

Note Méthodologique

Projet 7 - OpenClassrooms

Implémentez un modèle de scoring

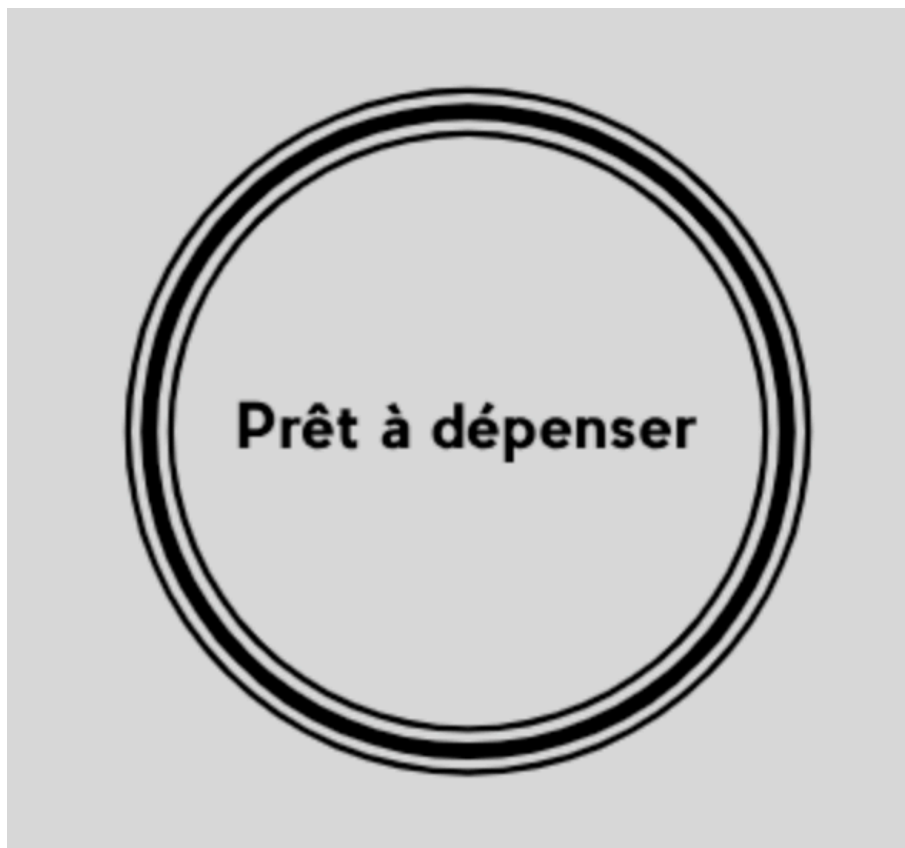


Table des matières

1. Présentation du jeu de données.....	2
2. Pré-traitement des données.....	3
Processus de pré-traitement	3
Nettoyage des données	3
Feature engineering.....	3
Assemblage des Datasets	4
Features selection.....	4
Imputation des données.....	4
3. Modélisation	5
Rééquilibrage de la variable cible	5
Métriques d'évaluation	5
Choix des modèles et optimisation des hyperparamètres	6
Modèle final	7
4. Interprétabilité du modèle.....	8
5. Limites et améliorations	8
6. Analyse du DataDrift.....	9
7. Liens du projet.....	10

OBJECTIF

Prêt à dépenser est une société financière spécialisée dans les crédits à la consommation pour des individus avec peu ou sans historique de prêt.

Notre mission consiste à **développer un outil de 'scoring crédit'** basé sur un algorithme de classification. Cet outil fournira la probabilité de défaut de paiement d'un client, facilitant ainsi la prise de décision quant à l'octroi ou au refus de crédit.

En parallèle, la création d'un **dashboard interactif** permettra aux chargés de relation client d'expliquer de manière transparente et simple ces décisions aux clients concernés.

1. Présentation du jeu de données

L'entreprise met à disposition 7 fichiers CSV.

Ils contiennent des informations bancaires et personnelles de 307511 clients recueillies auprès de Home Crédit Group et d'autres institutions financières.

