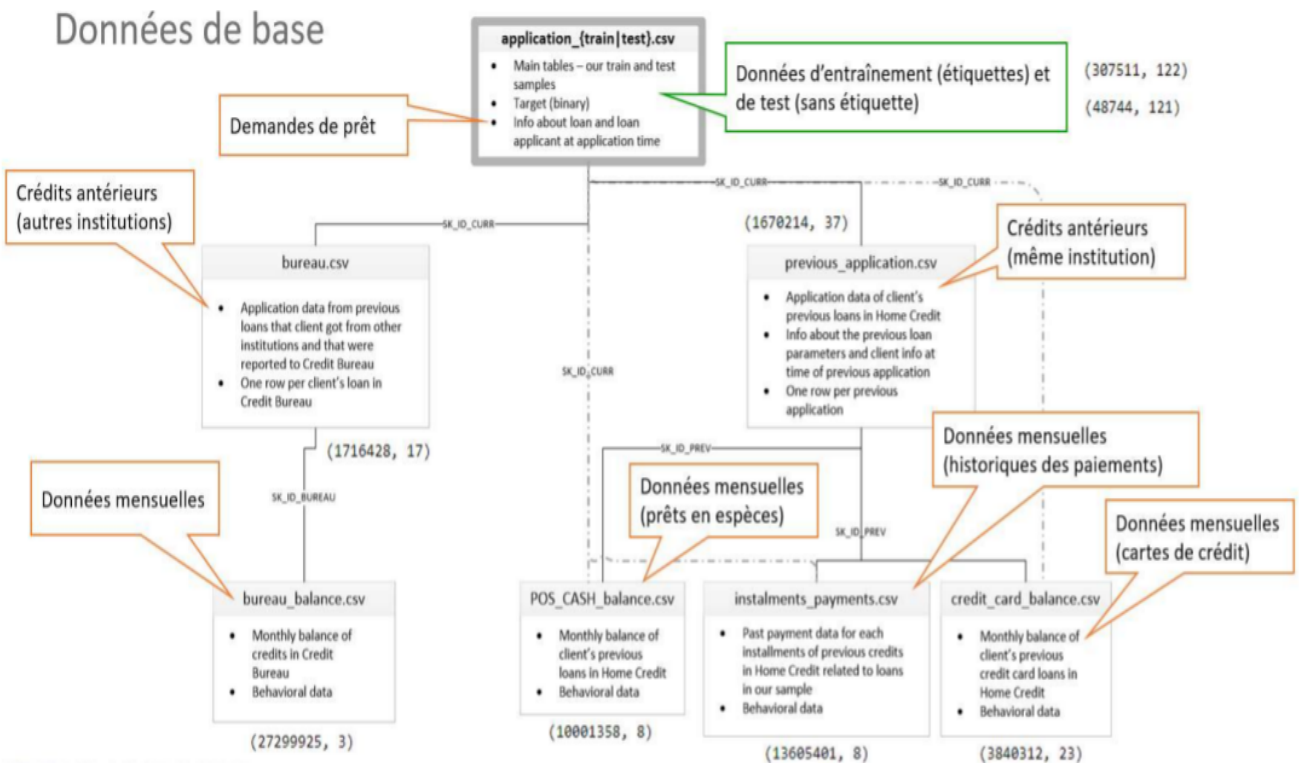
NOTE METHODOLOGIQUE

1. Contexte de la mission

Cette note constitue l'un des livrables du projet « Implémentez un modèle de scoring » du parcours Data Scientist d'Openclassrooms. Il présente le processus de modélisation et d'interprétabilité du modèle mis en place dans le cadre du projet. Le projet consiste à développer pour la société « Prêt à Dépenser », une société de crédit à la consommation, un modèle de scoring de la probabilité de défaut de paiement d'un client avec pas ou peu d'historique de prêt. Les données utilisées pour ce projet sont une base de données de 307 000 clients comportant 121 « features » (âge, sexe, emploi, logement, revenus, informations relatives au crédit, notation externe, etc.).

La probabilité de classification est essentielle car nous voulons être très sûrs lorsque nous classons quelqu'un en tant que non-défaut, car le coût d'une erreur peut être très élevé pour l'entreprise.



