

Implémentez un modèle de scoring :

Note méthodologique

Projet n° 7 - Data Scientist
OpenClassrooms - Felipe PEREIRA DE LIMA
<https://github.com/feliplim/credit-scoring>

Contexte et objectifs

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.

- Implémentation d'un modèle de scoring :

L'entreprise souhaite mettre en œuvre un outil de scoring credit qui calcule la probabilité qu'un client rembourse ou pas son crédit. En se basant sur ces probabilités, la demande en crédit est classée en accordée ou refusée. L'entreprise souhaite développer un algorithme de classification en s'appuyant sur des données issues de plusieurs sources (données personnelles, données comportementales, données financières, données issues d'autres institutions financières, données sur d'autres demandes de crédit, etc.).

Les données originales sont disponibles sur Kaggle à cette [adresse](#).

- Création d'un dashboard interactif avec Streamlit :

De plus, les chargés de relation client ont fait remonter le fait que les clients sont de plus en plus demandeurs de transparence vis-à-vis des décisions d'octroi de crédit. Cette demande de transparence des clients est en accord avec les valeurs que l'entreprise veut adopter. *Prêt à dépenser* décide donc de développer un dashboard interactif pour que les chargés de relation client puissent à la fois expliquer les décisions d'octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

Le dashboard réalisé avec Streamlit est accessible en [cliquant ici](#).

Tous les programmes sont accessibles sur [Github](#).

Sommaire

- 1- Les étapes préalables à la modélisation
- 2- La méthodologie d'entraînement du modèle
- 3- La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation
- 4- L'interprétation globale et locale du modèle
- 5- Les limites et les améliorations possibles
- 6- L'analyse du data drift

Note méthodologique

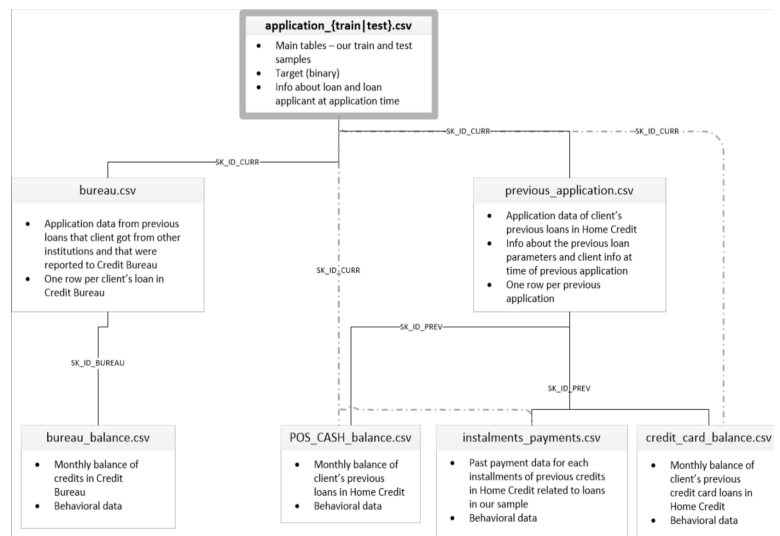
Cette note méthodologique présente les techniques employées pour la construction du modèle de scoring, ainsi que les outils mis en place pour son interprétation.

Pour répondre à cet objectif, les points suivants seront développés succinctement :

- *La méthodologie d'entraînement du modèle*
- *La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation*
- *L'interprétation globale et locale du modèle*
- *Les limites et les améliorations possibles*

1- Les étapes préalables à la modélisation

Les données sont disponibles dans 7 fichiers liés entre eux selon le schéma à gauche. La table “application” regroupe les informations personnelles des clients actuels, ainsi que les données relatives au crédit qu’ils demandent. Cette table est séparée en 2 jeux de données : application “train” regroupant 307.511 clients, dont la décision d’octroi de crédit (variable “Target”) est connue, et application “test” avec 48.774 clients, dont on ne connaît pas la décision.



Les autres fichiers contiennent les données historiques de prêt de ces mêmes clients : Les tables “bureau” et “balance_bureau” contiennent les informations des crédits passés dans d’autres institutions financières. La table “previous_application” reporte les données des crédits passés auprès de *Prêt à dépenser*.

Quoi que l’on puisse penser qu’il suffit d’un grand nombre de données pour obtenir un algorithme performant, les données mises à disposition sont souvent non adaptées. Il faut donc réaliser un traitement préalable pour pouvoir les utiliser : l’étape de *preprocessing*.

