

# Présentation du projet 7 : Implémentez un modèle de scoring



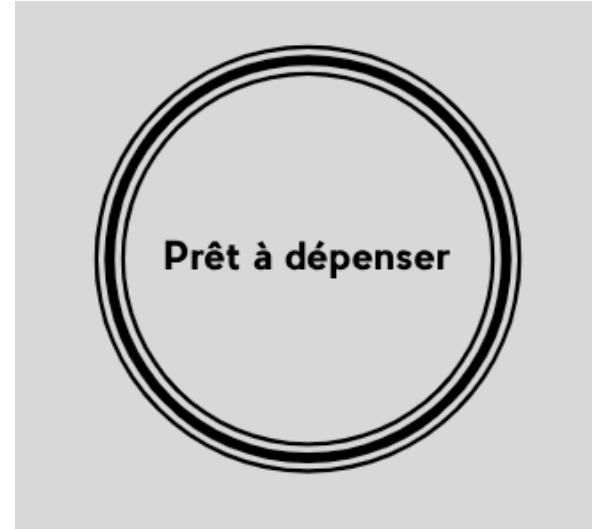
Préparé par :  
**MAGHLAZI Hanane**

# Projet 7 : Implémentez un modèle de scoring

- **Contexte**
- **Mission**
- **Exploration du jeu de données :**
  - ✓ Présentation des données
  - ✓ Description des données
  - ✓ Gestion des données manquantes
  - ✓ Feature engineering
- **Analyse exploratoires des données**
- **Modélisation:**
  - ✓ Tester différents modèles
  - ✓ Métriques
  - ✓ La fonction coût métier
  - ✓ Interprétabilité du modèle
- **Réalisation du Dashboard interactif :**
  - ✓ Réalisation du Dashboard en local
  - ✓ Mise en production de l'API
- **Conclusion**

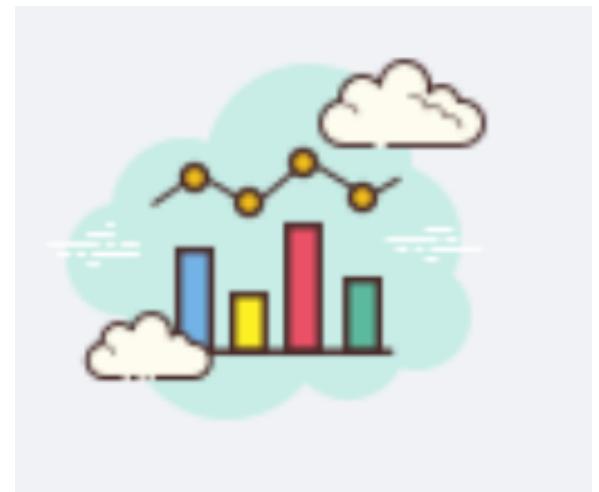
# Contexte :

- 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.
- Elle 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
- **Prêt à dépenser** décide donc de **développer un Dashboard interactif** pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d'octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

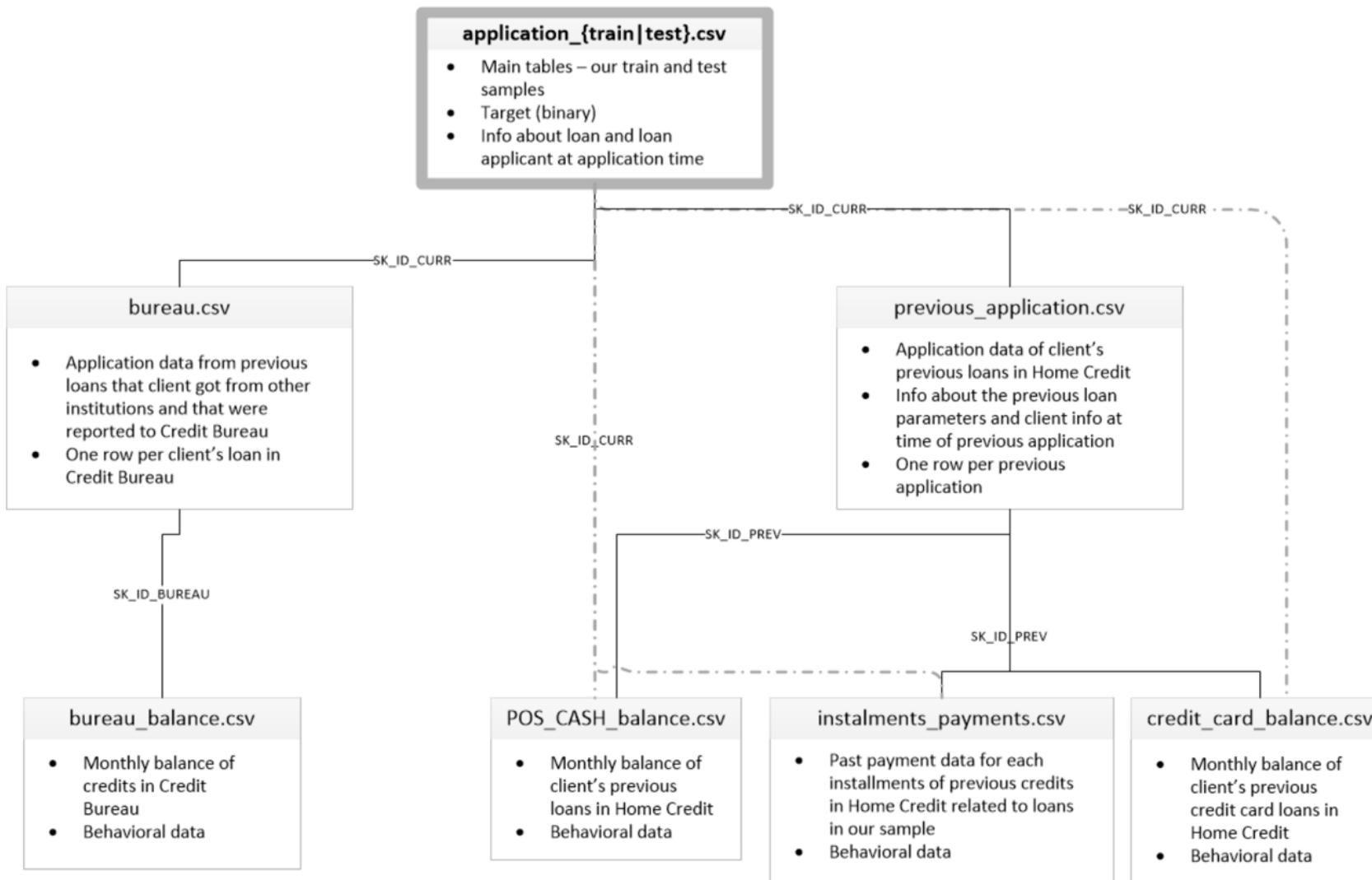


# Mission :

- Construire un **modèle** de **scoring** qui donnera une **prédition** sur la **probabilité** de **faillite** d'un client de façon automatique.
- Construire un **Dashboard** interactif à destination des gestionnaires de la relation client permettant **d'interpréter** les prédictions faites par le modèle, et d'améliorer la connaissance client des chargés de relation client.
- Mettre en **production** le modèle de scoring de prédition à l'aide d'une API, ainsi que le Dashboard interactif qui appelle **l'API** pour les prédictions



# Présentation des données :



# Description des données :

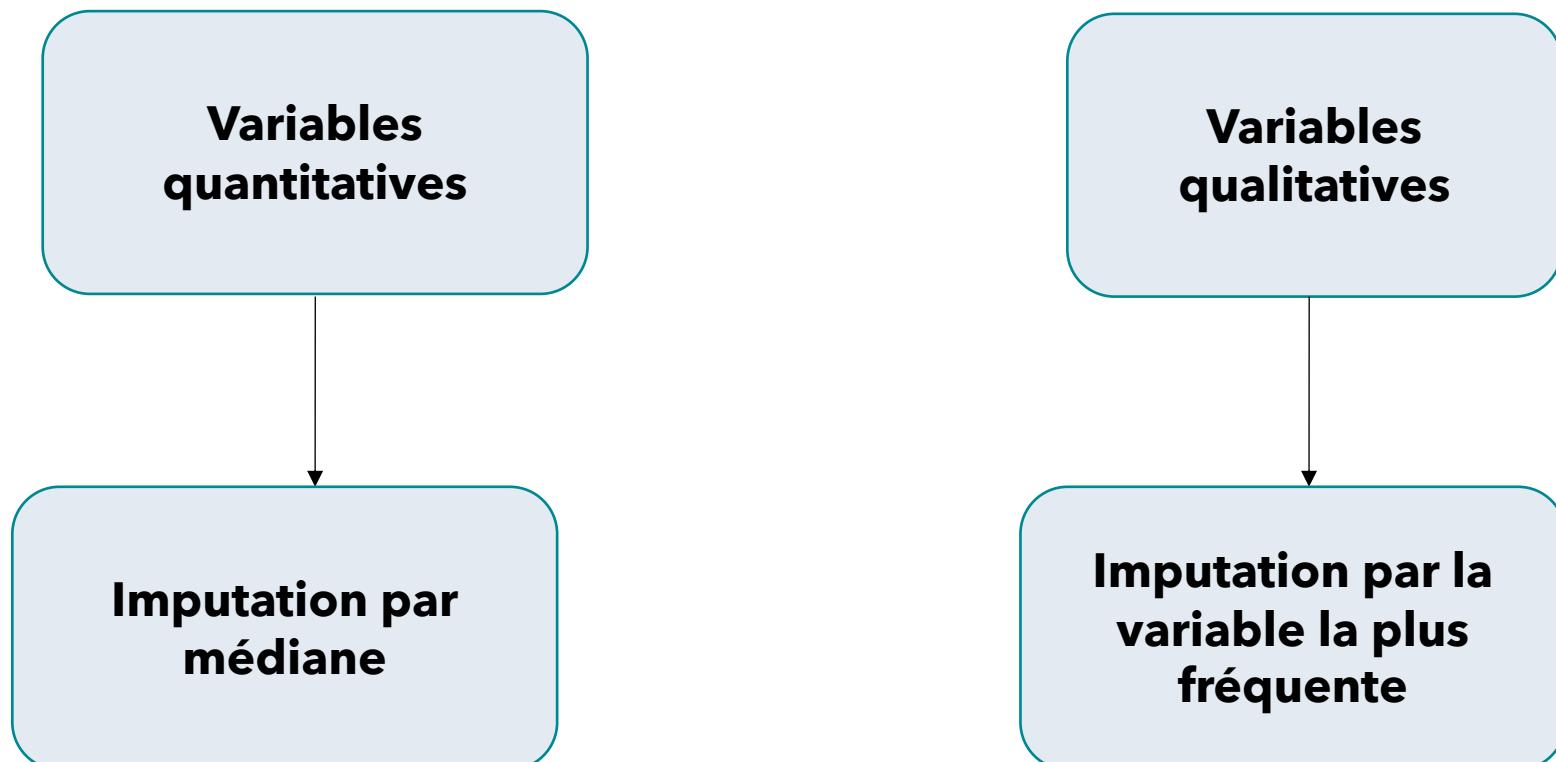
|                                | Dimensions    | NB_colonnes | NB_lignes | Total<br>remplissage | Description  |
|--------------------------------|---------------|-------------|-----------|----------------------|--|
| application_train              | (307511, 122) | 122         | 307511    | 28363877             | Données statiques pour toutes les applications. Une ligne représente un prêt dans notre échantillon de données avec TARGET   |
| application_test               | (48744, 121)  | 121         | 48744     | 4493605              | Données statiques pour toutes les applications. Une ligne représente un prêt dans notre échantillon de données sans TARGET   |
| bureau                         | (1716428, 17) | 17          | 1716428   | 25239329             | Tous les crédits antérieurs du client fournis par d'autres institutions financières qui ont été signalés au bureau de crédit (pour les clients qui ont un prêt dans notre échantillon) |
| bureau_balance                 | (27299925, 3) | 3           | 27299925  | 81899775             | Soldes mensuels des crédits antérieurs au bureau de crédit   |
| POS_CASH_balance               | (10001358, 8) | 8           | 10001358  | 79958706             | Aperçus mensuels du solde des points de vente précédents et des prêts en espèces que le demandeur avait avec Home Credit   |
| credit_card_balance            | (3840312, 23) | 23          | 3840312   | 82449820             | Aperçus mensuels du solde des cartes de crédit précédentes que le demandeur a avec Home Credit   |
| installments_payments          | (13605401, 8) | 8           | 13605401  | 108837398            | Historique de remboursement des crédits précédemment décaissés en Crédit Immobilier liés aux prêts de notre échantillon  |
| previous_application           | (1670214, 37) | 37          | 1670214   | 50688582             | Toutes les demandes précédentes de prêts Home Credit des clients qui ont des prêts dans notre échantillon  |
| HomeCredit_columns_description | (219, 5)      | 5           | 219       | 962                  | Ce fichier contient les descriptions des colonnes des différents fichiers de données   |

