

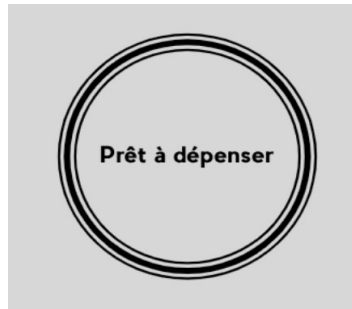
# Note Méthodologique

## Table of Contents

1. Présentation.....	2
2. Analyse exploratoire et prétraitement.....	2
2.1. Les données.....	2
2.2. Analyse Exploratoire.....	3
2.3. Prétraitement.....	3
3. La méthodologie d'entraînement du modèle.....	3
3.1. Division du données :.....	3
3.2. Remplacements le données nulles :.....	4
3.3. Standardisation du données ou Mise en échelle.....	4
3.4. Choix des modèles.....	4
3.5. Score d'évaluation des modèles.....	4
3.6. Déséquilibre de données.....	5
3.7. Modèle sélectionné.....	5
4. La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation.....	6
4.1. Coût métier.....	6
4.2. Score Bank (métrique d'évaluation).....	6
4.3. Seuil de solvabilité.....	7
4.4. Optimisation des hyperparamètres et du seuil de solvabilité.....	7
5. L'interprétabilité globale et locale du modèle.....	7
5.1. Interprétation globale.....	8
5.2. Interprétation locale.....	8
6. Les limites et les améliorations possibles.....	9
7. Les liens vers "Github", "API" et "Dashboard".....	9

# 1.Présentation

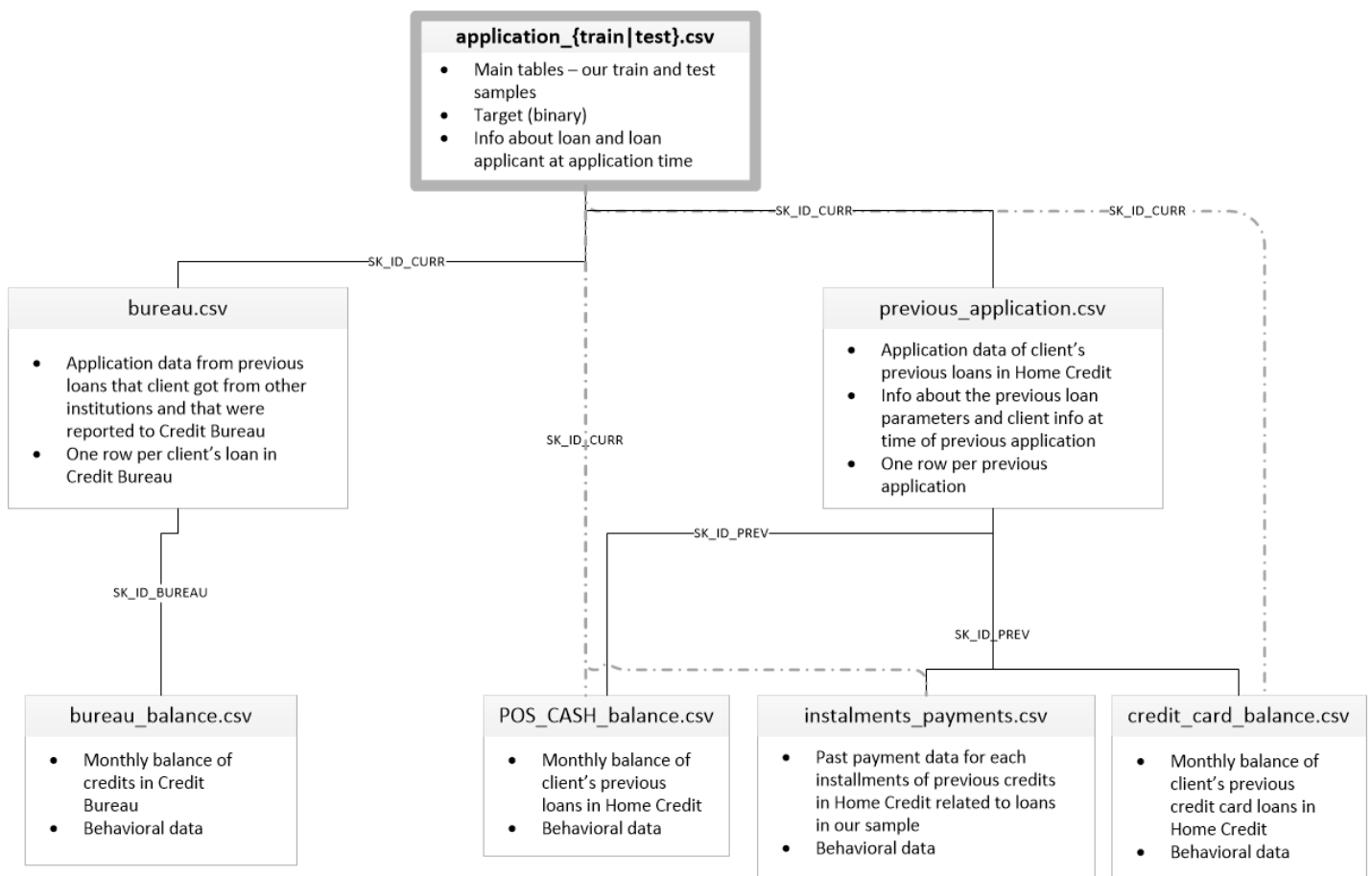
La société financière, "**Prêt à dépenser**", qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.