Data cleaning

La première étape consiste en un nettoyage des données incorrectes, incomplètes ou manquantes. Cette étape a été réalisée sur chaque table indépendamment des autres. Les individus contenant des données incorrectes ont été supprimés. Les anomalies ont été remplacées par NaN. Les données manquantes ont été remplies avec la médiane.

Data reduction

Lors de la construction d’un modèle prédictif, plusieurs variables dans l’ensemble de données peuvent être utilisées pour former le modèle. Cependant, la présence d’une variable ne justifie pas forcément sa pertinence pour le modèle, donc son utilité. De ce fait, les variables ayant plus de 95% de corrélation ont été ignorées. Le nombre de variables est passé de 681 à 574.

Data transformation

Cette étape de prétraitement regroupe les changements effectués sur la structure même de la donnée. Les méthodes employées ont été :

- L’encodage des variables catégorielles, soit par one-hot encoding pour les variables avec plusieurs catégories, soit par binary encoding pour les catégories ayant deux catégories (telle que le genre).
- La standardisation des données qui ramènent les données numériques à une échelle plus petite, ce qui permet de centrer la moyenne et de réduire la variance.

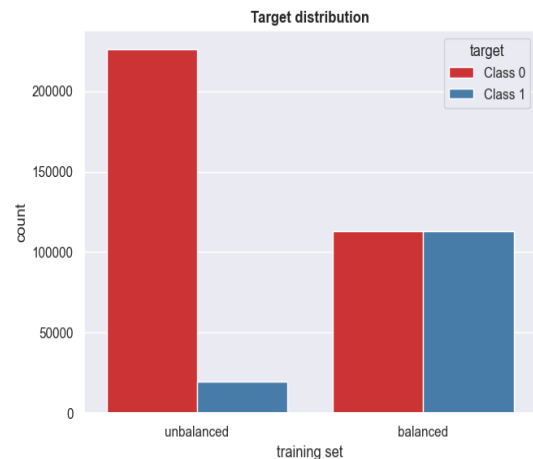
2- La méthodologie d'entraînement du modèle

Les clients en difficulté de paiement sont largement sous-représentés (8,77%) dans les données d'entraînement. Or, les méthodes de *machine learning* classiques ne sont pas toujours adaptées pour la **classification des données déséquilibrées**. Le résultat est souvent mauvais et peut induire en erreur avec des scores trop optimistes. La principale raison de ce dysfonctionnement est que les points de la classe minoritaires sont traités comme des *outliers* qui ne contiennent pas d'information.

Afin de remédier ce déséquilibre des classes, deux méthodes ont été appliquées :

- Oversampling, ou suréchantillonnage, avec la méthode SMOTE (Synthetic Minority Oversampling Technique)
- Undersampling, ou sous-échantillonnage, avec RandomUnderSampler

Le but d'appliquer les deux méthodes était d'équilibrer les deux classes en diminuant la classe majoritaire et en augmentant la classe minoritaire pour les ramener aux mêmes niveaux.



Avant d'appliquer ces deux méthodes, on doit d'abord choisir le type de modèle de prédiction le plus adapté à nos données. Pour ce faire, le jeu de données a été séparé en 2 : un sous-échantillon d'entraînement regroupant 80% des données et 20% pour le test, chaque sous-échantillon contenant le même mélange d'exemples par classe (environ 92% de classe 0 et 8% de classe 1).

Sur le sous-échantillon d'entraînement, nous avons évalué les modèles candidats à l'aide d'une validation croisée stratifiée k-fold avec k=5. La procédure de validation croisée fournit une bonne estimation générale de la performance du modèle qui n'est pas trop biaisée, par rapport à une seule séparation train-validation. De plus, l'étape de sous/sur échantillonnage a été intégrée à la pipeline pour qu'elle soit réalisée sur chaque fold afin d'éviter le déséquilibre.

Six modèles ont été testés : un classificateur naïf (Dummy Classifier), une régression logistique, un classificateur par forêt aléatoire et trois classificateurs ensemblistes, XGBoost, LightGBM et CatBoost. Le classificateur naïf effectue des prédictions qui ignorent les entités d'entrée, servant de référence simple à comparer avec les autres classificateurs.

3- La fonctions coût métier, l'algorithme d'optimisation et la métrique d'évaluation

Bien choisir les métriques pour mesurer les performances des modèles

Généralement en *machine learning*, il faut se baser sur le choix des métriques de mesure des performances. Cette règle est d'autant plus importante lorsque l'on travaille sur des données déséquilibrées.