# Données manquantes :

|                                  | %NaN    | %Duplicate | object_dtype | float_dtype | int_dtype | bool_dtype |
|----------------------------------|---------|------------|--------------|-------------|-----------|------------|
| <b>application_test.csv</b>      | 23.8100 | 0.0000     | 16           | 65          | 40        | 0          |
| <b>POS_CASH_balance.csv</b>      | 0.0700  | 0.0000     | 1            | 2           | 5         | 0          |
| <b>credit_card_balance.csv</b>   | 6.6500  | 0.0000     | 1            | 15          | 7         | 0          |
| <b>installments_payments.csv</b> | 0.0100  | 0.0000     | 0            | 5           | 3         | 0          |
| <b>application_train.csv</b>     | 24.4000 | 0.0000     | 16           | 65          | 41        | 0          |
| <b>bureau.csv</b>                | 13.5000 | 0.0000     | 3            | 8           | 6         | 0          |
| <b>previous_application.csv</b>  | 17.9800 | 0.0000     | 16           | 15          | 6         | 0          |
| <b>bureau_balance.csv</b>        | 0.0000  | 0.0000     | 1            | 0           | 2         | 0          |
| <b>sample_submission.csv</b>     | 0.0000  | 0.0000     | 0            | 1           | 1         | 0          |

# Gestion des données manquantes :

```
1 # Retirer les colonnes de plus de 80% de nan  
2  
3 print("colonnes avant suppression", data.shape[1])  
4 data = data.loc[:, data.isnull().mean() < 0.8]  
5 print("colonnes après suppression", data.shape[1])
```



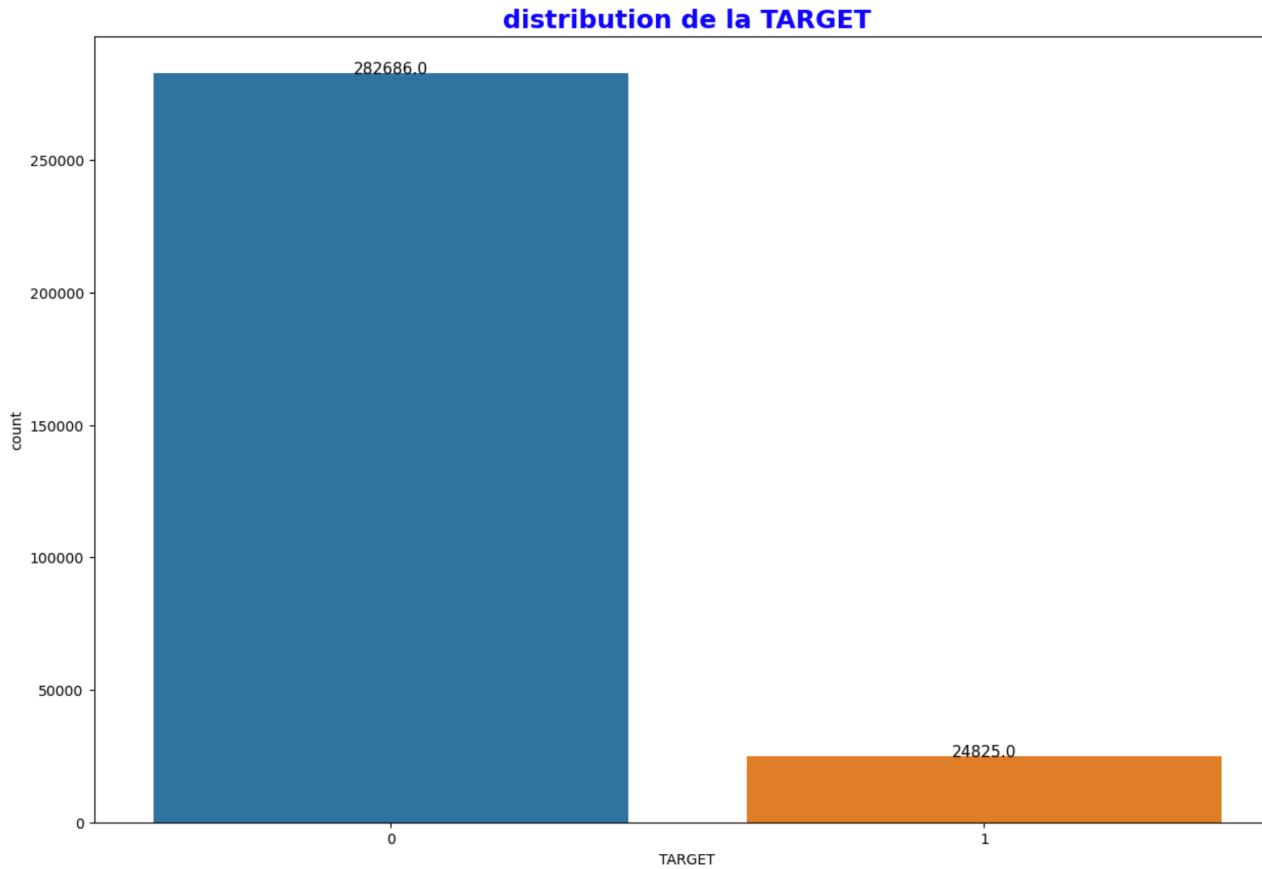
# Feature engineering :

- Dans le cadre de ce projet, nous utiliserons **un kernel Kaggle existant** pour faciliter la préparation des données nécessaires.
- Le kernel fourni permet la jointure des 7 fichiers, la création de nouvelles variables synthétiques ainsi que l'encodage des variables **qualitatives** par la méthode **get\_dummies**.

```
1 | data.shape
```

```
(356251, 798)
```

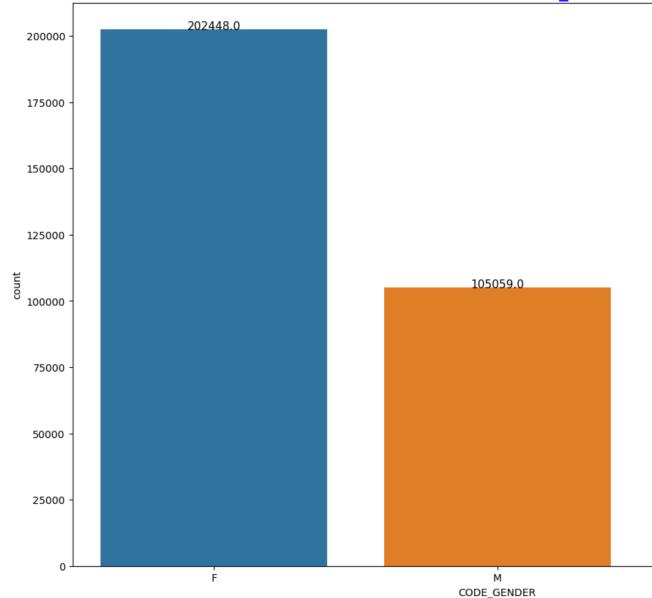
# Analyse exploratoire :