2. Utilisation de la librairie Pycaret lors des travaux de modélisation

- PyCaret est une bibliothèque d'apprentissage automatique open source en Python qui automatise les flux de travail d'apprentissage automatique. Il s'agit d'un outil d'apprentissage automatique et de gestion de modèles de bout en bout qui accélère le cycle d'expérimentation de manière exponentielle et le rend plus productif.
- En comparaison avec les autres bibliothèques d'apprentissage automatique open source, PyCaret est une bibliothèque alternative qui peut être utilisée pour remplacer des centaines de lignes de code par quelques lignes de code seulement. Cela rend les expériences exponentiellement rapides et efficaces.
- PyCaret est essentiellement un wrapper Python autour de plusieurs bibliothèques et frameworks d'apprentissage automatique tels que scikit-learn, XGBoost, LightGBM, CatBoost, spaCy, Optuna, Hyperopt, Ray et bien d'autres.

3. Mode opératoire :

- ***Etape 1 : choix des modèles***

Présélection de quelques modèles candidats => entraînement de tous les modèles disponibles dans la bibliothèque de modèles avec notation à l'aide de la validation croisée Kfold (10 folds) pour l'évaluation des métriques (AUC, F1-Score, Log-Loss) et du temps d'entraînement.

- ***Etape 2 : création et notation du modèle choisi***

A l'aide de la validation croisée K-fold (10 plis).

- ***Etape 3 : Optimisation du modèle et réglage des hyperparamètres***

La fonction tune_model règle automatiquement les hyperparamètres d'un modèle à l'aide de la recherche par grille aléatoire sur un espace de recherche prédéfini. Possibilité de passer le paramètre custom_grid dans la fonction tune_model. Utilisation possible de fonctions de recherche optimisées avec Hyperopt et Optuna.

- ***Etape 4 : Prédiction sur le jeu de données entraîné avec une validation croisée***

Prédire l'ensemble test/hold-out (30 %) du jeu d'entraînement initial.

- ***Etape 5 : Finaliser le modèle pour le déploiement en production***

La fonction finalize_model() ajuste le modèle sur l'ensemble de données d'entraînement complet, y compris l'échantillon test/hold-out (30%).

- ***Etape 6 : Prédictions sur le 'unseen df'***

Jeu de test non traité et représentant 20 % du jeu de données complet.

4. Formulation du problème d'apprentissage automatique :

- Il s'agit d'un problème de classification d'apprentissage supervisé, qui contient les points de données d'apprentissage ainsi que leurs étiquettes de classe. Ici, les étiquettes de classe indiquent si un candidat donné est un défaillant ou non. Ainsi, pour une application donnée d'un client, en utilisant les fonctionnalités données, nous devons prédire l'étiquette de classe associée à ce client.

- Il s'agit d'un problème de classification binaire, c'est-à-dire qu'il ne contient que 2 classes, à savoir. Positif (1) et Négatif (0).

- La ***classe négative (Target = 0)*** se réfère ici aux non-défaillants et la classe positive (Target = 1) aux défaillants.

