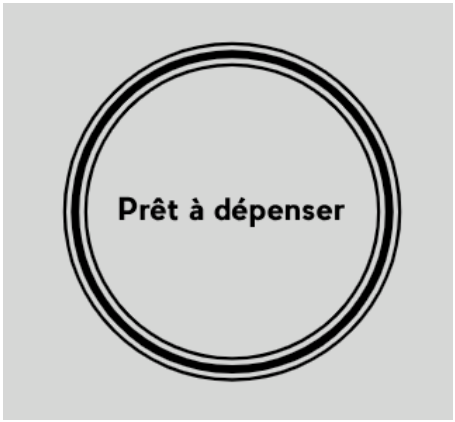


Création d'un modèle de scoring

Auteur :
Cyril REGAN

Client :
Prêt à dépenser

16 septembre 2020

Résumé

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 développer un modèle de scoring de la probabilité de défaut de paiement du client.

De plus, elle décide de développer un tableau de bord (dashboard) interactif pour que les chargés de relation client puissent expliquer les décisions d'octroi de crédit et permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement. Le tableau de bord réalisé avec FLASK (serveur) et React (client) ne sera pas présenté dans ce rapport mais est consultable à cette adresse : **dashboard**.

Ce rapport présente la construction d'un modèle de scoring, ses résultats et l'interprétation des prédictions.

Table des matières

Chapitre 1

Construction modèle de scoring

Les données sont accessibles à cette adresse : **kaggle**. Le notebook **a-gentle-introduction** a été utilisé pour pré-traiter les données. Le jeu de donnée contient 307511 clients, 244 caractéristiques + 1 cible qui est le remboursement ou non par les clients des prêts. 8% des clients seulement n'ont pas remboursé totalement leur prêts. Le jeu de donnée est donc fortement déséquilibré suivant la variable cible qualitative.

L'évaluation des techniques de gestion de classe et l'optimisation de la métrique spécifique sont réalisées sur un jeu d'entraînement représentant 4/5 de l'ensemble du jeu de donnée. Le reste des données (1/5) est réservé pour le jeu de test sur lequel sera évalué le modèle final sur les métriques classiques et la métrique spécifique. Ces 2 ensembles ont la même répartition de classe.

1.1 Gestion de classes déséquilibrées

Il arrive souvent dans les cas de classification en apprentissage machine (Machine Learning) que l'une des classes soit minoritaire par rapport à la population globale. Dans notre cas, 92% des observations appartiennent à la classe des négatifs (clients ayant remboursé leur prêt) et les 8% restants à la classe des positifs.

Le déséquilibre provoque des difficultés aux modèles d'apprentissage machine pour prédire la classe minoritaire, ce qui est souvent celle pour laquelle l'intérêt est porté. Dans notre cas, les clients positifs sont ceux qui n'ont pas recouvert leur prêt. Bien prédire la catégorie des positifs est donc primordial pour évaluer le risque sur l'octroi d'un prêt.

1.1.1 Techniques d'échantillonnage

Nous présentons ci dessous les différentes techniques d'échantillonnage pour aider les modèles à mieux détecter les clients positifs. Un schéma de ces méthodes est présenté ci dessous :

