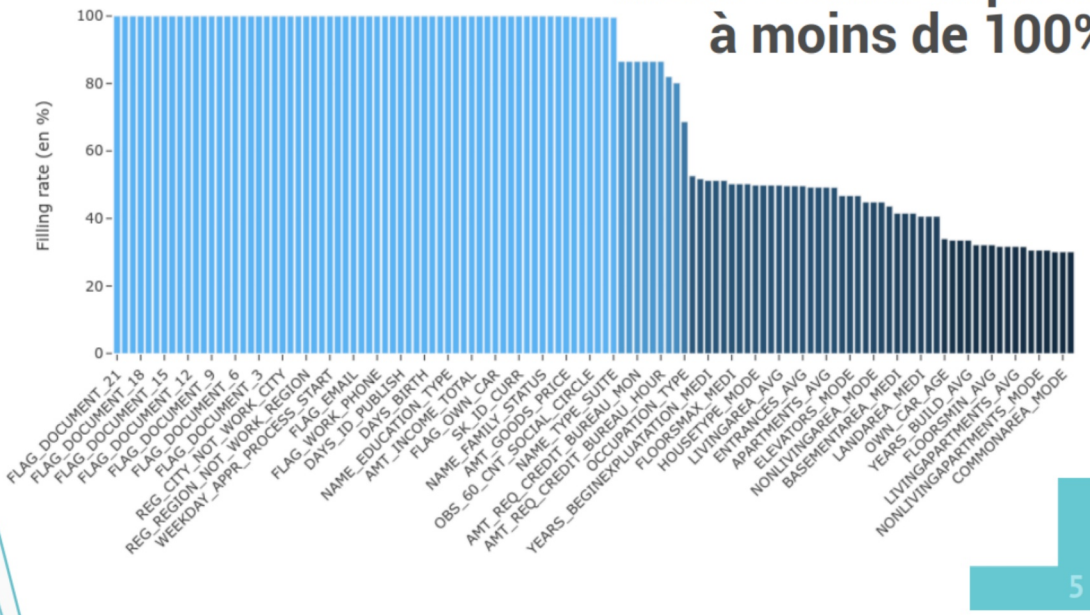
Il existe 7 fichiers source décrivant chacun des dossiers par 220 variables dont 122 se trouvent dans le fichier principal (application_train.csv).

- 38 variables booléennes dont TARGET
- 11 variables catégorielles
- 73 variables numériques

Présentation du jeu de données

Prêt à dépenser

Le jeu de données initial contient 307.511 dossiers décrits par 122 colonnes dont la moitié sont remplies à moins de 100%



Analyses exploratoires

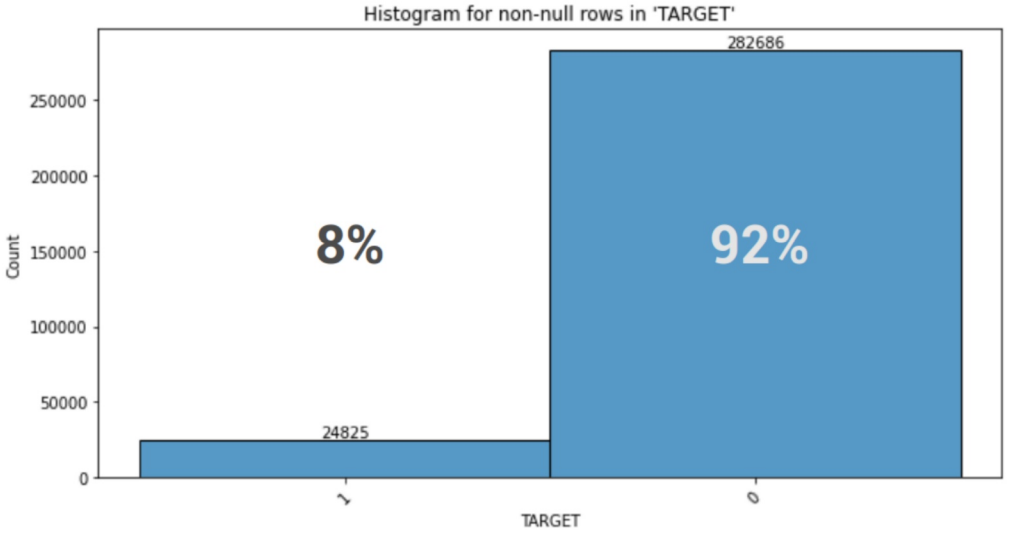
Prêt à dépenser

Les vérifications des erreurs les plus fréquentes
(valeurs manquantes, doublons, outliers, erreurs de format, erreurs lexicales, contenus multiples)
puis les analyses univariées et multivariées ont permis de déceler quelques problèmes qui ont été corrigés avec les actions suivantes:

- Suppression des colonnes inutiles
- Traitement des outliers
- Imputation des valeurs manquantes
- Encodage des colonnes catégorielles
- Normalisation

Analyses exploratoires

Par ailleurs, l'analyse à révélé que la variable cible est très déséquilibrée !



Recherche des modèles adaptés

Prêt à dépenser

Il existe de nombreux algorithmes de ML, et il n'est pas évident de savoir celui qui convient le mieux à un problème donné...

Nous allons donc en tester plusieurs pour trouver les deux plus prometteurs et les affiner.

- **Naïf :** DummyClassifier
- **Linéaire :** LogisticRegression
- **Non-linéaire :** DecisionTree / KNN / SVM
- **Ensemble :** RandomForest / XGBClassifier

Recherche des modèles adaptés

Prêt à dépenser

Les métriques

Pour un problème de classification, les mesures de référence sont:

- **Accuracy** : taux de bonnes prédictions au total
- **Precision** : taux de vrai-positifs parmi les positifs
- **Recall** : capacité à détecter les vrai-positifs
- **Specificity** : capacité à éviter les faux-positifs

- **F1-Measure** : moyenne harmonique entre Precision et Recall -> 2*(P*R)/(P+R)

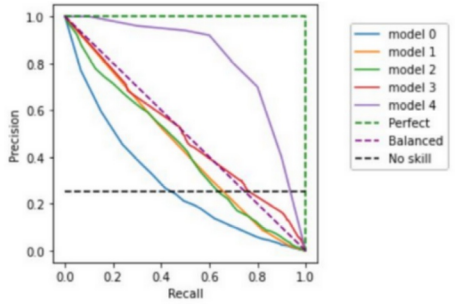
Recherche des modèles adaptés

Prêt à dépenser

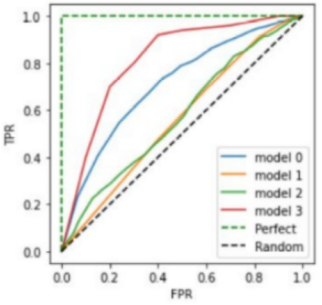
Les métriques

Mais les métriques que nous venons de voir sont assez sensibles aux déséquilibrage des jeux de données...