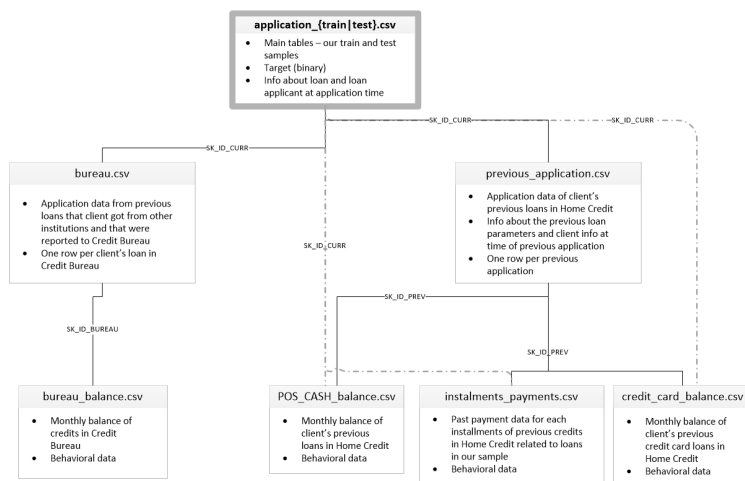
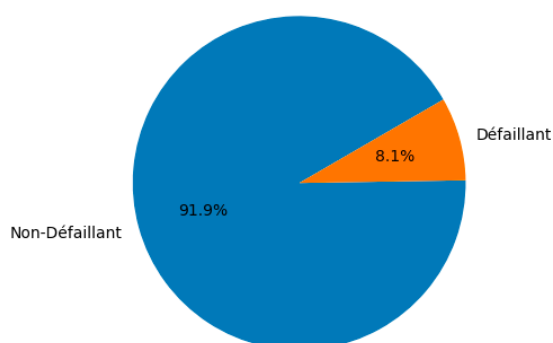


Figure 1 : Arborescence du jeu de données

Le fichier principal, désigné sous le nom d'application_train, contient une variable cible.

Cette dernière, indique le statut d'acceptation ou de refus du crédit pour le client concerné, elle constitue la cible de notre outil de 'scoring crédit', qui devra mettre en oeuvre un modèle de classification binaire :

- **Valeur 0 - client non défaillant :**
indique que le client a remboursé entièrement son crédit.
- **Valeur 1 - client défaillant :**
Indique que le client n'a pas remboursé le crédit partiellement ou totalement.



La variable cible est **fortement déséquilibré** (8.1/91.9).

- Certains algorithmes étant sensible au déséquilibres de classes, nous devons prendre en compte ce paramètre pendant leur entraînement.

Figure 2 : Répartition de la variable cible

2. Pré-traitement des données

Processus de pré-traitement

Pour cette phase du projet, nous avons utilisé un Kernel Kaggle ([source](#)) afin de gagner du temps, étant donné que ce n'était pas l'objectif principal de notre démarche.

Le but final de cette étape était d'assembler un Dataset global, structuré de manière à ce que chaque ligne représente un client, tout en incluant de manière exhaustive toutes les informations pertinentes issues de chaque Dataset.



Figure 3 : Processus de pré-traitement

Nettoyage des données

Durant cette étape,

- Nous avons harmonisé les types d'objets pour garantir leur cohérence. (ex : transformation des données catégorielles de type "Y/N" en valeurs binaires « 0/1 »).
- Nous avons également corrigé ou éliminé (lorsqu'aucune correction appropriée n'était envisageable) les valeurs aberrantes (ex: sexe Masculin / Féminin / XNA).

Feature engineering

Cette étape consiste en la création de nouvelles variables permettant une potentielle augmentation des performances du modèle.

- **OneHotEncoder** : utilisation de cette méthode pour l'ensemble des variables catégorielles, permettant de n'obtenir que des variables numériques afin d'assurer la compatibilité avec les modèles de machine learning
- **Création automatique** de variables (moyenne, minimum, maximum, nombre d'occurrence, écart-type...) lors de l'agrégation des Datasets pour n'obtenir qu'une ligne par client.
- **Création manuelle** à partir de la « compréhension métier » (variables s'appliquant principalement au domaine bancaire).

