

Project 7

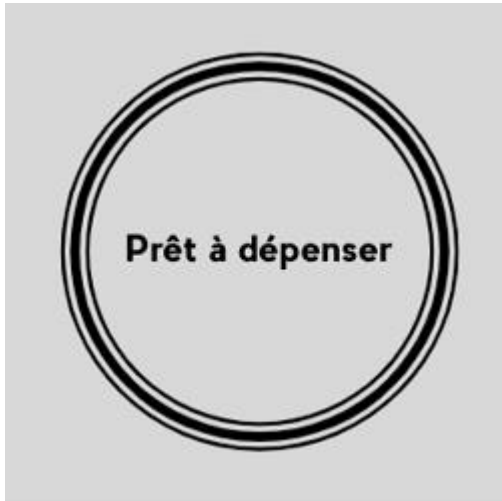
Implémentez un modèle de scoring

Tetiana Lemishko

Sommaire:

- Mission et objectifs principaux
- Présentation des données
- Analyse exploratoire des données
- Modélisation
- Présentation du dashboard
- Conclusion

Mission et objectifs principaux



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

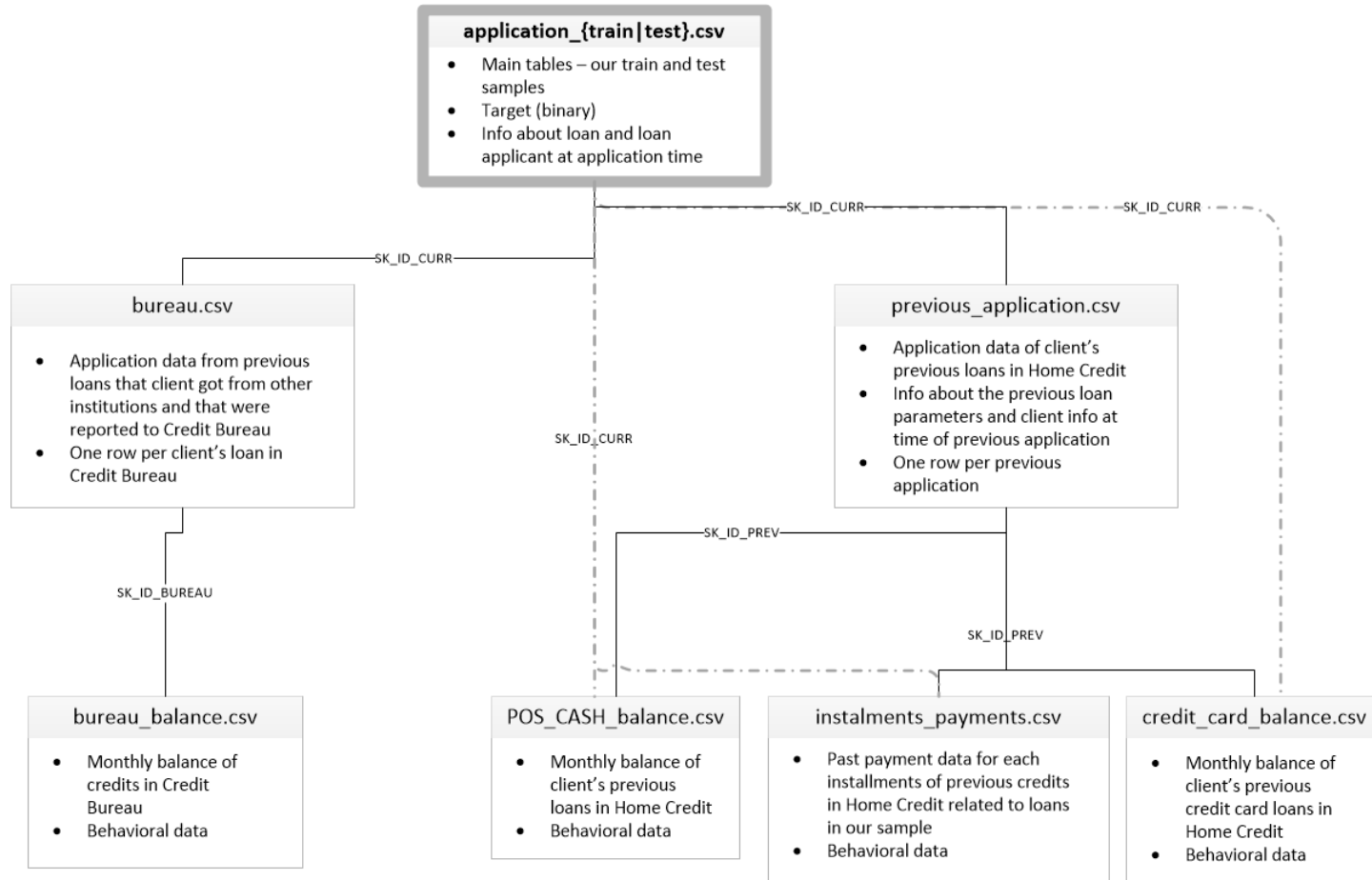
L'entreprise souhaite mettre en œuvre un outil de "scoring crédit" pour calculer la probabilité qu'un client rembourse son crédit, puis classifie la demande en crédit accordé ou refusé.

