# **Note méthodologique du projet 7**

## Objectif

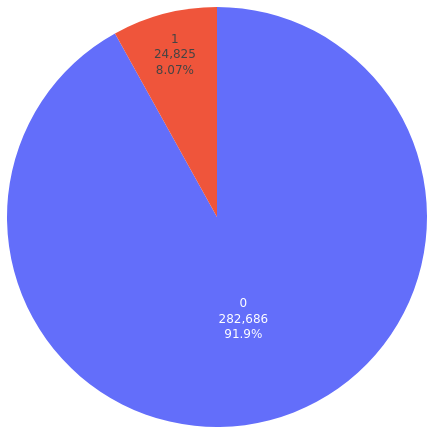
Pour la société « Prêt à dépenser », nous devons établir un modèle de Machine Learning permettant de classifier des demandes de crédit en 2 classes :

* + 0 : crédit accordé
  + 1 : crédit refusé

Pour ce faire, nous avons des données d’entraînement dont l’étude exploratoire montre :

* Une quantité importante de données non remplies (ex : le client n’a pas d’historique de crédit)
* Un grand déséquilibre des données au niveau de la classe cible :
  + - 282686 individus avec la classe TARGET=0
    - 24825 individus avec la classe TARGET=1

Soit un un rapport de 11.4 entre la classe majoritaire et la classe minoritaire.



Nous savons aussi que les prêts accordés à des clients non solvables coûtent au créditeur plus que les autres clients, mais nous n’avons pas de coût exact.

Les demandes de prêts se prêtent bien aux arbres de décision. On va donc se focaliser sur ce type de modèles.

Etant donné la présence de valeurs nulle (qui ne signifie pas forcement une valeur manquante), nous allons nous concentrer sur les modèles de classification qui gèrent les valeurs nulles :

* HistGradientBoostingClassifier
* CatBoost
* LightGBM
* XGBoost

# **Méthodogie d’entraînement**

L’entraînement des modèles se fera en 3 étapes :

1. La recherche des hyper-paramètres les plus adaptés
2. L’apprentissage avec les données d’entraînement
3. La validation avec les données de test

Les deux premières étapes se font dans le cadre d’une validation croisée.

Les données sont séparées en :

* Données d’entraînement : 206032 individus
* Données de test : 101475 individus

Afin de faciliter la traçabilité des différents modèles, la bibliothèque MLFlow est utilisée. Celle-ci permet de retenir sans intervention de notre part les paramètres utilisés pendant l’entraînement mais aussi les métriques mesurées. MLFlow enregistre aussi les modèles élaborés et permet un accès rapide à ceux-ci, ce qui facilite leur comparaison.

#### **Recherche des hyper-paramètres : Bayesian Optimization**

|  |  |
| --- | --- |
|  | Cette technique utilise des processus gaussiens itératifs pour trouver les meilleurs hyper-paramètres minimisant la fonction de coût spécifiée. On trouve ainsi la combinaison d’hyper-paramètres optimale en un temps moindre que les Grid Search.  Cette technique nécessite en entrée d’avoir pour chaque hyper-paramètre à chercher un couple (limite inférieure, limite supérieure). |

#### **Apprentissage par validation croisée**

|  |  |
| --- | --- |
| La validation croisée consiste à diviser les données d’entraînement en un nombre n de sous-ensembles de même cardinalité (appelés folds) et d’utiliser n combinaisons d’apprentissage avec pour chaque combinaison un fold pour la validation et les autres folds pour l’apprentissage. Chaque combinaison est appelée split.  L’utilisation de cette technique permet d’augmenter le nombre d’apprentissages avec le même ensemble de données d’entraînement et ainsi d’avoir plusieurs modèles qui sont proches. Pour chaque modèle trouvé, une évaluation est réalisée avec le fold de validation. Le modèle fournissant la meilleure évaluation sera retenu | Données d’entraînement |

La fonction d’évaluation pendant l’apprentissage est la fonction custom\_score (cf xxx).

