Modèle de Scoring

Prédiction sur la probabilité de faillite d'un client de façon automatique

|  |
| --- |
| **Résumé**  Mise en place d’un modèle de classification pour la prédiction de défaillance d’un prêt bancaire  Ce document décrit la méthodologie d'entraînement du modèle, le traitement du déséquilibre des classes, la définition de la fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation. Le document comprend en outre un tableau de synthèse des résultats et les métriques utilisées. Il inclut des graphiques permettant l’interprétabilité globale et locale du modèle, L’analyse du Data Drift ainsi que les limites et les points d’améliorations possibles du travail. |

Table des matières

[INTRODUCTION 3](#_Toc135611979)

[REFERENCES 3](#_Toc135611980)

[DONNEES D’ENTREES 4](#_Toc135611981)

[FEATURE ENGINEERING 5](#_Toc135611982)

[1. Kernel kaggle 5](#_Toc135611983)

[2. RESAMPLING 6](#_Toc135611984)

[3. PREPROCESSING 8](#_Toc135611985)

[CONSTRUCTION MACHINE LEARNING 9](#_Toc135611986)

[1. Modèle Lgbm 9](#_Toc135611987)

[ACCURACY, PRECISION, RECALL, F1 SCORE 9](#_Toc135611988)

[LA MATRICE DE CONFUSION 10](#_Toc135611989)

[COURBE ROC ET SCORE AUC 11](#_Toc135611990)

[FEATURES IMPORTANCE 12](#_Toc135611991)

[2. OPTIMISATION du modele Lgbm 14](#_Toc135611992)

[PIPELINE ET HYPERPARAMETRES 14](#_Toc135611993)

[Fonction de cout metier 16](#_Toc135611994)

[Data Drift 18](#_Toc135611995)

[AXES D’AMELIORATIONS A aPPORTER 19](#_Toc135611996)

# INTRODUCTION

L’entreprise « Prêt à dépenser » souhaite développer un modèle de scoring de la probabilité de défaut de paiement du client pour étayer la décision d'accorder ou non un prêt à un client potentiel en s’appuyant sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.).

A partir d’un kernel Kaggle existant, qui a permis de faciliter la préparation des données nécessaires, nous avons procédé à l’élaboration du modèle de Scoring.

Nous avons également défini une fonction de minimisation de coût de l’entreprise à partir d’un seuil optimal de décision. Nous avons aussi effectué une analyse du data drift et intégré un module de versioning permettant d’avoir un historique des différents modèles testés, de même que les métriques de performance et les hyper paramètres des différents modèles testés.

# REFERENCES

[1] Lien détail du projet 7 OPENCLASSROOMS :<https://openclassrooms.com/fr/paths/164/projects/632/assignment>

[2] Lien Kernel Kaggle :<https://www.kaggle.com/code/willkoehrsen/start-here-a-gentle-introduction/notebook>

[3] Lien téléchargement des données : <https://www.kaggle.com/c/home-credit-default-risk/data>

[4] Librairie Python « Imblearn » pour équilibrage des données : <https://github.com/scikit-learn-contrib/imbalanced-learn>