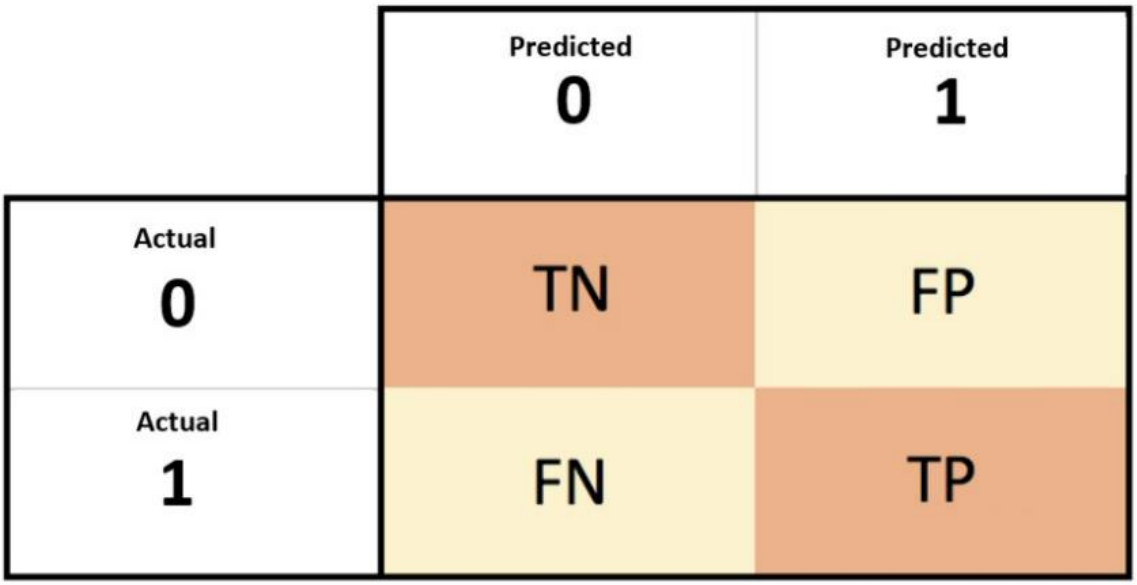
- L'ensemble de données fourni est un ***ensemble de données déséquilibré*** (8 % de clients en défaut contre 92 % de clients sans défaut). Ainsi, nous avons besoin de corriger ce déséquilibre car certains algorithmes de ML sont sensibles au déséquilibre des données. Ce ***déséquilibre*** sera ***corrigé*** à l'aide ***du paramètre « fix_imbalance » de la fonction setup de Pycaret***. ***L'approche SMOTE*** (Synthetic Minority Oversampling Technique) ***est appliquée par défaut*** pour créer des points de données synthétiques pour la classe minoritaire.

5. Indicateurs de performances :

Étant donné que les données dont nous disposons sont un ensemble de données déséquilibrées, nous ne pouvons pas simplement utiliser la précision comme mesure pour évaluer les performances du modèle. Certaines métriques fonctionnent bien avec des ensembles de données déséquilibrées, parmi lesquelles les métriques mentionnées ci-dessous.

La matrice de confusion est à la base de toutes les métriques de performance d'un modèle de classification qui s'appuie sur les TN, FN, FP et TP que l'on voit sur la matrice ci-dessous.

- ***Matrice de confusion*** *: La matrice de confusion nous aide à visualiser les erreurs commises par le modèle sur chacune des classes, qu'elles soient positives ou négatives. Par conséquent, elle nous informe sur les erreurs de classification pour les deux classes.*



Visualisation de la matrice de confusion (Source)

Une chose importante à noter ici est que nous ***voulons un score de rappel élevé*** même s'il conduit à un score de précision faible (conformément au compromis précision-rappel). C'est parce que nous nous soucions davantage de minimiser les faux négatifs, c'est-à-dire les personnes qui ont été prédites

comme non-défaillants par le modèle mais qui étaient en fait des défaillants. Nous ne voulons pas passer à côté d'un défaillant comme étant classé comme non défaillant car le coût des erreurs pourrait être très élevé. Cependant, même si certains des non-défaillants sont classés comme défaillants, ils peuvent postuler à nouveau et demander une vérification de profil spéciale par des experts. De cette façon, nous réduisons le coût des erreurs commises par le modèle.

- ***Accuracy :*** L'accuracy permet de connaître la proportion de bonnes prédictions par rapport à toutes les prédictions. L'opération est simplement : Nombre de bonnes prédictions / Nombre total de prédictions.

$$\text{Accuracy} = \frac{\mathbf{tn + tp}}{\mathbf{tn + fp + fn + tp}} = \frac{\mathbf{10}}{\mathbf{11}} = \mathbf{0.9}$$

- ***Score ROC-AUC :*** cette métrique est insensible au déséquilibre des classes. Il fonctionne en classant les probabilités de prédiction de l'étiquette de classe positive et en calculant l'aire sous la courbe ROC qui est tracée entre **True Positive Rates** et **False Positive Rates** pour chaque valeur de seuil.

