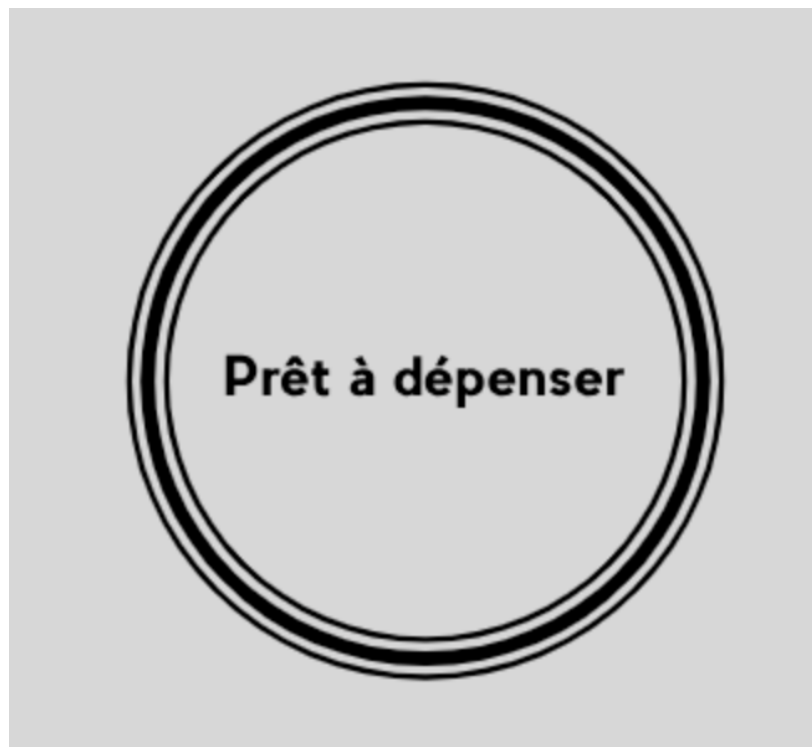


Note Méthodologique

Implémentez un modèle de scoring



Zeynep Erdem
14-10-2023

Liens

Le dossier Github : https://github.com/githubzey/p7_Home_Credit

Api : <https://apihomecredit-861d00eaed91.herokuapp.com/>

Dashboard : <https://dashboardhomecredit-1913c1e69feb.herokuapp.com/>

Le contexte

Je suis Data Scientist au sein d'une société financière, nommée "Prêt à dépenser", qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

"Prêt à dépenser", souhaite développer un outil de "scoring crédit" pour évaluer la probabilité de remboursement des crédits et classer les demandes en crédits accordés ou refusés. L'entreprise veut également répondre à la demande croissante de transparence de la part des clients. Pour ce faire, elle prévoit de créer un tableau de bord interactif permettant aux chargés de relation client d'expliquer les décisions d'octroi de crédit de manière transparente et de fournir aux clients un accès facile à leurs informations personnelles.

La mission:

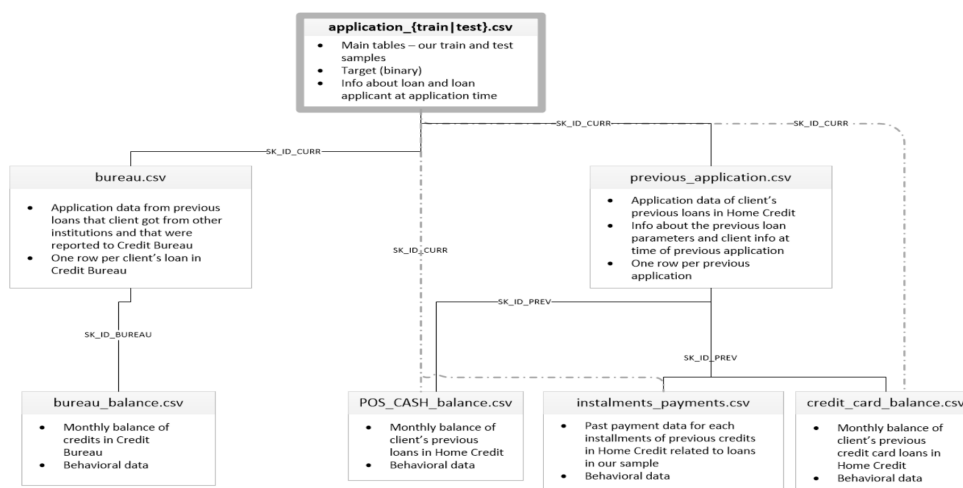
- Construire un modèle de scoring qui donnera une prédiction sur la probabilité de faillite d'un client de façon automatique.
- Construire un dashboard interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d'améliorer la connaissance client des chargés de relation client.
- Mettre en production le modèle de scoring de prédiction à l'aide d'une API, ainsi que le dashboard interactif qui appelle l'API pour les prédictions.