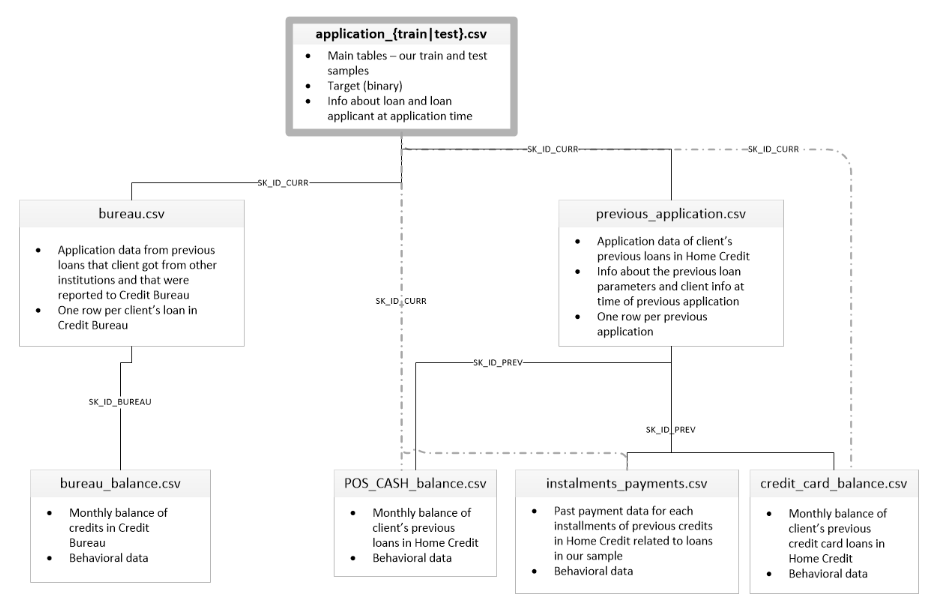
# DONNEES D’ENTREES

L’entreprise met à disposition 7 fichiers CSV contenant des données spécifiques et un fichier description de la base des données.

* application\_train/test: Il s'agit du tableau principal, divisé en deux fichiers pour Train (avec TARGET) et Test (sans TARGET)
* bureau: renseigne sur tous les crédits précédents du client fournis par d'autres institutions financières.
* bureau\_balance: donne les soldes mensuels des crédits précédents.
* POS\_CASH\_balance:donne un instantané des soldes mensuels des prêts POS (point of sales) et cash précédents que le demandeur a eu avec Home Credit.
* credit\_card\_balance: donne un instantanés des soldes mensuels des cartes de crédit précédentes que le demandeur détient auprès de Home Credit
* previous\_application: indique toutes les demandes précédentes de prêts pour le crédit immobilier des clients qui ont des prêts
* installments\_payments: donne l'historique des remboursements pour les crédits précédemment déboursés par Home Credit
* HomeCredit\_columns\_description: contient les descriptions des colonnes des différents fichiers de données.

Nous avons utilisé principalement le fichier « applic ation\_train » pour la modélisation en y ajoutant des features pertinentes d’autres fichiers.

**Description des liens des différentes base de données :**



*Figure 1 : source :* [*https://www.kaggle.com/code/willkoehrsen/start-here-a-gentle-introduction/notebook*](https://www.kaggle.com/code/willkoehrsen/start-here-a-gentle-introduction/notebook)

# FEATURE ENGINEERING

Dans cette partie, nous décrivons le Kernel Kaggle que nous avons utilisé pour pré-processing, l’analyse exploratoire et le feature ingeniering, quelques modélisations de base avant les modélisations pour le choix du modèle efficient.

## Kernel kaggle

L’analyse exploratoire et le feature ingeniering réalisé sur ce Kernel utilisent uniquement la base de données « train.csv »

## RESAMPLING

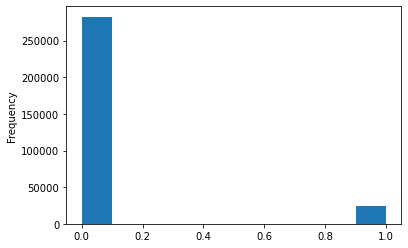
Le principal problème dans notre cas de figure se situe dans le déséquilibre des *targets*. Ici, nous avons à faire à une classification binaire dans laquelle la classe 0 représente les personnes qui ont payé leur crédit, et la classe 1, les personnes ayant rencontré des problèmes pour rembourser leur prêt.

Afin d’éviter de construire un modèle ne prédisant que la classe majoritaire, il convient d’entraîner ce modèle sur un jeu de données équilibré. Pour cela nous avons utilisé une librairie Python nommée *Imblearn [4]*.