Ex :

- ratio du revenu par membre (revenu du demandeur / membres de la famille)
- variables spécifiques en fonction du contexte :
 - pour les crédits actifs/clos
 - pour les demandes approuvées / refusées.
-

Assemblage des Datasets

Une fois les étapes précédentes terminées pour chaque fichiers, nous pouvons alors créer un fichier unique contenant l'ensemble des informations et représentant un client par ligne.

Cette opération a été réalisée en utilisant la variable 'SK_ID_CURR' représentant l'identifiant unique de chaque client.

- Nous obtenons ainsi un fichier global comprenant 307511 clients et 786 variables explicatives (dont une étant notre cible).

Features selection

Etant donné les **dimensions importantes** de notre espace (dimensions de notre Dataset).

Il paraît **nécessaire** de réaliser une **sélection de variables** et de ne retenir que les plus pertinentes pour répondre efficacement à notre problématique.

Cette démarche permettra une optimisation des performances et une réduction de la complexité de notre modèle.

Nous avons employé la méthode de **feature importance** de l'algorithme **LightGBM**. Cette technique attribue un score d'importance à chaque variable en fonction de sa contribution à la prédiction du modèle.

Le choix de cet algorithme a été motivé par **sa robustesse face aux valeurs manquantes** encore présentes à ce stade dans notre ensemble de données.

Initialement, nous avons réalisé une **première sélection** en ne retenant que les **100 variables** les plus importantes selon LightGBM.

Ensuite, afin d'assurer la cohérence et de **réduire la redondance**, nous avons comparé les corrélations entre ces variables. Toutes les paires de variables dont **la corrélation était supérieure à 0,6** ont été examinées, et parmi celles-ci, nous n'avons conservé que la variable présentant la feature importance la plus élevée.

Cette méthode, nous a permis de réduire considérablement les dimensions de notre espace, tout en conservant une pertinence des variables retenues pour l'utilisation de notre outils de 'scoring'.

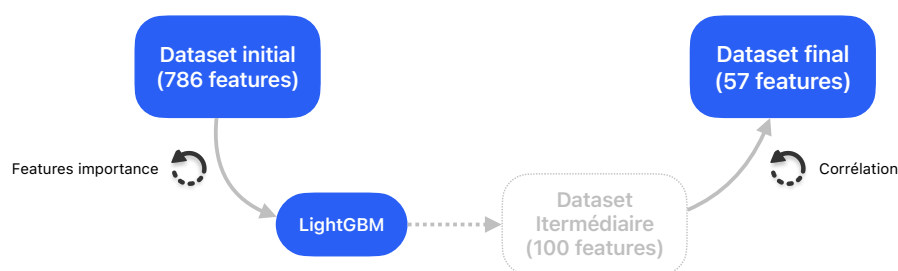


Figure 4 : Features selection

Imputation des données

Les valeurs manquantes des features sélectionnées ont été complétées en utilisant trois méthodes distinctes (moyenne, mode et médiane). Cette approche vise à évaluer la pertinence de chaque méthode lors de la modélisation ultérieure.

3. Modélisation

Rééquilibrage de la variable cible

Comme évoqué précédemment, le déséquilibre de classe au sein de notre jeu de données peut avoir des conséquences négatives sur la performance du modèle, le prédisposant à être sur-entraîné sur la classe majoritaire.

Afin de remédier à cette situation, nous avons intégré la technique **SMOTE (Synthetic Minority Over-sampling Technique)** dans notre processus d'optimisation des modèles. Cette approche vise à sur-échantillonner la classe minoritaire, contribuant ainsi à maintenir un équilibre informatif dans l'ensemble de nos données.