Objectifs principaux:

- Construire un modèle de scoring qui donnera une prédiction 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.

Présentation des données

On a à notre disposition une base de données bancaires anonymisées présentée par 8 fichiers (***application_{train|test}.csv*** son les fichiers principaux) => plus de 300000 clients



Types d'informations:

- Informations clients
- Historique de prêts
- Données cartes de crédit
- Données liquidités
- Autres informations

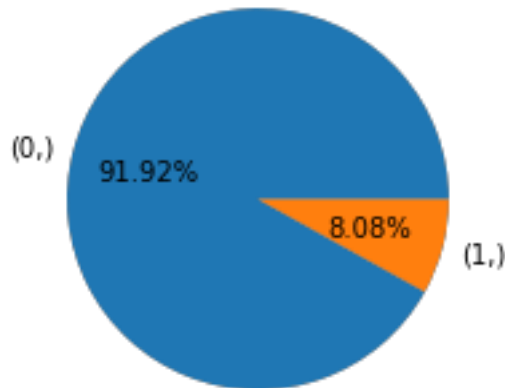
Target (variable binaire) - statut de remboursement du prêt :

1 - le prêt n'a pas été remboursé

0 - le prêt a été remboursé

Analyse exploratoire des données

RÉPARTITION DE TARGET



On peut voir que la variable **Target** est fortement déséquilibrée:

