OC-DS : Projet 7 – Note méthodologique

Problématique

Le projet consiste à mettre en œuvre un outil de « scoring crédit » pour calculer la probabilité qu'un client puisse rembourser le crédit qu'il demande puis, utiliser cette donnée pour accorder ou refuser sa demande. De plus, ce projet doit aussi répondre à une demande grandissante de transparence de la part des clients vis-à-vis des décisions d'octroi de crédit.

Objectifs

1. Cet outil doit permettre de définir la probabilité de défaut de remboursement d'un crédit sur la base d'informations relatives au client.

2. Il doit également offrir un certain niveau de transparence concernant les données et leurs traitements en vue d'implémenter des méthodes d'interprétabilité des features sous la forme d'un dashboard interactif.

3. Une simulation du data drift est aussi demandé en vue de déterminer une période de maintenance pour conserver des performances de prédictions optimales.

I) Données

1) Structure et caractéristiques

Le jeu de données est constitué de toute sorte d'informations relatives aux clients comme leurs crédits en cours, leurs genres ou encore leurs dates de naissances. Il se découpe en de multiples sous-datasets qui va falloir regrouper et traiter. Pour ce faire le kernel Kaggle revu de jsaguar a été utilisé comme fondation à ce travail pour sa clarté et son efficacité apparente.

Une fois tous les sous-datasets aggrégés, en un dataset global avec des features supplémentaires ajoutées par le kernel de jsaguar (ramenant le nombre de features à un total de 622), une dizaine de features se retrouvent complétement vides et à peu près 40 % des features sont à moins de 50 % remplies.
Le nombre de features étant important seules les features remplies à plus de 75 % sont conservées ramenant leur nombre à 360. Ceci permet à la fois de réduire le temps de traitement des données par le modèle ainsi que de limiter le biais induit par l'imputation des valeurs manquantes restantes.

⇨ On obtient ainsi de meilleures performances :

- Un temps de traitement réduit.
- Une certaine fiabilité entre les données et les clients physiques qu'elles représentent.

Une partie du dataset concerne des crédits échus et comprend donc également des étiquettes (une classe binaire attribuée à chaque dossier client pour indiquer si le crédit a été remboursé ou non) qui pourront être utilisées pour entraîner le modèle de classification.
Toutefois, une autre partie du jeu de données, moins conséquente, ne contient pas d'étiquette. Il s'agit des dossiers en cours pour lesquels il faut réaliser un classement prédictif avec le modèle entraîné.

2) Déséquilibre des classes

Parmi les données correspondantes au dataset d'entraînement, on peut remarquer que le taux de crédits remboursé est 10 fois supérieur au taux de défaut de remboursement.

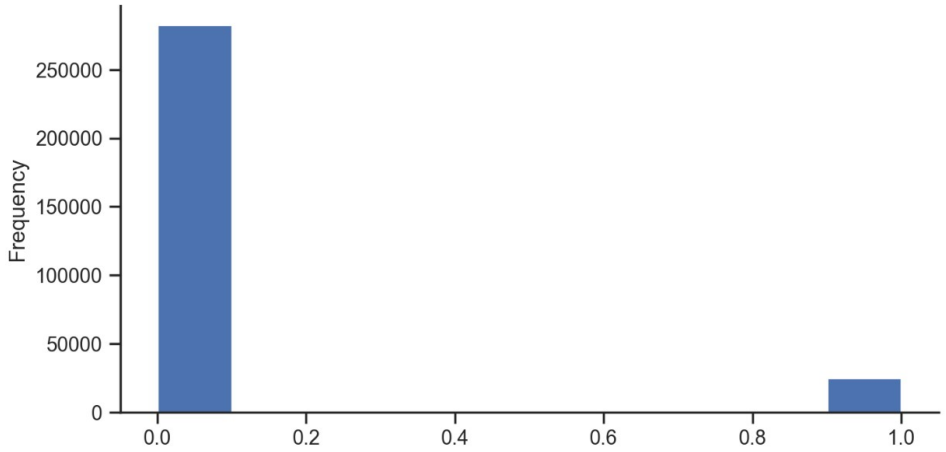

***Figure 1** : Déséquilibre entre le nombre de clients qui ont remboursé leurs prêts (classés 0) et ceux qui sont en défaut (classés 1).*

Deux approches sont possibles pour traiter cette différence lors de l'entraînement du modèle :

- L'attribution d'un poids qui donnera davantage d'importance aux clients en défaut de remboursement, moins nombreux. L'avantage de cette méthode est la rapidité de traitement pour le modèle et un jeu de données inchangé donc, certainement plus fiable.

- L'utilisation d'une méthode de ré-échantillonnage (resampling) qui, contrairement à l'attribution de poids, va générer des données fictives (ou en retirer des réelles) à l'une ou l'autres des classes pour les rééquilibrer l'une envers l'autre. Cette étape supplémentaire modifie le dataset et allonge le temps de traitement particulièrement dans un cas d'oversampling (ajout d'échantillons fictifs basés sur ceux déjà existant) ou retire des données potentiellement utiles du jeu d'entraînement dans un cas d'undersampling (retrait de données).
  L'algorithme SMOTE et ses variantes permettent de jouer un peu sur les 2 tableaux pour tenter de bénéficier des avantages de ces 2 formes opposées de resampling.

