Note méthodologique

1. La méthodologie d'entraînement du modèle

2. Le traitement du déséquilibre des classes

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

4. Un tableau de synthèse des résultats

5. L’interprétabilité globale et locale du modèle

6. Les limites et les améliorations possibles

7. L’analyse du Data Drift

La méthodologie d'entraînement du modèle

La méthodologie d'entraînement du modèle pour la détection des bons et mauvais clients a été réalisée en suivant les étapes suivantes :

1. Objectif : L'objectif principal est de développer un modèle capable de classifier les clients en bons et mauvais, afin d'identifier les risques potentiels.

2. Données d'entraînement : Les données utilisées pour l'entraînement du modèle étaient un dataframe contenant de nombreuses variables. Ces données ont été collectées et nettoyées, en incluant également du data Engineering.

3. Prétraitement des données : Les données ont subi plusieurs étapes de prétraitement, notamment l'encodage des variables catégorielles, l'utilisation de la méthode "get_dummies" pour le reste des variables catégorielles, l'imputation des valeurs manquantes et la mise à l'échelle des variables numériques à l'aide d'un scaler approprié.

4. Taille du jeu de données : Pour l'entraînement du modèle, 30% du jeu de données ont été utilisés, qui ont été divisés en ensembles d'entraînement et de test à l'aide de la méthode de séparation "train_test_split".

5. Métriques d'évaluation : Différentes métriques ont été utilisées pour évaluer les performances du modèle, notamment le recall score, l'AUC , le F-score, le precision score et le F-beta score. Ces métriques permettent de mesurer la capacité du modèle à identifier correctement les bons et mauvais clients.

6. Modèles évalués : Plusieurs modèles ont été évalués, notamment LightGBM, Random Forest, Decision Tree et Logistic Regression. Suite à ces évaluations, le modèle LightGBM a montré les meilleures performances.

En résumé, la méthodologie d'entraînement du modèle pour la détection des bons et mauvais clients a impliqué l'évaluation de différents modèles, avec LightGBM se révélant le plus performant. Les données ont été prétraitées en utilisant des techniques telles que l'encodage, l'imputation et la mise à l'échelle, avant d'être divisées en ensembles d'entraînement et de test. Les performances du modèle ont été évaluées à l'aide de plusieurs métriques, dont le recall score, l'AUC, le F-score, le precision score et le F-beta score.

Traitement du déséquilibre des classes

Le traitement du déséquilibre des classes dans la détection des bons et mauvais clients d'une banque a été réalisé en suivant les étapes suivantes :

1. Déséquilibre des classes : Le jeu de données initial présentait un déséquilibre important entre les bons clients (282,686) et les mauvais clients (24,825).

2. Problèmes associés au déséquilibre : En raison de la faible représentation des mauvais clients, il y a un risque de mauvais entraînement du modèle sur cette classe et une tendance à prédire majoritairement les bons clients.

3. Utilisation du paramètre "class_weight" : Nous avons utilisé le paramètre "class_weight = 'balanced'" lors de l'entraînement des modèles. Cette technique attribue automatiquement des poids aux classes en fonction de leur proportion dans les données, ce qui aide à corriger le déséquilibre des classes.

4. Avantages et inconvénients : L'utilisation du paramètre "class_weight = 'balanced'" a conduit à une amélioration des résultats en prenant en compte le déséquilibre des classes.

5. Évaluation des performances : Les performances du modèle ont été évaluées à l'aide d'une validation croisée avec GridSearchCV. Cela nous a permis de sélectionner les hyper paramètres optimaux pour chaque modèle et d'obtenir de meilleurs résultats.

6. Les résultats obtenus après le traitement du déséquilibre des classes ont montré une amélioration des performances des modèles dans la détection des bons et mauvais clients. Cela indique que la prise en compte du déséquilibre des classes a permis d'améliorer la capacité des modèles à détecter les mauvais clients.

La fonction coût métier et métrique d'évaluation

La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation ont été pris en compte dans notre approche pour la détection des bons et mauvais clients de la manière suivante :

1. Le precision et recall score ne sont pas assez précis/utile de par le déséquilibre des classes

2. Fonction coût métier : Pour tenir compte des conséquences financières spécifiques de la classification incorrecte des clients, nous avons utilisé le f-beta score comme métrique principale. Cela nous a permis de mettre davantage l'accent sur les faux négatifs (mauvais clients classés comme bons clients) par rapport aux faux positifs (bons clients classés comme mauvais clients).