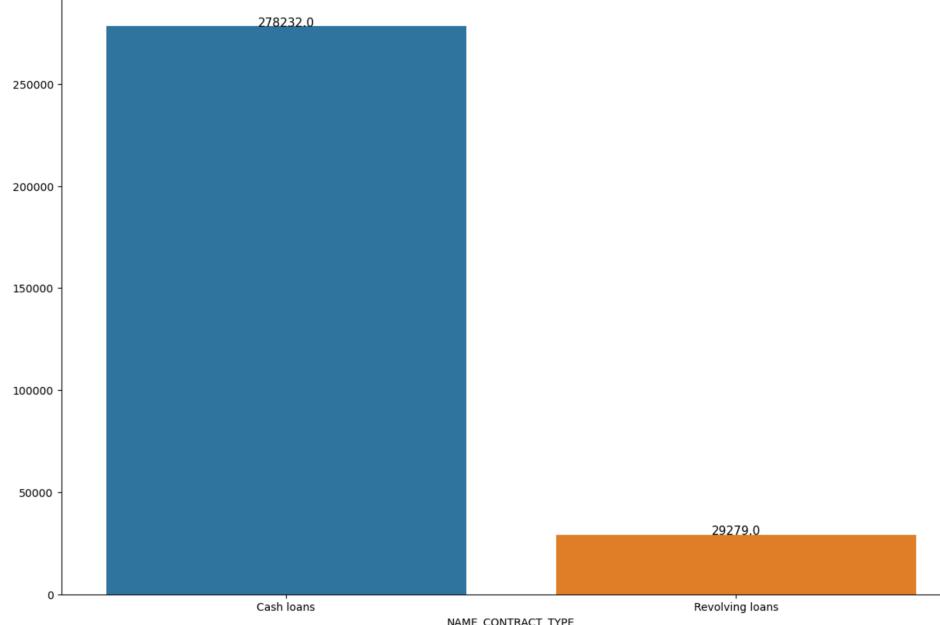
- La classe **0** représente les clients qui n'ont **pas de difficulté de paiement** et la classe **1** les clients ayant des **difficultés de paiement**.
- On remarque un **déséquilibre** entre les deux classes .
- La classe 0 représente **92%** des données et la classe 1 que **8%**

# Analyse exploratoire :

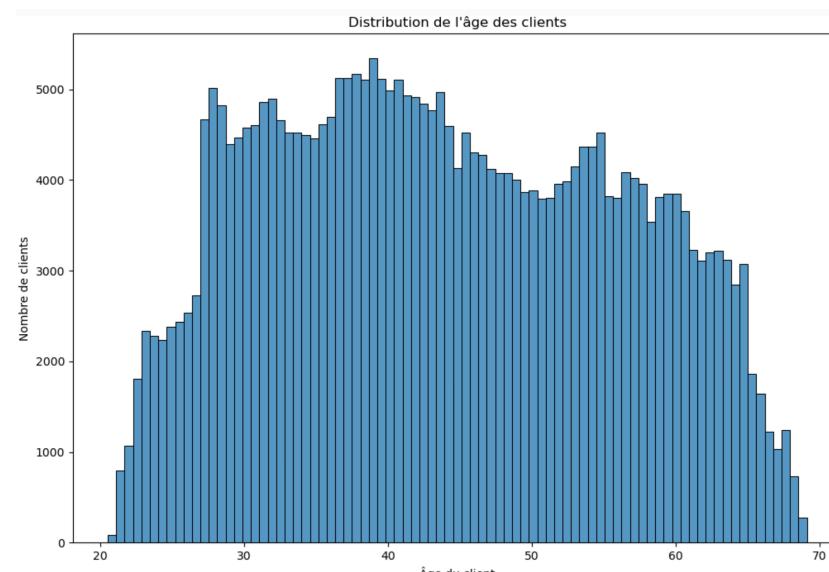
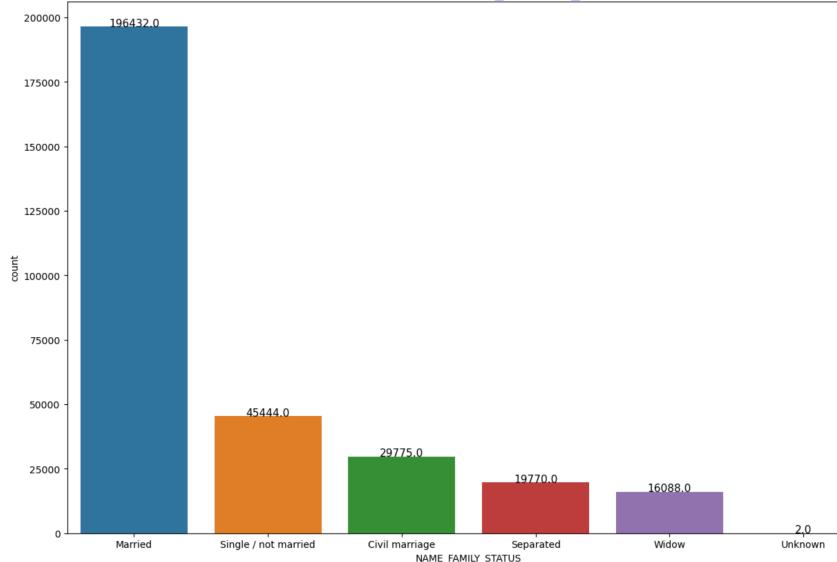
distribution du CODE\_GENDER



distribution du NAME\_CONTRACT\_TYPE



Distribution du NAME\_FAMILY\_STATUS



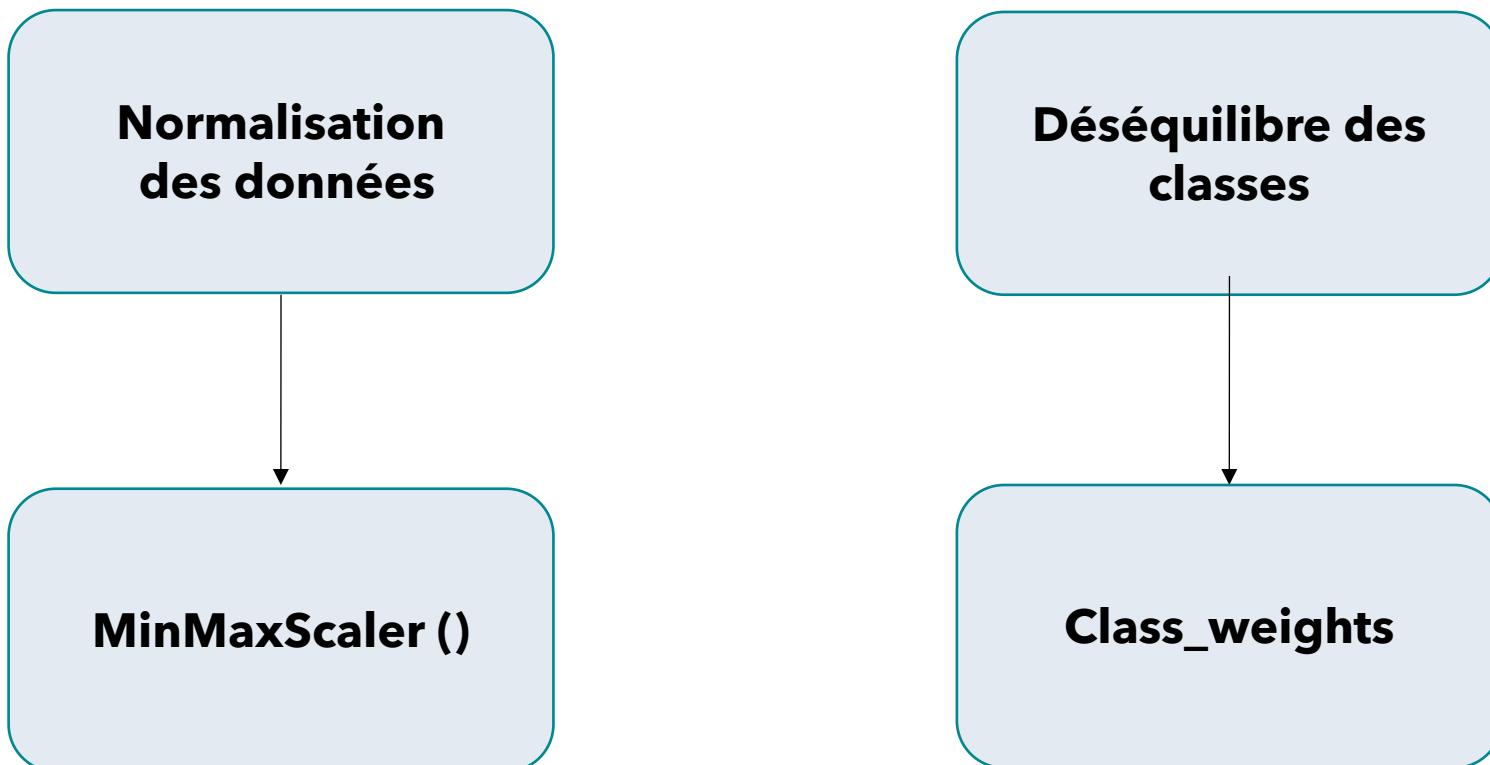
# Modélisation :

## Préparation des données :

- Avant la modélisation, il est nécessaire de :
  - ✓ **Normaliser** les données
  - Résoudre le problème de **déséquilibre** des classes : Il existe plusieurs techniques pour résoudre ce problème : class\_weights, SMOTE, Borderline-SMOTE, Adaptive Synthetic Sampling (ADASYN), SVM-SMOTE, ...
  - ✓ Pour ce projet , voici les techniques testées : SMOTE, SMOTE-NC et class\_weights.

# Modélisation :

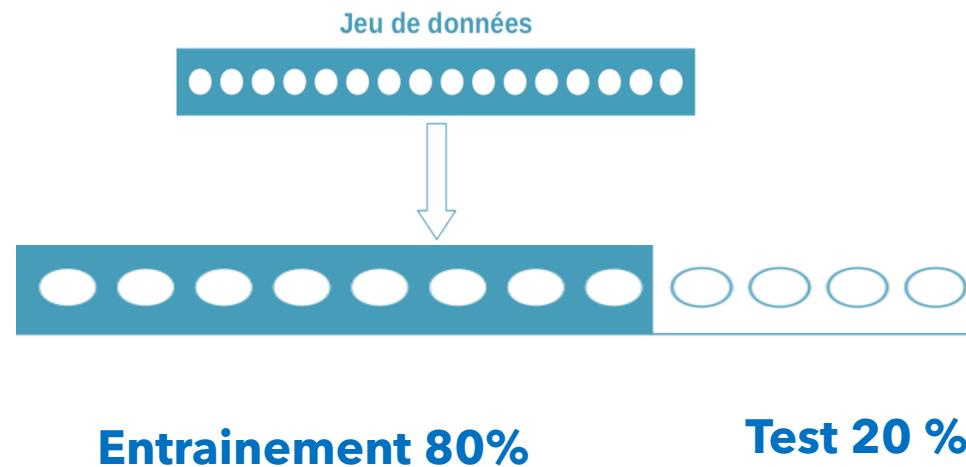
- **Préparation des données :**



*Attribuer des poids différents aux classes majoritaires et minoritaires. La différence de poids influencera le classement des classes lors de la phase d'entraînement. Le but est de fixer un poids de classe plus élevé et en même temps en réduisant le poids de la classe majoritaire.*