L'entreprise souhaite **mettre en œuvre un outil de "scoring crédit"** pour calculer la probabilité qu'un client rembourse son crédit, puis classifie la demande en crédit accordé ou refusé. Elle souhaite donc développer un **algorithme de classification** en s'appuyant sur des sources de données variées (données comportementales, données provenant d'autres institutions financières, etc.).

## 2.Analyse exploratoire et prétraitement

### 2.1.Les données



Les données viennent d'une compétition "**kaggle**", on peut diviser les données en trois grandes parties :

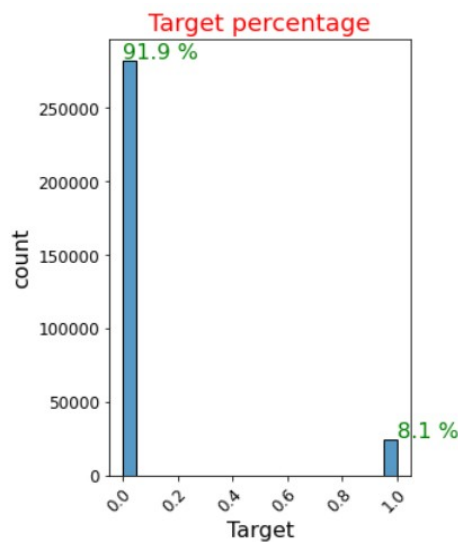
- liste des clients avec leurs demandes actuelles au sein de l'entreprise "prêt à dépenser"
- les anciens demandes et prêt au sein de l'entreprise "prêt à dépenser", avec des bilans comportementales
- les autres demandes et prêts auprès d'autres établissements financières, avec des bilans comportementales

la liste des clients est divisé en deux parties :

- une avec un décision, accordé ou non accordé, sur le prêt et l'emprunteur qu'on va l'utiliser pour le développement de notre modèle de classification,
- une deuxième partie sans décision qu'on va l'utiliser pour la présentation de notre Dashboard.

## 2.2.Analyse Exploratoire

Le point les plus important qui a été constaté par l'analyse exploratoire c'est qu'il y a un déséquilibre entre les clients qui ont été accordé le crédit et non. (91,9 % accordé 'valeur 0', contre 8,9 % non accordé 'valeur 1').