~ 92 % des prêts ont été remboursés

~ 8 % de prêts non remboursés

Target en fonction de l'âge

Distribution de la variable d'âge des clients

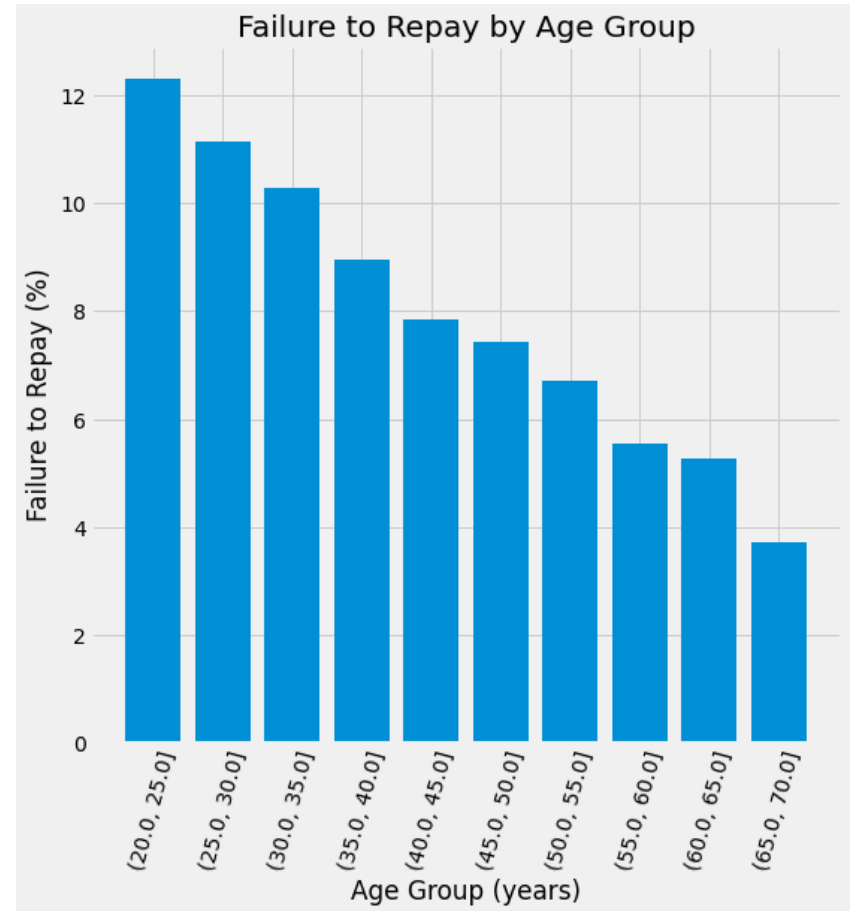


Il y a une tendance claire :

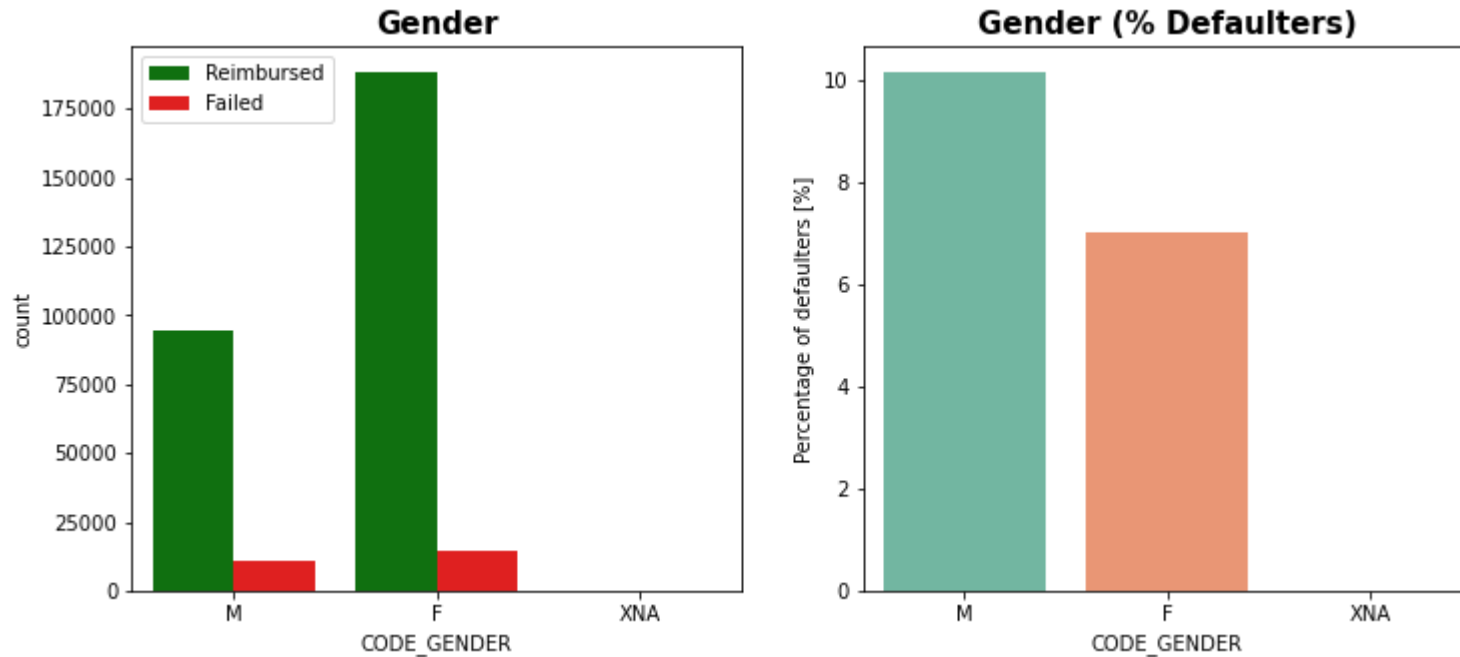
les jeunes demandeurs sont plus susceptibles de ne pas rembourser le prêt.

Le taux d'impayés est supérieur à 10 % pour les trois tranches d'âge les plus jeunes et inférieur à 5 % pour la tranche d'âge la plus élevée.