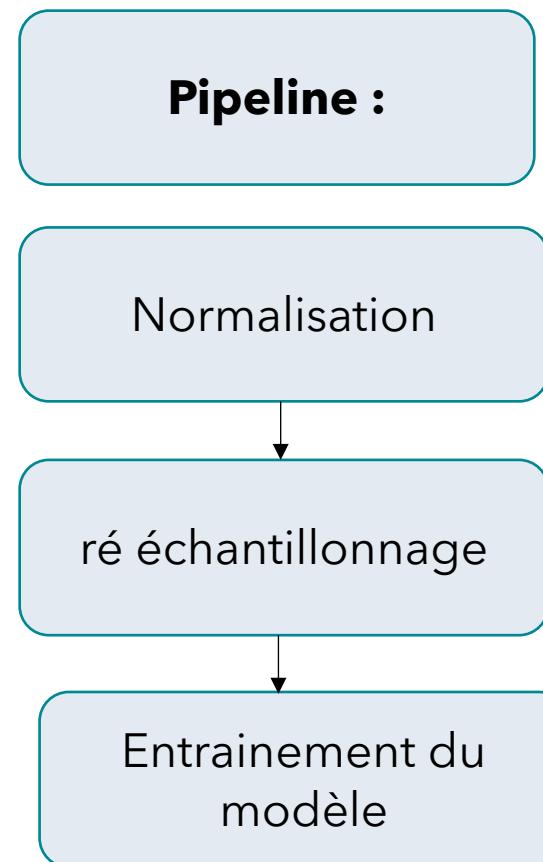
# Modélisation :

- Il existe plusieurs algorithmes de machine Learning, pour ce projet trois algorithmes de gradient boosting ont été utilisés : **GradientboostClassifier**, **CatboostClassifier** et le **XgboostClassifier** en raison de leur précision, rapidité, en particulier pour les données complexes et volumineuses .
- La première étape dans le cadre de la modélisation est de couper notre jeu de données en deux parties : un jeu **d'entraînement**, et un jeu de **test**



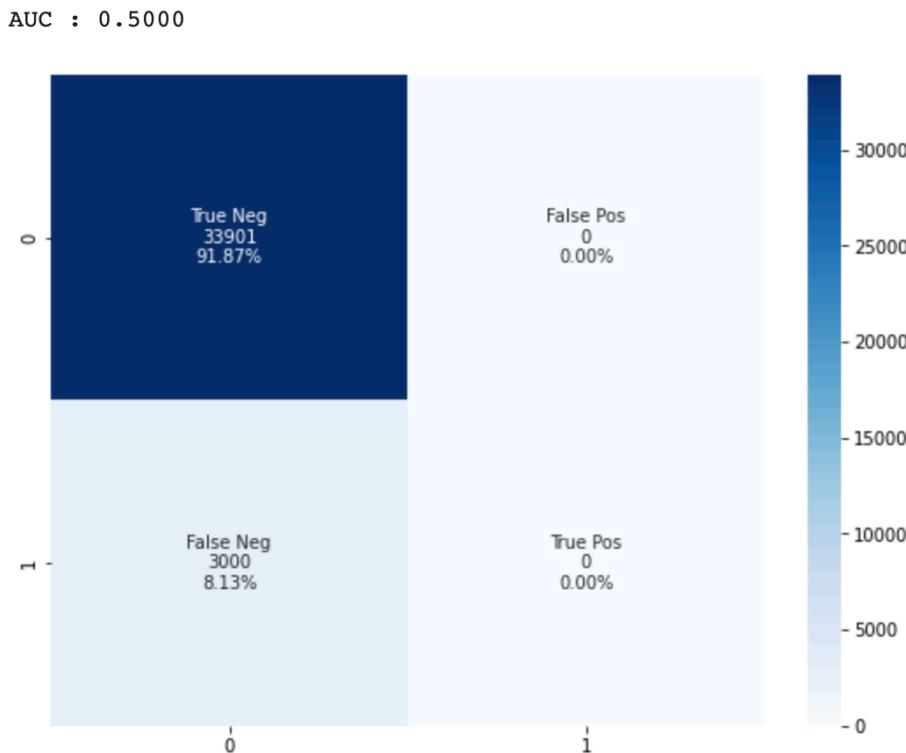
# Modélisation :

- La construction des modèles cités avant a été faite en passant par une pipeline qui intègre : la normalisation et le ré échantillonnage.



# Modélisation :

- Afin de mieux comprendre puis d'améliorer les performances de notre modèle, nous devons d'abord établir une **base** de référence pour les données dont nous disposons, le **DummyClassifier** sera notre Baseline.



*C'est un modèle de classificateur qui fait des prédictions sans essayer de trouver des modèles dans les données. Le modèle par défaut examine essentiellement quelle étiquette est la plus fréquente dans l'ensemble de données d'apprentissage et effectue des prédictions basées sur cette étiquette*

Le modèle a correctement prédit la classe 0 (TN, quadrant supérieur gauche) 92 % du temps, mais les prédictions pour la classe 1 (TP, quadrant inférieur droit) sont à 0 %

# Modélisation :

- Métriques :

Les métriques qui nous intéressent sont la **précision** et le **recall** qui se basent sur la **matrice de confusion** qui est indispensable pour définir les différentes métriques de classification

➤ La matrice de confusion : C'est un tableau à 4 valeurs représentant les différentes combinaisons de valeurs réelles et valeurs prédites comme dans la figure ci-dessous :

| Confusion matrix |              | Reality             |                     |
|------------------|--------------|---------------------|---------------------|
|                  |              | Negative : 0        | Positive : 1        |
| Prediction       | Negative : 0 | True Negative : TN  | False Negative : FN |
|                  | Positive : 1 | False Positive : FP | True Positive : TP  |

# Modélisation :

Voici une explication de ce que représente ces éléments dans notre cas :

**TP** : Le client a des difficultés de paiement et le modèle a prédit qu'il a des difficultés de paiement.

**TN** : Le client n'a pas de difficultés de paiement et le modèle a prédit qu'il n'a pas de difficultés de paiement.

**FP** : Le client n'a pas de difficultés de paiement et le modèle a prédit qu'il a des difficultés de paiement. (**Erreur de type 1**)

**FN** : Le client a des difficultés de paiement et le modèle a prédit qu'il n'a pas de difficultés de paiement. (**Erreur de type 2**)

# Modélisation :

- **La précision** : indique quelle fraction des positifs prédicts est réellement positive.

$$Precision = \frac{TP}{TP + FP}$$

*De tous les cas que nous avons qualifiés de classe 1 (dans notre exemple, de tous les cas prédicts comme client avec défaut de paiement) combien nous avons eu raison (c'est-à-dire combien ils étaient vraiment des clients qui ne remboursent pas).*

- **Le recall**: mesure la capacité du modèle à prédire les vrais positifs.

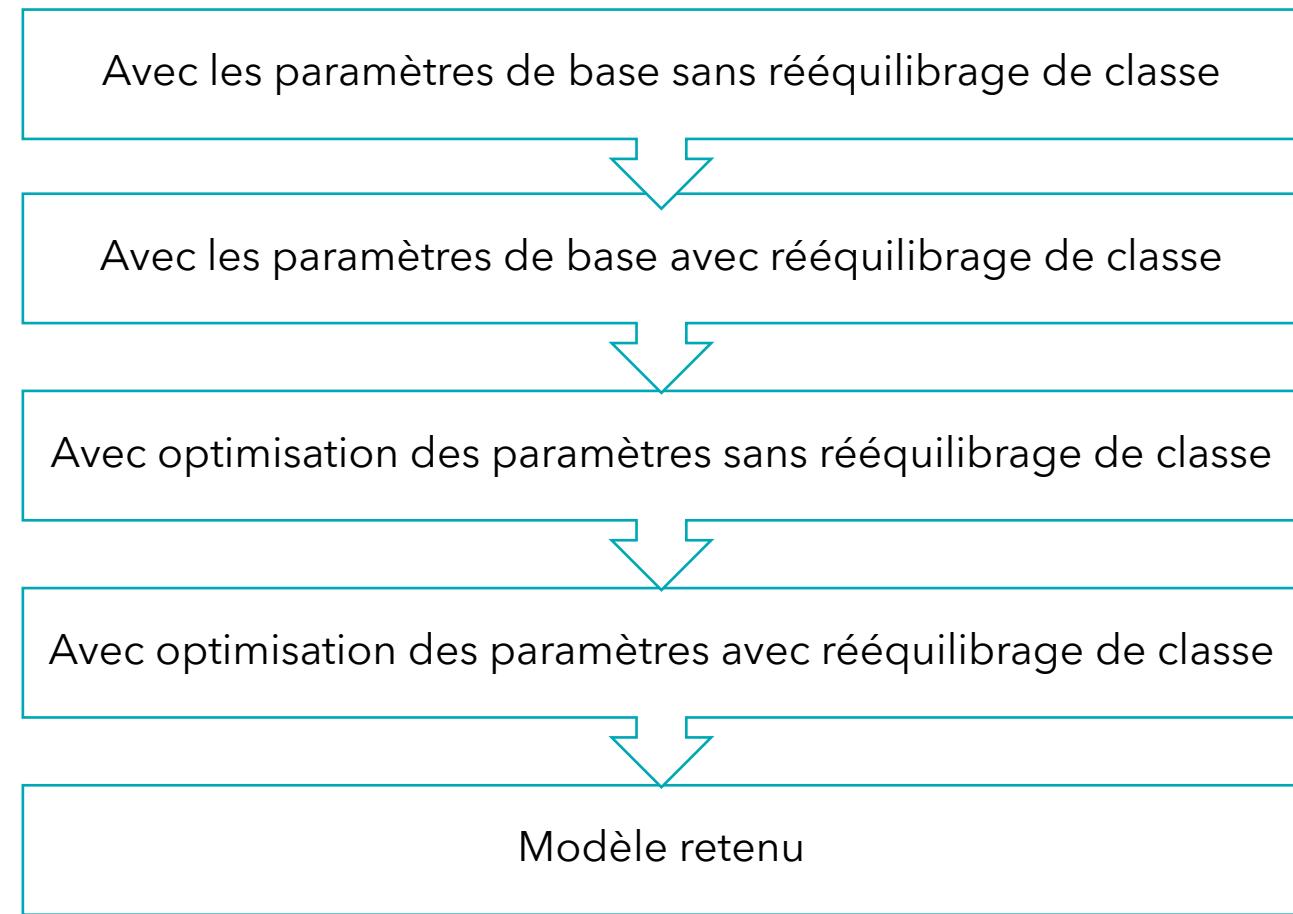
$$Recall = \frac{TP}{TP + FN}$$

*De tous les cas possibles où la cible était 1, combien nous pourrions capturer, dans notre exemple serait, de tous les cas possibles de client avec problème de paiement, combien notre modèle pourrait identifier.*

- **La courbe ROC et score AUC** : Il s'agit d'un graphique du taux de faux positifs par rapport au taux de vrais pour un certain nombre de valeurs de seuil candidates différentes comprises entre 0,0 et 1,0. En d'autres termes, il trace le taux de fausses alarmes par rapport au taux de succès.