La validation croisée est aussi utilisée dans le cadre de la recherche des hyper-paramètres.

#### **Traçabilité des expérimentations**

Pendant la modélisation, de nombreuses expérimentations sont réalisées. Ceci implique pour l’utilisateur une gestion rigoureuse de l’enregistrement des résultats de ces expérimentations.

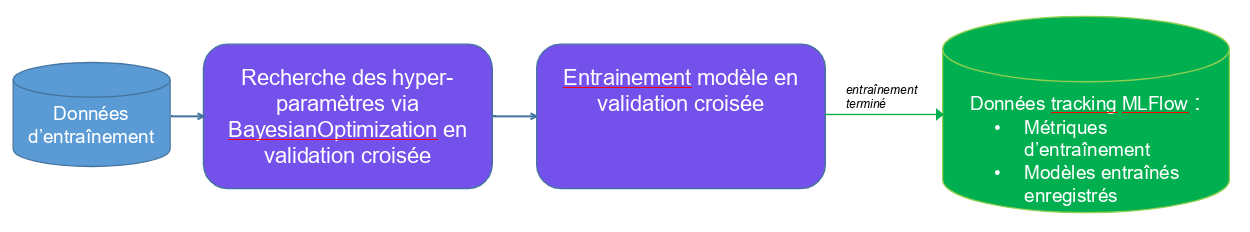
MLFlow est un outil développé pour gérer cette tâche de manière transparente et automatique. Parmi les nombreuses possibilités offertes par l’outil, on peut citer :

* Enregistrement et mise à disposition des modèles générés por utilisation
* Enregistrement des métriques, temps de calculs
* Générations de graphiques tels que Recall-Precision, ROC
* Fourniture d’un dossier d’export pour utiliser le modèle généré dans d’autres applications

Nous allons donc utiliser cet outil pour pouvoir naviguer entre les différentes expérimentations réalisées au cours de ce projet.

#### **Workflow**

La méthodologie utilisée pour entraîner les modèles peut donc être décrite de la manière suivante :



# **Traitement du déséquilibre des classes**

Comme évoqué en préambule de ce document, les données fournies pour l’entraînement sont sont déséquilibrées au niveau de leurs classes. Cela apporte un biais au niveau de l’entraînement des modèles : la classe majoritaire est plus fréquemment rencontrée lors de l’apprentissage.

Parmi les solutions pour pallier à ce problème : réduire le nombre de cas dans la classe majoritaire, augmenter le nombre de cas dans la classe minoritaire ou encore utiliser quand le modèle le permet un facteur de correction interne

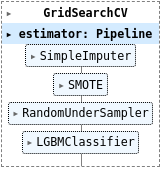
#### Réduire le nombre de cas dans la classe majoritaire

La technique la plus simple consiste à diminuer la classe majoritaire afin qu’elle ait la même cardinalité que la classe minoritaire. On peut peut par exemple effectuer une suppression arbitraire des individus jusqu’à parité des classes.

#### Augmenter le nombre de cas dans la classe minoritaire

Une autre technique est d’augmenter le nombre d’éléments de la classes minoritaire jusqu’à obtenir le ratio classe majoritaire/classe minoritaire désiré.

Pour cela on utilise des outils tels que ceux proposés par la librairie Imbalanced Learn : il s’agit ici de trouver les plus proches voisins des d’éléments de la classe minoritaire dans la même classe et d’effectuer des combinaisons pour créer de nouveaux éléments de la même classe. Différentes variantes au sein de cette librairie existent (SMOTE, ADASYN, SMOTE-borderline, KMEANS-SMOTE...) mais sont toutes basées sur le même principe.



La littérature montre que l’utilisation conjointe d’une augmentation de la classe minoritaire et une réduction de la classe majoritaire fournit généralement de meilleurs résultats.

L’utilisation de SMOTE nécessite l’absence de valeurs non renseignées. Nous avons donc remplacé les valeurs manquantes par leur médiane dans le cadre d’un pipeline.