  ⇨ In fine, le modèle choisi est basé sur la méthode des poids. Plus simple à mettre en place, plus rapide avec des performances légèrement supérieures à celles obtenues par le resampling des classes.

II) Modélisation

1) Méthode de classification

a) Principe

Rappelons que la probabilité de remboursement d'un crédit est la donnée déterminante pour accepter ou non son attribution à un client qui en fait la demande.

En appliquant un seuil à cette valeur, il est possible de classer la demande du client de façon négative ou positive en lui attribuant une valeur binaire (0 ou 1) selon la probabilité de remboursement qui lui a été prédite par rapport à ce seuil.
Si la probabilité est inférieure, on considère que le crédit sera remboursé et le dossier est classé négatif (0). Inversement, si la probabilité est supérieure au seuil, on considère que le crédit ne sera pas remboursé, le dossier est positif (1).

⇨ Il s'agit donc d'un problème de classification binaire dont le résultat dépend du seuil fixé.

b) Mesures

Pour évaluer le modèle on peut comparer les valeurs prédites avec les valeurs réelles et mettre en place un tableau indiquant le nombre de valeurs correctement prédites ou non (matrice de confusion).

⇨ Ainsi, notre tableau d'évaluation montre le nombre de valeurs binaires (0 ou 1) bien ou mal classées selon un seuil de classification prédéfini.

⇨ Pour un modèle de classification optimal, il est donc nécessaire de bien déterminer ce seuil par rapport aux prédictions du modèle et, indirectement, des données relatives aux clients.

c) Indicateurs techniques de performance

Les valeurs de ce tableau peuvent être converties en indicateurs qui caractérisent le modèle :

- La **sensibilité** (= TP / (TP + FN)) : représente la capacité du modèle à détecter les dossiers de crédits non-remboursés (codés 1).
- La **spécificité** (= TN / (TN + FP)) : représente la capacité du modèle à détecter les dossiers de crédits remboursés (codés 0).
- La **précision** (= TP / (TP + FP)) : représente la capacité du modèle à détecter les vrais dossiers non-remboursés (codés 1)

Grâce à ces indicateurs, il est possible de représenter les évolutions de la sensibilité et de la spécificité en fonction du seuil de classification avec une courbe ROC.

d) AUROC

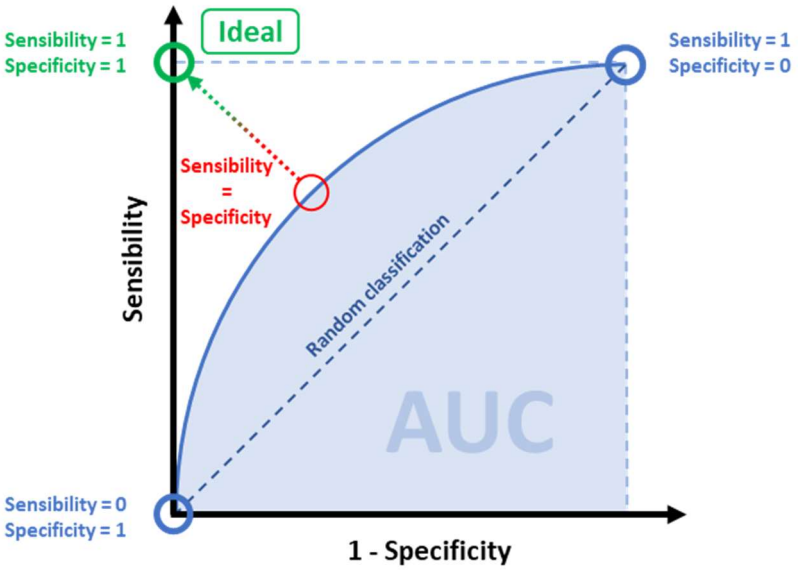

***Figure 2** : Principe d'une courbe ROC et du calcul de son aire (AUC) associée.*

Une courbe ROC caractérise le classifieur qui a produit les résultats sous forme de probabilités par variation du seuil de classification.

Un modèle idéal a une sensibilité et une spécificité de 1. Autrement dit, meilleurs sont les indicateurs, plus la courbe se rapproche de cet idéal (indiqué en vert) et plus le modèle est performant.
Il est possible de mesurer cette performance par l'aire sous la courbe qui prend l'acronyme anglais AUC pour *Area Under the Curve*.

⇨ C'est pourquoi, la mesure AUC est utilisée pour évaluer des modèles de classification et représente à ce titre un critère de sélection de ces modèles.

e) Optimisation métier

Cette approche, bien que pertinente dans l'hypothèse où les éléments de la matrice de confusion sont de même importance, l'est un peu moins dans le cas de notre problématique.

Si la demande d'un client est refusée à tort, à cause d'un risque de défaut de paiement jugé trop élevé, l'entreprise perd ce client et donc le bénéfice potentiel qui lui est associé. En revanche, si elle accorde un crédit à un client qui finalement s'avère ne pas le rembourser alors, la perte est beaucoup plus importante puisqu'elle correspond au total du prêt qui reste à rembourser.

⇨ Il s'agit donc, de trouver le meilleur compromis entre le nombre de crédits accordés à tort, coûteux pour l'entreprise (les faux négatifs) et le nombre de crédit refusés à tort sur des clients solvables (les faux positifs) qui représentent un manque à gagner certain.