- **PR AUC** : qui supporte bien le déséquilibrage, mais va favoriser la classe positive



- **ROC AUC** : qui supporte les déséquilibrage jusqu'à un certain point, mais va donner autant de poids aux deux classes

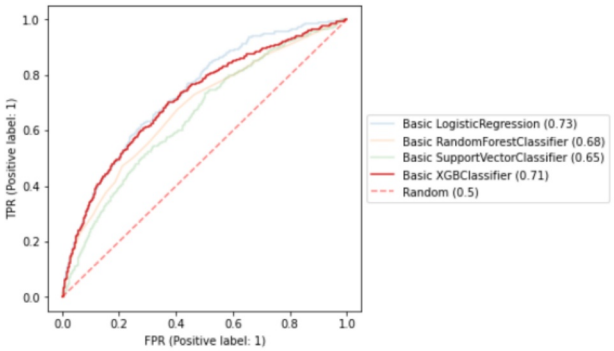


Recherche des modèles adaptés

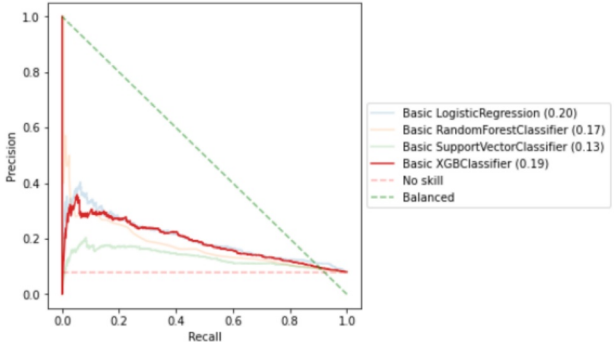
Prêt à dépenser

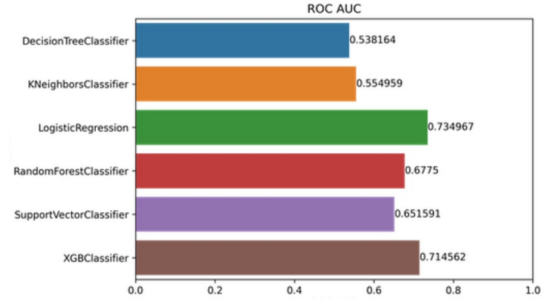
1. Évaluations en cross-validation mais sans aucun paramétrage

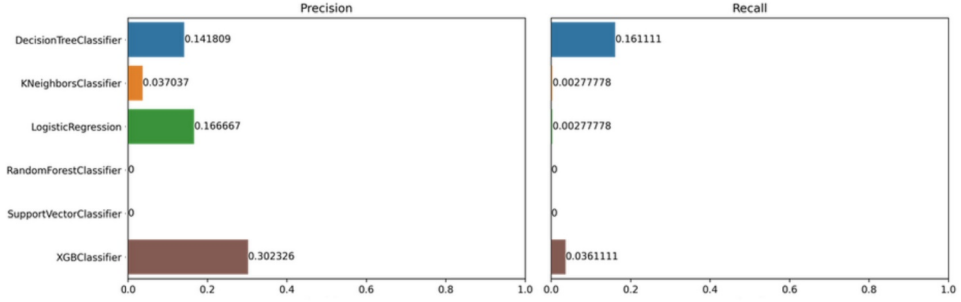
--- ROC AUC ----------



--- PRECISION RECALL AUC ----------







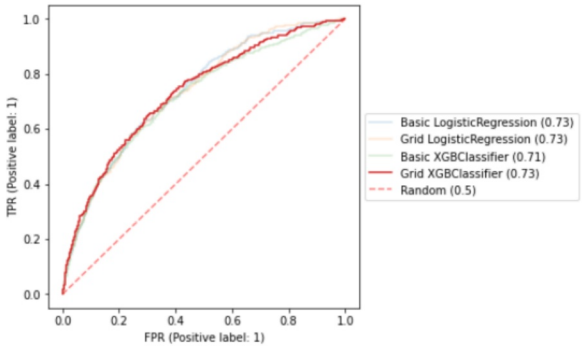
Prezi

Recherche des modèles adaptés

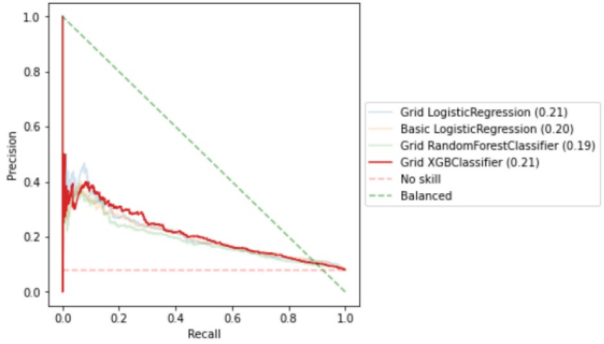
Prêt à dépenser

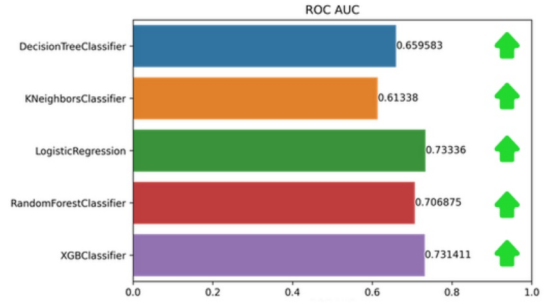
2. Recherche des hyper-paramètres via RandomizedSearchCV

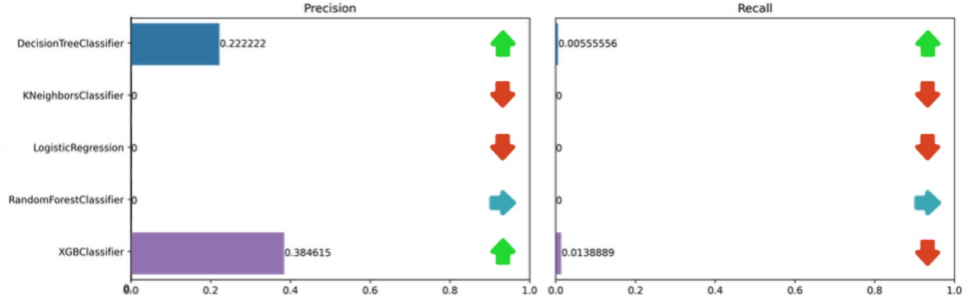
--- ROC AUC ---



--- PRECISION RECALL AUC ---





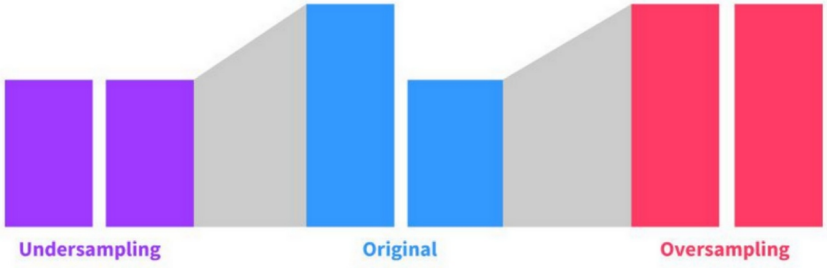


Recherche des modèles adaptés

Prêt à dépenser

3. Rééquilibrage de la variable cible

Il existe plusieurs approches pour tenter de résoudre un déséquilibre du jeu de données:



- **Undersampling : pandas df.sample(num)**
- **Oversampling : SMOTE**
- **Pondération : class_weight=balanced**