## 2.3.Prétraitement

J'ai adapté un kernel de chez kaggle(source :<https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features>) pour faire le prétraitement de données :

- les majorités de feature engineering (création des variables) est basé sur l'aggrégation (min, max, moyenne, somme), création de pourcentages et de proportions.
- Le données catégories ont été transformé en numériques via OneHoteEncoder
- les valeurs numériques 365243 (équivalent à 100 ans ont été considéré comme abérants) on les a transformé en NAN qui vont être remplacer par la moyenne avant standardisation et modélisation L

## 3.La méthodologie d'entraînement du modèle

### 3.1.Division du données :

J'ai divisé le données (avec Target) en deux parties : une pour l'entrainement de modèle (70%) et deuxième pour la validation(30%)

### 3.2. Remplacements le données nulles :

Pour chaque variable (Feature), les valeurs NaN ont été remplacés par la moyenne des valeurs non nulles de tous les individus (observations, lignes) de la même variable.

### 3.3. Standardisation du données ou Mise en échelle

Pour que l'algorithme traite tous les variables de la même ordre nous avons fait une mise à échelle entre 0 et 1, pour ne pas favoriser un variable sur un autre.

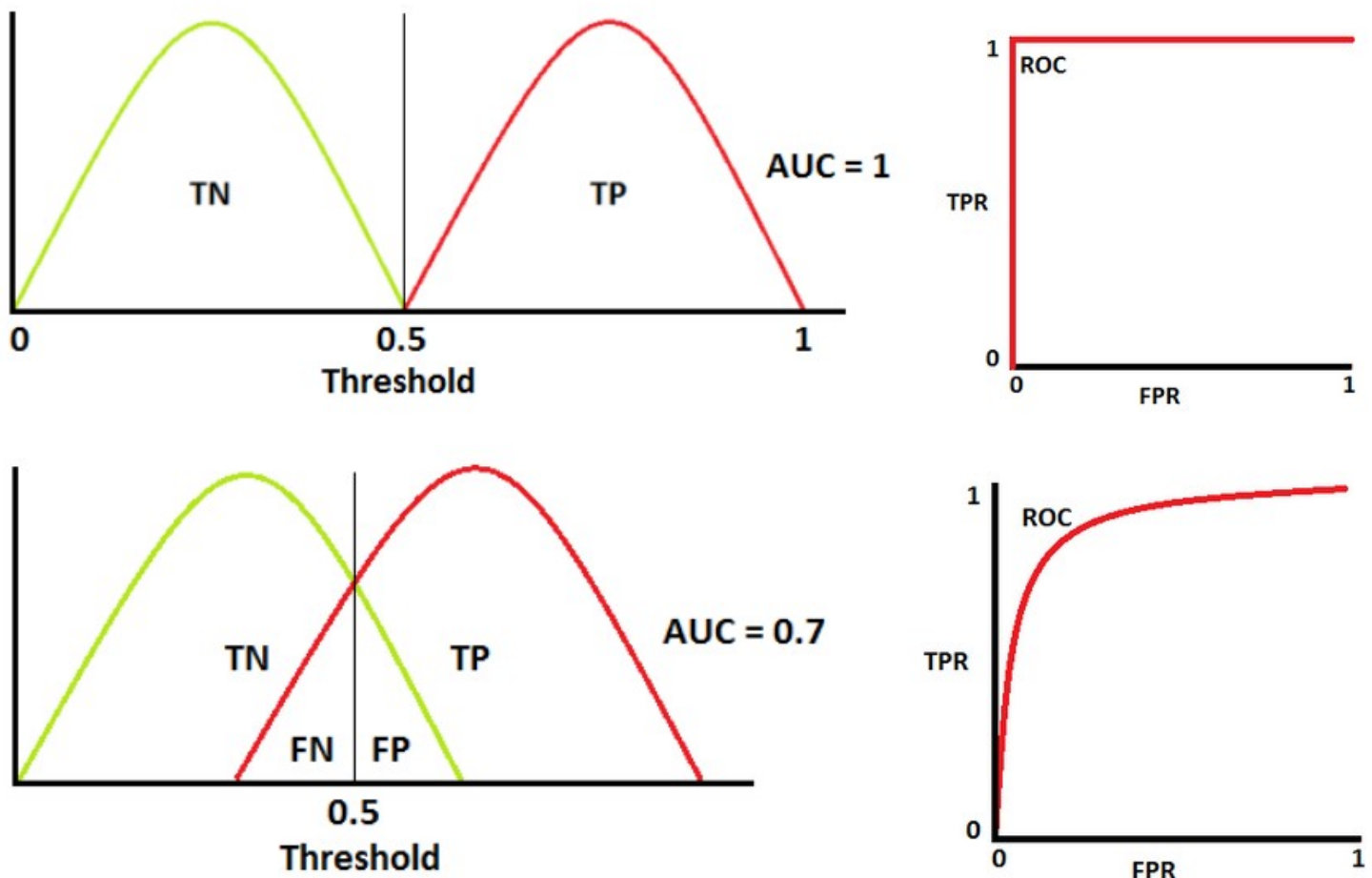