#### Utilisation du facteur de correction inhérent à l’algorithme d’apprentissage

Certains algorithmes pallient au problème de données non équilibrées en donnant la possibilité de définir des poids à prendre en compte pour chaque classe.

#### Comparaison

Des tests ont été réalisés avec les données du projet entre

1. l’association d’une augmentation (via SMOTE) et d’une réduction (via RandomOverSampler) des données
2. l’utilisation des poids des classes dans la définition du modèle.

Pour le 1er cas :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Total avant | Initialement | SMOTE | RandomOverSampler | Total après |
| Classe 0 | 206 032 | 189 324 | 189 324 | 113 594 | 227 188 |
| Classe 1 | 16 708 | 113 594 | 113 594 |

Pour le 2ème cas : nous avons utiliser le paramètre scale\_pos\_weight = 189324/16708

Les résultats de la comparaison sont dans le tableau de synthèse des résultats

# **Optimisation et évaluation des modèles**

Comme pour tout problème de classification, nous utilisons la matrice de confusion :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Valeurs prédites | |  |
|  |  | 0 | 1 |  |
| Valeurs  réelles | 0 | TN | FP | *Specificite =* |
| 1 | FN | TP | *Recall* = |
|  |  |  | *Precision* = |  |

Dans notre cas, un client non solvable prédit comme solvable coûte cher. Nous allons donc chercher à maximiser le score Recall.

Une autre contrainte réside dans le fait que la société veut minimiser le nombre de clients solvables prédits comme non solvables. Nous devons donc maximiser le score Specificite.

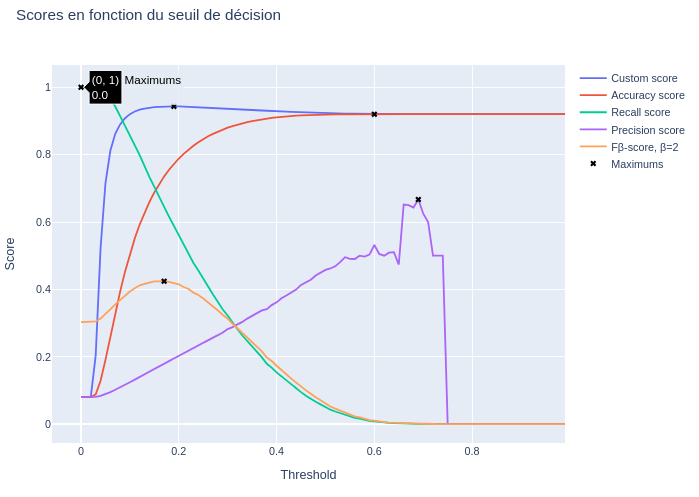
Pour conjuguer ces 2 contraintes, nous utilisons un score métier :

*Custom\_score = 4TN - 14FN - FP*

C’est ce custom\_score qui sera utilisé pour décider du seuil optimal délimitant les classes 0 et 1. Pour confirmer ou infirmer les résultats, nous utiliserons aussi un score donnant le revenu généré par chacun des modèles sur les données de test.

*Score\_revenu = 40TN - 140FN*

Chaque modèle sera d’abord étudié sous l’angle des métriques générales : Durée d’apprentissage et Score ROC AUC, et d’autre part en prenant le seuil pour atteindre le custom\_score optimal :Recall, Precision, Accuracy et Custom\_score