Technique utilisée dans la librairie *Imblearn* [4] : **Random Under Sampler**

Il existe plusieurs techniques de sampling de données. Parmi les plus efficaces, on trouve le Tomek links pour l’under-sampling et le SMOTE pour l’over-sampling. Dans notre cas, nous avons combiné les deux techniques le **Random Under Sampler** et le **SMOTE**.

Sur la figure 2 ci-dessous, on constate le déséquilibre initial entre les deux classes. Tandis que sur la figure 3, nous avons rééquilibré parfaitement les 2 classes.

*Figure 2*

*Figure 3*

**Nombre d’individus dans chaque classe avant et après rééquilibrage des données :**

|  |  |  |
| --- | --- | --- |
|  | **Classe 0** | **Classe 1** |
| **Avant Rééquilibrage** | 282686 | 24825 |
| **Après Rééquilibrage** | 17377 | 17377 |

## PREPROCESSING

Pour rappel, l’entraînement du modèle se fait uniquement sur le fichier « train.csv ». Le preprocessing est constitué de deux traitements :

* Une transformation des données comprenant :
  + Une imputation des valeurs manquantes détectées dans la base de données et celles créées lors du traitement des outliers en particulier pour la variable « age du client ».
  + Une application d’un MinMaxScaler et du SimpleImputer sur le dataset.
* Un échantillonnage du dataset. Cette partie a été réalisée avec la méthode « Train\_Test\_Split » de la librairie Scikit-learn. Nous avons retenu un ratio de 70% pour les données d’entraînement et 30% pour les données de tests.

Train

Preprocessing

Data\_train

70%

Data\_test

30%

Echantillonage

*Figure 4*

# CONSTRUCTION MACHINE LEARNING

## Modèle Lgbm

La détermination du modèle efficient a été effectué en utilisant un pipeline de déploiement continu. Nous avons testé 4 modèles (LogisticRegression, RandomForestClassifier, Xgboost et LightGBMClassifier) sur trois optimisateurs (RandomizedSearchCV, GridSearchCV et Optuna).

La comparaison des métriques de performance et le temps d’entrainement du modèle nous ont conduit à retenir le modèle LightGBM. Les tableaux suivant donnent les résultats des différents tests des 4 modèles sur les 3 optimisateurs.

### ACCURACY, PRECISION, RECALL, F1 SCORE

**PRINCIPE :** Les deux métriques qui nous intéressent ici sont la *Précision*, et le *Recall* :

* La Précision : Ce coefficient détermine que, quand le classifieur déclare que la prédiction est un 1, il a raison à X%.
* Le Recall : Ce coefficient détermine le pourcentage de détection des 1 du classifier.
* **Sur RandomizedSearchCV**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ACCURACY | PRECISION | RECALL | F1 SCORE | AUC |
| LogisticRegression | 0.68 | 0.15 | 0.66 | 0.25 | 0.729 |
| RandomForestClassifier | 0.64 | 0.14 | 0.68 | 0.24 | 0.720 |
| Xgboost | 0.68 | 0.16 | 0.68 | 0.26 | 0.747 |
| LightGBMClassifier | 0.71 | 0.16 | 0.63 | 0.26 | 0.732 |

* **Sur GridSearchCV**

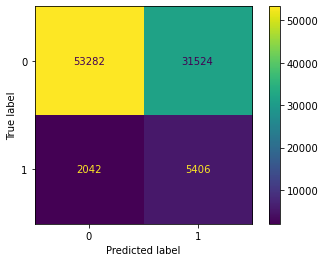
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ACCURACY | PRECISION | RECALL | F1 SCORE | AUC |
| RandomForestClassifier | 0.67 | 0.15 | 0.68 | 0.25 | 0.734 |
| Xgboost | 0.68 | 0.16 | 0.68 | 0.26 | 0.742 |
| LightGBMClassifier | 0.68 | 0.15 | 0.68 | 0.25 | 0.745 |

* **Sur Optuna**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ACCURACY | PRECISION | RECALL | F1 SCORE | AUC |
| LogisticRegression | 0.92 | 0.54 | 0.01 | 0.01 | 0.734 |
| Xgboost | 0.68 | 0.16 | 0.68 | 0.26 | 0.682 |
| LightGBMClassifier | 0.68 | 0.16 | 0.68 | 0.26 | 0.741 |

**ANALYSE :** Les 4 modèles testés présentent des résultats similaires, à l’exception des résultats du modèle LogisticRegression dans l’optimisateur Optuna. Le pouvoir de prédiction des modèles est de 75% mais la précision des modèles est faible (15%) c’est-à-dire que quand le modèle prédit une classe 1, il n’a raison que dans 15% des cas.