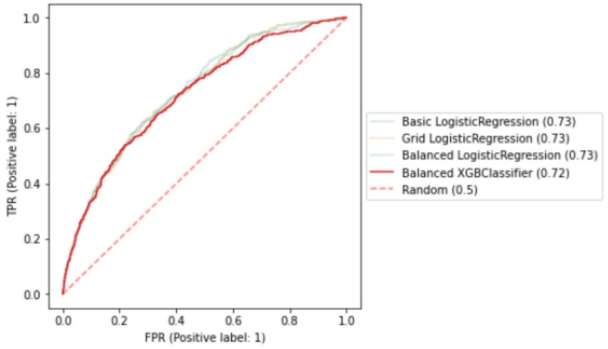
poids_j = n_samples / (n_classes * n_samples_j)

Recherche des modèles adaptés

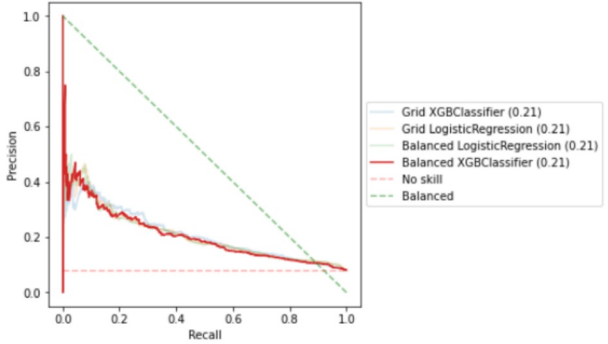
Prêt à dépenser

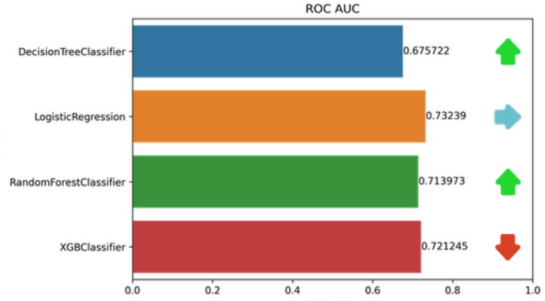
3. Rééquilibrage de la variable cible à l'aide de class_weight

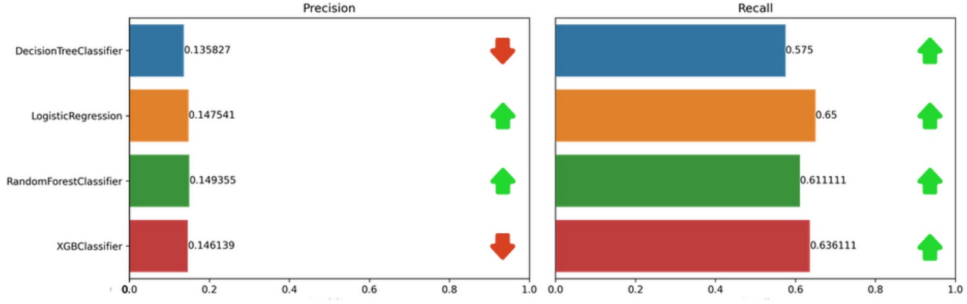
--- ROC AUC ---



--- PRECISION RECALL AUC ---







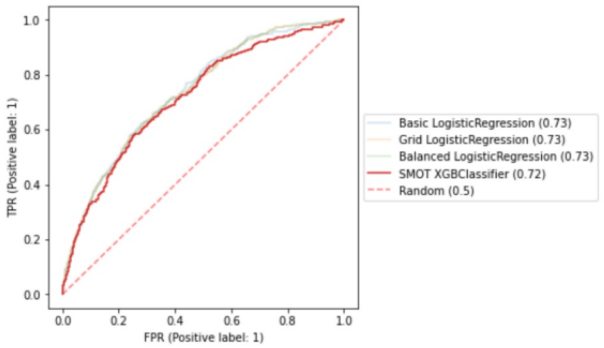
Prezi

Recherche des modèles adaptés

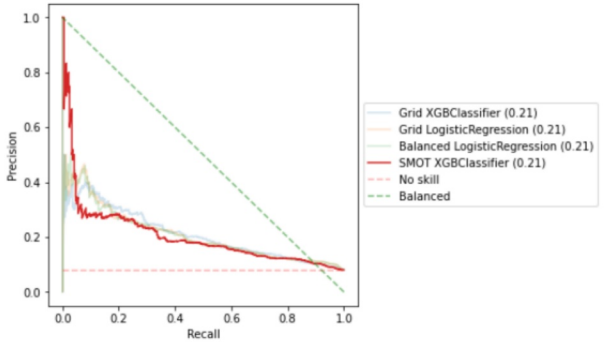
Prêt à dépenser

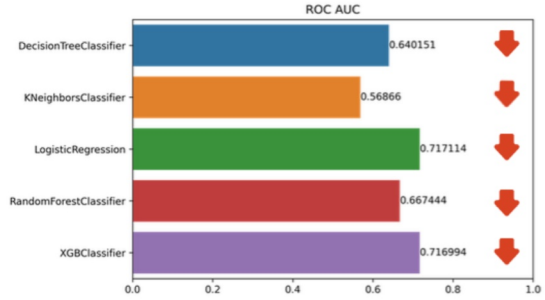
3. Rééquilibrage de la variable cible à l'aide de SMOTE

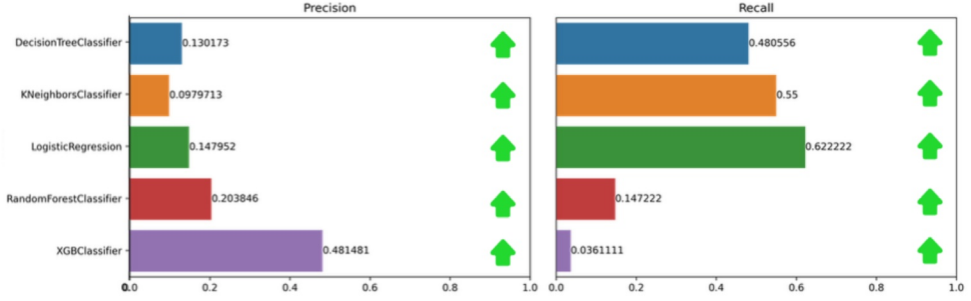
--- ROC AUC ---



--- PRECISION RECALL AUC ---







Recherche des modèles adaptés

Prêt à dépenser

4. Sélection des modèles les plus prometteurs

Method	ROC AUC	F1 score	Recall
Balanced LogisticRegression	0.732390	0.240493	0.650000
Balanced XGBClassifier	0.721245	0.237675	0.636111
SMOT LogisticRegression	0.717114	0.239061	0.622222
SMOT XGBClassifier	0.716994	0.067183	0.036111
Balanced RandomForestClassifier	0.713973	0.240044	0.611111
Balanced DecisionTreeClassifier	0.675722	0.219745	0.575000
SMOT RandomForestClassifier	0.667444	0.170968	0.147222
SMOT DecisionTreeClassifier	0.640151	0.204855	0.480556
SMOT KNeighborsClassifier	0.568660	0.166317	0.550000