En classification binaire, il est d'usage d'utiliser le pourcentage de bonnes prédictions comme score, sauf que ce pourcentage peut être élevé si une grande partie des points de la classe minoritaire sont mal classifiés, le score va lui aussi être impacté par le déséquilibre des classes. Pour éviter ce problème, il faut se tourner vers des métriques qui ne seront pas impactées par la mauvaise répartition. Ainsi, les métriques **sensibilité (recall)** et **F-score** sont de bons candidats.

Matrice de confusion

Dans la mission de classification, la **matrice de confusion** est le principal indicateur de la qualité d'un modèle. Il s'agit d'un tableau à double entrée avec la correspondance les classes réelles et les classes prédites par le modèle. La matrice de confusion sert de base à tous les calculs d'indicateurs.

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

Comme l'exactitude (accuracy) ne distingue pas le type des erreurs commises, il faut compléter son interprétation par deux autres indicateurs : le rappel (recall) et la précision (precision)

Choix de la fonction coût

La mission est centrée sur la classe positive (=1, les clients ayant des difficultés de paiement). La précision et le rappel sont de bons points de départ, il faut donc :

- **Maximiser la précision pour minimiser les faux positifs** : le client n'a pas de difficulté de paiement, mais il est prédit comme tel, ce qui signifie une perte de client pour la société de crédit (erreur Type I).
- **Maximiser le rappel pour minimiser les faux négatifs** : le client a des difficultés de paiement, mais il est prédit comme n'en ayant pas, ce qui cause une perte de chiffre d'affaires pour la société de crédit (erreur Type II).

Le problème ici demeure dans le fait que les **faux négatifs sont plus problématiques que les faux positifs**. Les faux négatifs sont des cas où un mauvais client est marqué comme un bon client et se voit accorder le crédit, alors que les faux positifs sont des cas où un bon client est marqué comme mauvais client et la société lui refuse le crédit. Cependant, les faux négatifs représentent un coût beaucoup plus cher à l'entreprise (perte du montant emprunté vs perte de bénéfice grâce à un prêt).

Ainsi, on s'intéresse à la mesure qui résumera la capacité d'un modèle de minimiser les erreurs de la classification pour la classe positive tout en privilégiant les modèles qui minimisent les faux négatifs plutôt que les faux positifs, c'est-à-dire, qui pénalisent plus les faux négatifs (10 fois plus). Ceci peut être réalisé en utilisant le *business score* suivant :

$$tn, fp, fn, tp = \text{confusion matrix}(y_{true}, y_{pred}).\text{ravel}()$$

$$tn_{rate}, tp_{rate}, fp_{rate}, fn_{rate} = 1, 1, -1, -10$$

$$\text{total default} = tp + fn$$

$$\text{total not default} = tn + fp$$

$$\text{total gain} = tn * tn_{rate} + tp * tp_{rate} + fp * fp_{rate} + fn * fn_{rate}$$

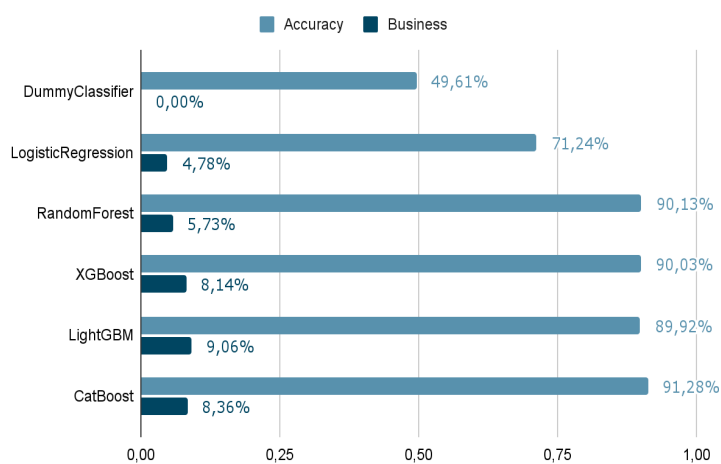
$$\text{min gain} = \text{total not default} * tn_{rate} + \text{total default} * fn_{rate}$$

$$\text{max gain} = \text{total not default} * tn_{rate} + \text{total default} * tp_{rate}$$

$$\text{business score} = (\text{total gain} - \text{min gain}) / (\text{max gain} - \text{min gain})$$

La valeur du *business score* varie de 0 à 1, où 1 est la meilleure valeur.