Sommaire

1. Méthodologie d'entraînement du modèle
2. Traitement du déséquilibre des classes
3. Fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation
4. Tableau de synthèse des résultats
5. Interprétabilité globale et locale du modèle
6. Les limites et les améliorations possibles
7. Analyse du Data Drift

1. Méthodologie d'entraînement du modèle

Les [données](#) nécessaires pour ce projet proviennent de la compétition Kaggle "Home Credit Default Risk", comprenant 10 fichiers avec 346 colonnes. Ces fichiers sont liés par des clés.



Pour simplifier le processus d'analyse, de préparation des données et le feature engineering, nous nous appuyons sur [un kernel](#) Kaggle déjà existant.

Les étapes du nettoyage et préparation des données

Voici un schéma de la méthodologie appliquée pour la préparation des données.



La démarche de modélisation

Après le nettoyage des données, nous éliminons les colonnes contenant plus de 30% de valeurs manquantes. Ensuite, nous divisons le jeu de données en ensembles d'entraînement (80%) et de test (20%) pour l'évaluation. Nous utilisons la stratégie de l'imputation médiane pour traiter les valeurs manquantes. Puis, nous éliminons les colonnes de variance nulle et celles fortement corrélées à la variable cible, réduisant le nombre de variables à 409. Les données sont ensuite

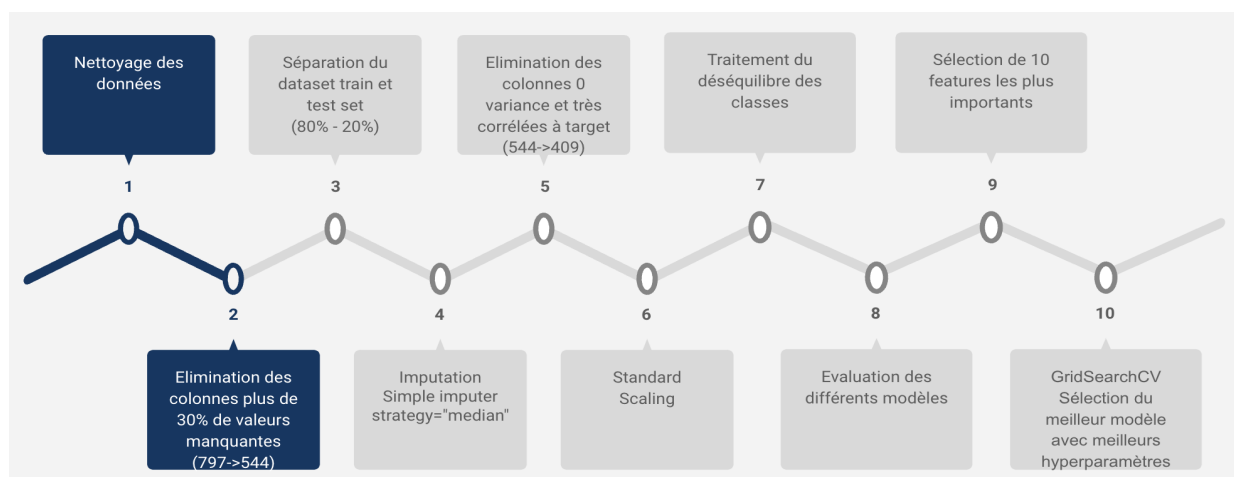
standardisées avec Standard Scaler. Nous abordons le déséquilibre des classes avec des techniques appropriées. Après avoir évalué plusieurs modèles, nous identifions les 10 features les plus importantes, puis optimisons les hyperparamètres avec GridSearchCV. Enfin, nous sélectionnons le meilleur modèle avec les paramètres optimaux pour la mise en production.