⇨ Un score métier peut ainsi être créé pour tenir compte de cette problématique.

f) Fonction de coût métier

Il est ainsi possible de définir une fonction de coût en accordant des poids différents aux éléments de la matrice de confusion lors de l'entraînement du modèle.
Un poids de -1 est affecté à chaque faux positifs (FP) et un poids de -10 aux dossiers identifiés comme faux négatifs (FN) car estimés 10 fois plus néfastes pour l'accroissement des bénéfices de l'entreprise. Quant aux dossiers identifiés comme réellement négatifs (TN) ou réellement positifs (TP), ils sont considérés normaux et voit donc leurs poids associés rester nuls.

Le gain de classification est alors défini par la fonction suivante :

$$Gain = 0 \times TP + 0 \times TN - 1 \times FP - 10 \times FN$$

Ainsi, l'optimisation métier du classifieur consiste à maximiser le Gain (normalisé pour la circonstance) en faisant varier le paramètre seuil de classification mais également les paramètres du classifieur.

Cette opération est réalisée lors de l'optimisation des hyperparamètres avec la méthode Hyperopt par l'intermédiaire d'un dictionnaire de paramètres (dont chacun possède une gamme de valeurs à combiner) passé à la fonction objective avec pour métrique d'évaluation, le Gain normalisé à maximiser.

⇨ L'optimisation métier résulte en un Gain normalisé de 0.70 et une valeur AUC de 0.77.

2) Modèle

a) Architecture du modèle

Le modèle doit :

- être de capable gérer des données déséquilibrées.
- posséder un classifieur qui intègre des méthodes de classification binaires et qui est capable de traiter un nombre important de données dans un temps d'exécution raisonnable.
- avoir une partie capable de traiter et d'interpréter l'importance des features par rapport au résultat de la classification.

b) Optimisation des hyperparamètres

Deux techniques d'optimisation ont été mises en place :

1. Une première basée sur une optimisation aléatoire au moyen de la fonction RandomizedSearchCV() de scikit-learn pour comparer les différents modèles testés entre-eux. De cette façon, on obtient rapidement une première combinaison d'hyperparamètres pour lesquels chaque modèle peut afficher un score proche de l'optimal.

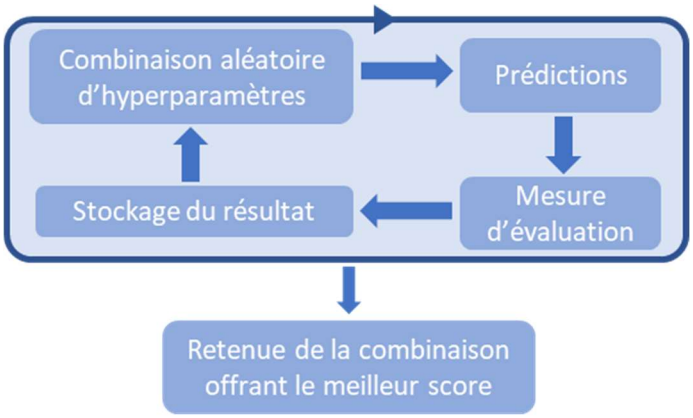

Figure 3 : *Principe de fonctionnement d'un algorithme d'optimisation aléatoire d'hyperparamètres.*

2. Une fois un modèle sélectionné, il passe une seconde étape d'optimisation pour cette fois-ci essayer d'atteindre la meilleure combinaison d'hyperparamètres possible. Cependant, l'optimiseur n'est plus aléatoire. Il tient compte des résultats obtenus avec la combinaison d'hyperparamètres précédentes pour définir la nouvelle combinaison à essayer.

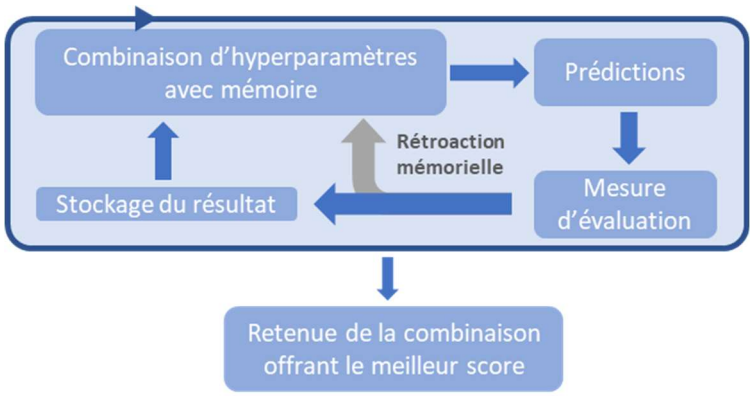

Figure 4 : *Principe de fonctionnement d'un algorithme d'optimisation bayesien d'hyperparamètres.*

⇨ Ainsi, pour cette étape, la fonction d'optimisation aléatoire de scikit-learn est remplacée par la fonction d'optimisation bayesienne d'Hyperopt.

c) Modèles testés

Au cours de la phase de développement plusieurs types de modèles ont été testés : le Dummy Classifieur comme base line, la Régression Logistique, la Random Forest, le XGBoost, et le LightGBM. Après plusieurs essais, il s'avère que les algorithmes les plus adaptés à notre problématique sont basés sur les arbres de décision.