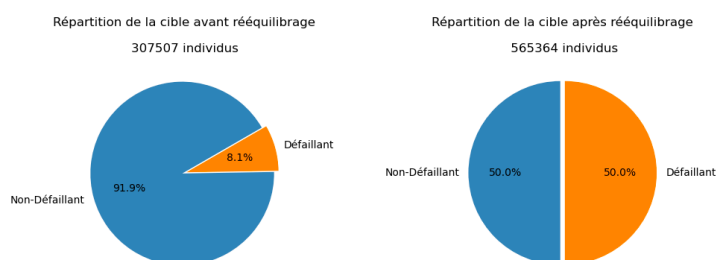


Figure 4 : Répartition des classes avant/après rééquilibrage

Métriques d'évaluation

Le choix des métriques, dépend de la problématique et permet à la fois d'évaluer la performance du modèle et de garantir la qualité de la classification.

Pour notre problématique, l'enjeu principal réside dans la **minimisation des pertes financières** pour la société de financement de crédit :

- **les faux négatifs**, c'est-à-dire prédire à tort qu'un client défaillant est non défaillant, entraînent des coûts considérables (perte de la somme prêtée).
- **les faux positifs**, prédire à tort qu'un client non défaillant est défaillant, occasionnent des coûts significatifs mais moins impactants (perte limitée aux intérêts du remboursement).

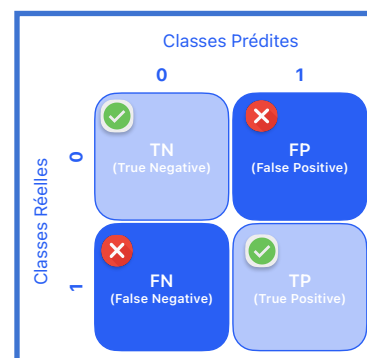


Figure 5 : Matrice de confusion

Pour répondre spécifiquement à cet objectif, nous avons introduit un « **score_métier** ».

Celui-ci attribue un **coût dix fois plus élevé aux faux négatifs**, représentant ainsi les cas où notre modèle prédit à tort qu'un client ne présente pas de risque de défaillance alors qu'il en présente.

Cette approche cible précisément la réduction des erreurs les plus coûteuses pour la société, et sera donc la **métrique de référence pour l'optimisation du modèle**.

$$score = \frac{10 * FN + FP}{TP + TN + FP + 10 * FN}$$

En parallèle nous complétons notre évaluation avec d'autres métriques standard pour une vue d'ensemble de la performance du modèle.

Choix des modèles et optimisation des hyperparamètres

Pour notre problématique, nous avons sélectionné 4 modèles :

	LogisticRegression	RandomForestClassifier	XGBoostClassifier	DecisionTreeClassifier
Avantages	<ul style="list-style-type: none"> ▸ Simplicité et interprétabilité. ▸ Faible risque de surajustement. 	<ul style="list-style-type: none"> ▸ Bonne performance prédictive. ▸ Robuste face au surajustement. 	<ul style="list-style-type: none"> ▸ Haute performance grâce au boosting. 	<ul style="list-style-type: none"> ▸ Intuitif et facile à comprendre. ▸ Gère les données de manière non linéaire.
Inconvénients	<ul style="list-style-type: none"> ▸ Ne capture pas les relations complexes entre les variables. 	<ul style="list-style-type: none"> ▸ Peut être gourmand en ressources computationnelles. 	<ul style="list-style-type: none"> ▸ Plus de complexité par rapport à d'autres modèles. 	<ul style="list-style-type: none"> ▸ Sensible aux variations dans les données d'entrée.

Nous avons entraîné chaque modèle de manière spécifique sur nos données afin d'**optimiser leurs performances**, en utilisant une **validation croisée** pour garantir la robustesse des résultats.

Cette optimisation a été réalisée avec la librairie **Hyperopt**, exploitant un algorithme de recherche bayésienne pour explorer efficacement l'espace des hyperparamètres. En définissant un espace de recherche pour chaque modèle, nous avons laissé Hyperopt identifier les combinaisons les plus performantes de paramètres.

Afin d'adapter nos modèles aux exigences spécifiques de **notre problématique métier**, nous avons intégré la métrique **"score_métier"** dans le processus d'optimisation. Cette métrique prend en compte les coûts associés aux faux positifs et aux faux négatifs, comme décrit précédemment. De plus, le **seuil de décision optimal** pour la classification binaire a été ajusté pendant le processus d'optimisation.