Les différents modèles testés

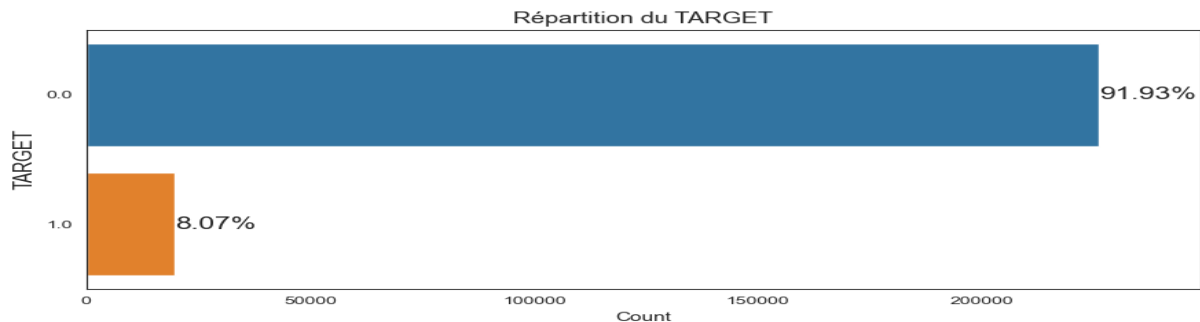
Les modèles ont été évalués en utilisant l'ensemble de données de 409 colonnes sur les ensembles d'entraînement et de test. Les résultats ont été comparés en utilisant diverses métriques d'évaluation, notamment le score métier. Le meilleur modèle (Lgbm) a été sélectionné, et les 10 features les plus importantes pour ce modèle ont été identifiées. Ensuite, un GridSearchCV a été utilisé pour optimiser les hyperparamètres de ce modèle.

- **DummyClassifier**: un modèle simple utilisé comme point de référence. Il attribue des prédictions en utilisant des règles simples, comme le choix de la classe majoritaire dans notre cas.
- **Régression Logistique** : un algorithme de classification qui utilise une fonction logistique. Elle modélise la probabilité d'une observation appartienne à une classe particulière.
- **Random Forest** : un modèle d'ensemble basé sur les arbres de décision. Il combine plusieurs arbres de décision pour améliorer la précision de la prédiction et réduire le surajustement. Chaque arbre est construit sur un échantillon aléatoire des données et des caractéristiques.
- **Light GBM** : un algorithme de boosting qui utilise une méthode d'entraînement en gradient, similaire à d'autres méthodes de boosting. Cependant, il se distingue par sa vitesse et son efficacité. Il est souvent utilisé pour des ensembles de données de grande taille et offre des performances exceptionnelles.

Voici un schéma de la démarche de modélisation.



2. Traitement du déséquilibre des classes



On constate clairement un déséquilibre dans notre variable cible, avec seulement 8% de la classe 1 (clients non solvables) par rapport à 92% de la classe 0 (clients solvables). Pour éviter d'obtenir des résultats biaisés, nous allons explorer différentes techniques pour équilibrer les données et comparer les résultats avec les données d'origine.

RandomUnderSampler : une technique de sous-échantillonnage qui vise à équilibrer les classes en réduisant le nombre d'instances de la classe majoritaire. Il supprime aléatoirement des échantillons de la classe majoritaire jusqu'à atteindre un équilibre entre les classes.

SMOTE : une technique de suréchantillonnage qui vise à équilibrer les classes en créant des échantillons synthétiques de la classe minoritaire. Cela augmente la taille de la classe minoritaire et peut améliorer la performance du modèle en réduisant le biais.

Model(class_weight="balanced") : l'utilisation de l'argument `class_weight="balanced"` lors de la création d'un modèle, cela signifie que le modèle attribuera automatiquement des poids plus importants aux classes minoritaires pendant l'entraînement, réduisant ainsi l'impact du déséquilibre sur la performance du modèle.