### 3.4. Choix des modèles

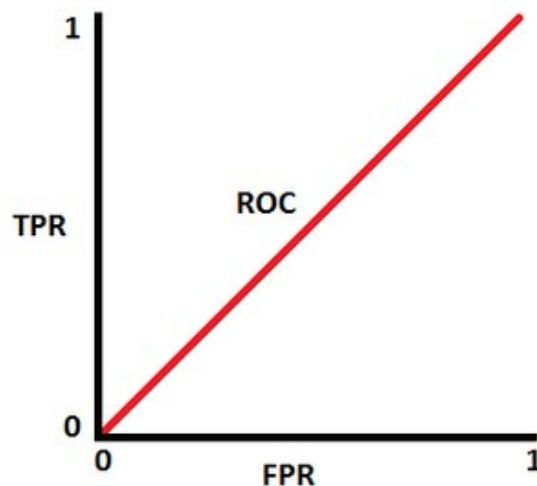
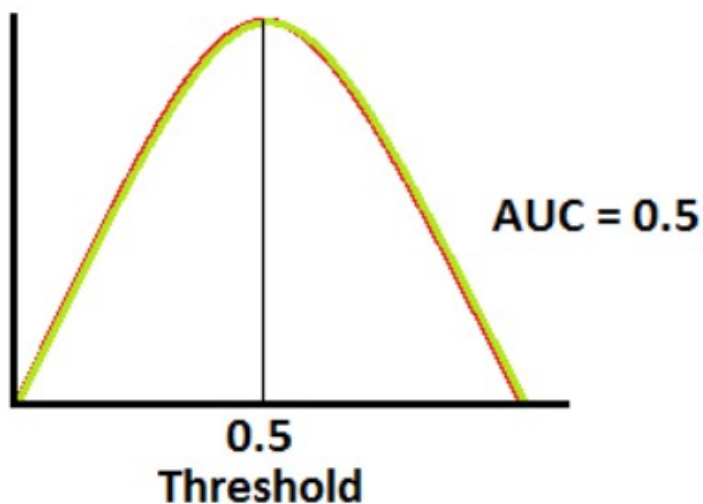
La modélisation du **"scoring crédit"** pour calculer la **probabilité** qu'un client rembourse son crédit, puis classifie la demande en crédit accordé ou refusé. C'est un problème de classification binaire. Dans notre cas c'est un problème de classification en apprentissage supervisé. Plusieurs familles d'algorithmes existent en Machine Learning pour traite cette problématique, la famille des algorithmes de Gradient Boosting ont montrés leurs efficacités dans la littérature, rapport accuracy/time élevé, nous avons opté pour tester 3 algorithmes de cette famille : XGBoost, CatBoost et LGBM. L'algorithme de Regression Logistic, a été considéré comme un modèle Baseline pour l'étude.

### 3.5. Score d'évaluation des modèles

Le roc\_auc\_score a été choisi pour l'évaluation des modèles. le AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve c'est l'un de meilleur métrique d'évaluation pour la performance des modèles de classification.

ROC c'est un courbe de probabilité et AUC représente la mesure de la séparabilité. Il exprime la capacité de modèle de séparer les classes. Plus que AUC est élevé, mieux la capacité de modèle de prédire le classe 0 en tant que 0 et le classe 1 en tant que 1.