- ***Score de rappel*** *(Recall):* C'est le rapport entre le True Positives prédit par le modèle et le nombre total de Actual Positives (Vrais Positifs). Il est également connu sous le nom **True Positive Rate**.

- ***Score de précision*** *:* C'est le rapport **True Positives** et **le Total Positives** prédit par le modèle.

$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives} \qquad precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Formules de rappel et de score de précision

- ***F1-score : Le F1-Score combine subtilement la précision et le rappel. Il peut être plus*** intéressant que l'Accuracy car le nombre de vrais négatifs (TN) n'est pas pris en compte.

Dans les situations d'imbalanced class comme c'est le cas ici, nous avons une majorité de vrais négatifs qui faussent complètement notre perception de la performance de l'algorithme. Un grand nombre de vrais négatifs (TN) laissera le F1-Score de marbre.

$$\text{F1-Score} = 2\,\frac{\text{precision} * \text{recall}}{(\text{precision} + \text{recall})}$$

- ***En conclusion, nous privilégierons ici l'AUC et le F1-Score.***

6. <u>Fonction de coût</u> :

- S'agissant d'un problème de classification binaire, nous utiliserons la ***fonction de Cross-Entropy Loss*** (Log Loss).

- L'entropie croisée mesure la divergence entre deux distributions de probabilité, si l'entropie croisée est grande, cela signifie que la différence entre deux distributions est grande, tandis que si l'entropie croisée est petite, cela signifie que les deux distributions sont similaires.

- Dans le cas d'une classification binaire, chaque probabilité prédite est comparée à la valeur de sortie de classe réelle (0 ou 1) et un score est calculé qui pénalise la probabilité en fonction de la distance par rapport à la valeur attendue.

- Formule :

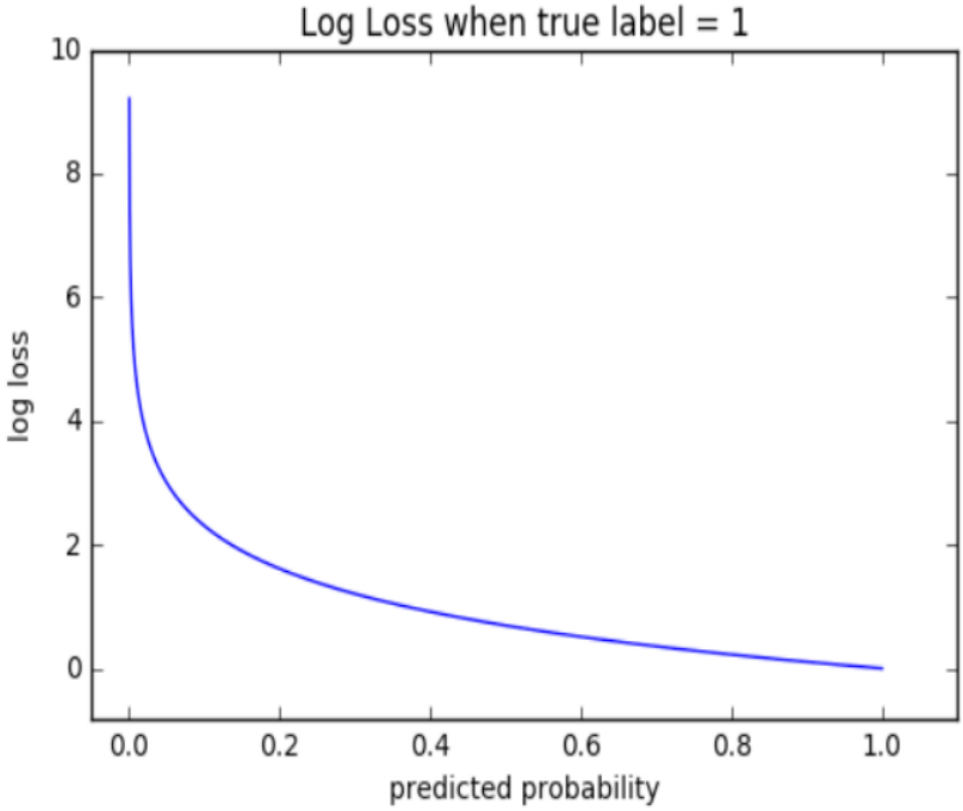
$$CE = -(y\log(p) + (1-y)\log(1-p))$$

  - y : un indicateur binaire (0 ou 1).