Choix de modèle



Nous pouvons observer que les meilleurs business scores ont été obtenus pour les modèles ensemblistes : **LightGM (9,1%)**, CatBoost (8,4%) et XGBoost (8,1%). Pour les autres modèles, les business scores ont été inférieur à 5% (voir zéro pour le classificateur naïf).

Le modèle final sélectionné est le Light Gradient Boosted Machine ou LightGM.

Optimisation de l'algorithme LightGBM

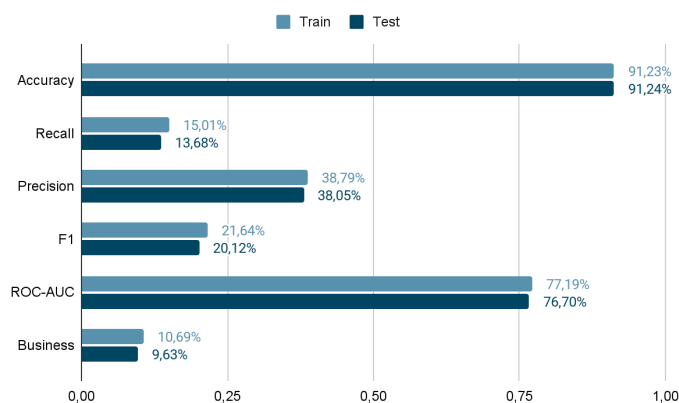
Il existe de nombreux hyperparamètres qui peuvent être évalués pour LightGBM. Ces hyperparamètres peuvent être regroupés en 4 catégories :

- Paramètres affectant la structure et l'apprentissage des arbres de décision
- Paramètres affectant la vitesse d'entraînement
- Paramètres pour une meilleur précision
- Paramètres pour combattre le surajustement

Généralement, ces catégories de paramètres se confondent et l'augmentation de l'efficacité dans l'une peut engendrer une diminution dans l'autre. C'est pour cette raison que **l'on doit éviter de régler manuellement les hyperparamètres**.

Nous avons testé certains paramètres, tels que le nombre d'estimateurs, la profondeur de l'arbre de décision, le taux d'apprentissage, le sous-échantillonnage par itération, et le nombre de feuilles. Avec ces paramètres, nous avons réalisé 120 essais avec comme objectif la minimisation de la perte.

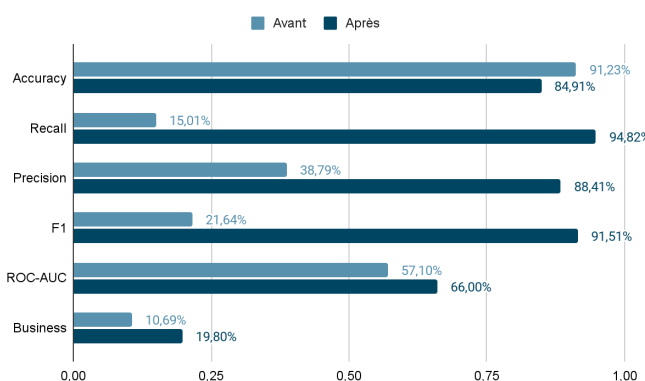
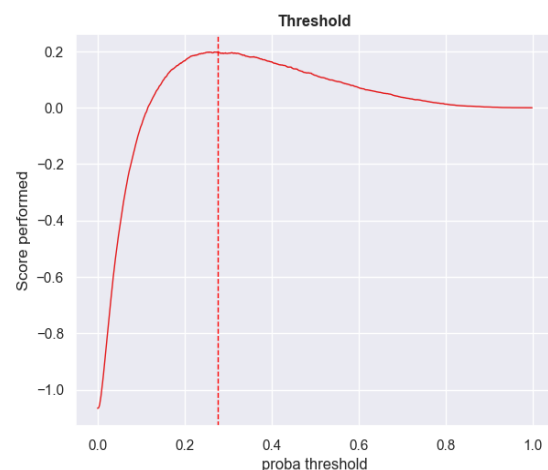
Les meilleurs paramètres obtenus :



Le business score a augmenté de 9,1% à 10,7% sur le jeu d'entraînement.

Sélection de seuil

Après le choix du modèle et de ses meilleurs hyperparamètres, la minimisation du coût métier doit passer par l'optimisation du seuil qui détermine, à partir d'une probabilité, la classe 0 ou 1. Ceci doit être réalisé, car une prédiction suppose un seuil à 0,5 ce qui n'est pas forcément l'optimum.