# Modélisation :

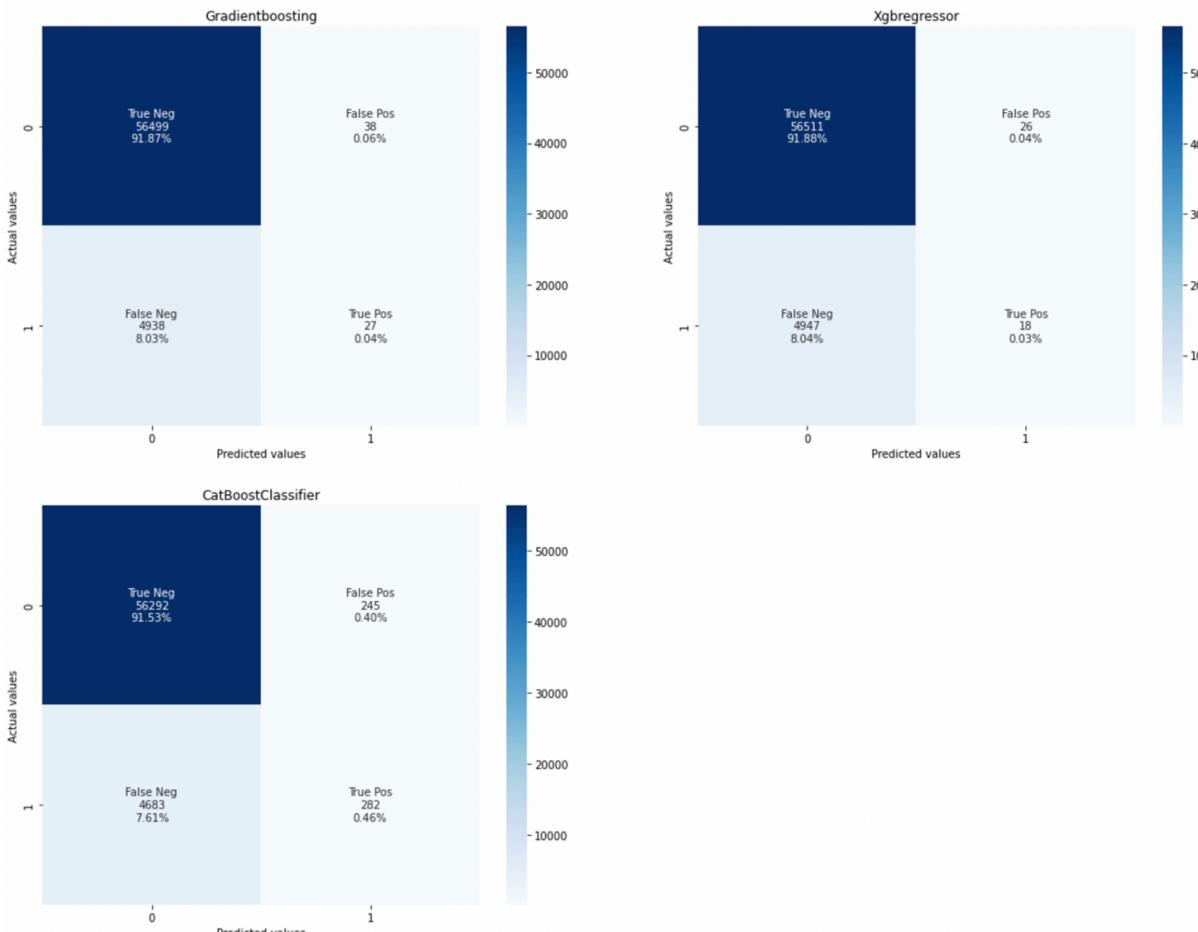
- Les modèles ont été testé de cette façon :



# Modélisation :

- Ici les premiers résultats ( de base):

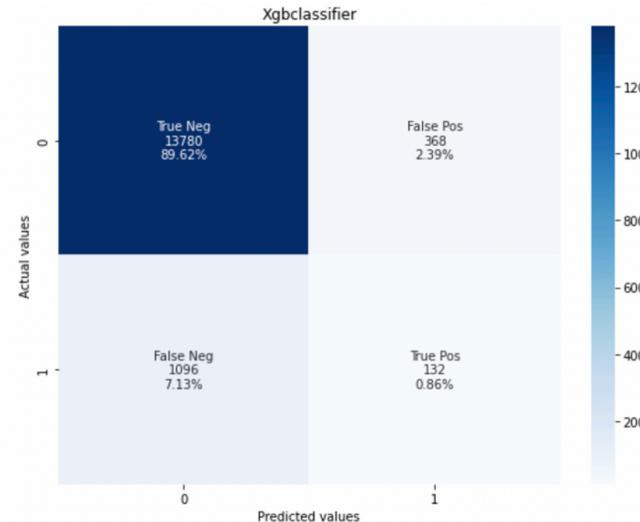
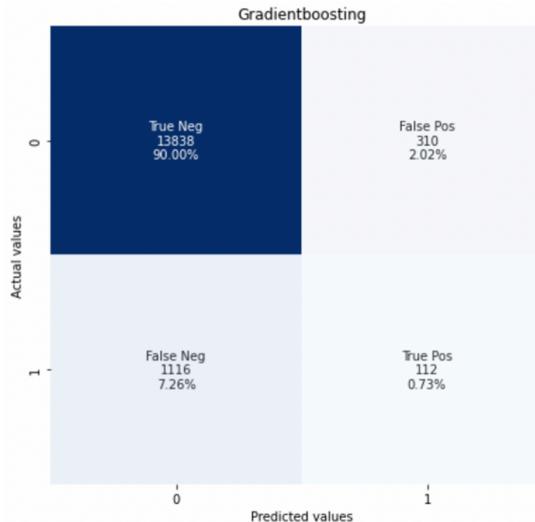
|   | Models             | Auc    | Accuracy | Precision | Recall | F1     |
|---|--------------------|--------|----------|-----------|--------|--------|
| 0 | Gradientboosting   | 0.7413 | 0.9191   | 0.4154    | 0.0054 | 0.0107 |
| 1 | Xgbregressor       | 0.7436 | 0.9191   | 0.4091    | 0.0036 | 0.0072 |
| 2 | CatBoostClassifier | 0.7846 | 0.9199   | 0.5351    | 0.0568 | 0.1027 |



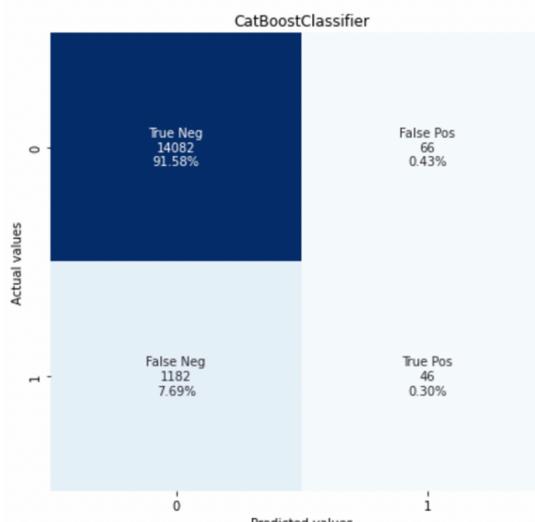
**Pour le dernier modèle on peut traduire le résultat en disant que le modèle prédit 5% de cas positifs, mais que, quand il prédit une classe 1, il a raison dans 53% des cas.**

# Modélisation :

- Optimisation des modèles ave des **Gridsearchs** en optimisant les **hyper paramètres** :



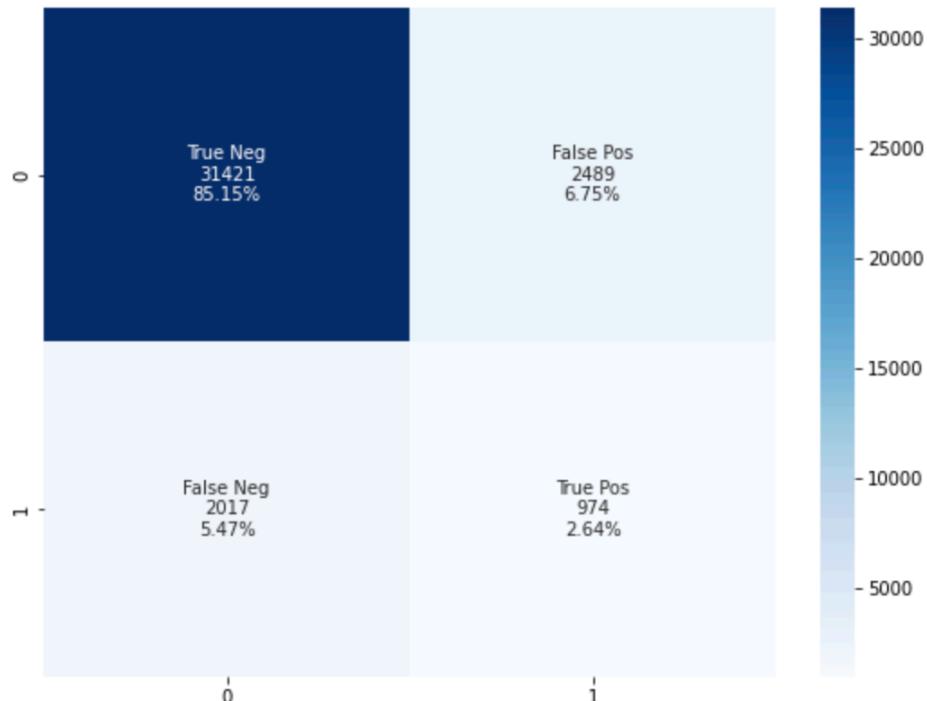
| Models             | Auc    | Accuracy | Precision | Recall | F1     |
|--------------------|--------|----------|-----------|--------|--------|
| Gradientboosting   | 0.7071 | 0.9073   | 0.2654    | 0.0912 | 0.1358 |
| Xgbclassifier      | 0.7079 | 0.9048   | 0.2640    | 0.1075 | 0.1528 |
| CatBoostClassifier | 0.7474 | 0.9188   | 0.4107    | 0.0375 | 0.0687 |