### 3.6.Déséquilibre de données

Comme indiqué en haut il y a un important déséquilibre dans les données entre les clients solvables (Target=0) et les clients non solvables (Target=1). Ce déséquilibre aura un impact sur la performance de modèle et les prédictions seront faussées car l'algorithme attribuera beaucoup plus fréquemment la classe de Target clients la plus présente. Pour remédier à ce problème trois approches ont été testées :

- **ClassWeight Balanced** est une méthode directement gérée par les modèles et qui permet de pénaliser la classe sous représentée (Target=1 – clients non solvable)
- **UnderSampling** est une méthode qui va enlever de façon aléatoire des observations de classe sous représentée pour que les deux classes aient les mêmes nombres d'observations.
- **SMOTE** est une méthode qui va augmenter les nombres d'observations de classe sous représentée en créant d'autres observations de celles qui existent pour que les deux classes aient les mêmes nombres d'observations

### 3.7.Modèle sélectionné

Nous avons entraîné les quatre modèles choisis avec les trois approches adoptées ci-dessus pour contrecarrer le déséquilibre des données. Dans notre choix nous avons pris en compte le score AUC et le temps d'entraînement. Le modèle LGBMClassifier avec l'approche Class Weight Balanced a présenté le compromis le plus intéressant score/temps entre les trois modèles de gradient boosting.

	Model	AUC	Accuracy	Precision	Recall	F1	Bank	Time
1	LGBMClassifier	0.780693	0.731109	0.185389	0.68745	0.292026	0.713867	20.68185
3	CatBoostClassifier	0.779134	0.788668	0.213082	0.601451	0.314679	0.714731	73.507263
0	LogisticRegression	0.729228	0.673984	0.152063	0.664606	0.247498	0.67028	6.749535
2	XGBClassifier	0.724876	0.37756	0.103182	0.873152	0.184555	0.573282	94.959764

L'optimisation des hyperparamètres de ce modèle va être faite après la présentation de coût métier et métrique d'évaluation.

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

### 4.1. Coût métier

La banque cherche à minimiser sa perte en cherchant les personnes qui ne peuvent pas rembourser leur prêt et lorsque ce but est atteint l'évènement est positif. Le défaut de remboursement sera donc considéré comme l'évènement positif. Positif= 1.

La matrice de confusion entre classes prédites et classes réelles se représente comme ainsi :

		Predicted value	
		0 (Negative)	1 (Positive)
Actual value	0 (Negative)	TN	FP
	1 (Positive)	FN	TP

- **TN (True Negatives)** : Le prêt est accordé et le client peut rembourser le prêt.
- **TP (True Positives)** : Le prêt n'est pas accordé et le client ne peut pas rembourser le prêt.
- **FP (False Positive)** : Le prêt n'est pas accordé mais le client peut rembourser le prêt.
- **FN (False Negative)** : Le prêt est accordé mais le client ne peut pas rembourser le prêt.

Une banque cherche à ne pas accorder de prêt à des clients qui ne peuvent pas le rembourser. La banque ne veut pas accorder un prêt à un client qui ne peut pas le rembourser (FN à minimiser). La banque fait un déficit si elle n'accorde pas de prêt à des clients qui peuvent rembourser le prêt (FP à minimiser).

### 4.2. Score Bank (métrique d'évaluation)

On va créer un score en pondérant les différents cas possibles en pénalisant négativement les prédictions néfastes pour le banque comme FN et FN et valorisant positivement les prédictions bénéfiques pour le banque:

- $\text{gain\_total} = \text{COEF\_TN} \times \text{TN} + \text{COEF\_FP} \times \text{FP} + \text{COEF\_FN} \times \text{FN} + \text{COEF\_TP} \times \text{TP}$
- $\text{gain\_max} = (\text{TN} + \text{FP}) \times \text{COEF\_TN} + (\text{TP} + \text{FN}) \times \text{COEF\_TP}$
- $\text{gain\_min} = (\text{TN} + \text{FP}) \times \text{COEF\_FP} + (\text{TP} + \text{FN}) \times \text{COEF\_FN}$
- $\text{score} = (\text{gain\_total} - \text{gain\_min}) / (\text{gain\_max} - \text{gain\_min})$

coefficients données arbitrairement en respectant la métrique métier à valider par retour d'expériences :

- FN = perte d'argent pour la banque ==>  $\text{COEF\_FN} = -100$
- TP = refus de prêt ==>  $\text{COEF\_TP} = 0$
- TN = prêt accordé, gain d'argent pour la banque ==>  $\text{COEF\_TN} = +10$
- FP = client perdu, moins du gain d'argent pour la banque ==>  $\text{COEF\_FP} = -1$