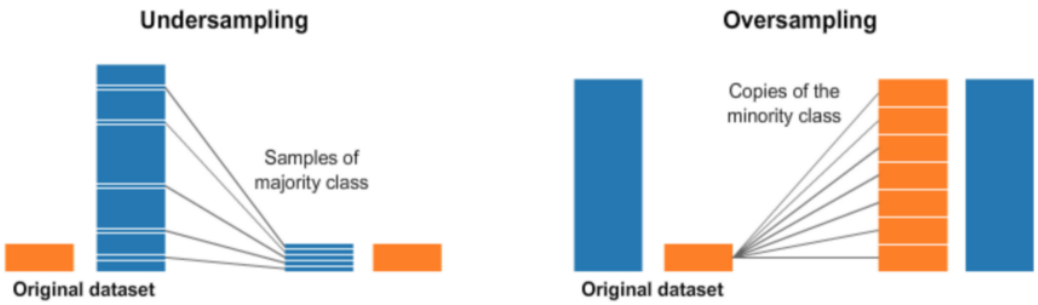


FIGURE 1.1 – Méthodes d'échantillonage

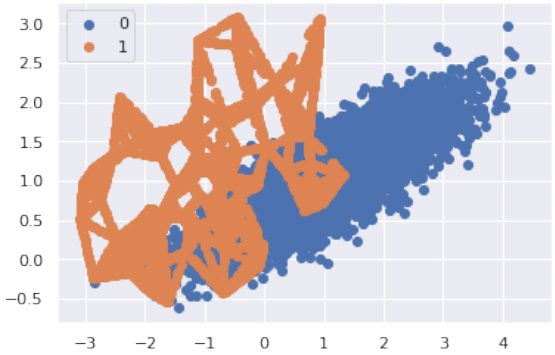
L'undersampling (1.1 : gauche) fonctionne en enlevant des données de la classe majoritaire aléatoirement.

L'oversampling (1.1 : droite) fonctionne en ajoutant des données de la classe minoritaire avec par exemple la technique SMOTE (Synthetic Minority Over-Sampling TEchnique) qui consiste à générer de nouveaux échantillons en combinant les données de la classe minoritaire avec celles de leurs voisins proches.

La figure 1.2 représente 2 classes déséquilibrées de données 2D sous forme de nuage de point sur laquelle des techniques d'échantillonage sont appliquées. La figure de gauche 1.2a représente les nuages de points originaux, la figure 1.2b représente les mêmes nuages avec SMOTE et 1.2c avec l'undersampling. Avec SMOTE, on observe

une reconstruction de points de la classe minoritaire basée le long des lignes créées par les points originaux de la classe minoritaire. Avec l'undersampling, on observe une diminution radicale du nombre de point de la classe majoritaire.

(a) Nuage de points 2D sans SMOTE



(b) Nuage de points 2D avec SMOTE

(c) Nuage de points 2D avec undersampling

FIGURE 1.2 – Effet de SMOTE et undersampling sur 2 classes déséquilibrées en 2D

1.1.2 Class-weighting

Il est aussi possible de traiter le déséquilibre en attribuant plus de poids à certaines observations via le paramètre class_weight (dans **sklearn**). Pour class_weight = « balanced », l'attribution des poids se fera automatiquement en fonction de la proportion des classes.

1.2 Modélisation

1.2.1 Validation croisée

Pour utiliser les méthodes d'échantillonage, il est important de transformer par under ou oversampling uniquement **le jeu d'entraînement** sans modifier le jeu de test. Pour une validation croisée, cette vigilance doit être maintenue pour chaque pli (fold), ce qui impose de construire sa propre validation croisée (sans utiliser **GridSearch de Sklearn**).

1.2.2 Classification binaires

Pour la classification binaires (appartenance ou pas à une classe), une métrique couramment utilisée est la justesse (accuracy) qui calcule la proportion de prédiction exacte. Dans le cas d'un fort déséquilibre de classe, un modèle qui prédit que des négatifs aura une très bonne justesse (92% pour notre étude), ce qui n'est pas satisfaisant pour statuer sur l'octroi d'un prêt. C'est pourquoi il est nécessaire d'utiliser d'autres métriques, en particulier sur un jeu de données déséquilibré. Il est présenté ci dessous quelques métriques classiques sur des valeurs binaires :

- $Justesse(accuracy)$: proportion de prédictions exactes,
- $Rappel(recall) = \frac{TP}{TP+FN}$: taux de vrais positifs. Proportion de positifs que l'on a correctement identifiés,
- $Precision = \frac{TP}{TP+FP}$: proportion de prédictions correctes parmi les points que l'on a prédits positifs,