⇨ Le classifieur répondant le mieux à l'ensemble des critères est le LightGBM qui affiche, à la fois, les meilleurs scores techniques et métiers avec à un temps d'entraînement et de prédiction court.

d) Optimisation plus fine du modèle choisi

Après 150 itérations on peut remarquer une légère augmentation des performances du modèle (de l'ordre de 0.5 %) ainsi que du temps de prédiction. On constate que les effets de l'optimisation sont bien présents, bien qu'ils soient minimes, dans cette configuration.
Ces observations tendent à montrer que la combinaison de paramètres obtenue grâce à la fonction de scikit-learn est déjà proche de l'optimal.

⇨ Par conséquent, il semble possible d'améliorer davantage les performances du modèle avec un nombre d'itérations à l'entraînement plus important. Toutefois, le temps de prédiction reste à surveiller. Effectivement, il n'est pas nécessairement avantageux d'augmenter légèrement les performances de prédiction au détriment du temps d'exécution.

e) Synthèse des résultats

Tableau 1 : *Résultats obtenus par cross validation du jeu de données d'entraînement.*

Modèles	Seuil	Score métier	AUROC	F2.5	Temps (s)	Sensibilité	Spécificité	Précision
Poids & classifieur seul								
wt_dummy_clf	1.000	0.532	0.498	0.078	0.677	0.000	1.000	0.919
wt_logi_reg_clf_opt	0.500	0.613	0.653	0.286	10.808	0.642	0.587	0.592
wt_rf_clf_opt	0.360	0.687	0.747	0.360	1053.115	0.703	0.673	0.676
wt_xgb_clf_opt	0.430	0.667	0.721	0.336	4.330	0.673	0.661	0.662
wt_lgbm_clf_opt	0.435	0.699	0.768	0.380	6.977	0.734	0.669	0.674
Normalisation Min Max + poids & classifieur								
scl_wt_dummy_clf	1.000	0.532	0.498	0.078	2.054	0.000	1.000	0.919
scl_wt_logi_reg_clf_opt	0.490	0.703	0.769	0.382	11.483	0.706	0.700	0.700
scl_wt_rf_clf_opt	0.380	0.687	0.746	0.362	1198.790	0.671	0.700	0.698
scl_wt_xgb_clf_opt	0.440	0.668	0.722	0.338	5.530	0.662	0.672	0.672
scl_wt_lgbm_clf_opt	0.440	0.702	0.766	0.380	7.095	0.697	0.707	0.706
Normalisation Min Max + rééchantillonnage + classifieur								
scl_resp_dummy_clf	1.000	0.532	0.498	0.200	8.920	0.000	1.000	0.919
scl_resp_logi_reg_clf_opt	0.415	0.691	0.756	0.369	24.826	0.719	0.668	0.672
scl_resp_rf_clf_opt	0.290	0.673	0.728	0.344	99.860	0.680	0.667	0.668
scl_resp_xgb_clf_opt	0.425	0.660	0.701	0.329	12.146	0.700	0.624	0.630
scl_resp_lgbm_clf_opt	0.395	0.655	0.697	0.324	14.058	0.654	0.656	0.656
Normalisation Min Max + poids & classifieur + optimisation bayesienne								
scl_wt_lgbm_clf_fine_opt	0.450	0.705	0.771	0.438	11.722	0.707	0.702	0.703

Bien que les scores techniques et métier (AUROC et gain) montrent les mêmes tendances, on peut aussi noter quelques particularités :

- La régression logistique tire avantage de la mise à l'échelle des données puisqu'elle rattrape les performances obtenues avec LightGBM (meilleur classifieur jusqu'ici) qui, quant à lui, ne semble pas tirer parti de la normalisation min max.

- On remarque également que le suréchantillonnage apporte des résultats inattendus : LightGBM devient le modèle le moins performant tandis que le régresseur logistique est le meilleur. Cette observation s'opposant à celles faites sur les classifieurs seuls, elle suggère

que les nouveaux échantillons créés ne sont pas forcément très représentatifs de la réalité. De plus, les scores AUC de tous les modèles sont inférieurs avec cette méthode par rapport à ce que nous obtenons sans rééchantillonnage.

- La Random Forest demande clairement beaucoup plus de temps pour s'entraîner et renvoyer une prédiction. Cette caractéristique l'élimine d'office.
  La régression logistique est 25 % plus lente que LightGBM dans tous les cas de figures.
  Le rééchantillonnage ralentit considérablement tous les temps d'entraînement et de prédiction des modèles de 100 à 200 %. On pouvait s'y attendre puisque des opérations de suréchantillonnage sont ajoutées et davantage de données doivent être traitées ensuite. Néanmoins, la forêt aléatoire fait exception puisqu'elle bénéficie étonnamment d'un boost de 1000% en termes de vitesse.