Recherche des modèles adaptés

Prêt à dépenser

4. Sélection des modèles les plus prometteurs

Method	ROC AUC	F1 score	Recall
Balanced LogisticRegression [w/thresholding]	0.732390	0.276182	0.413889
Balanced XGBClassifier [w/thresholding]	0.721245	0.265700	0.305556
SMOT LogisticRegression [w/thresholding]	0.717114	0.268595	0.361111
SMOT XGBClassifier [w/thresholding]	0.716994	0.211073	0.169444
Balanced RandomForestClassifier [w/thresholding]	0.713973	0.259512	0.369444
Balanced DecisionTreeClassifier [w/thresholding]	0.675722	0.223041	0.336111
SMOT RandomForestClassifier [w/thresholding]	0.667444	0.171244	0.147222
SMOT DecisionTreeClassifier [w/thresholding]	0.640151	0.216263	0.347222
SMOT KNeighborsClassifier [w/thresholding]	0.568660	0.125771	0.141667

Optimisation des modèles sélectionnés

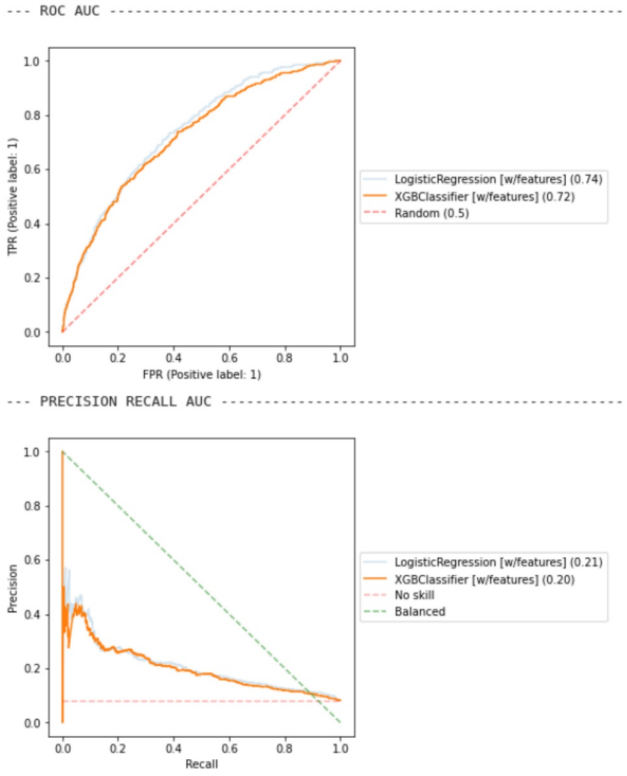
1. Création de features

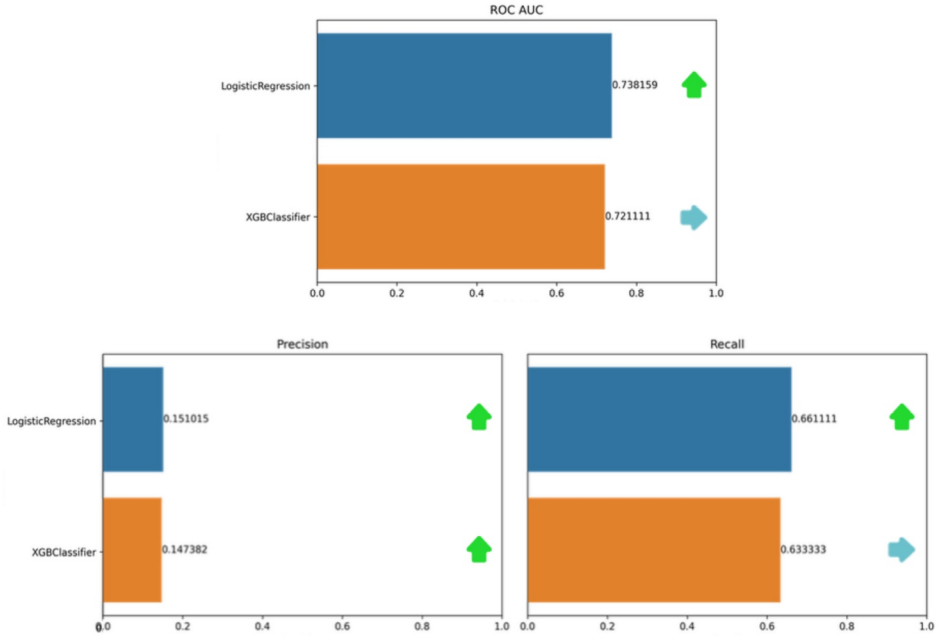
- FRAC_GOODS_PRICE : Quelle fraction du bien convoité représente le montant du crédit ?
- AGE_EMPLOYEMENT : Quel age (en années) avait le client quand il a obtenu son emploi actuel ?
- INCOME_PER_FAM_MEMBER : Quel montant de revenu est disponible pour chaque membre de la famille ?
- AMT_REQ_CREDIT_BUREAU_TOTAL : Combien de demandes de crédits ont été faites au total ?
- ANNUITY_INCOME_RATIO : Quelle fraction des revenus représente chaque annuité ?
- DIFF_GOOD_PRICE_CREDIT : Quel est le reste à payer sur le bien à l'origine de la demande de crédit ?
- DAYS_EMPLOYED_PERC : Quelle fraction de son temps de vie représente son dernier emploi ?
- AGE_INT : Quel est l'age en années ?

Optimisation des modèles sélectionnés

Prêt à dépenser

1. Création de features





Optimisation des modèles sélectionnés

Prêt à dépenser

2. Sélection de features

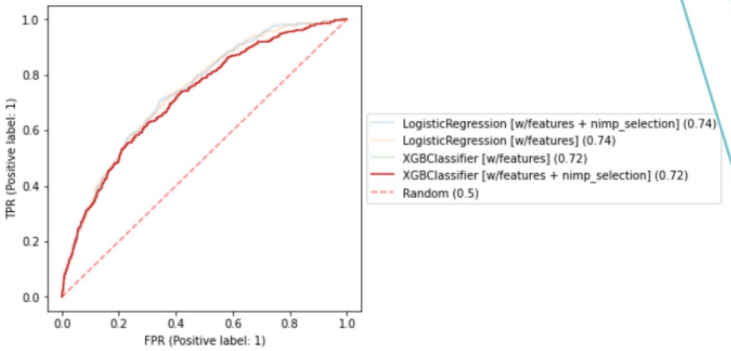
- **SelectFromModel** : selon l'importance des features dans le modèle associé
- **SelectKBest**, chi2 : les K meilleurs scores chi2
- **RFECV** : par élimination récursive des features selon leur importance
- **Null Importance** : par comparaison avec une importance calculée sur une TARGET nulle