### LA MATRICE DE CONFUSION



**INTERPRETATION**

La matrice de confusion est un outil utile pour évaluer les performances d'un modèle de classification. Elle fournit une répartition des prédictions du modèle, ce qui permet d'évaluer sa capacité à classer correctement les clients solvables et défaillants.

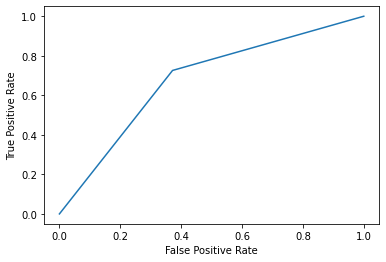
* Vrais positifs (VP) : 53282
  + Cela représente le nombre de cas où le modèle a prédit correctement un résultat positif.
* Faux positifs (FP) : 31524
  + Cela représente le nombre de cas où le modèle a prédit à tort un résultat positif.
* Faux négatifs (FN) : 2042
  + Cela représente le nombre de cas où le modèle a prédit à tort un résultat négatif.
* Vrais négatifs (VN) : 5406
  + Cela représente le nombre de cas où le modèle a prédit correctement un résultat négatif.

En résumé :

* Le modèle a correctement prédit 53282 clients comme étant solvables et qui le sont effectivement (Vrais positifs).
* Le modèle a prédit à tort 31524 clients comme étant solvables alors qu'ils étaient en réalité défaillants c’est-à-dire non solvables (Faux positifs).
* Le modèle a prédit à tort 2042 clients comme étant non solvables alors qu'ils étaient en réalité solvables (Faux négatifs).
* Le modèle a correctement prédit 5406 clients comme étant non solvables et effectivement, ils sont défaillants (Vrais négatifs).

Notre modèle idéal serait de retrouver 100% de TP, car ce sont les individus qui ne remboursent pas leur prêt. C’est aussi la catégorie la plus difficile à prédire car étant minoritaire par rapport à la catégorie *0 (Cf. FEATURE ENGINEERING – Partie 2).*

### COURBE ROC ET SCORE AUC



**PRINCIPE :** La courbe ROC (Receiver Operating Characteristic) est un outil communément utilisé avec les classifieurs binaires. Elle croise le taux de TP avec le taux de FP.

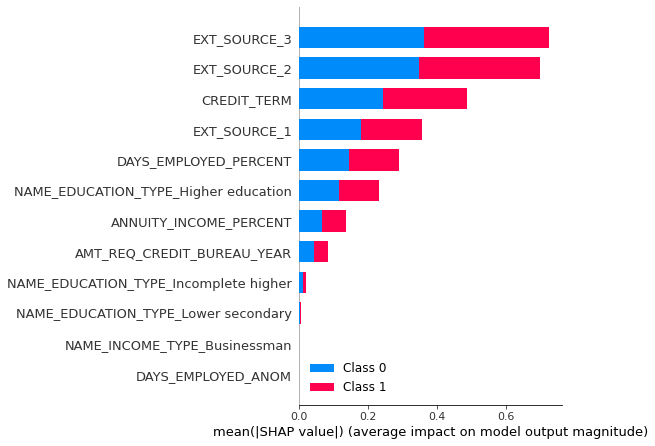
Une façon de comparer des classifieurs consiste à mesurer l’aire sous la courbe (Area Under the Curve ou AUC). Un classifieur parfait aurait un score AUC égal à 1, tandis qu’un classifieur purement aléatoire aurait un score AUC de 0.5.