Tableau 2 : *Résultats obtenus sur le jeu de données de test par la méthode des poids.*

Modèles	Seuil	Score métier	AUROC	F2.5	Temps (s)	Sensibilité	Spécificité	Précision
Normalisation Min Max + poids & classifieur								
scl_wt_dummy_clf	0.000	0.529	0.502	0.084	0.516	0.084	0.920	0.853
scl_wt_logi_reg_clf_opt	0.500	0.700	0.765	0.373	2.983	0.696	0.704	0.703
scl_wt_rf_clf_opt	0.350	0.687	0.752	0.364	302.130	0.728	0.650	0.656
scl_wt_xgb_clf_opt	0.445	0.672	0.727	0.341	1.308	0.649	0.691	0.688
scl_wt_lgbm_clf_opt	0.485	0.703	0.767	0.380	1.800	0.658	0.743	0.736
Normalisation Min Max + poids & classifieur + optimisation bayesienne								
scl_wt_lgbm_clf_fine_opt	0.460	0.701	0.768	0.432	10.753	0.676	0.723	0.719

On peut aussi observer une bonne généralisation des modèles dans tous les cas.

⇨ Compte tenu de ses performances techniques et métier ainsi que de vitesse, LightGBM, couplé à la méthode d'attribution des poids, semble être le classifieur le plus prometteur de ceux testés.

III) Interprétation du modèle

NB : *Les graphiques présentés ci-dessous sont tirés du dashboard. Il a été jugé que pour des personnes non averties, il est plus intuitif de comprendre que plus la probabilité est élevée plus elles ont de chances d'obtenir un prêt.*

⇨ Ainsi, les concepts vus jusqu'ici, de dossiers classés positifs (pas assez fiables) ou négatifs (assez fiables), voient leurs valeurs inversées.

1) Importance des features

Les modèles testés offrent tous une méthode permettant d'extraire les features ayant le plus d'influence sur la classification. Cependant, après de multiples recherches, il s'avère qu'il existe des techniques plus générales (adaptées à la plupart des modèles) et, semble-t-il, souvent plus fiables comme SHAP par exemple, qui est utilisé dans ce projet.

La méthode SHAP (SHapley Additive exPlanations) consiste à calculer la valeur de Shapley pour toutes les features de tous les individus. Autrement dit, la moyenne de l'impact d'une feature sur la prédiction renvoyée par le classifieur pour toutes les combinaisons de variables possibles.

⇨ La somme des effets de chaque variable explique alors la prédiction.

Ce score permet de disposer d'une information relative à l'importance globale des features dans le modèle qui peut à la fois être débattue et validée avec les experts métier pour améliorer le modèle et favoriser la transparence vis-à-vis des clients sur les éléments qui ont le plus influencés le modèle sur sa prédiction.

Un autre avantage de SHAP est sa capacité à pouvoir offrir en plus d'une analyse locale une analyse globale. En effet, en moyennant les valeurs absolues des valeurs de chaque feature (au niveau local), il est possible de remonter à l'importance globale de celles-ci.

2) Interprétabilité globale

Le jeu de données, décrit plus haut, a été utilisé pour l'optimisation des paramètres du classifieur LightGBM avec pour objectif la maximisation du gain selon une fonction d'évaluation définie précédemment.

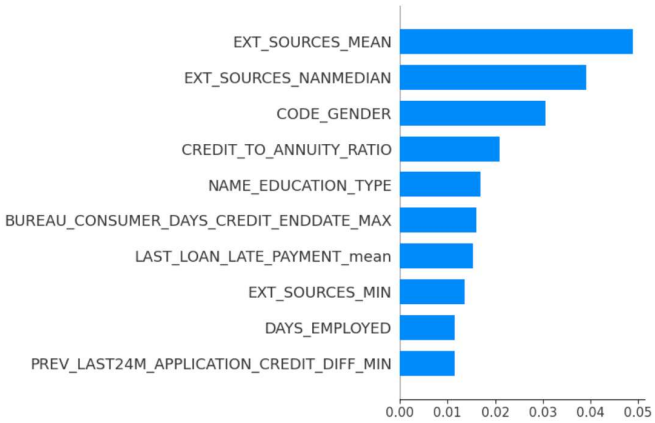

Figure 3 : *Extrait des features globalement les plus influentes sur les prédictions du modèle, classées selon leurs scores attribués par l'algorithme SHAP.*

Cette classification des features, par leur degré d'influence sur le classifieur, représente un premier niveau d'interprétation du modèle.

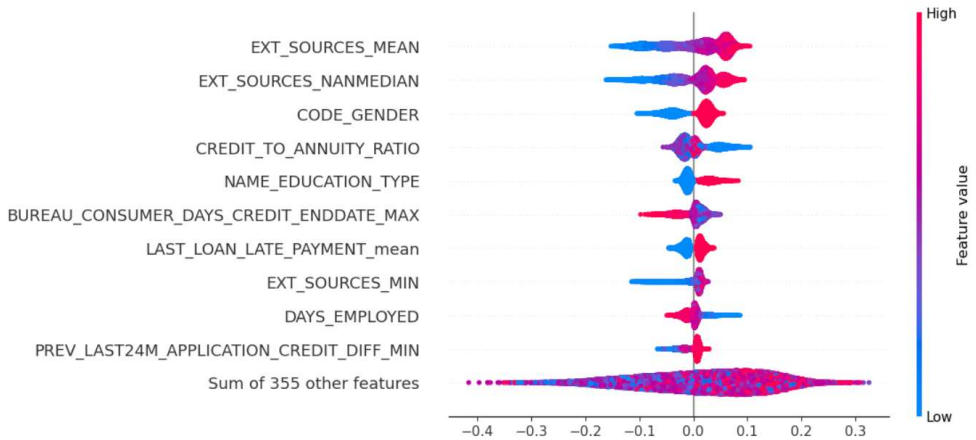

Figure 4 : *Distribution des dossiers des clients du dataset selon les features globalement les plus influentes sur les prédictions du modèle. Les couleurs représentent les valeurs que peuvent prendre ces features.*