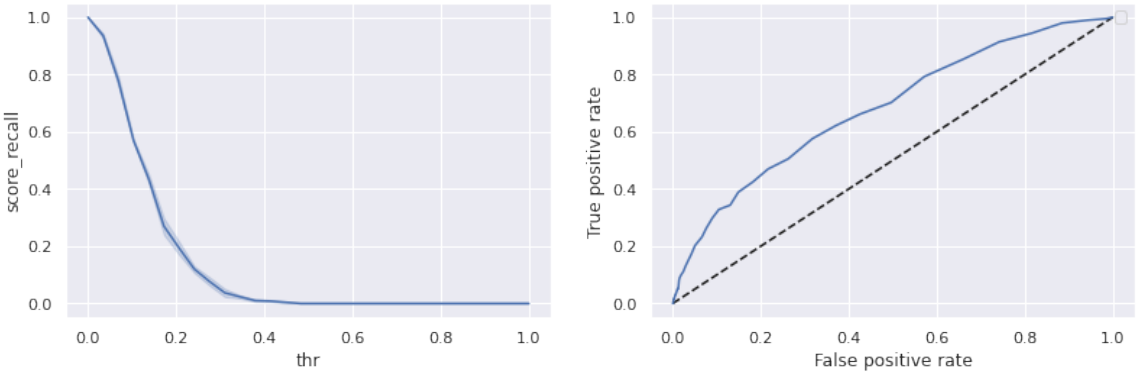
— $F - mesure = 2 \times \frac{Precision \times Rappel}{Precision + Rappel}$: moyenne harmonique entre rappel et précision,
— $Specificite = \frac{TN}{FP+TN}$: taux de vrais négatifs.
TN, FN, FP, TP sont respectivement les vrais négatifs, faux négatifs, vrais positifs et faux positifs. La figure ci dessous présente la matrice de confusion :

		Classe réelle	
		-	+
Classe prédite	-	True Negatives *(vrais négatifs)*	False Negatives *(faux négatifs)*
	+	False Positives *(faux positifs)*	True Positives *(vrais positifs)*

FIGURE 1.3 – Matrice de confusion

Dans cette étude, la métrique du **rappel** est particulièrement importante, car elle permet d'identifier le taux de vrais positifs, c'est à dire l'erreur sur les clients n'ayant pas recouvert leur prêt.

1.2.3 Scores



(a) Rappel(seuil) (b) ROC Curve

FIGURE 1.4 – Rappel(seuil) et ROC Curve de RF sur données déséquilibrées

La plupart des algorithmes retournent en fait un score (souvent une estimation de la probabilité qu'un point est positif) sur lequel un seuil est défini pour obtenir une prédiction binaire : si le score retourné est supérieur au seuil, alors l'algo prédit positif ; s'il est inférieur, il prédit négatif. Par défaut, la valeur du seuil est 0.5 pour les scores compris entre 0 et 1, mais il est possible d'optimiser le seuil pour améliorer le score sur une certaine métrique.

Par exemple, la figure 1.4a présente l'évolution du rappel en fonction de la valeur du seuil pour les résultats d'un algorithme de forêt aléatoire ou RF (Random Forest) avec les paramètres par défaut sur les données initiales déséquilibrées. On observe que le $recall = 1$ pour $thr = 0$. C'est normal car un seuil nul donne forcément que des positifs : le rappel (taux de vrai positifs) est donc égal à 1. On observe aussi que le rappel est presque nul pour un seuil de 0.5 : ce modèle détecte très peu les positifs par défaut (cad : seuil à 0.5).

la courbe ROC pour Receiver-Operator Characteristic permet montrer comment la sensibilité évolue en fonction de la spécificité (en fait 1 - la spécificité) suivant tous les seuils compris entre 0 et 1. Sur la figure 1.4b, la courbe bleue est la courbe ROC du même modèle que 1.4a. La droite en pointillée représente les résultats d'un classifieur aléatoire. La courbe ROC d'un modèle parfait dessine le coin supérieur gauche du carré. On évalue la performance d'un modèle avec l'aire sous la courbe ROC : c'est L'AUROC (=1 si modèle parfait, =0.5 si aléatoire). Dans ce cas particulier, l'AUROC = 0.68.