3. Estimation des coûts : Pour quantifier les conséquences financières, nous avons estimé qu'un faux négatif valait entre 10 et 20 fois plus qu'un faux positif. Cette estimation a été basée sur des considérations spécifiques liées aux coûts associés aux mauvaises prédictions et aux impacts financiers potentiels.

4. Algorithme d'optimisation : Nous avons utilisé la courbe ROC pour déterminer le seuil de décision optimal. La courbe ROC nous a permis d'analyser la relation entre le taux de vrais positifs et le taux de faux positifs à différents seuils de décision, afin de trouver le point de compromis approprié entre sensibilité et spécificité pour notre problème.

En utilisant la courbe ROC, nous avons pu déterminer le seuil de décision qui maximise nos prédictions, en tenant compte de la fonction coût métier et de l'importance relative des faux négatifs par rapport aux faux positifs.

Cette approche nous a permis d'optimiser la performance du modèle en termes de détection des mauvais clients tout en prenant en compte les coûts associés aux erreurs de classification.

En résumé, le f-beta score a été utilisé comme métrique principale pour tenir compte des coûts associés aux faux négatifs et aux faux positifs. L'estimation des coûts a permis de quantifier l'importance relative des erreurs de classification. En utilisant la courbe ROC, nous avons déterminé le seuil de décision optimal qui maximise nos prédictions, en prenant en compte les conséquences financières spécifiques. Cette approche a permis d'optimiser la performance du modèle dans la détection des bons et mauvais clients tout en tenant compte de la fonction coût métier.

Tableau de synthèse des résultats

Résultat sans traitement des classes

	ROC AUC	f1 score	precision	recall
Logistic Reg	0.747	0.0125	1	0.006
Random forest	0.765	0	0	0
Decision tree	0.636	0	0	0
LightGBM	0.744	0.005	0.5	0.0025

Très mauvais f1, precision et recall score a cause du mauvais équilibre des classes et donc le mauvais entraînement sur les mauvais clients

Résultat avec equilibre des classes

	ROC AUC	f1 score	precision	recall
Logistic Reg	0.743	0.25	0.154	0.658
Random forest	0.758	0.245	0.151	0.637
Decision tree	0.564	0.159	0.087	0.867
LightGBM	0.745	0.257	0.15	0.661

Amélioration de tous les scores globalement surtout f1, precision et recall score
Nous avons donc garder le modele LightGBM

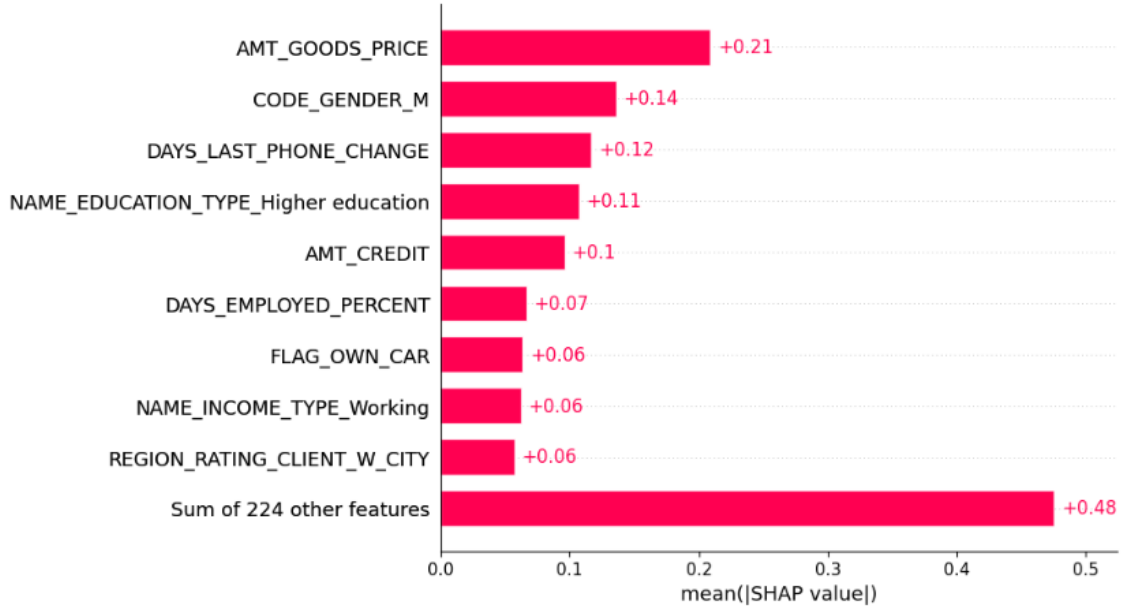
Résultat avec data Engineering et fbeta score