Optimisation des modèles sélectionnés

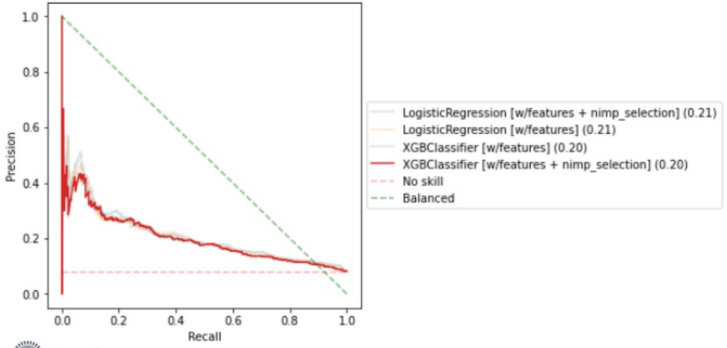
Prêt à dépenser

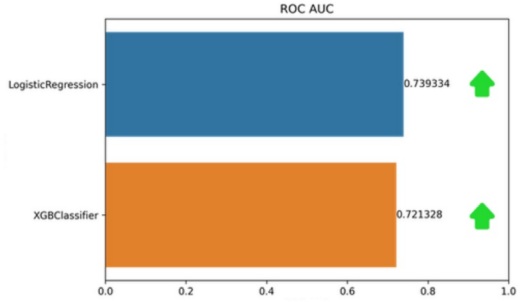
2. Sélection de features (with null Importances)

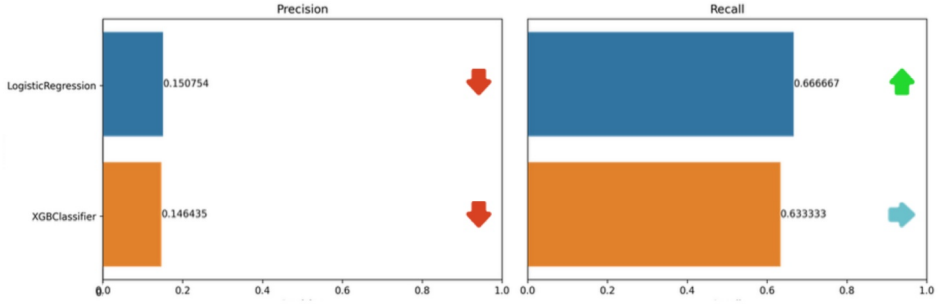
--- ROC AUC ----------



--- PRECISION RECALL AUC ----------







Optimisation des modèles sélectionnés

Prêt à dépenser

3. Création d'une mesure adaptée au projet

- **(TP) Vrai Positif** : on ne prête pas et on **évite** une perte.
- **(TN) Vrai Négatif** : on prête et on **gagne** de l'argent
- **(FP) Faux Positif** : on ne prête pas et on **loupe** de l'argent
- **(FN) Faux Négatif** : on prête et on **perd** de l'argent

		Actual class 1	Actual class 0
Predicted class	1	TRUE POSITIVE	FALSE POSITIVE
Predicted class	0	FALSE NEGATIVE	TRUE NEGATIVE

Optimisation des modèles sélectionnés

Prêt à dépenser

3. Création d'une mesure adaptée au projet

- (TP) Vrai Positif : on ne prête pas et on évite une perte.
- (TN) Vrai Négatif : on prête et on gagne de l'argent
- (FP) Faux Positif : on ne prête pas et on loupe de l'argent
- (FN) Faux Négatif : on prête et on perd de l'argent

Donc nos objectifs par ordre d'importance sont :

1. minimiser les FN pour éviter les difficultés financières
2. maximiser les TP pour éviter les difficultés financières
3. maximiser les TN pour gagner de l'argent
4. minimiser les FP pour éviter de louper de l'argent

Optimisation des modèles sélectionnés

3. Création d'une mesure adaptée au projet

1. minimiser les FN pour éviter les difficultés financières
2. maximiser les TP pour éviter les difficultés financières
3. maximiser les TN pour gagner de l'argent
4. minimiser les FP pour éviter de louper de l'argent

L'une des façons d'atteindre ces objectif d'accentuer le poids du Recall (qui correspond à nos deux premiers objectifs puisqu'il favorise les TP au détriment des FN)

La F-measure se décline en plusieurs variantes;

- F0.5 : qui donne plus de poids à la precision
- F1 : qui donne autant de poids aux deux
- F2 : qui donne plus de poids au recall

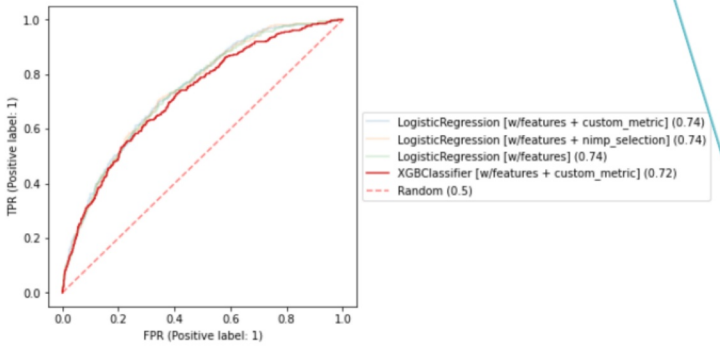
Nous allons donc Fbeta_score avec beta = 2 mais cette valeur peut être ajustée

Optimisation des modèles sélectionnés

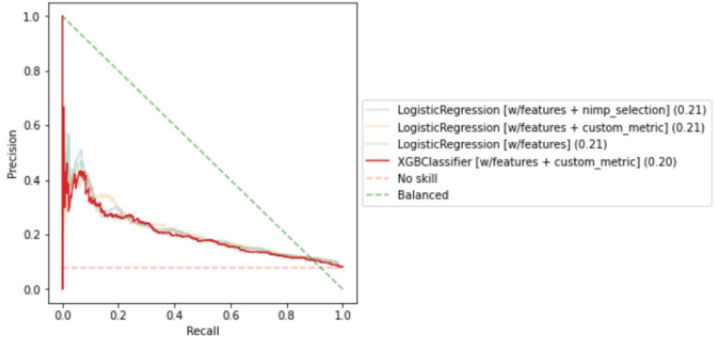
Prêt à dépenser

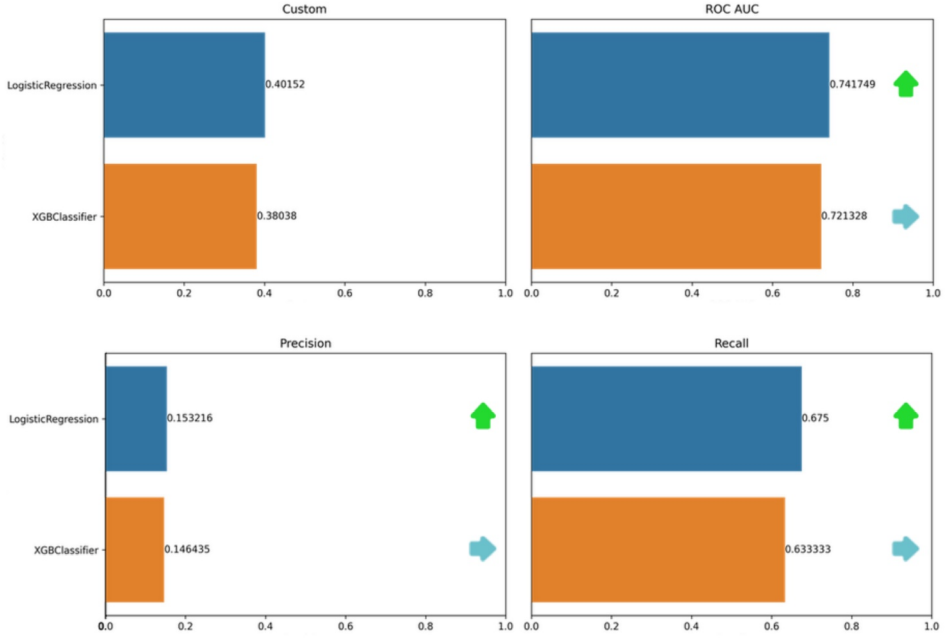
3. Création d'une mesure adaptée au projet