1.2.4 Évaluation des méthodes de gestion du déséquilibre

Pour l'évaluation des techniques de gestion du déséquilibre comme l'under-over-sampling ou le class-weighting, les techniques seront comparées en validation croisée avec un modèle de forêt aléatoire (avec paramètres par défaut) sur les métriques classiques (présentées précédemment) avec une attention particulière sur le rappel.

1.2.5 Métrique spécifique

La métrique choisie est basée sur un principe de gain et de perte. On définit le gain (G) comme la somme des montants des crédits (variable AMT_CREDIT) des clients prédits solvables ayant recouvert leur prêts (vrais négatifs : TN). Le gain maximal est calculé sur tous les clients ayant recouvert leur prêts (négatifs : N)

$$\begin{aligned} G &= \sum_{i \in TN} AMT\_CREDIT[i] \\ G_{max} &= \sum_{i \in N} AMT\_CREDIT[i] \end{aligned} \tag{1.1}$$

On définit la perte (L) comme la somme des montants des crédits des clients prédits solvables n'ayant pas recouvert leur prêts (faux négatifs : FN).

$$L = \sum_{i \in FN} AMT\_CREDIT[i] \tag{1.2}$$

On considère que les pertes sont 10 fois plus importantes que les gains :

$$\text{score} = (G - 10.L)/G_{max} \tag{1.3}$$

1.2.6 Optimisation de la métrique spécifique

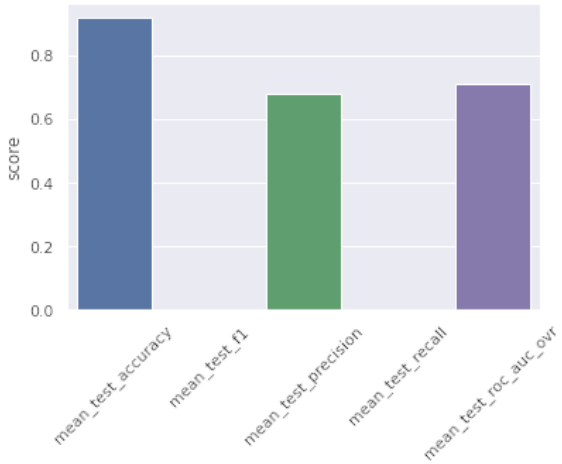
L'évaluation de la métrique spécifique sera effectuée sur un Light Gradient Boosting Machine (LightGBM) en validation croisée avec optimisation des hyper-paramètres et du seuil.