En résumé, notre approche d'optimisation des hyperparamètres, combinée à la personnalisation du "score_métier" et au réglage du seuil de décision, renforce la capacité de nos modèles à répondre de manière précise et efficace aux exigences particulières de notre problématique.

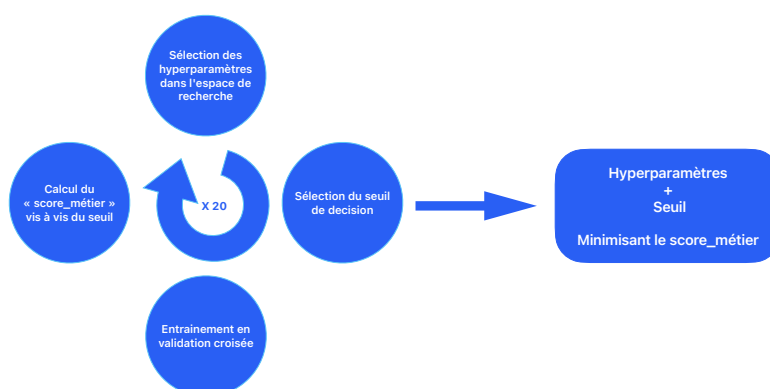
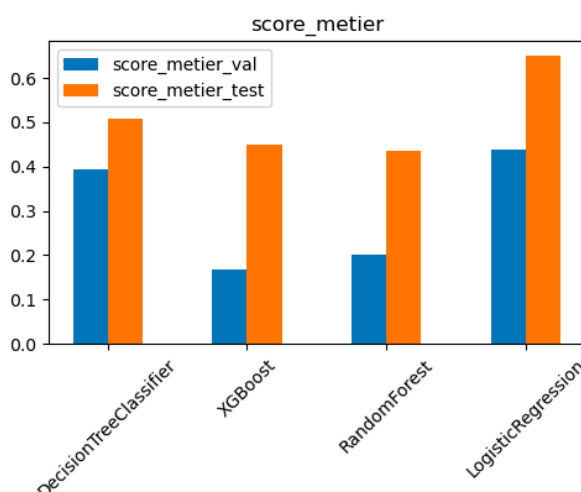


Figure 6 : Cycle d'optimisation des modèles

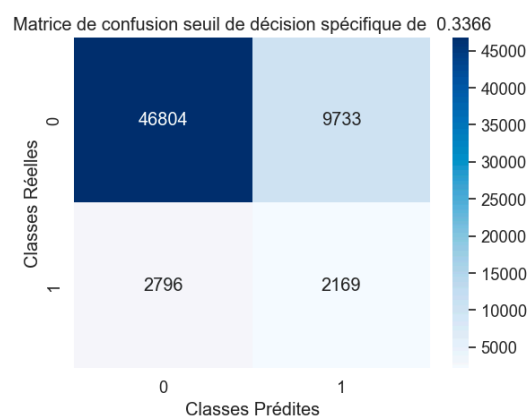
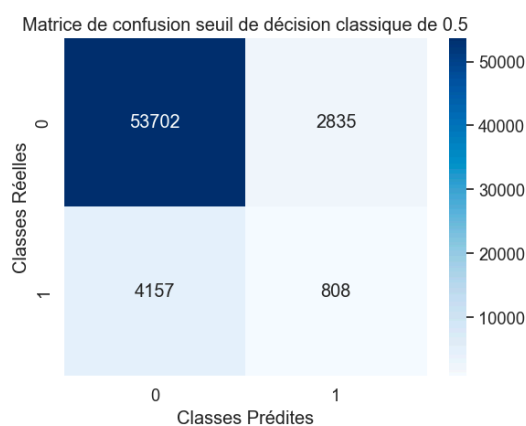
Modèle final

Une fois l'optimisation des hyperparamètres effectuée et nos modèles entraînés en conséquence, nous pouvons les comparer directement en fonction de leurs métriques :



Notre décision se base sur le score_métier, spécialement conçu pour répondre à la problématique de notre projet. Nous sélectionnons ainsi le modèle **RandomForestClassifier** qui minimise au maximum cette métrique (tout en ayant les autres scores standards les plus pertinents), avec un **seuil de décision spécifique** trouvé par HyperOpt fixé à **0,3366**.