	ROC AUC	f1 score	precision	recall	fbeta score
LightGBM	0.68	0.223	0.135	0.632	0.636

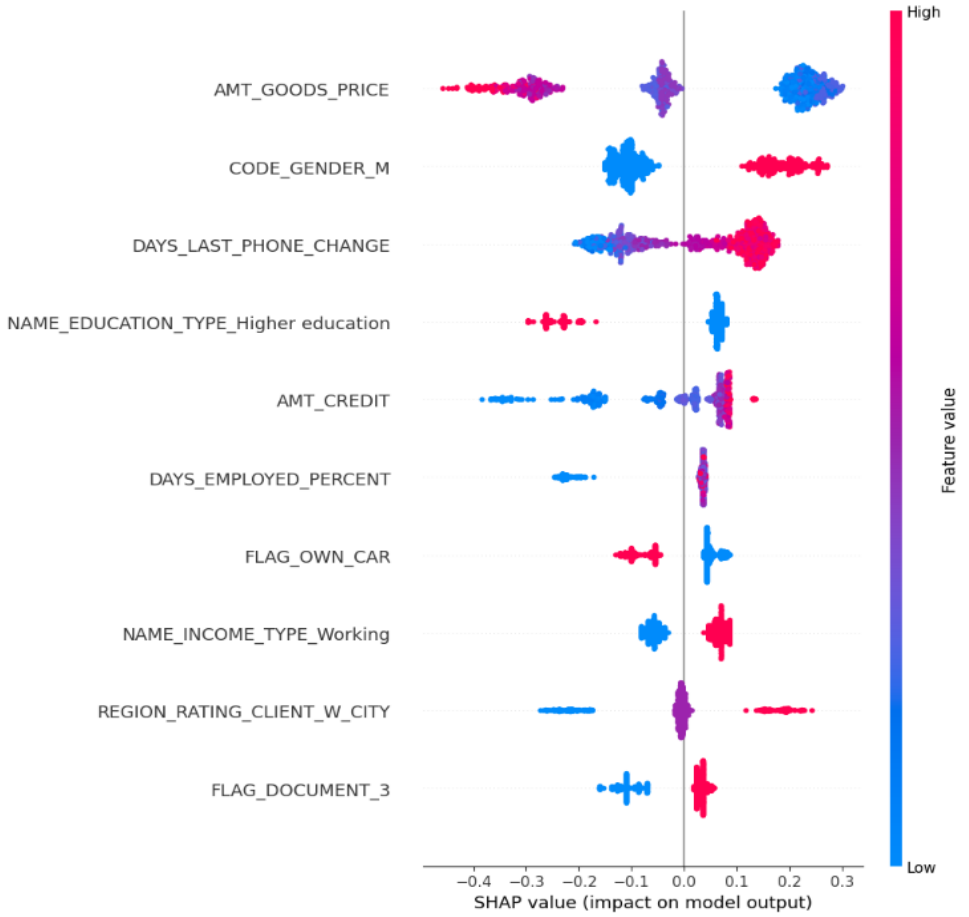
Nous constatons une baisse des scores globaux, mais cela est cohérent avec notre métrique métier et notre objectif de réduire le nombre de faux négatifs tout en maintenant la cohérence du modèle.

L’interprétabilité globale et locale du modèle

Features les plus importantes :



Influence des valeurs sur le modèle :



Les limites et les améliorations possibles

Limites :

1. Manque de connaissances dans le domaine de la banque : En raison d'un manque de connaissances spécifiques dans le domaine de la banque, il peut y avoir des aspects importants ou des nuances qui ne sont pas pris en compte dans la méthodologie développée. Cela pourrait limiter la précision.

2. Données manquantes : Certaines colonnes du jeu de données présentent des données manquantes. Cela peut entraîner des lacunes dans la compréhension et la modélisation des caractéristiques pertinentes pour la détection des bons et mauvais clients.