**ANALYSE :** On constate que la performance du modèle est relativement bonne. Pour rappel, nous cherchons une courbe ROC se rapprochant le plus du coin supérieur gauche du graphique.

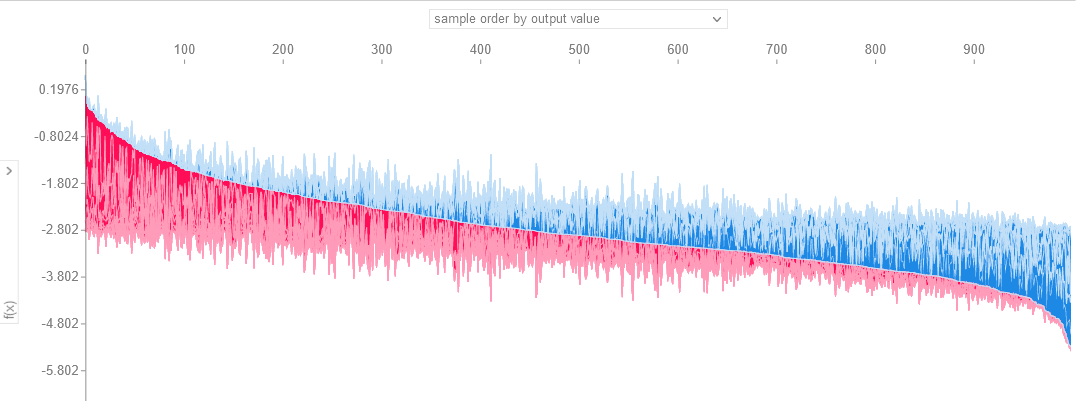
En effet l’aire en dessous de la courbe (Area Under the Curve ou AUC) vaut 0.75 sur un total de 1. Ce qui est acceptable.

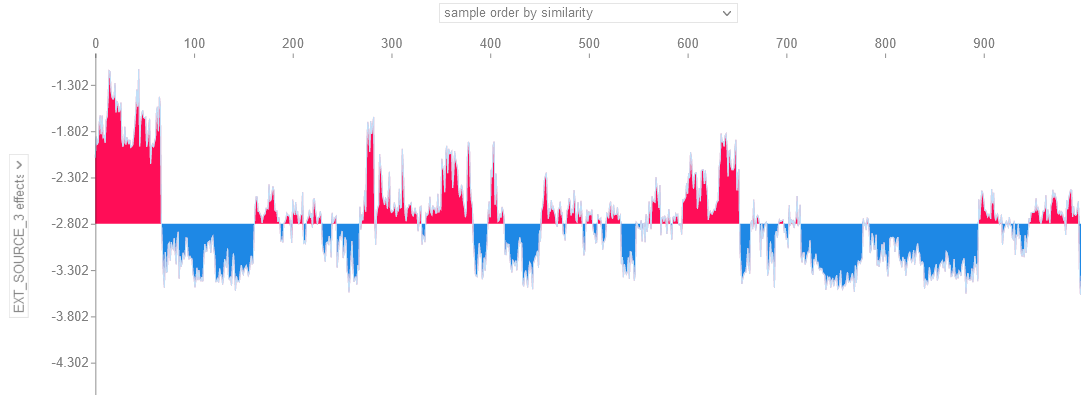
### FEATURES IMPORTANCE

**PRINCIPE :** L’analyse de l’importance des variables nous permet de visualiser sur quelles variables s’appuie le modèle pour effectuer ses prédictions.

Sur la figures ci-après, l’importance des variables se lit en pourcentage (En multipliant les valeurs des abscisses par 100). 