Chapitre 2

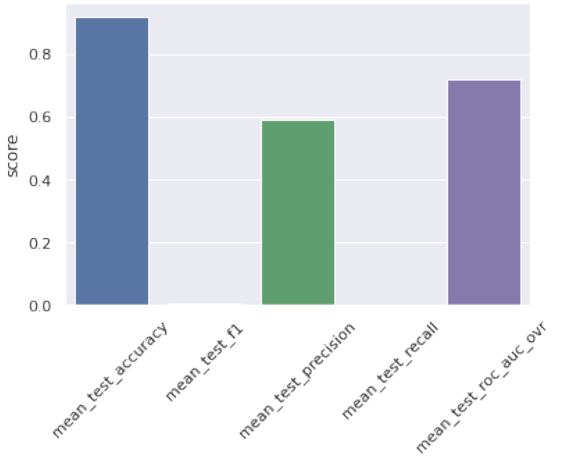
Résultats

2.1 Évaluation gestion du déséquilibre

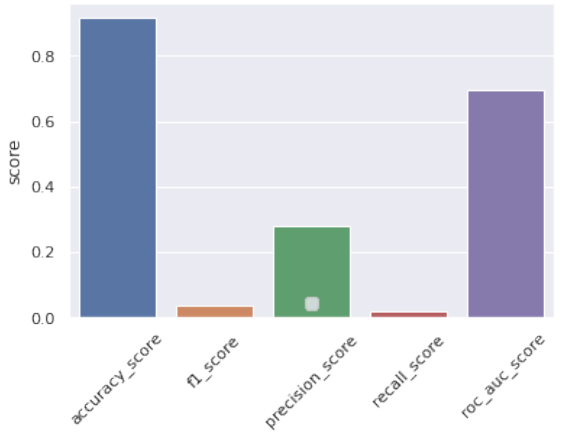
La figure ci dessous représente les résultats d'un modèle de forêt aléatoire non optimisé en validation croisée avec les métriques classiques en appliquant plusieurs méthodes de gestion du déséquilibre.



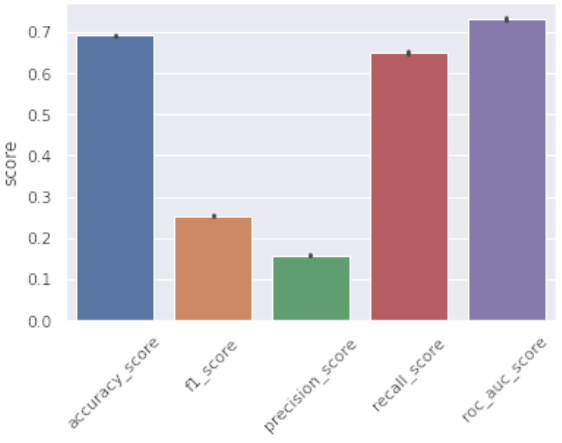
(a) Scores avec déséquilibre



(b) Scores avec class-weighting



(c) Scores avec oversampling



(d) Scores avec undersampling

FIGURE 2.1 – Scores pour différentes méthodes de gestion du déséquilibre

La figure 2.1a représente les résultats de référence sans gestion du déséquilibre. Ils sont mauvais sur le rappel (et donc la f-mesure) : le taux de vrais positifs est proche de zéro.

Le class-weighting (2.1b) n'améliore que très peu les résultats du modèle sans gestion du déséquilibre. La bibliographie sur le sujet évoque un possible bug de class-weighting avec les forêts aléatoires de la librairie sklearn. En l'état, le class-weighting est écarté comme candidat pour les techniques de gestion du déséquilibre.

l'oversampling (SMOTE) améliore très légèrement le rappel sans être significatif.

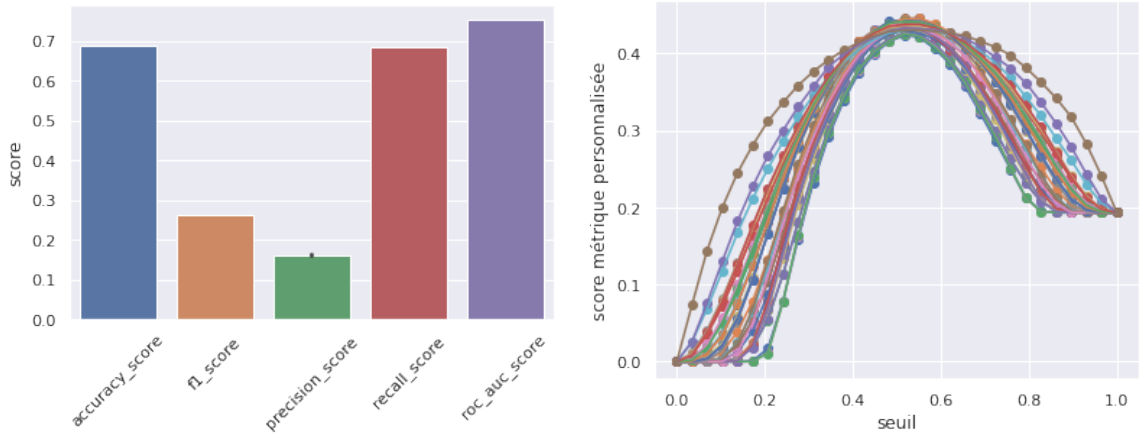
La méthode d'undersampling donne les résultats les plus précis pour le rappel (taux de vrais positifs) = 0.67. C'est la méthode qui est retenue pour l'optimisation de la métrique spécifique.