On a choisi la méthode d'utilisation de l'argument **class_weight="balanced"**, qui a conduit à de meilleures performances. Voici les résultats pour le modèle LightGBM.

model	accuracy	precision	recall	f1_score	roc_auc_score	custom_score	execution_time
lgbm_origine	0.92	0.55	0.03	0.06	0.78	0.55	14.535
lgbm_unders	0.70	0.17	0.70	0.28	0.77	0.70	5.846
lgbm_overs	0.92	0.52	0.03	0.05	0.77	0.54	28.776
lgbm_balanced	0.72	0.18	0.69	0.29	0.78	0.71	14.838

3. Fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation

a. La fonction de coût métier: On a le déséquilibre du coût métier entre un faux négatif et un faux positif. Donc on a créé un score métier en basant matrice de confusion pour pénaliser plus des faux négatifs.

FN - mauvais client prédit bon client : donc crédit accordé et perte en capital

FP - bon client prédit mauvais : donc refus crédit et manque à gagner en marge

TN - mauvais client prédit mauvais client : donc refus crédit et pas perte

TP - bon client prédit bon : donc crédit accordé et pas perte

	Predicted Positive (1)	Predicted Negative (0)
Actual Positive (1)	True Positive	False Negative
Actual Negative (0)	False Positive	True Negative

coeff_fn = -10

coeff_fp = -1

coeff_tn = 0

coeff_tp = 0

$$total = (coeff_tn * tn + coeff_fp * fp + coeff_fn * fn + coeff_tp * tp)$$

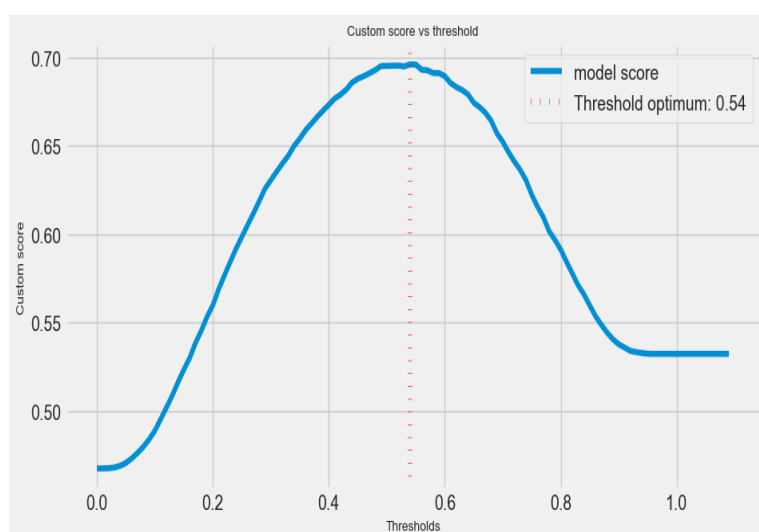
$$max_gain = (tn + fp) * coeff_tn + (tp + fn) * coeff_tp$$

$$min_gain = (tn + fp) * coeff_fp + (tp + fn) * coeff_fn$$

$$gain = (total - min_gain) / (max_gain - min_gain) \rightarrow \text{gain normalisé}$$

On va essayer de maximiser ce score de gain.

b. L'algorithme d'optimisation: Selon les résultats d'évaluations on a choisi LightGBM et effectué un GridSearchCV afin d'optimiser les hyperparamètres. Puis on a calculé le seuil optimum pour le score métier.



Les meilleurs hyperparamètres pour LightGBM avec 10 features

learning_rate: 0.05

n_estimators: 350

Le seuil optimal est 0.54 avec un score métier de 0.7

b. La métrique d'évaluation:

On a évalué nos modèles en utilisant le **score métier**, mais on a également examiné les métriques suivantes.

Accuracy : mesure combien de prédictions correctes le modèle a fait dans l'ensemble des prédictions. C'est un indicateur global de la performance d'un modèle, mais il peut être trompeur en présence de classes déséquilibrées.