Le diagramme de dispersion de la densité des SHAP valeurs montre l'impact de chaque feature sur les résultats du modèle pour les individus de l'ensemble de données de validation. Ici les features sont triées en fonction de la somme des SHAP valeurs de tous les échantillons. Il montre permet de visualiser l'importance de chaque feature par classe, c'est à dire l'importance de la feature selon que le client est solvable (classe "0") ou défaillant (classe "1"). Comme identifiées précédemment, les variables des sources extermes (Ext\_Source\_3&2) et la durée du crédit (Credit\_term) ont plus d'impact total sur le modèle que les annuités et le niveau d'éducation par exemple. Les variables des sources externes sont très déterminantes pour la catégorie des clients défaillants (classe 1) dans la mesure où ces variables donnent des informations sur des antécédents bancaires de ces clients.



 Les deux images ci-dessus sont extraits d’un graphique interactif qui permet de voir les contributions locales de chaque feature pour chaque client, par groupe de clients similaires ou encore, la contribution d'une seule feature, ou encore par type d'échantillons considérés.

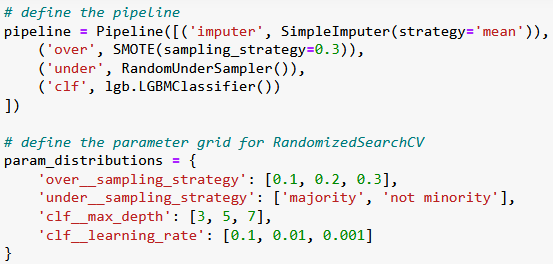
## OPTIMISATION du modele Lgbm

L’optimisation du modèle a été faite en utilisant un pipeline de déploiement continu. Nous y avons intégré la librairie Mlflow qui nous permets de sauvegarder les différents essais des optimisations. Nous avons aussi optimisé les modèles sur trois optimasteurs à savoir RandomizedSearchCV, GridSearchCV et Optuna. Les résultats sont similaires.

Il permet de gérer simultanément questions suivantes :

* L’imputation des valeurs manquantes
* La gestion des classes déséquilibrées par le rééquilibrage (Undersampling et Oversampling)
* l’optimisation des hyperparamètres du modèle

### PIPELINE ET HYPERPARAMETRES

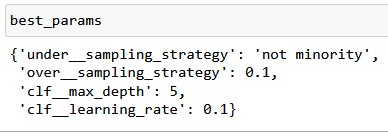


Il existe un grand nombre d’hyperparamètres pour ce modèle. Afin de réduire le temps d’optimisation, nous avons optimisé deux hyperparamètres importants à savoir le max\_depth et le learning\_rate. Les autres hyperparamètres sont conservés à leurs valeurs par défaut.

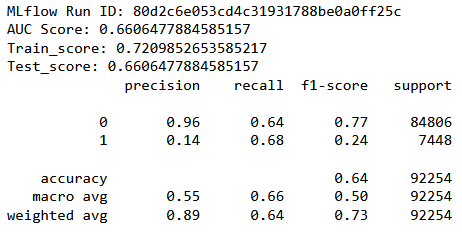
1. Learning\_rate.
2. Max\_depth : Profondeur maximale d'un arbre. Utilisé pour contrôler le sur-ajustement car une profondeur plus élevée permettra au modèle d'apprendre des relations très spécifiques à un échantillon particulier.

Les résultats reportés ici sont ceux obtenu avec l’optimiseur RandomizedSearchCV.

Meilleurs paramètres trouvés :



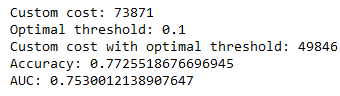
Résultats du modèle optimisé

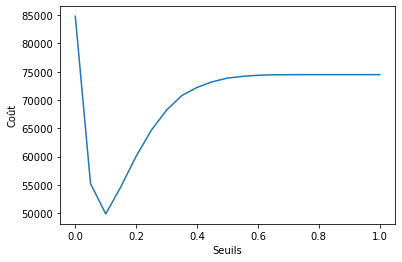


# Fonction de cout metier

L’objectif est de créer une fonction permettant de minimiser le coût d'erreur de prédiction des faux négatifs(FN) et des faux positifs (FP) sachant que le coût d’un FN est dix fois supérieur au coût d’un FP.