On observe que le seuil optimal est plutôt à **0,277** au lieu de 0,5, ce qui augmente le business score de **10,7% à 19,8%**.

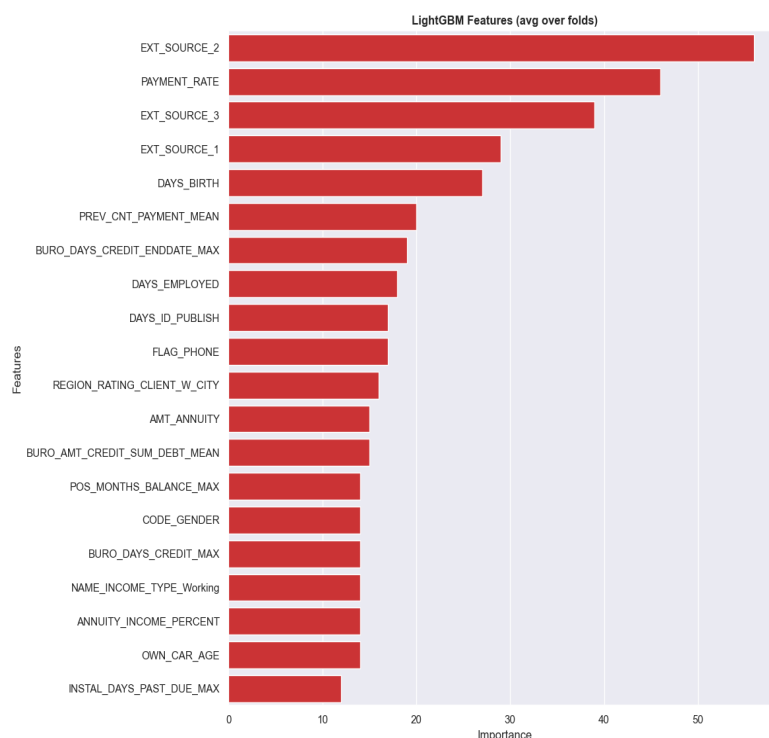
Avec le changement du seuil de 0,5 à 0,277, en plus de l'augmentation du business score, on observe une augmentation également pour **toutes les autres métriques**.

4- L'interprétation globale et locale du modèle

L'importance des variables mesure l'impact global de chaque *feature* dans le modèle. Elle peut être estimée en modélisation (sur l'échantillon d'apprentissage) ou en prédiction (sur l'échantillon de test). Dans les deux cas, les étapes sont les suivantes :

1. Calculer le taux d'erreur de référence
2. Calculer ensuite le même indicateur en neutralisant tour à tour chaque variable prédictive
3. Former le ratio entre les deux valeurs

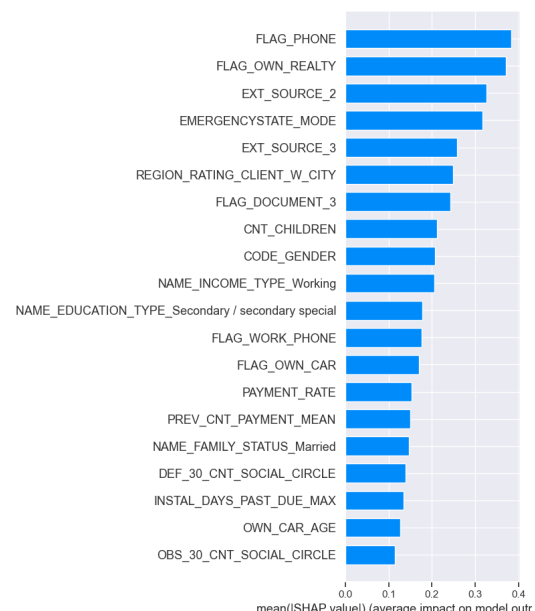
L'échantillon d'apprentissage permet de comprendre le rôle des variables dans la modélisation, alors que l'échantillon de test explique l'influence de la variable sur la performance en généralisation (quand le modèle est appliqué dans la population).



Les **valeurs de Shapley** calculent l'importance d'une variable en comparant ce qu'un modèle prédit avec et sans cette variable. Néanmoins, comme l'ordre dans lequel un modèle voit les variables peut impacter ses prédictions, cela est fait de façon aléatoire, afin que les fonctionnalités soient comparées équitablement.