### 4.3. Seuil de solvabilité

Le modèle retourne un score entre 0 et 1 et par défaut il attribue la classe 1 lorsque le score est supérieur à 0.5 et 0 sinon. Il a été décidé d'optimiser ce seuil qui permet de définir si un client est solvable ou non. Le seuil optimal sera déterminé par la méthode « hyperopt » pendant l'optimisation des hyperparamètres du modèle, i.e quand le score retourné par le modèle est supérieur au seuil optimal, le client est dit non solvable.

### 4.4. Optimisation des hyperparamètres et du seuil de solvabilité

chaque modèle de Machine Learning présente différents paramètres à fixer. Suivant la combinaison des paramètres choisis pour le modèle, les résultats seront plus ou moins performants.

Pour déterminer quelle combinaison d'hyper paramètres donne les meilleurs résultats sur notre jeu de données, plusieurs algorithmes existent tel que GridSearch, RandomSearch mais nous utiliserons pour ce projet l'algorithme d'optimisation « hyperopt ».

L'algorithme « hyperopt » utilise une approche Bayésienne pour déterminer les meilleurs paramètres d'une fonction. L'approche d'optimisation Bayésienne est un model probabilistique de type  $P(\text{score} | \text{configuration})$  qui se mis à jour itérativement dans le but de maximiser le score pour une configuration donnée.

HyperOpt nécessite 4 paramètres essentiels pour l'optimisation des hyperparamètres qui sont : l'espace de recherche (hyperparamètres et plages de recherche), la fonction de perte à optimiser de type (1-score), l'algorithme d'optimisation, une base de donnée pour conserver l'historique (Score| Configuration).

```
'n_estimators': hp.quniform('n_estimators', 100, 600, 100),
'learning_rate': hp.uniform('learning_rate', 0.001, 0.03),
'max_depth': hp.quniform('max_depth', 3, 15, 2),
'subsample': hp.uniform('subsample', 0.60, 0.95),
'colsample_bytree': hp.uniform('colsample_bytree', 0.60, 0.95),
'reg_lambda': hp.uniform('reg_lambda', 1, 20),
'reg_alpha': hp.uniform('reg_alpha', 1, 20),
'solvability_threshold': hp.quniform('solvability_threshold', 0.0, 1.0, 0.025)
```

Nous avons intégré l'optimisation de seuil de solvabilité en même temps que les hyperparametres. Comme le score, constituant la fonction de perte d'optimisation, et est dépend de la seuil de solvabilité.

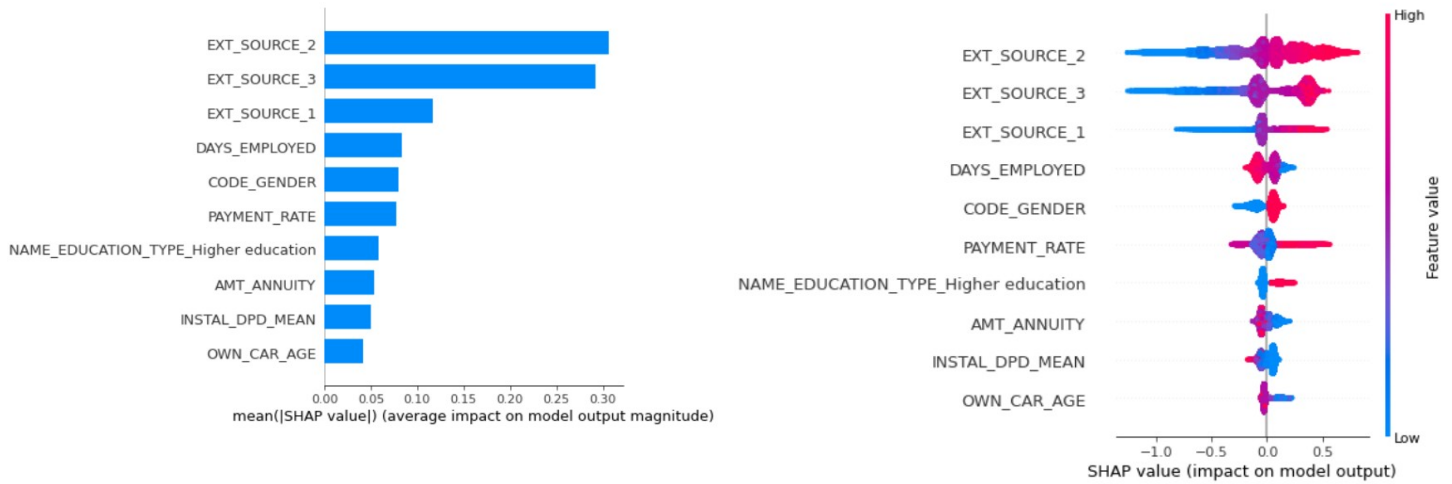
## 5. L'interprétabilité globale et locale du modèle