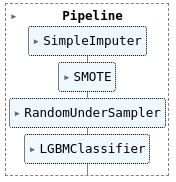
# **Résultats des modèles**

L’évaluation des différentes modèles donne avec les données de tests :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Modèle** | **Technique** | **Durée** | **Roc AUC** | **Recall** | **Precision** | **Accuracy** | **Score métier** | **Score**  **Revenu** |
| Baseline | Aug/Réduc | 4.2 | 0.502 | 0.507 | 0.499 | 0.499 | 0.876 | 1 301 020 |
| Poids | 1.6 | 0.500 | 0.081 | 0.081 | 0.916 | 0.945 | 2 390 320 |
| HistGradientBoosting | Aug/Réduc | 110.6 | 0.689 | 0.589 | 0.195 | 0.772 | 0.943 | 2 477 860 |
| Poids | 38.7 | 0.683 | 0.583 | 0.189 | 0.766 | 0.942 | 2 450 740 |
| CatBoost | Aug/Réduc | 25.5 | 0.675 | 0.558 | 0.203 | 0.788 | 0.940 | 2 453 560 |
| Poids | 17.7 | 0.651 | 0.488 | 0.186 | 0.789 | 0.937 | 2 462 240 |
| LightGBM | Aug/Réduc | 63.8 | 0.700 | 0.595 | 0.210 | 0.788 | 0.946 | 2 548 880 |
| Poids | 52.1 | 0.701 | 0.604 | 0.207 | 0.783 | 0.946 | 2 533 260 |
| XGBoost | Aug/Réduc | 74.9 | 0.692 | 0.583 | 0.202 | 0.783 | 0.944 | 2 516 560 |
| Poids | 74.7 | 0.700 | 0.608 | 0.203 | 0.777 | 0.945 | 2 514 320 |

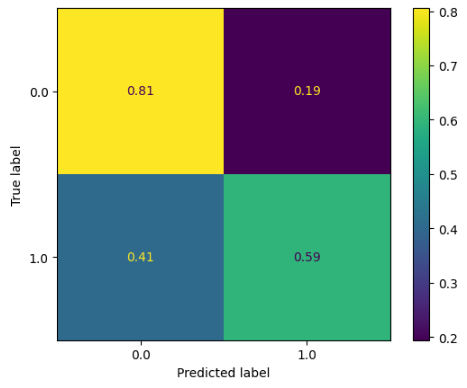
Les scores Recall, Precision, Accuracy, score métier, et score revenu sont calculés à l’issue de la détermination du seuil optimisant le score métier.

On constate que le modèle LightGBM fournit les meilleurs résultats, que ce soit en en utilisant l’augmentation et réduction des données ou en utilisant les poids associés aux classes lors de l’apprentissage.



L’utilisation du score revenu nous permet de distinguer les 2 modèles LightGBM : on retiendra le modèle LightGBM en utilisant l’imputation des valeurs manquantes, l’augmentation de la classe minoritaire via SMOTE et la réduction de la classe majoritaire via RandomUnderSampler.

Matrice de confusion



# **Interprétabilité du modèle retenu**

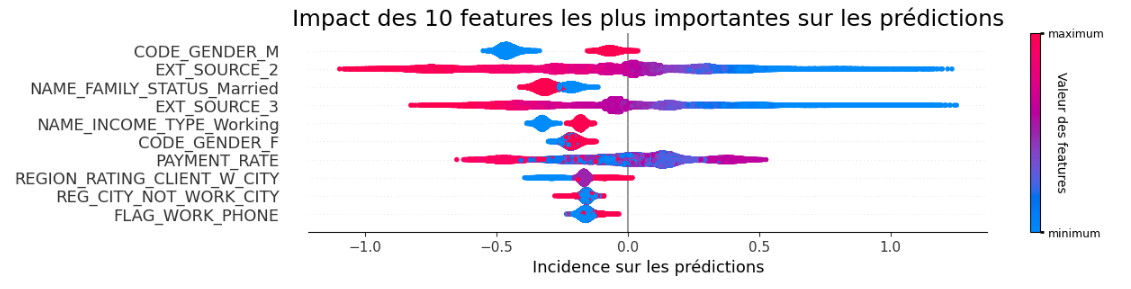
Une fois un modèle ML établi, il est primordial de comprendre comment celui-ci fonctionne pour plusieurs raisons :

* Au niveau métier : la compréhension du fonctionnement du modèle permet d’avoir une confiance sur sa cohérence
* Au niveau éthique : certains modèles doivent pouvoir être justifiés. C’est par exemple le cas dans notre projet, où le client peut demander à comprendre pourquoi il se voit refuser un crédit
* Au niveau légal : l’article 22 du RGPD prévoit qu’un personne a le droit de ne pas faire l'objet d'une décision fondée exclusivement sur un traitement automatisé. ... le responsable du traitement met en œuvre des mesures appropriées ... , au moins du droit de la personne concernée d'obtenir une intervention humaine de la part du responsable du traitement, d'exprimer son point de vue et de contester la décision.