2.2 Optimisation de la métrique spécifique

L'optimisation de la métrique spécifique s'effectue avec LightGBM dans une validation croisée en undersampling avec un seuil variant de 0 à 1 et les hyper-paramètres suivant :

"n_estimators"	[100, 1000]	"max_depth"	[5,7,-1]	"num_leaves"	[9,31,127]
"objective"	binary	"class_weight"	balanced	"learning_rate"	[0.03,0.05]
"reg_alpha"	0.1	"reg_lambda"	0.1	"subsample"	0.8
"n_jobs"	-1	"random_state"	50		

TABLE 2.1 – Paramètres LightGBM



(a) LightGBM : métriques classiques

(b) LightGBM : métrique spécifique pour tous les folds et paramètres en fonction du seuil

FIGURE 2.2 – Optimisation métrique spécifique

La figure 2.2a présente les résultats sur les métriques classiques du LightGBM avec undersampling et optimisation de paramètres en validation croisée. Ils sont légèrement meilleurs que la forêt aléatoire avec un rappel = 0.69 (0.67 pour la RF). Cette configuration d'échantillonage et algorithmique est utilisée pour l'optimisation de la métrique spécifique.

La figure 2.2b présente les résultats sur la métrique spécifique pour tous les plis et paramètres de la validation croisée en fonction du seuil variant entre 0 et 1. La forme concave du score en fonction du seuil permet d'identifier un unique maximum stable pour chaque pli et combinaison d'hyper-paramètres. Le meilleur score de la métrique spécifique est **0.44** pour un seuil de **0.52**

2.3 Validation sur le jeu de test

Un jeu de donnée test a été séparé dés le début de l'étude pour valider le modèle final sur des données sur lesquelles il n'a pas été du tout été entraîné.

La figure 2.3 présente les résultats du modèle final sur le jeu de test avec les métriques classiques et la métrique spécifique. On obtient des résultats similaires à ceux de l'entraînement 2.2a sur les métriques classiques, et un score de 0.45 sur le jeu de test, proche des 0.44 obtenus sur le jeu d'entraînement. Ces résultats valident le modèle final.

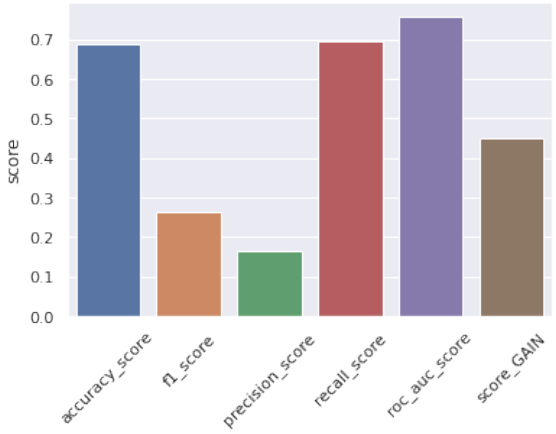


FIGURE 2.3 – Validation sur jeu de test

Chapitre 3

Interprétation

L'une des questions éthiques qui est très souvent évoquée actuellement est celle de la transparence. En effet, avec l'arrivée de la RGPD (Règlement général sur la protection des données) en mai 2018, il est nécessaire de pouvoir expliquer la prédiction ou décision d'une intelligence artificielle à un client ou simple utilisateur de l'Intelligence Artificielle (IA).