Défaut de prêt par groupe d'âge



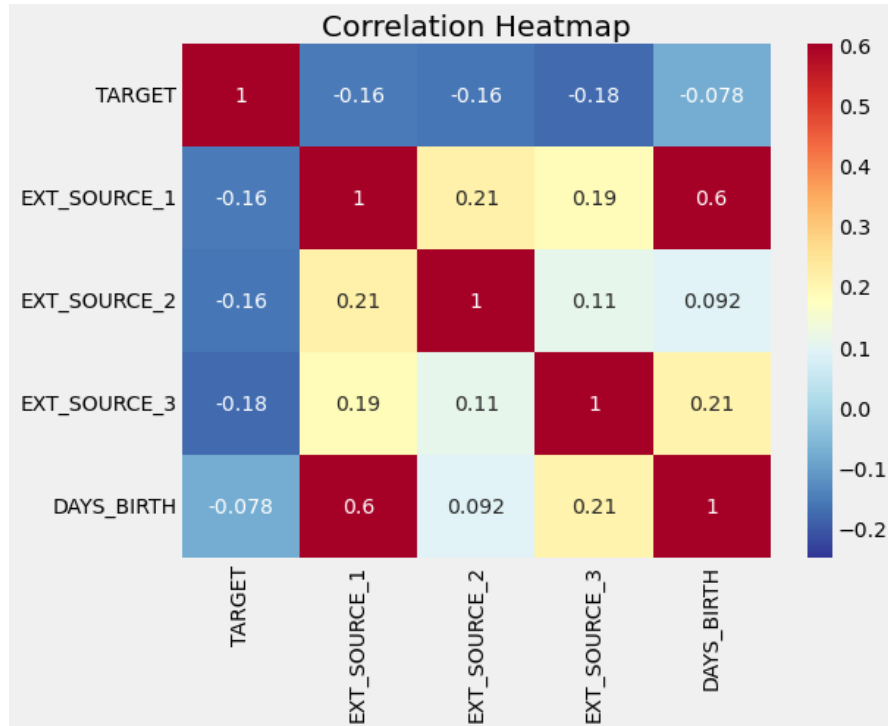
Target en fonction du genre



Le nombre total de clients féminins est presque le double du nombre de clients masculins.

Les hommes ont plus de risque de ne pas rembourser leurs prêts (~ 10 %), par rapport aux femmes (~ 7 %)

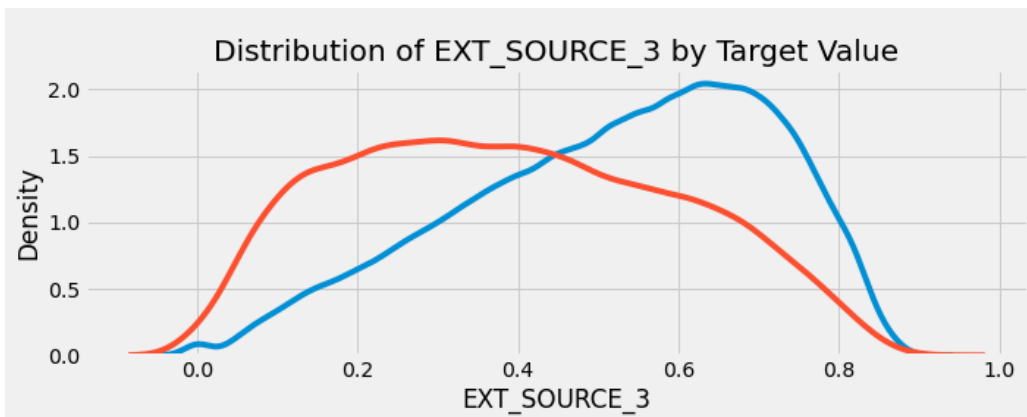
Corrélation (3 variables **EXT_SOURCE**, Days_BIRTH vs. **TARGET**)



- Les trois variables **EXT_SOURCE** ont des corrélations négatives avec la variable **TARGET**, ce qui indique que plus la valeur de la **EXT_SOURCE** augmente, plus le client est susceptible de rembourser le prêt.

- Selon la documentation, les variables **EXT_SOURCE** représentent un 'score normalisé provenant d'une source de données externe'.

- **EXT_SOURCE_3** présente la corrélation la plus forte avec le remboursement du prêt.