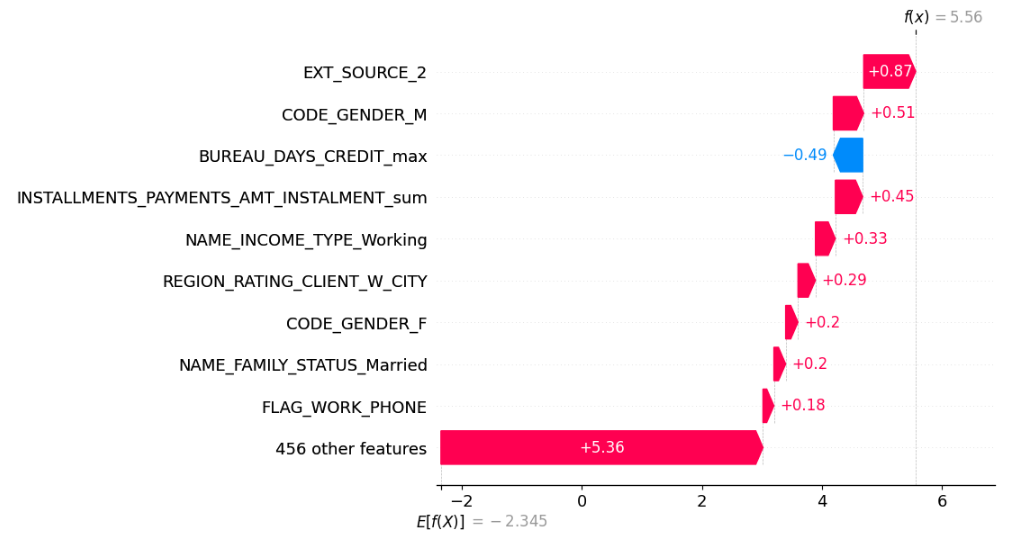
|  |  |
| --- | --- |
|  | **SHAP en 2 mots :**  Shap explique l’amplitude de contribution de chaque feature aux prédictions d’un modèle, de manière globale (pour un grand nombre d’observations) ou au niveau local (pour un nombre d’observations proches les unes des autres). |

#### Interprétation globale

Dans la figure ci-dessous est représentée l’incidence des 10 features les plus importantes sur les prédictions du modèle. On peut par exemple voir que la feature CODE\_GENDER\_M (valeurs 0 ou 1) tend à diminuer le score de prédiction (classer vers 0, donc crédit accordé) les personnes de sexe féminin. On constate aussi que plus les revenus correspondant à EXT\_SOURCE\_2 et/ou EXT\_SOURCE\_3 sont élévés, plus la prédiction tend vers la classe 0.



#### Interprétation globale



Ici, pour une observation particulière, sont affichées les features les plus importantes qui ont contribué à la prédiction correspondante. Ainsi, on constate que la feature qui a contribué le plus est « EXT\_SOURCE\_2 », celle-ci a contribué de manière à faire tendre la classe prédite vers 1.

Parmi les 10 features les plus importantes, on voit que seule la feature « Bureau\_DAYS\_CREDIT\_max » a contribué à faire tendre la classe prédite vers 0.

# **Limites et améliorations possibles**

#### Problèmes rencontrés



Lors de ce projet, la plus grosse difficulté vient de l’absence de connaissance du métier. Trop de données fournies n’étaient pas comprises, même avec l’explication. Ceci venant du fait qu’elles proviennent d’un domaine non connu.