  - p : probabilité prédite par le modèle pour l'observation

- Le graphique ci-dessous montre la plage des valeurs de perte de log possibles en fonction d'une observation vraie (y = 1). À mesure que la probabilité prédite se rapproche de 1, la

perte logarithmique diminue lentement. Cependant, à mesure que la probabilité prédite diminue, la perte de log augmente rapidement.



- ***Fonction de coût personnalisée*** : On peut définir une fonction de coût en accordant des poids différents aux éléments de la matrice de confusion. Nous avons ainsi affecté un ***poids de - 10*** à chaque dossier prédit négatif mais réellement positif (***FN***) et un ***poids de +100*** aux dossiers négatifs identifiés comme tels (***TN***). Les poids des dossiers FP et TP sont nuls.

  Le gain résultant pour une valeur donnée du paramètre seuil de classification est alors défini par la fonction d'évaluation du gain suivante : Gain = TP * (0) + TN * (100) + FP * (0) + FN * (-10) L'optimisation métier du classifieur consiste à maximiser le Gain (normalisé pour la circonstance) en faisant varier le paramètre seuil de classification mais également les paramètres du classifieur.

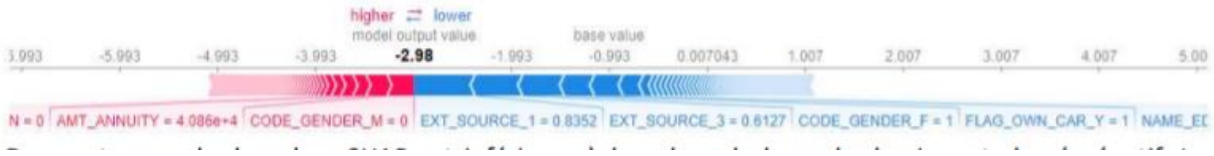
7. Interprétation du modèle :

- **Importance des variables :**

  - Les modèles testés offrent tous une méthode permettant ***d'extraire les variables ayant le plus d'influence sur la classification***. Un score leur est affecté à cet effet. Nous avons utilisé cette fonctionnalité pour la ***sélection de variables dans le cadre de l'optimisation du modèle par réduction de dimensions*** (méthode RFECV = Recursive Feature Elimination with Cross-Validation). Outre les gains en temps de calculs obtenus, cette approche permet également de disposer d'une information relative à l'importance globale des variables dans le modèle, qui peut être débattue et validée avec les experts métier.

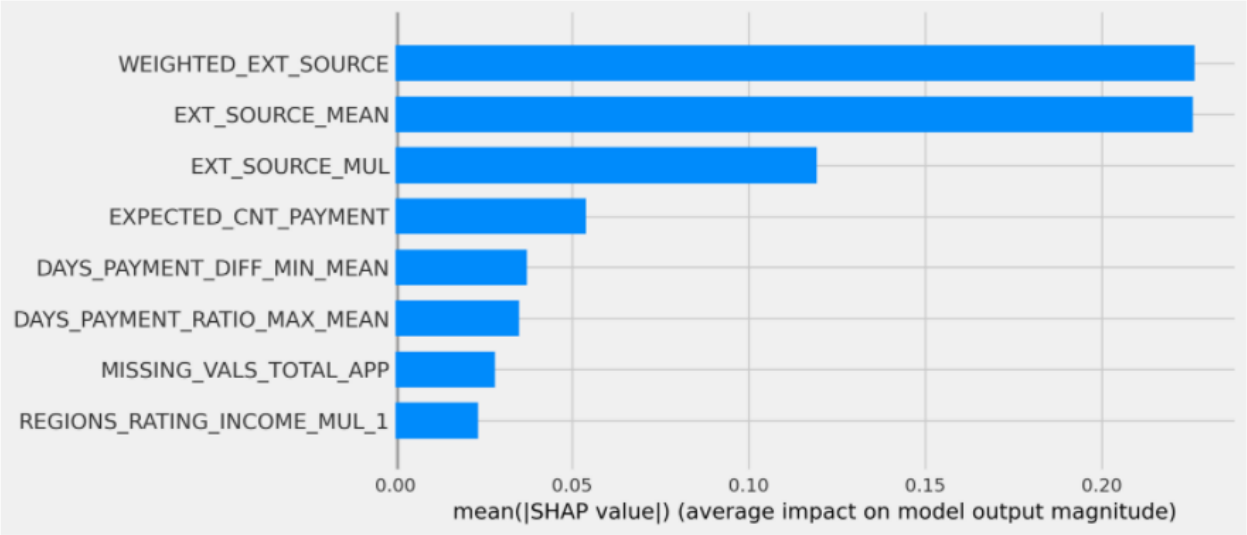
- **Implémentation de la méthode SHAP (SHapley Additive exPlanations)** :