Les points affichés sur la figure représentent les dossiers dont la couleur varie en fonction de la valeur de la feature (petite à grande valeur : bleu à rouge). Ces points sont positionnés sur l'axe des abscisses selon leur valeur SHAP caractérisant l'impact de la variable sur la prédiction.

Ex : Beaucoup de points rouges localisés sur la partie droite du graphique montre que les valeurs élevées contribuent positivement à l'estimation de remboursement d'un crédit et par conséquent, à l'attribution de ce-dernier.

3) Interprétabilité locale

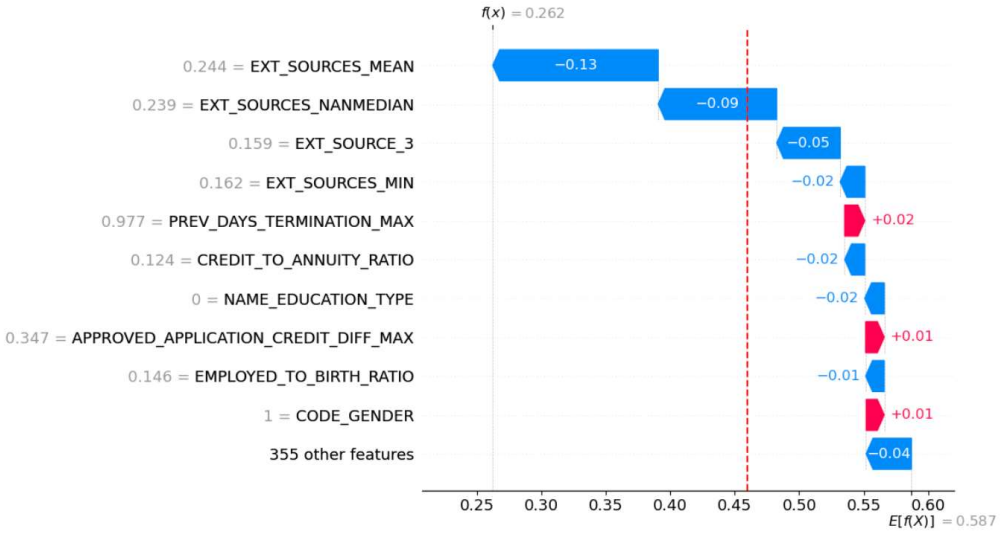

Figure 5 : *Extrait des features les plus influentes sur les prédictions du modèle pour un client anonyme qui se voit refuser sa demande de prêt, classées selon leurs scores attribués par l'algorithme SHAP. Le trait rouge représente le seuil de classification.*

Le graphique permet d'observer quelles sont les features qui semblent les plus influentes sur la recommandation du modèle pour un client. Il montre aussi de façon simple de quelle manière chaque feature a pu influencer le résultat délivré par le modèle :

- **En rouge :** les features qui ont influencé négativement l'attribution d'un crédit au client.
- **En bleu :** les features qui ont influencé positivement l'attribution d'un crédit au client.

IV) Analyse de data drift

1) Data drift

Entre le jeu de données d'entraînement celui des nouveaux clients à classer, on observe un drift d'environ 18 %. Il serait sans doute possible de le réduire et rester plus proche du profil des données d'entraînement en traitant les dossiers plus rapidement.

⇨ Cependant, on peut considérer que, pour le moment, cette valeur est normale.

2) Simulation de drift dans le temps

NB : *Une feature est considérée driftée si sont profil de valeurs change de plus de 5 %.*

La simulation se déroule sur un peu moins d'une quarantaine d'années. Elle débute à partir de 20 ans de données puis mesure l'influence sur le profil global des données déjà enregistrées lors de l'ajout de données supplémentaires (le data drift) de plus en plus récentes par itération de 2 ans.