J’ai donc été confronté à mes limites de connaissance du métier.

La deuxième difficulté éprouvée a été d’obtenir des résultats satisfaisant, en particulier au niveau des matrices de confusion, avec le fait que les classes ne soient pas équilibrées.



Autre problème trouvé : les processus d’apprentissage ont été limités par les capacités de calcul.

#### Améliorations possibles

Avec une ressource métier, une compréhension des données aurait permis de nous focaliser sur les features à priori plus pertinentes.

Cette ressource aurait aussi permis d’établir un score métier plus adéquat dans le cadre du projet.

Plus de ressources machine aurait aussi permis de faire tourner plusieurs apprentissages en parallèle et ainsi comparer plus de possibilité, par exemple approfondir les cas d’ augmentation/réduction de classes avec d’autres ratios et/ou d’autres techniques.

# **Analyse du datadrift**

#### Problématique

Au cours du temps , les données utilisées pour entraîner un modèle ML peuvent changer de comportement. On peut prendre comme exemple des données concernant les habitudes de voyages en avion avant la crise du COVID : ces données ne sont plus applicables depuis 2020.

On parle alors de Data Drift ou de dérive des données. Ces données ne sont plus à jour. Tout modèle entraîné sur des données qui ont dérivé nécessite un nouvel entraînement avec des données valides.

#### Méthode

Pour analyser le Data Drift entre 2 datasets de même nature, on analyse les différences de distribution entre les 2 populations. On prend comme référence le dataset sur lequel le modèle a été entraîné.

Afin de pouvoir analyser l’impact du data drift sur les features cibles, nous pouvons aussi les rajouter aux colonnes à analyser. Nous devons dans ce cas comparer ce qui est comparable et prédire la cible dans les 2 datasets.

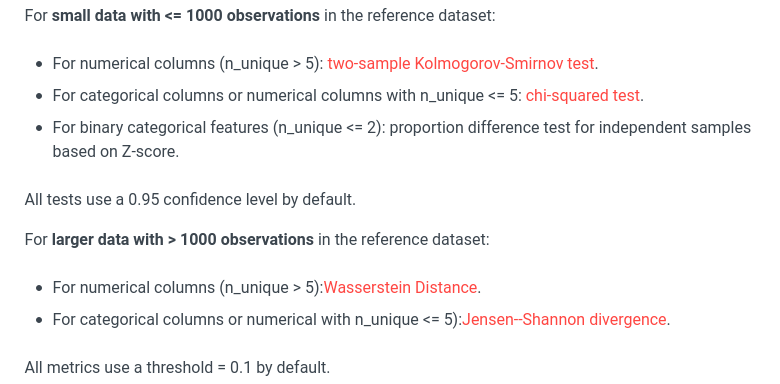
Si nous avons les cibles d’origine dans les 2 datasets, nous pouvons les ajouter pour l’analyse.

Si le dataset est beacoup trop grand (et donc prends trop de temps) :

* Si le nombre de features est trop important : on peut ne considérer que les features les plus importantes (en utilisant la propriété feature\_importance\_ ou en utilisant SHAP)
* Si le nombre d’individus est trop grand, on peut échantillonner nos datasets pour obtenir une taille gérable par les ressources à disposition.

#### Solution utilisée

Nous allons utiliser la librairie Evidently et en particulier la configuration DataDriftPreset qui fait le travail d’analyse du DataDrift. Cette librairie utilise différents scores en fonction de la nature des Datasets



#### Résultats

Pour notre projet, nous avons utilisé SHAP pour ne garder dans notre étude que les 80 features les plus importances (qui représentent 65% de l’explication globale). Nous y avons ajouter les prédictions de la cible (feature TARGET)

Le rapport nous indique que globalement (avec les seuils laissés par défaut), **il n’y a pas de dérive des données**, la cible prédite ne dérive pas non plus.