--- ROC AUC ---



--- PRECISION RECALL AUC ---



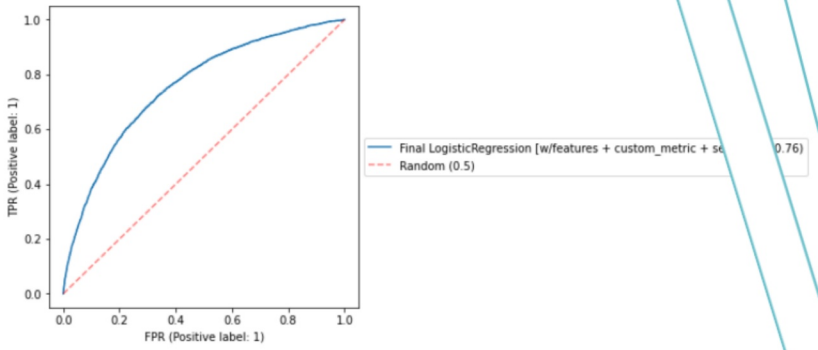


Optimisation des modèles sélectionnés

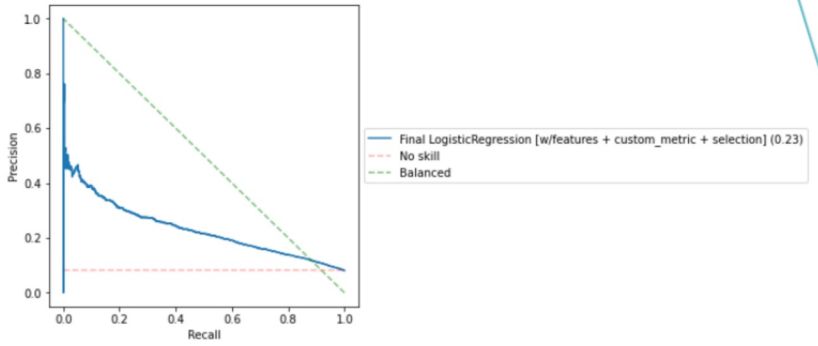
Prêt à dépenser

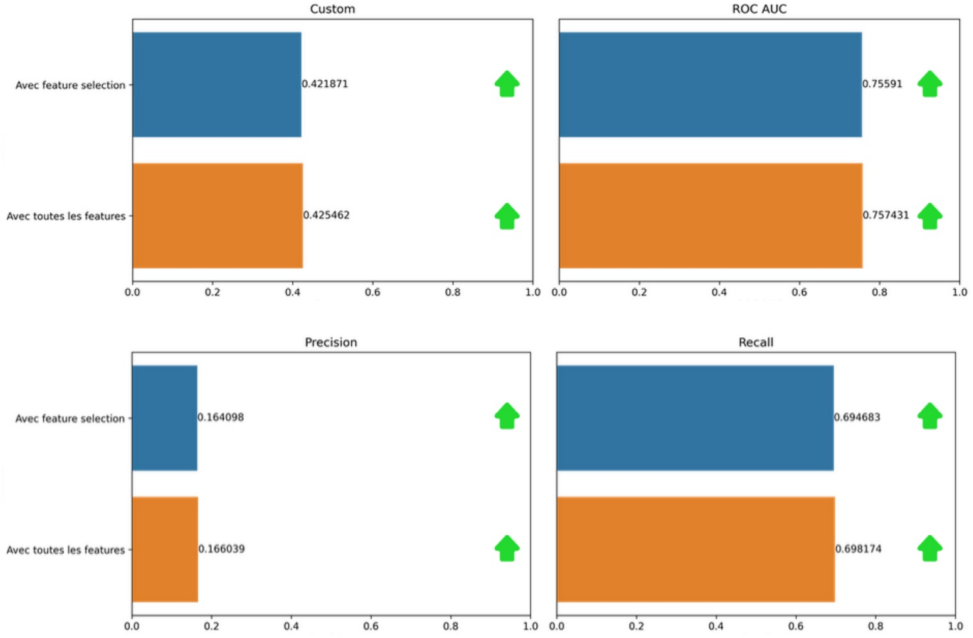
4. Ajout de nouvelles données (full retrain)

--- ROC AUC ----------



--- PRECISION RECALL AUC ----------



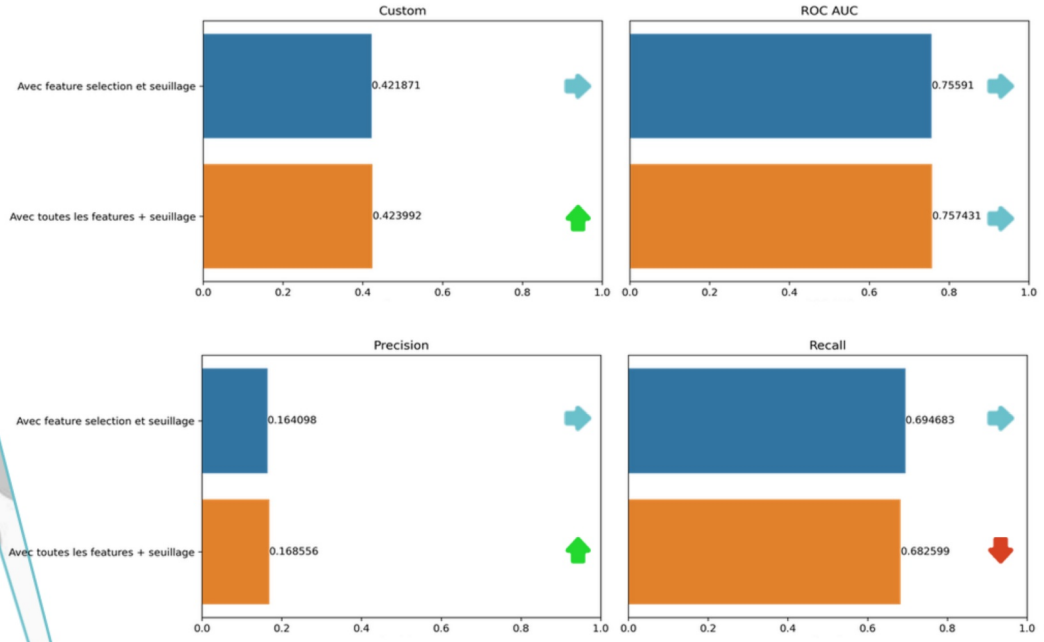


Prezi

Optimisation des modèles sélectionnés

Prêt à dépenser

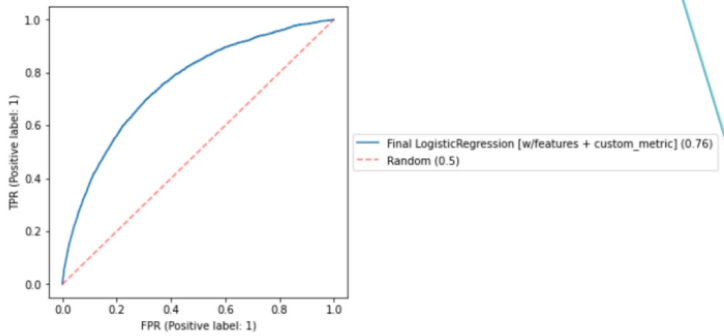
5. Seuillage (full retrain)