- Modèle retenu : **CatboostClassifier**

```
param_cat = {  
    "depth": [2,4, 6],  
    "learning_rate": [1,2],  
    "n_estimators": [50,100],  
    "l2_leaf_reg": [0.5,1]  
}
```

AUC : 0.7635

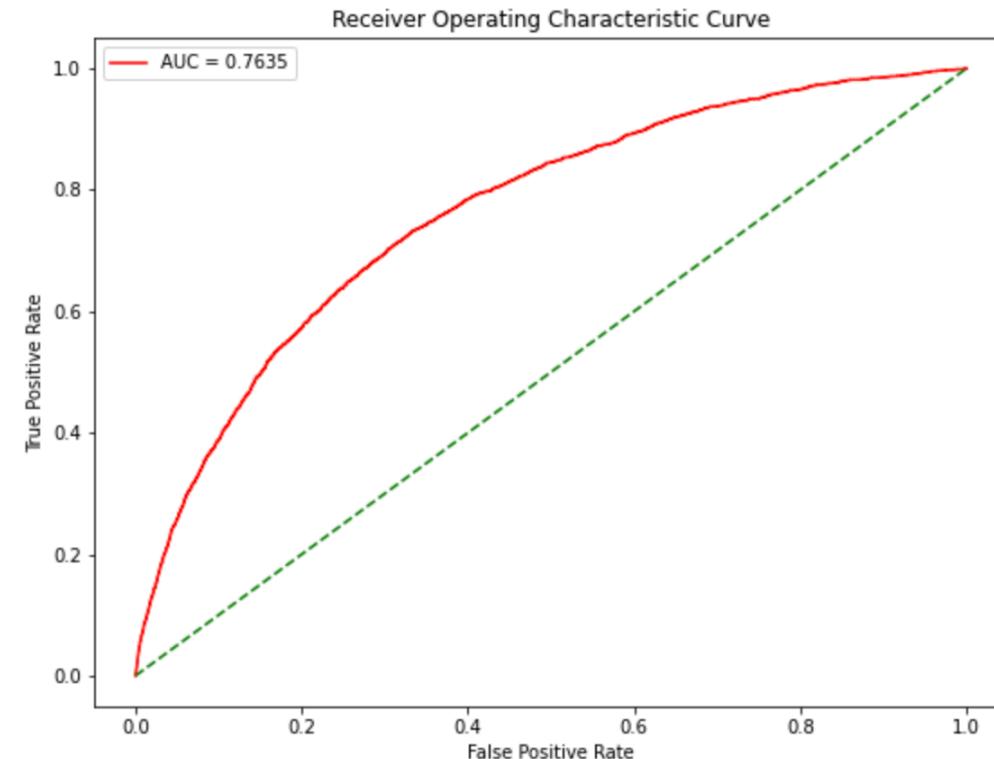


**Depth** : Profondeur de l'arbre

**Learning\_rate** : La fréquence à laquelle les pondérations du modèle sont mises à jour après avoir travaillé sur chaque lot d'exemples d'entraînement.

**N\_estimators** : Le nombre maximum d'arbres

**L2\_leaf\_reg** : Coefficient du terme de régularisation L2 de la fonction de coût.



# Modélisation :

| Models             | Auc    | Accuracy | Precision | Recall | F1     |
|--------------------|--------|----------|-----------|--------|--------|
| CatboostClassifier | 0.7635 | 0.8779   | 0.2813    | 0.3256 | 0.3018 |

- **Le modèle prédit 32% de cas positifs, mais que, quand il prédit la classe 1, il a raison dans 28% des cas.**

# La fonction coût métier :

L'entreprise Prêt à dépenser est une société **financière** qui propose des crédits et donc essayera de lutter contre les défauts de paiement et **diminuer les pertes financières**.

Le modèle ne permettra pas d'éviter complètement ce risque mais on peut **minimiser** les pertes en définissant une fonction coût afin de **pénaliser** les **erreurs** de prédiction qui peuvent **coûter** cher à l'entreprise.

**L'idée est d'éviter les clients avec un fort risque de défaut de paiement. Il est donc nécessaire de pénaliser les FN.**



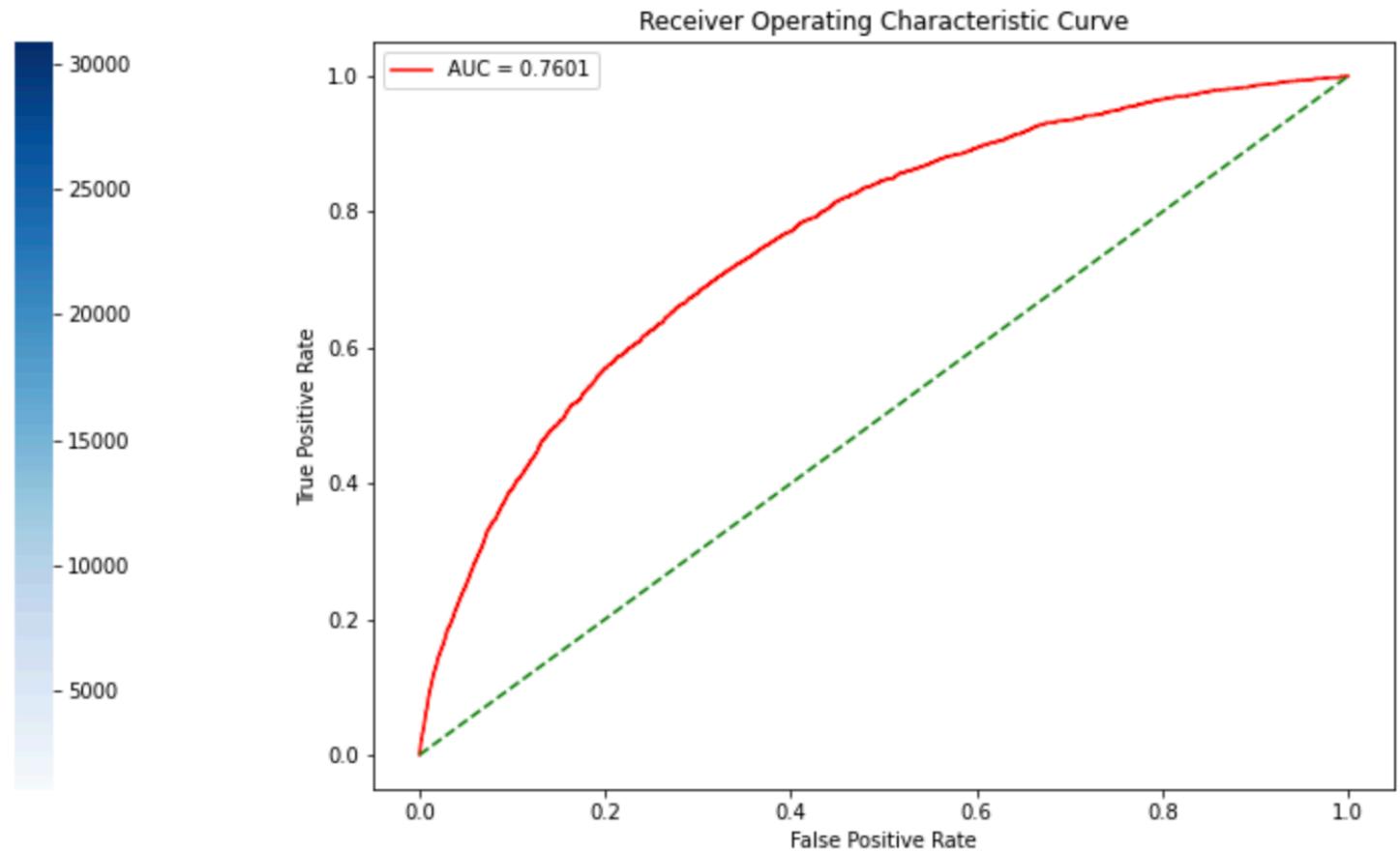
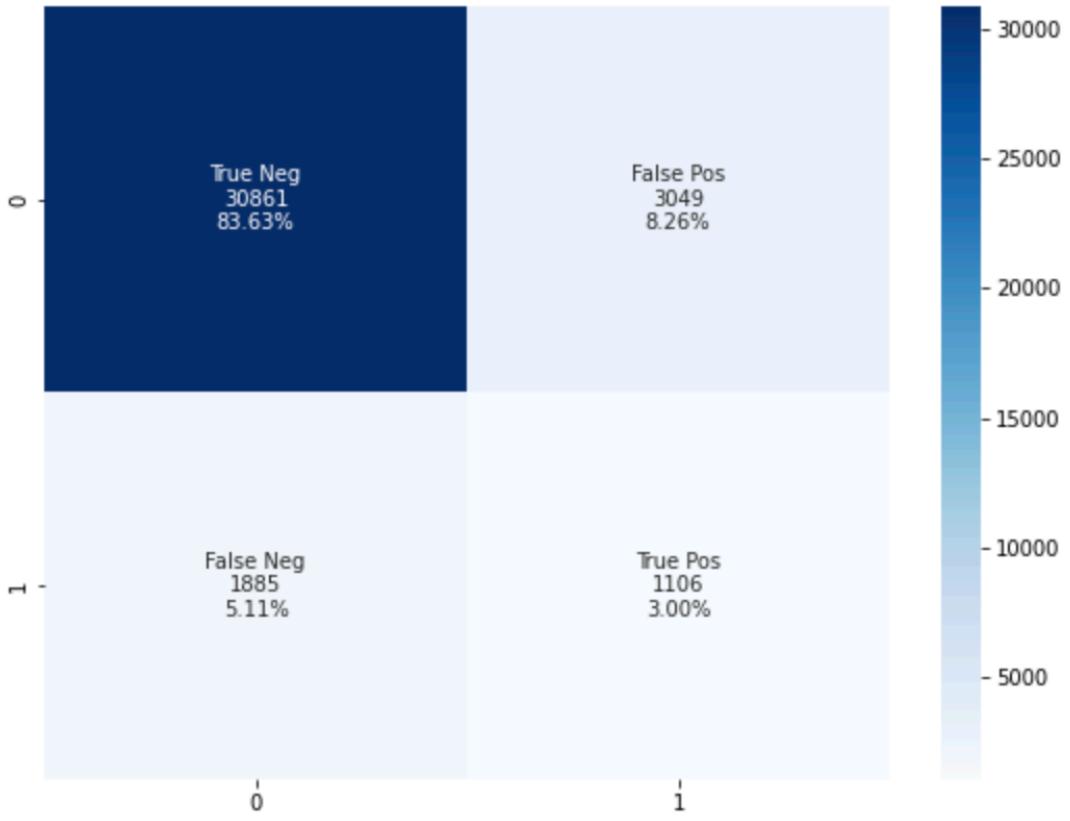
# La fonction coût métier :