Precision : elle est calculée en divisant le nombre d'observations correctement prédites comme positives par le nombre total d'observations prédites comme positives. La précision est utile lorsque les faux positifs sont coûteux.

Recall : il est calculé en divisant le nombre d'observations correctement prédites comme positives parmi toutes les véritables observations positives. Le rappel est important lorsque les faux négatifs sont coûteux.

F1-Score : est une métrique qui combine à la fois la précision et le rappel en une seule valeur. Il est particulièrement utile en cas de déséquilibre entre les classes.

ROC AUC Score : Le score AUC-ROC mesure la capacité d'un modèle à distinguer entre les classes en utilisant la courbe ROC. Il calcule l'aire sous la courbe ROC, qui varie de 0 à 1.

4. Tableau de synthèse des résultats

model	accuracy	precision	recall	f1_score	roc_auc_score	custom_score	execution_time
threshold_best_model	0.74	0.18	0.63	0.28	0.76	0.70	2.967
best_model_lgbm	0.70	0.17	0.68	0.27	0.76	0.70	2.678
lgbm_balanced	0.72	0.18	0.69	0.29	0.78	0.71	14.938
lr_balanced	0.70	0.17	0.69	0.27	0.76	0.70	16.531
lgbm_unders	0.70	0.17	0.70	0.28	0.77	0.70	5.609
lr_unders	0.69	0.17	0.69	0.27	0.76	0.69	3.971
rf_unders	0.69	0.16	0.68	0.26	0.75	0.69	20.563
lr_overs	0.71	0.17	0.65	0.26	0.75	0.68	33.442
lgbm_origine	0.92	0.55	0.03	0.06	0.78	0.55	15.161
lr_origine	0.92	0.50	0.02	0.04	0.76	0.54	16.700
rf_overs	0.92	0.35	0.03	0.05	0.72	0.54	316.575
lgbm_overs	0.92	0.52	0.03	0.05	0.77	0.54	27.332
dummy_origine	0.92	0.00	0.00	0.00	0.50	0.53	9.709
dummy_unders	0.92	0.00	0.00	0.00	0.50	0.53	2.472
dummy_overs	0.92	0.00	0.00	0.00	0.50	0.53	17.460
rf_origine	0.92	0.78	0.00	0.00	0.72	0.53	167.918
rf_balanced	0.92	0.59	0.00	0.00	0.73	0.53	128.659

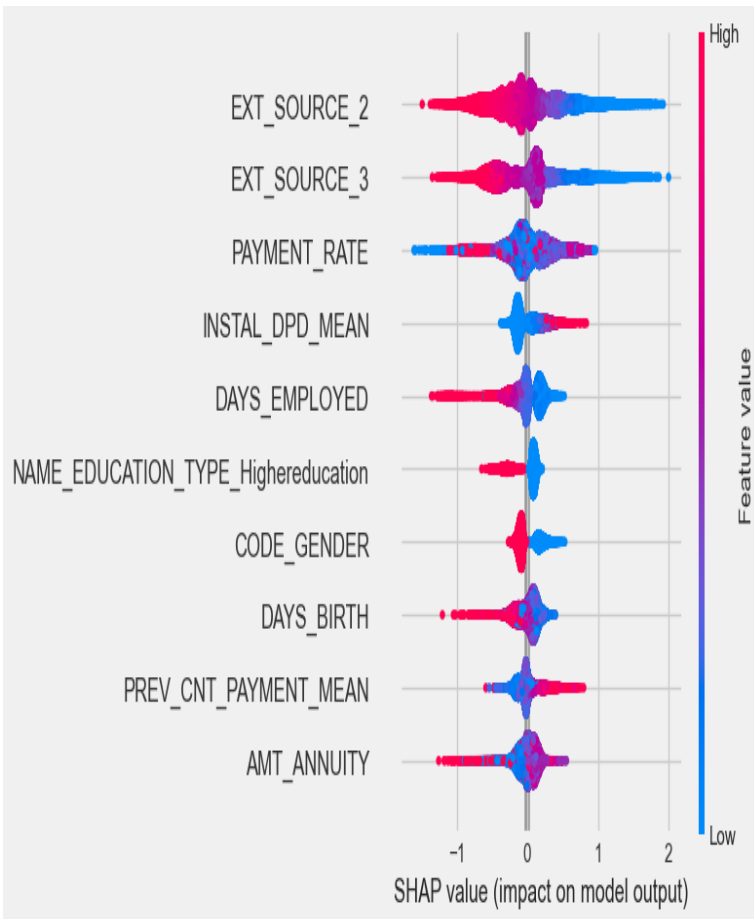
Nous avons testé les algorithmes, le Dummy Classifier, la Régression Logistique, le Random Forest et le Light GBM, en utilisant à la fois les ensembles de données d'origine déséquilibrés et les trois techniques d'équilibrage mentionnées précédemment. Les meilleurs résultats ont été obtenus avec le modèle Light GBM (lgbm_balanced) en utilisant la pondération de classe équilibrée (class_weight balanced).