En examinant les prédictions du modèle RandomForest sur l'ensemble d'entraînement, qui n'a jamais été utilisé pour l'optimisation, nous obtenons les matrices de confusion suivantes :



En analysant ces matrices, il est apparent que l'ajustement du seuil spécifique permet de **réduire significativement** le nombre de **faux négatifs (FN)**. Les faux négatifs représentent les situations où le modèle prédit à tort qu'un client ne présente pas de risque de défaillance alors qu'il en présente.

En minimisant ces faux négatifs, notre modèle est mieux calibré pour identifier de manière plus précise les clients réellement à risque, ce qui revêt une importance cruciale dans le contexte de la société de crédit.

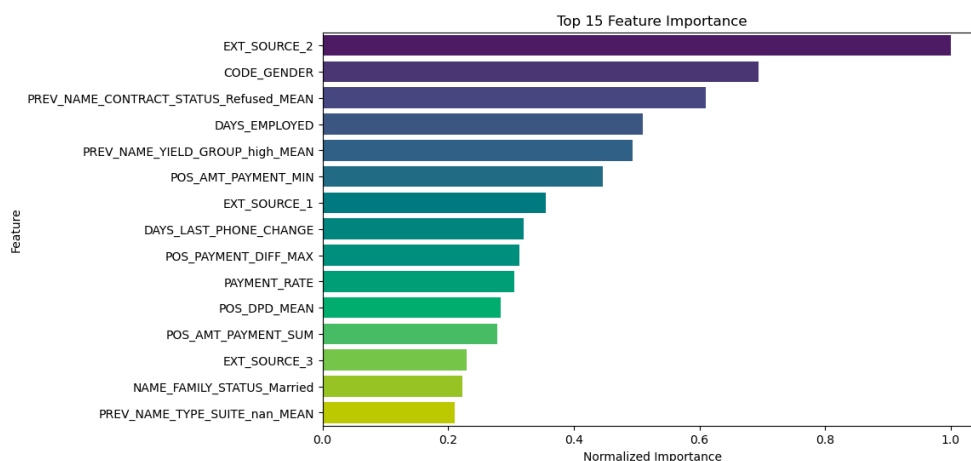
Il est important de noter que bien que cette adaptation du seuil puisse entraîner une **augmentation des faux positifs (FP)**, cette augmentation est compensée par le fait que le coût des FP est considérablement inférieur au coût des FN.

En appliquant **un rapport de coût de 1 pour 10 entre les FP et les FN**, ce seuil spécifique offre le meilleur compromis en termes de performance, en accord avec notre objectif principal tel que reflété par la métrique score_métier

4. Interprétabilité du modèle

Le modèle RandomForestClassifier propose une méthode qui permet de calculer l'importance de chaque variable pour la prise de décision.

Une fois ces importances normalisées, nous sommes en mesure de comparer la contribution relative de chaque variable :



Les variables les plus influentes dans notre modèle incluent :

- Des données provenant de **source externe** : EXT_SOURCE_2, EXT_SOURCE_1 et EXT_SOURCE_3.
- Des **informations personnelles** : CODE_GENDER, DAYS_EMPLOYED...
- Des informations sur les **crédits antérieurs** : PREV_NAME_CONTRACT_STATUS_Refused_MEAN, PREV_NAME_YIELD_GROUP_high_MEAN....

5. Limites et améliorations

Pour aborder au mieux la problématique de Prêt à Dépenser, nous avons adopté une approche diversifiée, impliquant une **agrégation spécifique des données** par client, un **rééquilibrage des classes**, la création de **nouvelles variables**, la **sélection** de variables pertinentes, et le développement d'une **métrique adaptée à la problématique métier**. Nous avons également ajusté le **seuil spécifique de décision** pour minimiser les coûts associés aux deux types d'erreurs de prédiction.