Pour ceci nous créons une **matrice de confusion**, mais au lieu de mettre les quantités prévues, nous allons mettre des pondérations pour chaque élément de la matrice.

|   |                            |                          |  |
|---|----------------------------|--------------------------|--|
|   | <b>True Négatif : + 1</b>  | <b>False positif : 0</b> |  |
| <b>Client qui rembourse et bien identifié (<u>gain pour l'entreprise</u>)</b>       |                            |                          | <b>Client rembourse et mal identifié</b>         |
| <b>Client ne rembourse pas et mal identité<br/>(<u>perte pour l'entreprise</u>)</b> | <b>False Négatif : -10</b> | <b>True positif : 0</b>  | <b>Client ne rembourse pas et bien identifié</b> |

# La fonction coût métier :

AUC : 0.7601



Avec l'implémentation de la métrique métier le taux des **False Négatifs** est de **5.11 %** au lieu de 5.47%, soit une détection de 132 cas de plus de clients ayant des difficultés de paiement

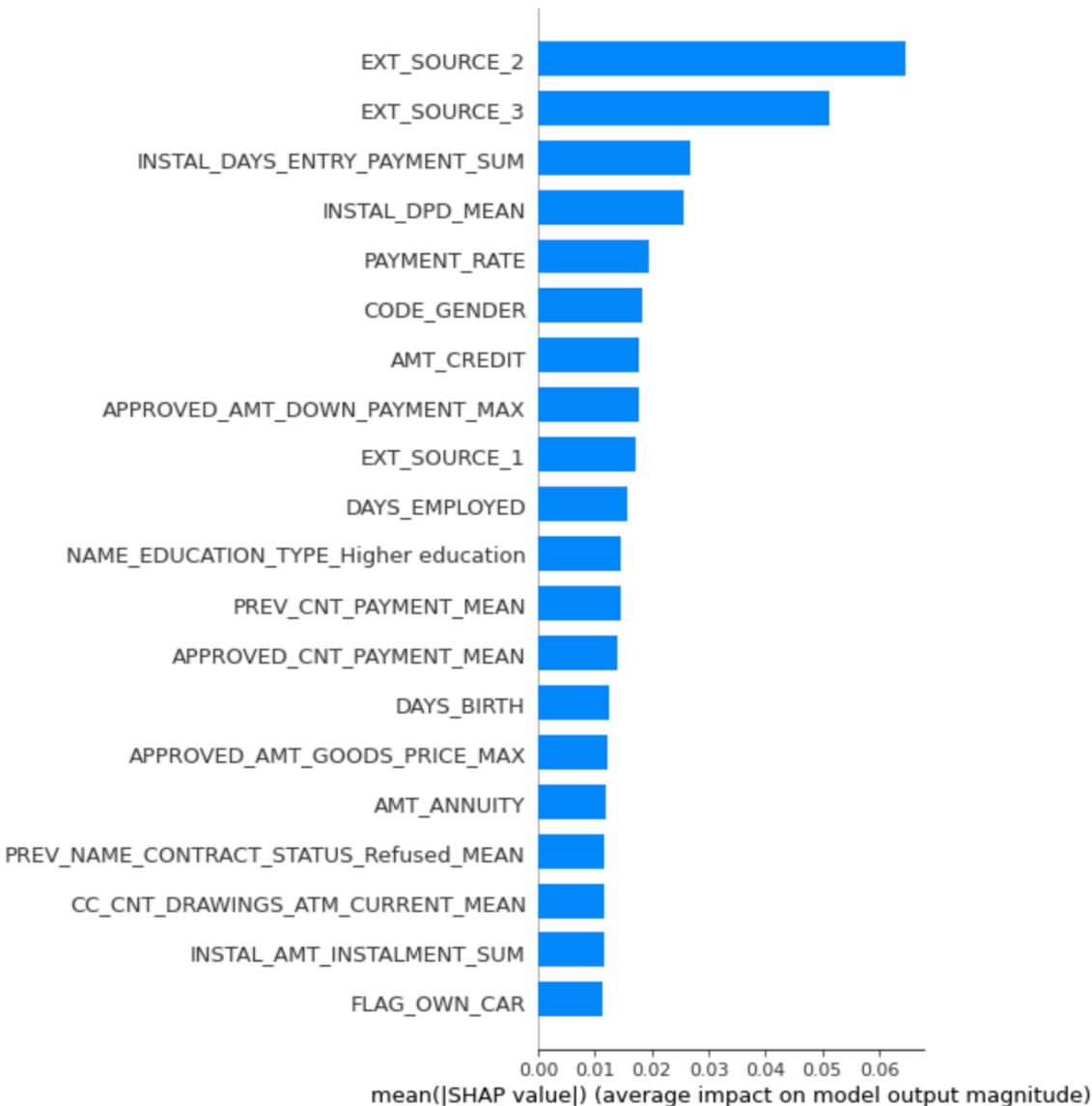
# Threshold : Seuil de probabilité

- Le threshold est le **seuil de décision** que nous fixons pour qu'un client appartient à la classe 0 ou 1.
- La matrice de confusion dépend du seuil de classification qui est par défaut de **50%**.
- Selon les besoins, le choix du seuil diffère. On peut privilégier la détection des clients ayant des difficultés de paiement en **abaissant le seuil**.
- Pour un même modèle, on obtient plusieurs matrices de confusion en fonction du seuil choisi.

# L'interprétabilité globale

- L'importance des features permet de **comprendre la relation** entre les **caractéristiques** et la variable **cible**. Elle fait référence à des techniques qui calculent un score pour toutes les variables d'entrée pour un modèle donné - les scores représentent simplement « **l'importance** » de chaque fonctionnalité. Un score plus élevé signifie que la variable spécifique aura un effet plus important sur le modèle utilisé pour prédire une certaine variable.
- L'importance des fonctionnalités est également utile pour **interpréter** et **communiquer** le modèle à d'autres parties prenantes.

# L'interprétabilité globale

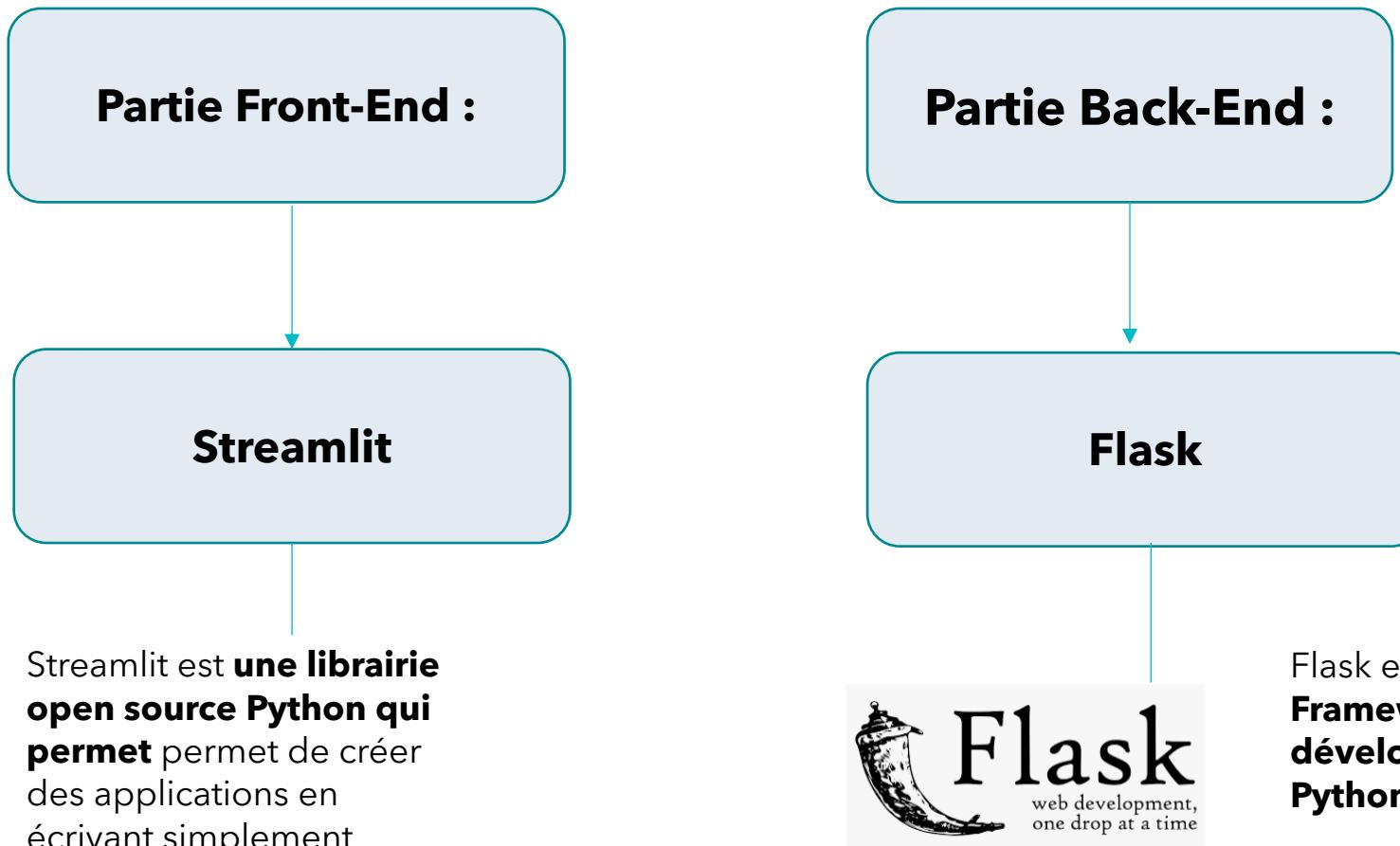


**SHAP** (*Shapley Additive explanations*) est une approche théorique pour expliquer la sortie de tout modèle d'apprentissage automatique.