  - ***La méthode SHAP consiste à calculer la valeur de Shapley pour toutes les variables de tous les individus c'est-à-dire la moyenne de l'impact d'une variable*** (sur la sortie, donc la prédiction) pour toutes les combinaisons de variables possibles. La somme des effets de chaque variable explique alors la prédiction. Le graphe ci-dessous représente les impacts des valeurs des variables importantes sur la prédiction. Les variables en rouge contribuent le plus à une prédiction positive (dossier non remboursé), les variables en bleu contribuent le plus à une prédiction négative (dossier remboursé).



Dans cet exemple, la valeur SHAP est inférieure à la valeur de base, le dossier est classé négatif. Les variables en bleu expliquent cette décision.

  - ***SHAP permet également une analyse globale***. En effet, en moyennant les valeurs absolues des valeurs de chaque variable (niveau local), on remonte à l'importance globale des variables.



8. Limites et améliorations possibles :

- ***Sélection des variables*** : Les informations disponibles relatives à l'importance des variables sont débattues avec les experts métier en vue de définir les stratégies techniques à tester dans les différents blocs concernés :
  - Valeurs manquantes
  - Corrélations entre variables
  - Seuil de variance
  - Réduction de dimensions (RFE)

- ***Equilibrage des données (SMOTE)*** : L'équilibrage des données introduit des données artificielles donc la possibilité d'incohérences. Des tests peuvent être réalisés en variant certains paramètres (ratios classes).

- ***Fonction d'évaluation du gain :*** Les règles métier et les critères financiers relatifs aux pertes et profits doivent être communiqués en vue d'établir une fonction d'évaluation du gain adaptée (fonction de coût personnalisée optimisée).

- ***Optimisation des hyper-paramètres :*** Nous avons testé plusieurs classifieurs et retenu XGBOOST pour ses meilleures performances, bien qu'il soit plus lent à exécuter que

Lightgbm. D'autres classifieurs comme Lightgbm ou Linear Discriminant Analysis peuvent potentiellement apporter de meilleures performances techniques et un gain de temps appréciable par rapport à XGBOOST. Il s'agit de les tester dans le pipeline complet si les contraintes de temps en production le permettent. Hyperopt et Optuna impliquent le réglage de nombreux hyper-paramètres à optimiser, en particulier les gammes de valeurs des paramètres du classifieur et le nombre d'évaluations.

- ***Interprétation :***
    - *La méthode SHAP offre des résultats intéressants*, en particulier pour l'interprétabilité locale. Par contre*, son implémentation dans une application web n'est pas toujours des plus aisées.*
    - *Concernant l'interprétabilité globale*, SHAP permet d'identifier les variables influentes. *D'autres approches comme les permutations de variables ou l*es *algorithmes basés sur les forêts aléatoires* peuvent être testées pour valider les résultats.
    - *S'agissant de la similarité des dossiers*, nous avons retenus des critères fondés sur l'âge, le sexe, le statut familial, le revenu, le montant du crédit qui nous ont permis de créer des masques de sélection. On pourra néanmoins optimiser la fonction de similarité basée sur la méthode des plus proches voisins.