Ensuite, nous avons sélectionné les 10 features les plus importantes et effectué une GridSearchCV. Selon les résultats, le modèle (best_model_lgbm) nous a donné un custom score de 0.70 et une AUC ROC (roc_auc) de 0.76. Le modèle précédent (lgbm_balanced) avant la sélection de features avait un custom score de 0.71 et une AUC ROC de 0.78. Bien que nous ayons légèrement perdu en performance, nous avons gagné en termes de durée d'exécution et l'interprétabilité du modèle.

Enfin, nous avons ajusté le seuil de décision du modèle (threshold_best_model) en le passant de 0.50 par défaut à 0.54, ce qui a eu très peu d'impact sur les résultats

5. Interprétabilité globale et locale du modèle

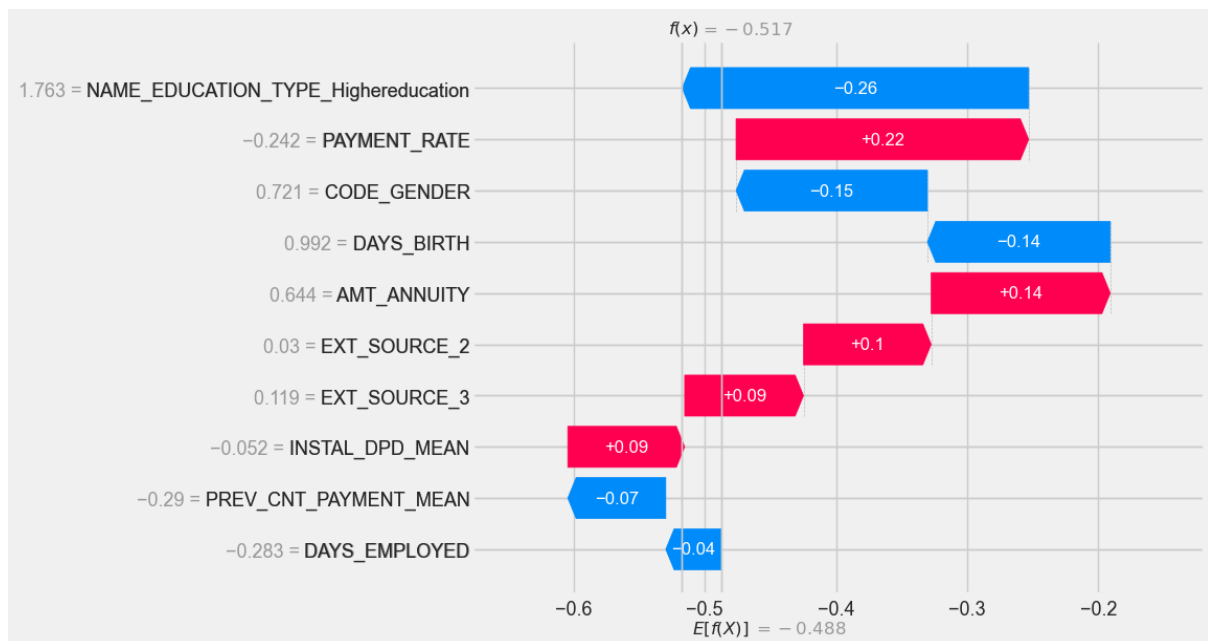
Interprétabilité globale



Feature	Explication
EXT_SOURCE_2 et 3	Score normalisé provenant d'une source de données externe
PAYMENT_RATE	Taux de paiement
INSTAL_DPD_MEAN	Nombre de jours de retard de paiement pour le crédit précédent (moyenne)
DAYS_EMPLOYED	Durée du travail (ans)
NAME_EDUCATION_TYPE_Highereducation	Niveau d'études le plus élevé (éducation supérieure)
CODE_GENDER	Genre female - 1 male - 0
PREV_CNT_PAYMENT_MEAN	Durée du crédit précédent (moyenne)
DAYS_BIRTH	Age (ans)
AMT_ANNUITY	Rente de prêt annuelle

Pour l'analyse globale de l'importance des features, nous utilisons 'summary_plot' pour identifier les features les plus importantes et comment chaque variable influence les prédictions. Sur l'axe vertical, un déplacement vers la gauche indique une réduction du risque de défaut de paiement, tandis qu'un déplacement vers la droite augmente ce risque. Les couleurs rouge et bleue indiquent respectivement des valeurs élevées et faibles. Par exemple, le genre masculin semble contribuer à un risque plus élevé de défaut, tandis qu'une faible valeur pour la EXT_SOURCE_2 et 3 est associée à un risque accru de défaut. De plus, de faibles valeurs pour la DAYS_EMPLOYED et DAYS_BIRTH augmentent le risque de défaut, tandis que des valeurs élevées pour INSTAL_DPD_MEAN augmentent également le risque de défaut.

Interprétabilité locale