Les algorithmes Machine Learning de classifications sont très puissants avec des fois des centaines des hyperparamètres à réglés et peuvent gérer des milliers des variables pour faire sortir un probabilité sur la décision même une réponse binaire 0 ou 1. l'intrépretation de ces résultats et la part de responsabilité de chaque variable dans la décision finale reste flou, on a affaire à presque une boîte noire. Les algorithmes de ces modèles ont souvent la capacité de donnée une liste d'importance globale des variables selon un critère sur leur décision.

Pour ce projet nous avons décidé d'utilisé la librairie SHAP pour, **"SHapley Additive exPlanations"** qui utilise l'approche de théorie de jeu pour expliquer les prédictions des modèles.



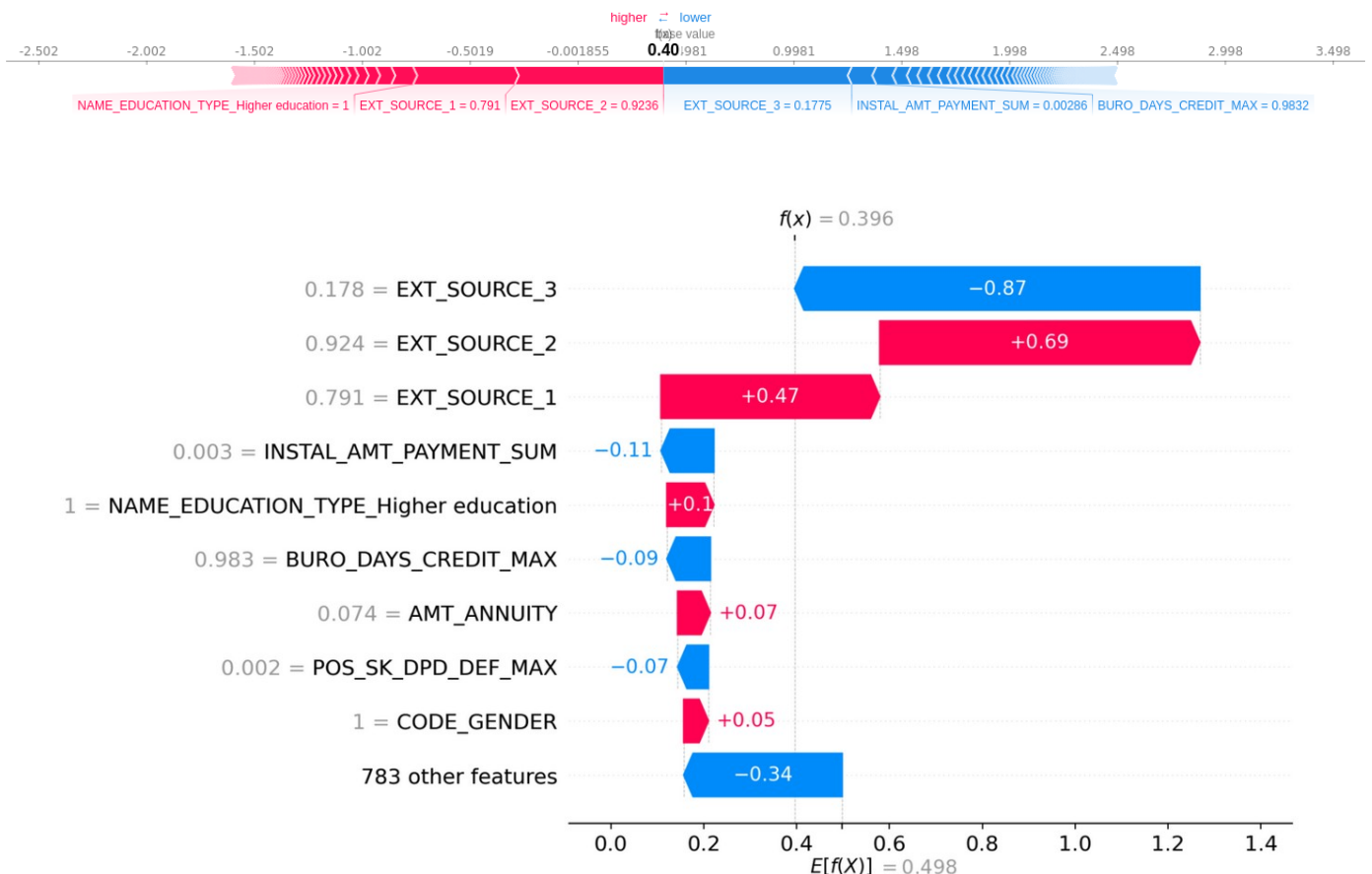
## 5.1. Interprétation globale



Les deux images ci dessus nous montrent l'importance globale (sur la totalité des individus de données) en ordre des features sur la décision du modèles. Les nombres de l'affichage c'est nous qu'on l'a décidé. La première à gauche nous montre simplement l'ordre des importances des features façon bar plot. La deuxième à droite elle nous indique en plus dans quelle sens ces features en reagit, les premiers trois montrent que plus que leur valeurs est importants ils réagissent dans le sens positif de décision, tandis que le quatrième dans le sens inverse. et ainsi de suite.

## 5.2. Interprétation locale

Le point qui est très intéressant dans la librairie de SHAP en plus d'une interprétation globale sur la totalité des individus comme vu en haut, il y a l'interprétation locale sur un individu.









Dans le même esprit les deux plots ci dessus montrent les importances des features et dans quel sens mais ici sur un seul individus de la population