Améliorations possibles :

1. Améliorer la fonction métier en connaissant le coût réel d'un faux négatif et faux positif : Actuellement, l'estimation du coût d'un faux négatif est basée sur une estimation approximative. Il serait bénéfique de mener une étude plus approfondie pour déterminer le coût réel associé à la classification incorrecte des mauvais clients en tant que bons clients. Cela permettrait de mieux quantifier l'impact financier et d'ajuster la fonction coût métier en conséquence.

2. Acquérir une meilleure compréhension du domaine de la banque : Il serait bénéfique de collaborer avec des experts du domaine de la banque afin d'acquérir une connaissance plus approfondie des facteurs spécifiques qui influencent la détection des bons et mauvais clients. Cela permettrait de prendre en compte des variables pertinentes et de développer des modèles plus précis et adaptés au domaine.

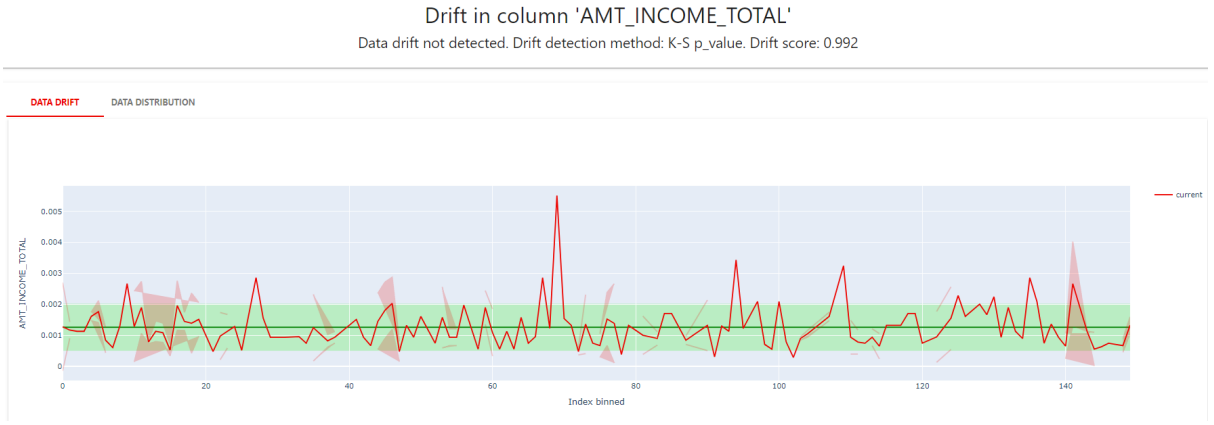
En prenant en compte ces limites et en mettant en œuvre les améliorations proposées, il est possible d'améliorer la précision et la pertinence de la méthodologie de détection des bons et mauvais clients. Cela permettrait une meilleure prise de décision et une réduction des risques financiers associés aux mauvaises classifications des clients.

Data Drift

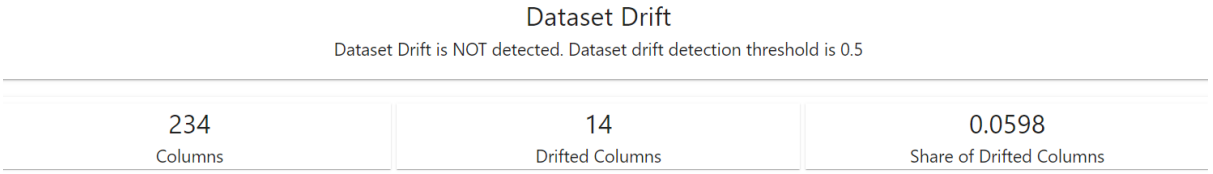
Le data drift, également appelé dérive des données, est un phénomène qui se produit lorsque les caractéristiques des données utilisées dans un modèle d'apprentissage automatique changent au fil du temps. En d'autres termes, les données sur lesquelles le modèle a été formé et les données réelles sur lesquelles il est déployé ne sont plus en adéquation.

Nous avons utilisé evidently pour détecter la présence de Data Drift:

- Vérification pour notre top 5 Features :
    - Aucun drift détecter sur les features



- Vérification sur toutes nos features pour plus d'information :



- Sur nos 234 colonnes, seulement 14 sont détectés avec de la dérivé ce qui représente 5% de la données :
    - Avec des vérifications supplémentaires, les colonnes contenant de la dérivé de données sont parmis les colonnes les moins importantes de notre modèle de machine Learning