Cependant, une collaboration directe avec la société aurait pu nous offrir une **compréhension plus approfondie** des répercussions réelles des faux positifs et des faux négatifs, ainsi que de leurs **coûts réels**, améliorant ainsi l'optimisation des hyperparamètres du modèle et du seuil de décision spécifique.

L'**assistance d'experts du domaine bancaire** aurait pu être bénéfique pour la création de variables spécifiques pertinentes et potentiellement pour une métrique plus efficace.

L'**origine** et la **signification** des scores provenant de **sources externes** restent un point d'interrogation, soulignant le besoin d'une meilleure explicabilité lors de la prise de décision de notre modèle.

Enfin, l'absence de retour de la société sur le Dashboard développé nous prive de précieuses informations sur sa **compréhension** et sa **facilité d'utilisation**. Cela aurait pu nous éclairer sur la convivialité du tableau de bord et sur la nécessité d'éventuelles améliorations fonctionnelles.

6. Analyse du DataDrift

Dans notre contexte, la surveillance du Data Drift (ou dérive des données) revêt une importance cruciale pour garantir la pertinence continue de notre modèle prédictif. En effet, notre modèle a été entraîné sur des données historiques, et une évolution significative des données au fil du temps peut conduire notre modèle vers des espaces moins familiers, entraînant ainsi des incertitudes de prédiction plus importantes.

C'est pourquoi il est important de surveiller cette évolution entre les données historiques d'entraînement et les données actuelles afin de prévenir au mieux toute dérive de prédiction. Un score de Data Drift trop important peut nécessiter une réévaluation du modèle et influencer les décisions de gestion. La détection proactive de ces changements contribue à maintenir la performance et la fiabilité du modèle dans un environnement dynamique.

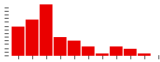
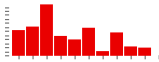




Pour cette étude, nous avons utilisé la bibliothèque Evidently en appliquant les seuils de décision standards : 0.5 pour le seuil global et 0.1 pour chaque variable.

L'analyse a été effectuée sur les fichiers train (représentant notre ensemble d'entraînement historique) et test (représentant l'ensemble des nouveaux clients).

Les résultats indiquent un DataDrift global de 15,8% sur l'ensemble des données, affectant 9 variables parmi les 57 initiales. :

57	9	0.158
Columns	Drifted Columns	Share of Drifted Columns

En se penchant spécifiquement sur les 9 variables affectées par le drift, nous constatons que parmi celles-ci, 3 font partie du top 10 des variables les plus influentes pour le modèle :

> PAYMENT_RATE	num			Detected	Wasserstein distance (normed)	0.57475
> EXT_SOURCE_1	num			Detected	Wasserstein distance (normed)	0.17996
> DAYS_LAST_PHONE_CHANGE	num			Detected	Wasserstein distance (normed)	0.138978

Comme démontré visuellement à travers les graphiques de répartition et confirmé par les scores de drift spécifiques à ces variables, PAYMENT_RATE présente une dérive notable, soulignant ainsi la nécessité d'accorder une attention particulière à cette variable.

Enfin, pour compléter au mieux cette étude, solliciter l'avis d'experts serait une démarche intéressante.

Leur contribution permettrait une analyse plus approfondie et un ajustement éventuel des seuils de décision, rendant ainsi le suivi du DataDrift plus pertinent.

7. Liens du projet

Dépôt GitHub global du projet : [P7_OCR](#)

Dépôt GitHub de la partie backend de l'API : [backend](#)

Dépôt Github de la partie frontend de l'API : [frontend](#)

Lien de l'API du projet : [API](#)

NB : En raison de limitations techniques de la plateforme Heroku (utilisée pour le déploiement), la modèle utilisé dans l'API sur le web est un DecisionTreeClassifier (ce dernier a montré de moins bon résultat mais est beaucoup plus léger en terme de stockage et permet donc de satisfaire les limitations de la plateforme).