Résultats :





Interprétation

* **Coût métier : 73871** - Il s'agit du coût total calculé à l'aide de la fonction de coût métier sur l'ensemble de test. Les coûts spécifiques pour les faux négatifs et les faux positifs ont été fixés à 10 et 1 respectivement dans ce calcul. Plus le coût métier est élevé, plus les classifications erronées (faux négatifs et faux positifs) sont coûteuses pour la banque.
* **Seuil optimal : 0.1** - Le seuil optimal est la valeur qui minimise le coût métier. Dans ce cas, le seuil optimal trouvé est de 0.1. Cela signifie que toute probabilité prédite égale ou supérieure à 0.1 est considérée comme une prédiction positive, tandis que les valeurs inférieures à 0.1 sont considérées comme des prédictions négatives.
* **Coût métier avec le seuil optimal** : **49846** - Le coût métier est recalculé en utilisant le seuil optimal. Avec l'application du seuil optimal, le coût est réduit à 49846, ce qui indique une amélioration potentielle des performances par rapport à l'utilisation d'un seuil de 0.5.
* Précision : 0.7725518676696945 - La précision est une mesure couramment utilisée qui indique la proportion d'échantillons correctement prédits (à la fois les vrais positifs et les vrais négatifs) par rapport au nombre total d'échantillons. Dans ce cas, la précision est d'environ 0.7725, ce qui signifie qu’environ 77,25 % des échantillons ont été classés correctement.
* **AUC : 0.7530012138907647** - L'AUC (Aire sous la courbe ROC) est une mesure utilisée pour évaluer la performance globale d'un classifieur binaire en considérant le compromis entre le taux de vrais positifs (sensibilité) et le taux de faux positifs (1 - spécificité) pour différents seuils de classification. Une valeur d'AUC plus élevée indique une meilleure performance du classifieur. Dans ce cas, l'AUC est d'environ 0.753, ce qui suggère un pouvoir discriminant raisonnable du classifieur.

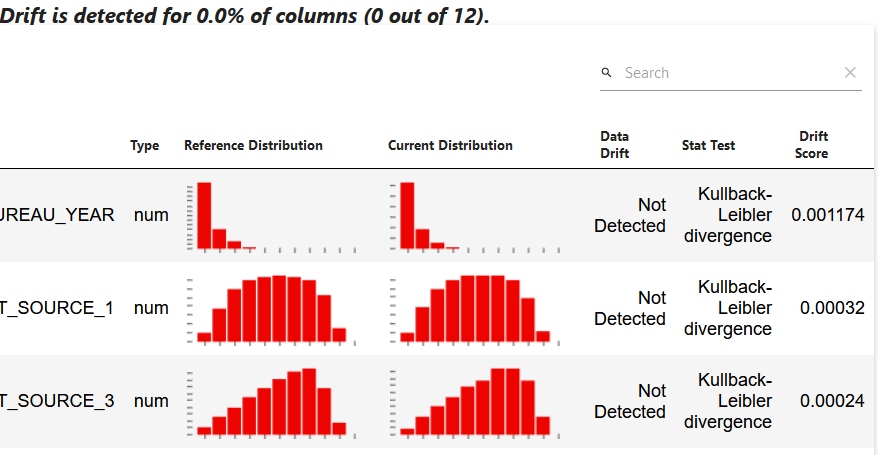
Ces métriques fournissent des informations sur la performance du classifieur binaire, notamment en termes de coût, de précision et de compromis entre le taux de vrais positifs et le taux de faux positifs. Le seuil optimal et le coût qui y est associé peuvent aider à prendre des décisions qui tiennent compte des coûts spécifiques et des exigences du domaine bancaire.