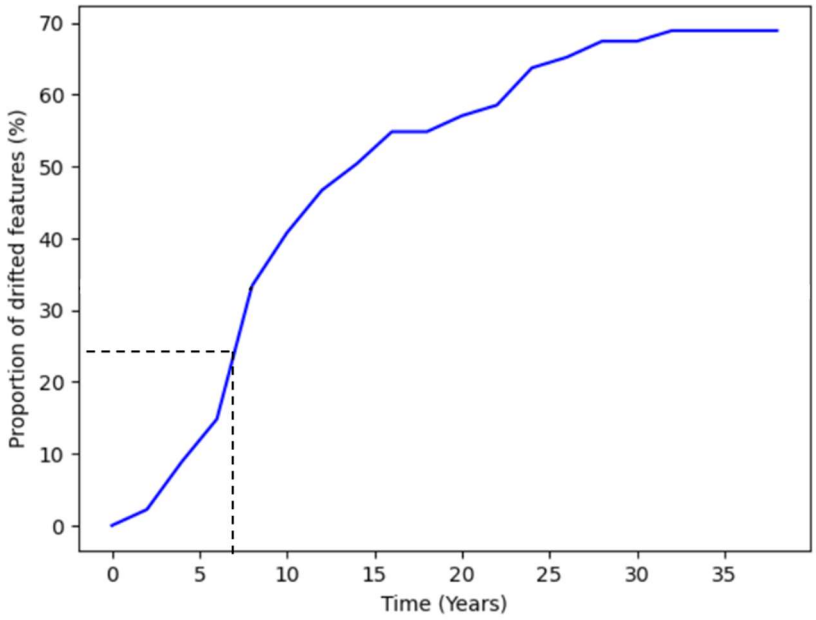

Figure 6 : *Simulation de data drift des 135 features les plus influentes sur les prédictions du modèle (influence relative supérieure à 0.1 % d'importance).*

Sur la figure, on peut remarquer que le profil des données change de plus en plus rapidement au cours des 10 premières années pour atteindre 33 % de drift aux alentours de 8 ans (dont 13 % entre la 6^{e} et la 7^{e} année).
Dans le but de conserver des performances optimales, il est nécessaire d'établir un seuil de drift maximal, qui une fois atteint, indique la nécessité d'effectuer une maintenance sur notre outil (i.e. de ré-étalonner/ré-entaîner le modèle sur le nouveau profil de données).

Une méthode pour estimer ce seuil consiste à observer, parmi les features détectées comme ayant driftées, celles faisant partie du top 10 des features les plus influentes. Il est ainsi possible de considérer que lorsque 2 ou 3 features très influentes pour le modèle ont déjà drifté, il est préférable d'effectuer une maintenance.

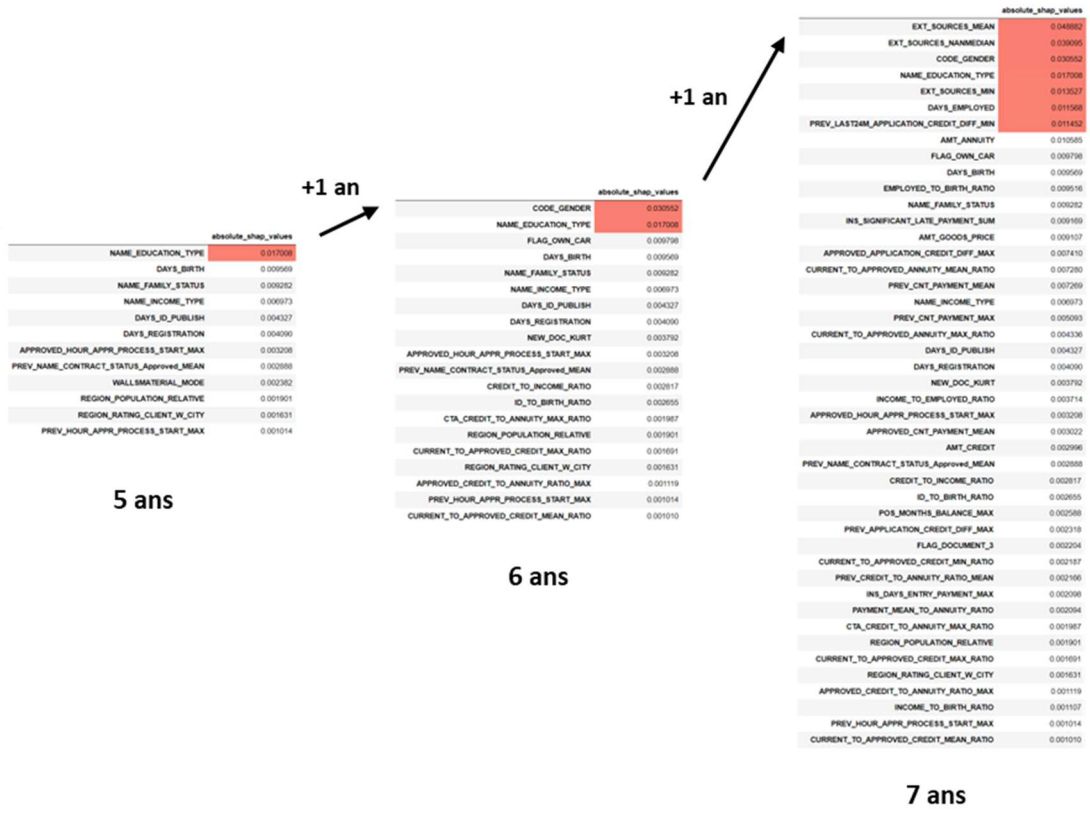

Figure 7 : *Features ayant driftées au cours des 7 premières années. En rouge les features faisant parties du top 10 des features les plus influentes sur les prédictions du modèle.*

Comme le montre cette figure, on peut remarquer qu'entre 6 et 7 ans beaucoup de features commencent à drifter et notamment une proportion significative des top 10 features.

⇨ D'après ces 2 figures, il serait certainement judicieux de fixer une première date de maintenance entre la 6^{e} et la 7^{e} année correspondant à un drift global d'environ 25 %.