L'**interprétation globale** permet d'expliquer le modèle dans sa globalité, c'est-à-dire, quelles sont les variables les plus importantes en moyenne pour le modèle. Dans le cas du modèle de scoring, quelles variables affectent le comportement général pour la décision d'octroi de prêt.

A contrario, l'**interprétation locale** consiste à expliquer la prévision d'un modèle pour un individu donné. Dans notre cas, quelles variables ont fait que la demande de crédit d'un client a été approuvée ou refusée.



5- Les limites et les améliorations possibles

La première limite dans ce projet provient de ma **connaissance limitée du milieu bancaire** et, par conséquent, du vocabulaire et des besoins. La sélection des variables, ainsi que les agrégations, ont été réalisées selon ma compréhension et en fonction de leur impact potentiel, sans forcément comprendre les implications. J'ai donc pu faire des sélections et des agrégations incohérentes d'un point de vue "métier".

Il aurait donc été nécessaire de disposer d'un dictionnaire des variables et de pouvoir d'échanger avec "*Prêt à dépenser*" pour vérifier **la cohérence du preprocessing**.

En plus, le **choix et l'optimisation du modèle** de scoring ont été réalisés sur la base d'une hypothèse forte concernant la **métrique d'évaluation** : le business score pénalise 10 fois plus les faux négatifs par rapport aux faux positifs, ce qui n'a pas été confirmé par le métier. L'axe principal d'amélioration serait donc de **définir plus finement la métrique d'évaluation et la fonction coût** en collaboration avec l'équipe métier.

Enfin, en termes d'amélioration du **dashboard**, il serait intéressant d'intégrer une partie interactive qui permettrait au client de voir quelle valeur sur quelle variable aurait pu lui permettre d'obtenir son crédit. Ainsi, on pourrait envisager de rajouter une page "scénario" où l'on pourrait changer manuellement une ou plusieurs valeurs du profil dudit client et voir l'impact sur la réponse. Il serait également intéressant d'y ajouter une fonctionnalité qui permettrait d'avoir quelle valeur (ou à partir de quelle valeur) le dit client aurait pu avoir son crédit accordé.

6- L'analyse du data drift

Le rapport de data drift permet d'évaluer l'état des données d'entrée, en vérifiant son évolution dans le temps.

Le data drift, ou la dérive des données, est le changement de la distribution de la donnée au fil du temps. Quand les caractéristiques ou propriétés de la donnée changent, cela peut impacter la performance du modèle qui l'utilise et l'exactitude de l'analyse et de la prise de décision.

Les métriques utilisées pour mesurer la dérive des données ont été :

- Distance Wasserstein mesure la distance entre deux distributions de probabilité, elle quantifie l'effort nécessaire pour "transformer" une distribution dans l'autre.
- Divergence Jensen-Shannon mesure la similarité entre deux distributions de probabilité.

Parmi les 574 colonnes présentes dans les jeux de données (entraînement et test), seulement **39 colonnes** ont dérivé, ce qui représente **6,8%** des colonnes. La colonne qui présente la dérive la plus importante a été “BURO_MONTHS_BALANCE_SIZE_SUM”, avec une **distance Wasserstein de 1,36**. Les autres variables présentent une métrique de dérive inférieure à **0,6**.

Dataset Drift

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

574

Columns

39





















Drifted Columns

0.0679

Share of Drifted Columns

Data Drift Summary

Drift is detected for 6.794% of columns (39 out of 574).

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
> BURO_MONTHS_BALANCE_SIZE_SUM	num			Detected	Wasserstein distance (normed)	1.364012
> PAYMENT_RATE	num			Detected	Wasserstein distance (normed)	0.574902
> BURO_MONTHS_BALANCE_SIZE_MEAN	num			Detected	Wasserstein distance (normed)	0.543194
> AMT_REQ_CREDIT_BUREAU_QRT	num			Detected	Wasserstein distance (normed)	0.486847
> CREDIT_INCOME_PERCENT	num			Detected	Wasserstein distance (normed)	0.29326
> AMT_REQ_CREDIT_BUREAU_MON	num			Detected	Wasserstein distance (normed)	0.288944
> BURO_STATUS_0_MEAN_MEAN	num			Detected	Wasserstein distance (normed)	0.250742
> BURO_STATUS_X_MEAN_MEAN	num			Detected	Wasserstein distance (normed)	0.244546
> AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.211138
> CC_CNT_INSTALMENT_MATURE_CUM_MIN	num			Detected	Wasserstein distance (normed)	0.196456