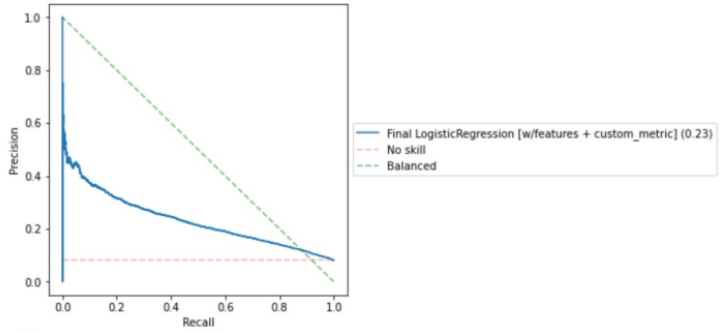
Modèle final

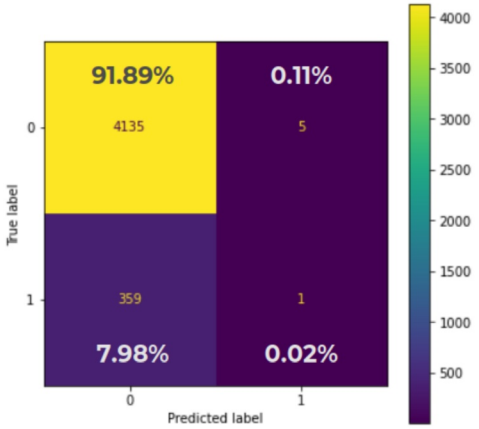
Prêt à dépenser

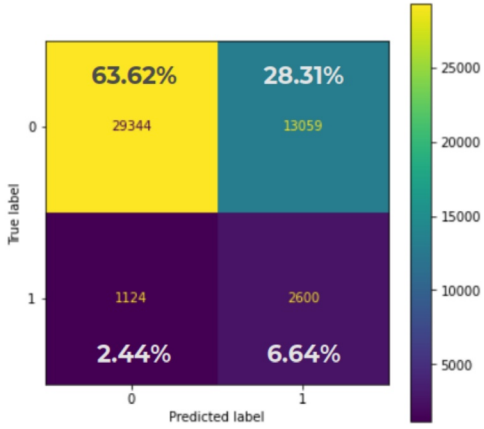
--- ROC AUC ----------



--- PRECISION RECALL AUC ----------







-- Basic LogisticRegression --

ROC_AUC : 0.7350
F1 : 0.0055
ACCURACY : 0.9191
PRECISION : 0.1667
RECALL : 0.0028
AVERAGE_PRECISION : 0.2003

-- Final LogisticRegression [w/features + custom_m

ROC_AUC : 0.7574
F1 : 0.2683
ACCURACY : 0.6925
PRECISION : 0.1660
RECALL : 0.6982
AVERAGE_PRECISION : 0.2329
CUSTOM : 0.4255

Explicabilité des résultats

Prêt à dépenser

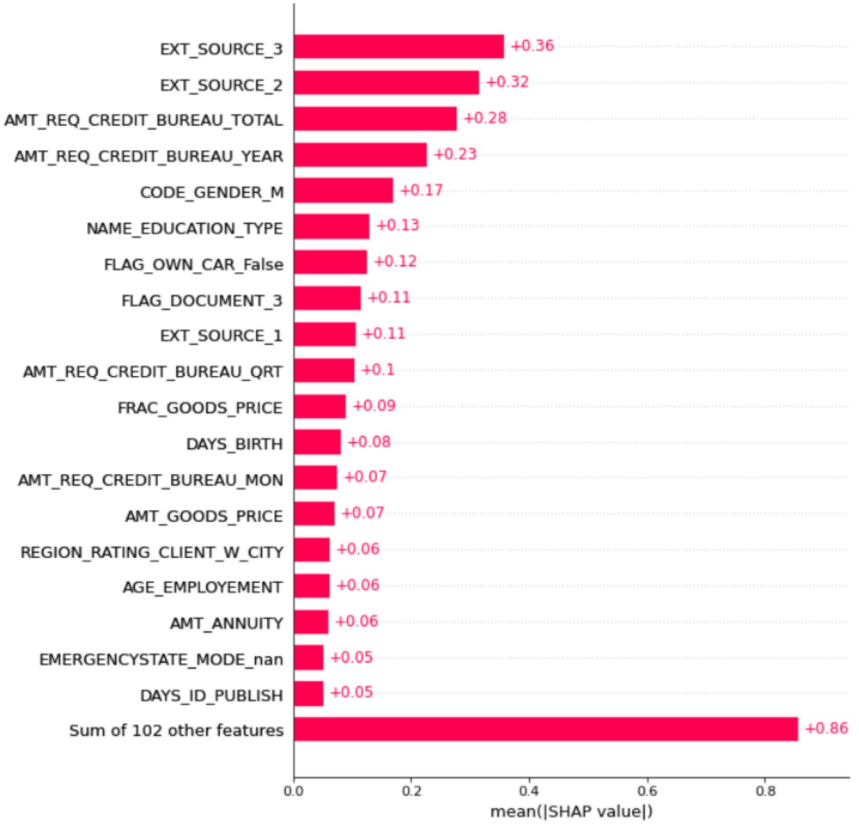
Problèmatique

- **Aspect légal**: l'article 22 du RGPD prévoit des règles pour éviter qu'un individu ne subisse des décisions émanant uniquement de machines.
- **Validation du modèle**: nous cherchons à connaître les variables influentes afin de vérifier la cohérence avec la connaissance métier du domaine.
- **Explication et recommandation**: il faut pouvoir expliquer les raisons d'une décision tant au client qu'à la banque.