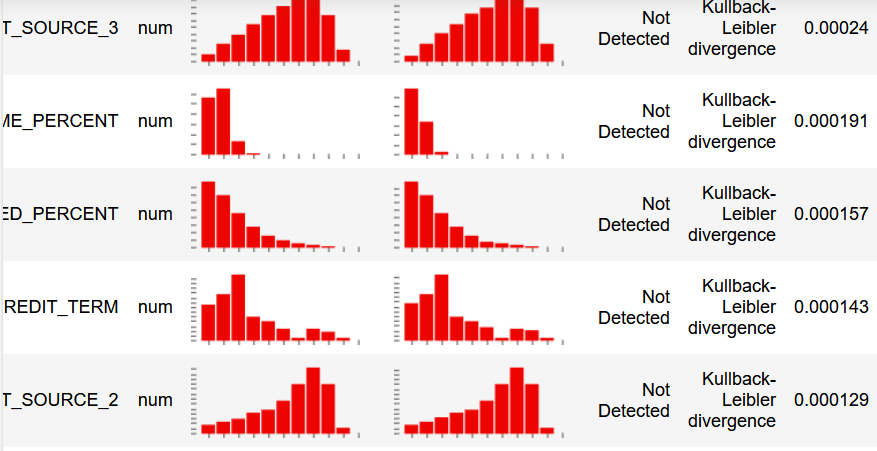
# Data Drift

Le Data Drift fait référence au phénomène qui se produit lorsqu'il y a des changements dans la distribution ou la qualité des données utilisées dans un modèle d'apprentissage automatique. En d'autres termes, le Data Drift se produit lorsque les caractéristiques des données d'entraînement ou de test changent au fil du temps par rapport à celles sur lesquelles le modèle a été initialement formé.

Le Data Drift peut avoir un impact significatif sur les performances d'un modèle d'apprentissage automatique, car celui-ci est optimisé pour fonctionner avec des données spécifiques qui peuvent ne plus être représentatives des données réelles une fois qu'elles ont changé. Les causes courantes du Data Drift comprennent les fluctuations des comportements des utilisateurs, les évolutions des environnements, les modifications des politiques ou des règlements, les erreurs dans les systèmes de collecte de données, etc.

En somme, le Data Drift est un concept essentiel dans le domaine de l'apprentissage automatique, car il permet de maintenir la pertinence et la fiabilité des modèles dans des environnements en constante évolution en identifiant les changements dans les données et en adaptant les modèles en conséquence.





Interprétation

Stabilité des données : Les résultats montrent l'absence de Data Drift. Cela suggère que les données utilisées dans le modèle restent stables et cohérentes au fil du temps. Cela est plutôt souhaitable, car cela signifie que les conditions ou les caractéristiques sous-jacentes des données n'ont pas changé de manière significative, ce qui permet de conserver la performance du modèle sur le long terme.

Cependant, l'absence de Data Drift dans un jeu de données ne peut jamais être considérée comme définitive. Les changements dans les données peuvent survenir à tout moment, et il est nécessaire de continuer à surveiller activement les performances du modèle et à mettre en place des mécanismes pour détecter les signes éventuels de Data Drift à l'avenir.

# AXES D’AMELIORATIONS A aPPORTER

On retient que le modèle retenu a un pouvoir de prédiction acceptable de 75% mais cette précision est inférieure à celle du premier de Kaggle chalenge qui est de 82%. Cela signifie que le modèle prédit 75% de la classes 1. Cependant, lorsqu’il en prédit une, il a raison à 16% ; ce qui est plutôt faible.

Nous pouvons l’expliquer par le features engineering qui est à améliorer avec une connaissance approfondie du domaine bancaire. Le nombre de features retenues est faible. En effet, en introduisant les features une à une dans les modèles, nous en avons retenues 12 car l’ajout d’autres features n’amélioraient pas les performances des modèles. Le nombre élevé des variables dans les bases de données (accrues avec le feature ingeniering et le one-hot-enconding) n’a pas permis de tester toutes les features. Nous avons peut-être omis de tester des features qui amélioreraient les performances du modèle. Il convient de trouver une technique efficace et peu coûteuse en temps qui permet de sélectionner les features pertinentes pour le modèle.