Nous examinons maintenant l'impact spécifique des features sur la décision du modèle par rapport à une observation particulière. Le graphique présente l'impact de chaque feature pour l'individu sélectionné et comment ces features influencent la prédiction. Les features ayant un impact négatif sont en bleu, tandis que celles ayant un impact positif sont en rouge. Par exemple, pour le client ayant l'ID 343913, on constate que les features "NAME_EDUCATION_TYPE_Highereducation" et "DAYS_BIRTH" ont réduit le risque de défaut de paiement, tandis que les features "PAYMENT_RATE" et "AMT_ANNUITY" ont augmenté le risque de défaut de paiement pour ce client.

6. Les limites et les améliorations possibles

Pendant l'évaluation de notre modèle, nous avons élaboré une métrique métier personnalisée. Cependant, il serait préférable de collaborer avec les équipes métier pour créer une métrique plus adaptée à leurs besoins spécifiques.

Nous pouvons également améliorer la sélection des features les plus importantes en explorant d'autres méthodes, tout en maintenant un dialogue constant avec les experts métier pour répondre aux exigences du domaine et augmenter l'interprétabilité du modèle.

Enfin, l'amélioration des performances du modèle peut être obtenue grâce à une recherche plus approfondie des hyperparamètres, ce qui contribuera à optimiser davantage le modèle.

7. Analyse du Data Drift

Le "data drift" est la modification de la qualité et de la distribution des données au fil du temps, ce qui peut affecter les performances d'un modèle de machine learning. Il nécessite une surveillance continue et des ajustements pour maintenir la précision du modèle.

On a testé la librairie evidently pour détecter dans le futur du Data Drift en production. Pour cela on a pris comme hypothèse que le dataset "application train" représente les datas pour la modélisation et le dataset "application test" représente les datas de nouveaux clients une fois le modèle en production.

On constate une légère data drift pour 9 colonnes de dataset.

Dataset Drift

Dataset Drift is NOT detected. Dataset drift detection threshold is 0.5











121
Columns

9
Drifted Columns

0.0744
Share of Drifted Columns

Data Drift Summary

Drift is detected for 7.438% of columns (9 out of 121).

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
> AMT_REQ_CREDIT_BUREAU_QRT	num			Detected	Wasserstein distance (normed)	0.359052
> AMT_REQ_CREDIT_BUREAU_MON	num			Detected	Wasserstein distance (normed)	0.281765
> AMT_GOODS_PRICE	num			Detected	Wasserstein distance (normed)	0.210785
> AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.207334
> AMT_ANNUITY	num			Detected	Wasserstein distance (normed)	0.161102