Il existe plusieurs méthode pour expliquer la prédiction d'un modèle comme l'importance des variables qui est définie de manière globale sur toutes les observations. Néanmoins, l'importance globale est insuffisante pour expliquer la prédiction sur un individu ou une partie de la population.

D'autres méthodes permettent d'observer l'influence des variables sur des individus particuliers. C'est le cas de la méthode LIME (Local Interpretable Model-Agnostic Explanations). Son fonctionnement global est de générer aléatoirement de nouveaux individus fictifs proches de l'individu sélectionné et de pondérer ces individus en fonction de leur proximité avec lui. Une régression linéaire est effectuée sur ce voisinage local pour déterminer l'influence des variables sur l'individu.

Bien que plus précis que l'importance des variables, LIME n'est pas complet et possède quelques défauts. En particulier, LIME s'appuie sur le voisinage des individus alors que les clients n'ayant pas rembourser leur prêts sont a priori loin des autres clients du fait en premier lieu du déséquilibre de classe.

Une autre méthode nommée SHAP plus performante a la particularité de connecter la théorie des jeux avec les explications locales en unifiant plusieurs anciennes méthodes comme LIME et la valeur de Shapley. C'est cette méthode qui est utilisée pour interpréter les prédictions du modèle final.

3.1 Valeurs SHAP

SHAP qui signifie "SHapley Additive exPlanations", est une approche pour expliquer un modèle de Machine Learning quelque soit le modèle, basée sur la théorie des jeux.

La théorie des jeux repose sur la résolution de ce problème : "*si nous (les joueurs) collaborons tous, comment diviser la récompense totale obtenue par le groupe*", et le calcul des valeurs de Shapley permet d'y répondre. SHAP récupère la formule des valeurs de Shapley pour transformer les joueurs en variable du modèle et leur récompense en sortie correspond à leur importance. La formule de LIME a permis d'adapter celle des valeurs de Shapley pour gagner en efficacité.

SHAP permet donc une interprétation globale et locale d'un modèle. Elle permet de définir l'importance des variables sur des parties de la population ou même sur un individu isolé ainsi que l'influence positive ou négative que ces variables ont sur la prédiction du modèle.