V) Perspectives

La réalisation du modèle a nécessité la conception de nombreux blocs de transformation et de traitement des données dans son architecture (comme en atteste les multiples notebooks). Chacun fait appel à des méthodes paramétrables et les résultats sont dépendants des paramètres choisis.

⇨ Ainsi, les choix des paramètres utilisés à chaque étape peuvent être changés ou affinés.

1) Sélection des variables

Les informations disponibles relatives à l'importance des variables devraient être débattues avec les experts métier en vue de définir les stratégies techniques à tester dans les différents blocs concernés :

- Seuil des valeurs manquantes.
- Techniques d'imputation.
- Corrélations entre variables.
- Réduction de dimensions et seuil de variance.

2) Equilibrage des données

- Changer la méthode d'attribution des poids.
- L'équilibrage des données introduit des données artificielles dans les cas de rééchantillonnage et la possibilité d'incohérences. Des tests peuvent être réalisés en variant les algorithmes et leurs paramètres.

3) Fonction d'évaluation du gain

Communiquer les règles métier et les critères financiers relatifs aux pertes et profits pourrait permettre d'affiner davantage la fonction d'évaluation du gain pour être plus proche des contraintes métiers.

4) Classifieurs et optimisation des hyperparamètres

Plusieurs classifieurs ont été testés avant de retenir LightGBM pour son couple rapidité d'exécution - performances. D'autres classifieurs comme XGBoost ou le stacking de modèles (Non-testé au cours de cette étude) peuvent potentiellement apporter de meilleures performances techniques. Il s'agit de les tester dans le pipeline complet si les contraintes de temps en production le permettent.

Pour encore affiner les performances du modèle, il possible de continuer son optimisation avec davantage d'itérations sur l'algorithme d'Hyperopt et de sélectionner d'autres plages de valeurs pour les hyperparamètres testés.

5) Interprétabilité

La méthode SHAP offre des résultats intéressants, en particulier pour l'interprétabilité locale. Par contre, son intégration dans une application frontend comme streamlit demeure assez mal prise en charge et demande souvent l'ajout de code, voir, l'import de bibliothèques supplémentaires non-officielles pour compenser ce manque de compatibilité afficher ses graphiques.

Concernant l'interprétabilité globale, les algorithmes basés sur les forêts aléatoires (mais également SHAP) permettent d'identifier les variables influentes. Cependant, d'autres approches comme les permutations de variables peuvent être testées pour valider ces résultats.

6) Dashboard

Séparer en 2 onglets différents l'application frontend avec :

- les interprétations locales relatives aux clients sur le premier onglet.
- les interprétations globales, davantage destinées aux chargés de communication, avec des fonctionnalités différentes comme la matrice de confusion par exemple.

Ajouter un panneau latéral affichant les caractéristiques propres au client (âge, genre, adresse …) pour en faciliter l'accès.

7) Simulation du data drift

Discuter avec les experts métiers et en apprendre davantage sur le coût monétaire d'une maintenance pour mieux adapter les seuils de drift de la simulation et définir une période de maintenance plus optimale.

VI) Annexes

Tableau 2 : *Résultats obtenus sur le jeu de données de test.*

Modèles	Seuil	Score métier	AUROC	F2.5	Temps (s)	Sensibilité	Spécificité	Précision
Poids & Classifieur seul								
wt_dummy_clf	0.000	0.532	0.528	0.498	0.501	0.078	0.084	0.677
wt_logi_reg_clf_opt	0.505	0.613	0.614	0.653	0.653	0.286	0.287	10.808
wt_rf_clf_opt	0.355	0.687	0.683	0.747	0.750	0.360	0.365	1053.115
wt_xgb_clf_opt	0.450	0.667	0.668	0.721	0.724	0.336	0.340	4.330
wt_lgbm_clf_opt	0.465	0.699	0.700	0.768	0.769	0.380	0.381	6.977
Normalisation Min Max + poids & classifieur								
scl_wt_dummy_clf	0.500	0.703	0.700	0.769	0.765	0.382	0.373	11.483
scl_wt_logi_reg_clf_opt	0.350	0.687	0.687	0.746	0.752	0.362	0.364	1198.790
scl_wt_rf_clf_opt	0.445	0.668	0.672	0.722	0.727	0.338	0.341	5.530
scl_wt_xgb_clf_opt	0.485	0.702	0.703	0.766	0.767	0.380	0.380	7.095
scl_wt_lgbm_clf_opt	1.000	0.532	0.532	0.498	0.500	0.200	0.080	8.920
Normalisation Min Max + ré-échantillonnage + classifieur								
scl_resp_dummy_clf	0.300	0.673	0.674	0.728	0.728	0.344	0.345	99.860
scl_resp_logi_reg_clf_opt	0.430	0.660	0.664	0.701	0.705	0.329	0.336	12.146
scl_resp_rf_clf_opt	0.365	0.655	0.659	0.697	0.700	0.324	0.332	14.058
scl_resp_xgb_clf_opt	0.460	0.705	0.701	0.771	0.768	0.438	0.432	11.722
scl_resp_lgbm_clf_opt	0.000	0.532	0.528	0.498	0.501	0.078	0.084	0.677
Normalisation Min Max + poids classifieur								
scl_wt_lgbm_clf_fine_opt	0.355	0.687	0.683	0.747	0.750	0.360	0.365	1053.115