## 6. Les limites et les améliorations possibles

Ce projet est à la carrefour de deux sciences /métiers très complexes La Finance et La Machine Learning (A.I.) avec beaucoup des lois mathématiques et beaucoup des facteurs métiers et techniques / technologiques les limites et les améliorations sont innombrables, nous arrêtons ici sur la citation de quelques idées :

- comme pour chaque projet de data scientist les nombres de features est très importants sur plusieurs niveaux : compréhension de phénomène, interprétation et présentation de modélisation et des résultats, efficacité d'algorithmes, temps de calculs, taille de données etc. Le points judicieux c'est de savoir combien et quoi utiliser comme features pour une modélisation efficace peut être sera intéressant de faire une sélection avant modélisation (ex. RFECV).
- Les Features Engineering ont été fait de façon très généraliste (min, max, moy, perc), peut être sans aucun lien avec le métier de Finance, des Features Engineering ciblés vers le métier de prêt bancaires sera plus judicieux.
- Les coefficients de pondération utilisés dans la création de score métier ont été choisi de façon arbitraire, avec une raisonnement logique de notre part, mais une expertise, correction et amélioration par des experts du métier est inévitable.
- De même le Dashboard est destinée pour une mission bien précise, communication des résultats et justifications pour un client via un employé de l'établissement financière. Les choix des données, la façon de les présenter, l'architecture de Dashboard tous à valider par les experts.
- Le modèle lui même LGBMClassifier présente des centaines des hyperparamètres dans un étape ultérieure sera nécessaire d'intégrer d'autres paramètres à l'optimisation du modèle.

## 7. Les liens vers "Github", "API" et "Dashboard".

- Pour accéder au dossier github\_api appuyez ici 
- Pour accéder au dossier github\_dashboard appuyez ici 
- Pour accéder à l'API appuyer ici 
- Pour accéder au Dashboard appuyer ici  Streamlit