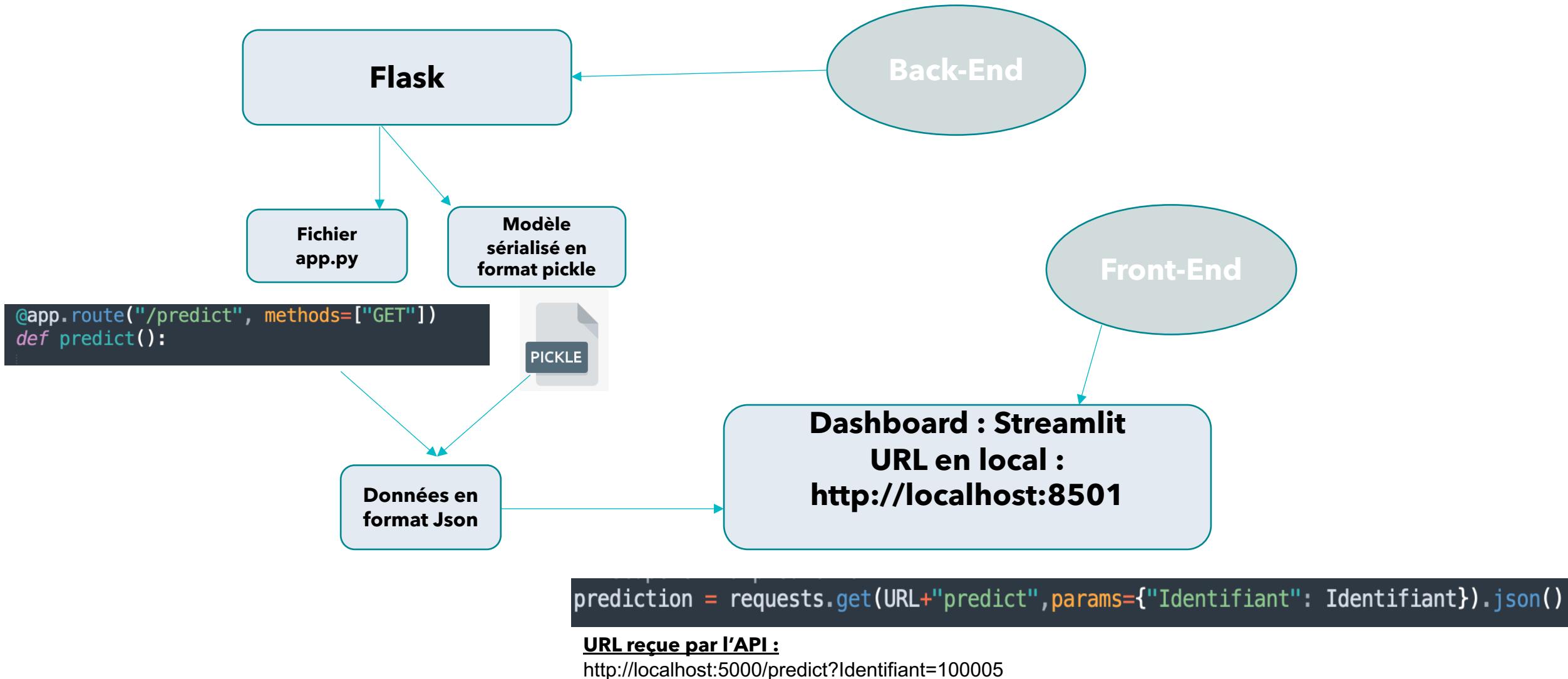
SHAP

# Réalisation du Dashboard interactif :



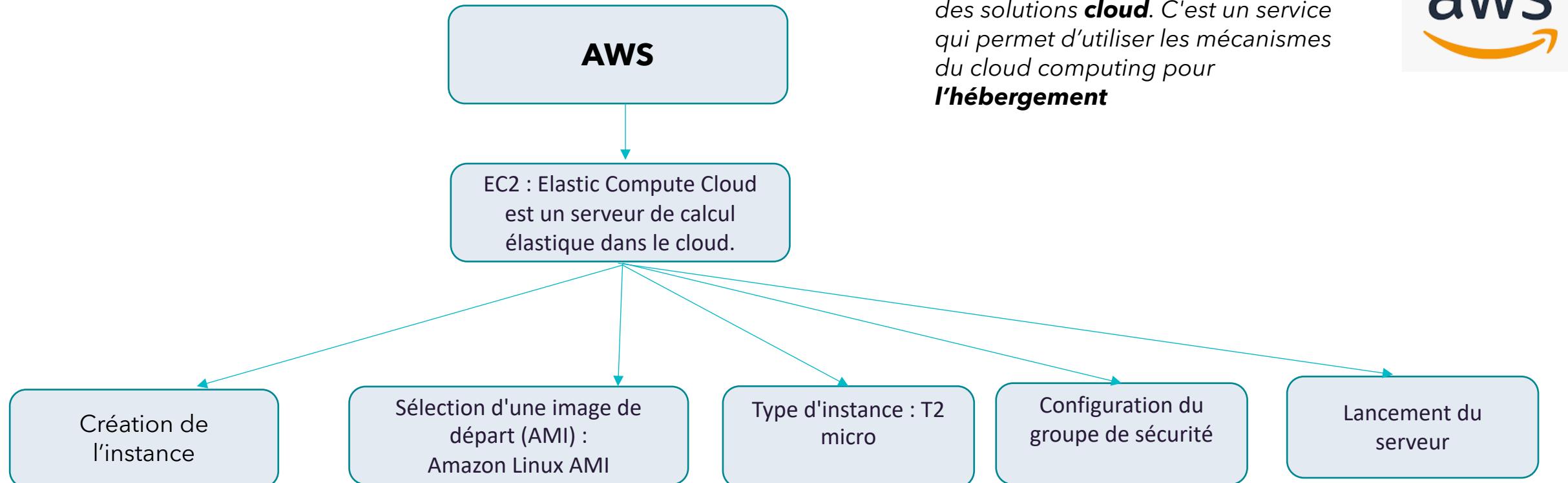
# Réalisation du Dashboard interactif :

## Fonctionnement en local :



# Réalisation du Dashboard interactif :

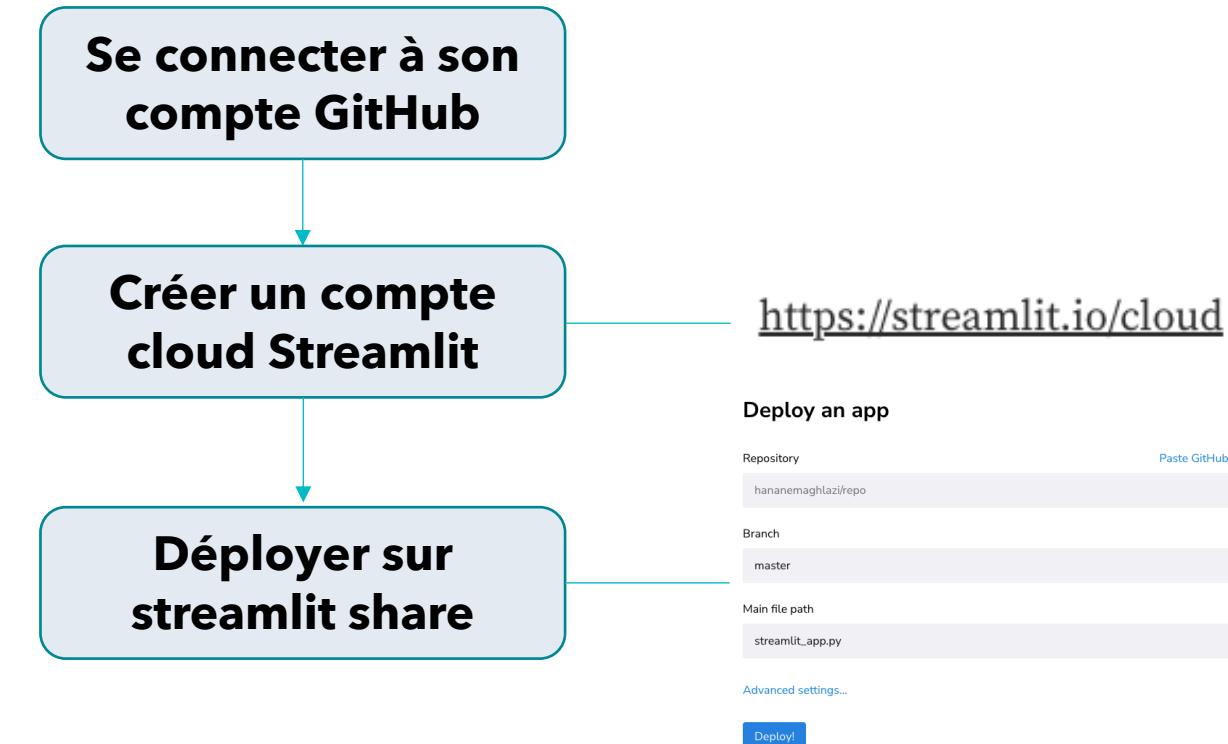
- **Déploiement de l'API dans le cloud:**



# Réalisation du Dashboard interactif :

- **Déploiement du Dashboard avec streamlit share :**

Streamlit propose également sa propre plate-forme cloud et le déploiement se fait selon les étapes suivantes:



# Lien du Dashboard interactif et GitHub

- **Déploiement dans le cloud:**

<https://hananemaghazi-p7-scoring-dashboardprediction-juf752.streamlit.app/>

- **L'outil de versioning:**

[https://github.com/HananeMaghlazi/P7\\_Scoring/tree/main](https://github.com/HananeMaghlazi/P7_Scoring/tree/main)

# Conclusion :

- Les résultats obtenus peuvent être améliorés en essayant d'autres hyper paramètres au niveau de l'optimisation du modèle qui peuvent augmenter ses performances. La taille du dataset et les temps de calcul n'ont pas permis de tester tous les hyper paramètres.
- L'hypothèse de la fonction métier personnalisée est à discuter avec l'équipe métier et donc peut être modifiée.
- L'ajustement du seuil de probabilité pour l'attribution du crédit doit-être aussi discuté aussi avec les équipes métier ce qui permettra de répondre aux besoins de l'entreprise.



# Discussion :



**FIN**

• **MERCI**