Explicabilité des résultats

Prêt à dépenser

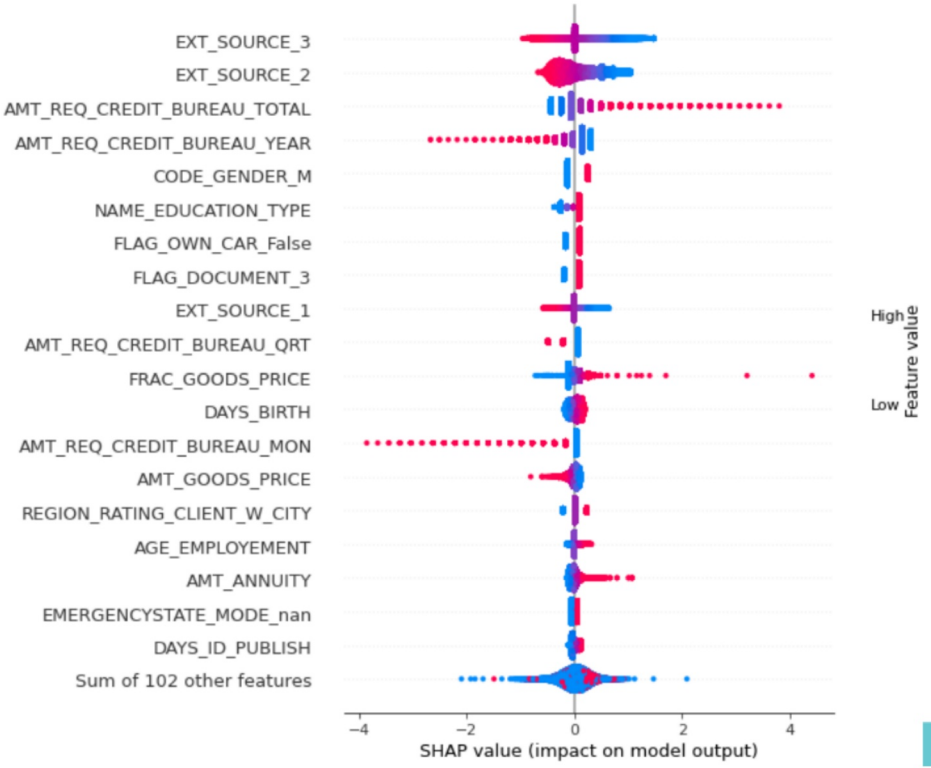
1. Interprétation globale



Explicabilité des résultats

Prêt à dépenser

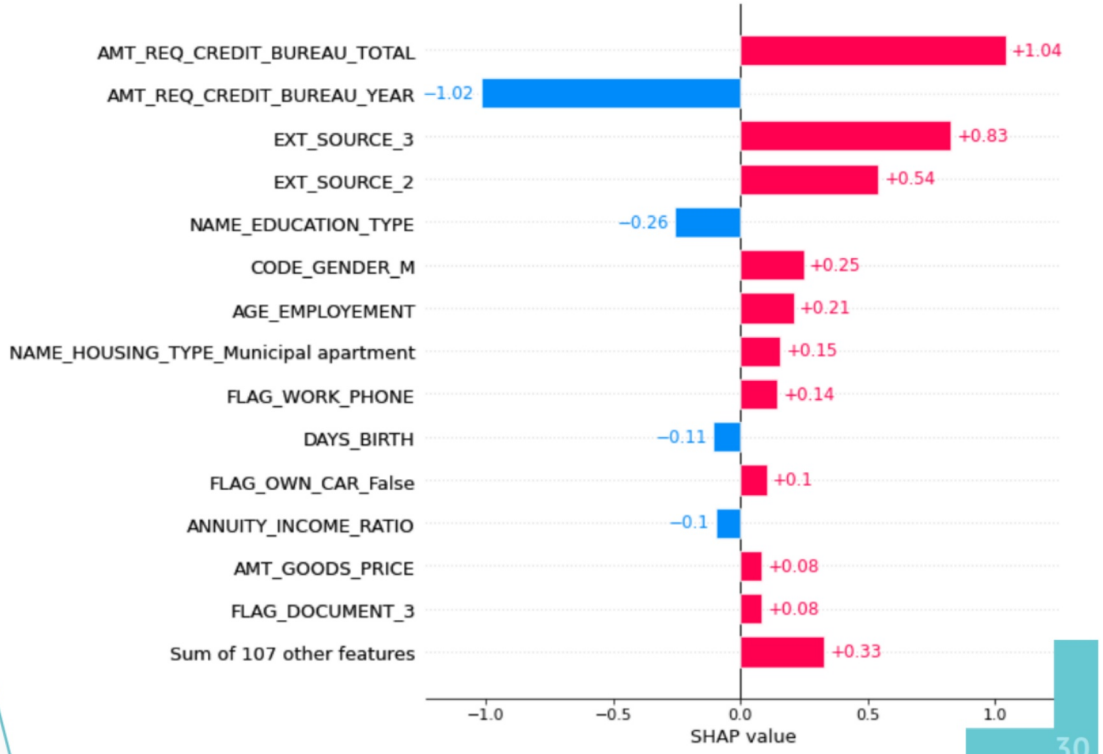
1. Interprétation globale



Explicabilité des résultats

Prêt à dépenser

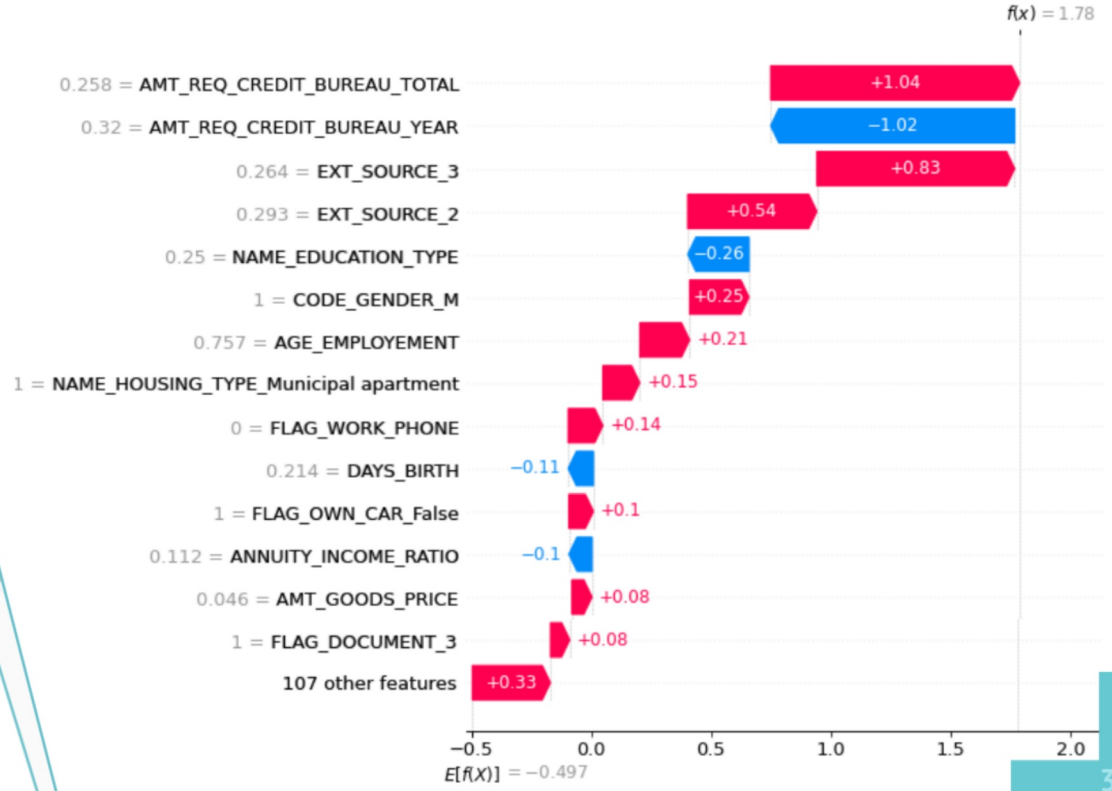
2. Interprétation locale



Explicabilité des résultats

Prêt à dépenser

2. Interprétation locale



Axes d'amélioration

Prêt à dépenser

- Créer une API
- Créer une Interface
- Tester davantage d'algorithmes
- Tester des variables polynomiales
- Utiliser les dataset périphériques

Merci de m'avoir écouté, évalué et conseillé.

Prêt à dépenser