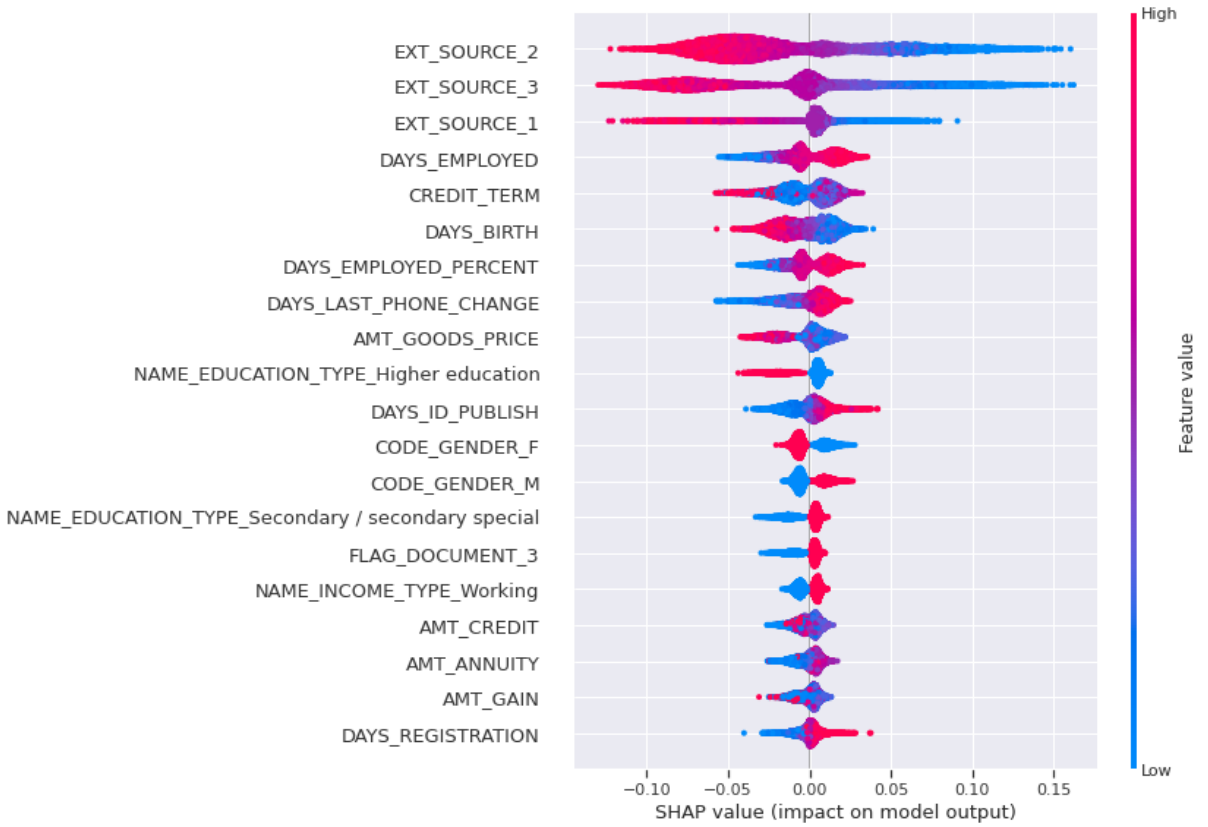


FIGURE 3.1 – Valeurs SHAP sur l'ensemble des observations

La figure 3.1 représente les valeurs SHAP de notre modèle final. On peut en déduire que ce sont les variables EXT_SOURCE_1, 2, 3 qui sont les 3 variables les plus importantes sur tous les individus. Ces variables représentent les sources de financement du client. De plus, on peut déduire aussi qu'une valeur élevée de EXT_SOURCE_2 (le client a une grande source de financement) tend à avoir un impact négatif sur la prédiction (diminue le risque de défaut de recouvrement). A l'inverse, les valeurs proche de 0 de DAY_EMPLOYED (peu de temps que le client a été embauché) augmentent le risque de défaut de recouvrement (impact positif sur la prédiction).

Chapitre 4

Conclusion

4.1 Conclusion

Ce projet a permis de construire un modèle de classification en appliquant des méthodes spécifiques pour gérer un déséquilibre de classe. Les méthodes testées sont le class-weighting, l'oversampling et l'undersampling. L'évaluation de la performance de ces méthodes via un modèle de forêt aléatoire a retenu l'undersampling comme étant la plus précise, en particulier sur le taux de vrais positifs.

Une métrique spécifique a été construite avec une fonction de gain et de perte se basant sur les probabilités calculées par le modèle LightGBM. Le modèle a été optimisé en validation croisée sur des hyper-paramètres et sur un seuil variant de 0 à 1. Une vérification sur un jeu de test a permis de valider le modèle sur les métriques classiques et la métrique spécifique.

L'interprétabilité du modèle a été réalisé en calculant les valeurs SHAP. Les caractéristiques de sources de financement du client (EXT_SOURCE_1, 2, 3) sont les 3 plus importantes en moyenne pour tous les individus. Les valeurs élevées de EXT_SOURCE_2 diminuent le risque de défaut de recouvrement. A l'inverse, les valeurs proches de 0 de DAY_EMPLOYED (peu de temps que le client a été embauché) augmentent le risque de défaut de recouvrement.

4.2 Perspective

La prédiction du modèle pourra être améliorée par plus de feature-engineering en s'inspirant des notebook sur **kaggle**. Un modèle de boosting plus précis (XGboost) pourra être entraîné pour améliorer les résultats.

On pourra construire une métrique plus adaptée à la problématique en calculant par exemple la somme des intérêts cumulés du prêt plutôt que le simple montant.