Modélisation. Étapes à suivre

- 1) Feature engineering
- 2) Séparation des données
- 3) Prétraitement des données (imputation des valeurs NaN et normalisation (scaling))
- 4) Entraînement des modèles et choix du meilleur modèle
- 5) Optimisation du modèle sélectionné d'un point de vue métier
- 6) Feature importance

Modélisation. Feature engineering

Cette partie a été entièrement inspirée du notebook Kaggle suivant:

<https://www.kaggle.com/code/willkoehrsen/start-here-a-gentle-introduction/notebook>

Label Encoding and One-Hot Encoding

Pour toute variable catégorielle (dtype == objet) avec 2 catégories uniques, nous utiliserons **Label Encoding**, et pour toute variable catégorielle avec plus de 2 catégories uniques, nous utiliserons **One-Hot Encoding**.

Pour **Label Encoding**, nous utilisons Scikit-Learn LabelEncoder et pour **One-Hot Encoding**, la fonction pandas get_dummies(df).

```
# Create a label encoder object
le = LabelEncoder()
le_count = 0

# Iterate through the columns
for col in app_train:
    if app_train[col].dtype == 'object':
        # If 2 or fewer unique categories
        if len(list(app_train[col].unique())) <= 2:
            # Train on the training data
            le.fit(app_train[col])
            # Transform both training and testing data
            app_train[col] = le.transform(app_train[col])
            app_test[col] = le.transform(app_test[col])

            # Keep track of how many columns were label encoded
            le_count += 1

print('%d columns were label encoded.' % le_count)
```

3 columns were label encoded.

Modélisation. Séparation et Prétraitement des données

Séparation:

Train set (75%)

Test set (25%)

Prétraitement :

- Imputation des valeurs manquantes (Stratégie: médiane)
- Normalisation des features (MinMaxScaler)

Modélisation

Modeles testés:

- Régression Logistique
- RandomForestClassifier
- Light Gradient Boosting Machine

Optimisation des hyperparamètres:

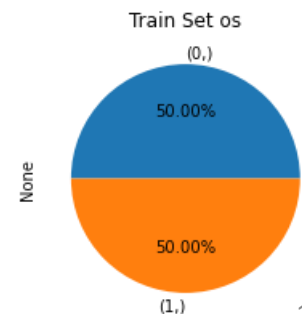
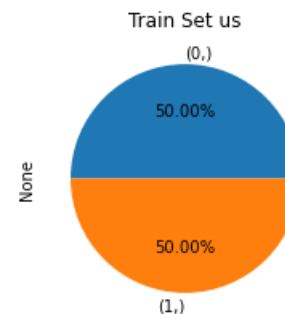
GridSearch

Évaluation:

roc_auc_score (0÷1 => plus roc_auc_score est élevée, meilleures sont les performances du modèle)

Gestion des données déséquilibrées:

- Unbalanced data
- Undersampling (supprimer des observations de la classe majoritaire afin de rééquilibrer le jeu de données (RandomUnderSampler))
- Oversampling(répéter des observations de la classe minoritaire afin de rééquilibrer le jeu de données (Synthetic Minority Oversampling Technique))



Modélisation. Choix du meilleur modèle

Modele/Stratégie de gestion des données déséquilibrées	roc_auc_score
Régression Logistique/ Imbalanced data	0.6881
Régression Logistique/ Undersampling	0.7221
Régression Logistique/ Oversampling	0.7352
RandomForestClassifier/ Imbalanced data	0.7150
RandomForestClassifier/ Undersampling	0.7437
RandomForestClassifier/ Oversampling	0.7121
LGBM/ Imbalanced data	0.7581
LGBM/ Undersampling	0.7560
LGBM/ Oversampling	0.7534

Le modèle LGBM appliqué aux données déséquilibrées donne les meilleurs résultats

Modélisation. Optimisation du modèle sélectionné d'un point de vue métier

Il y a les abréviations suivantes qu'on a utilisées dans cette étude que l'on va définir ci-dessous.

- **True Positive (TP):** Les Targets qui sont réellement vrais et que nous avons prédits vrais (Le prêt est (correctement) refusé : la banque ne gagne ni perd d'argent).
- **True Negative (TN):** Les Targets qui sont en réalité faux et que nous avons prédits faux (Le prêt est remboursé : la banque gagne de l'argent).
- **False Positive (FP):** Les Targets qui sont en fait faux mais que nous avons prédits vrais (Le prêt est refusé par erreur : la banque perd de l'argent qu'elle aurait pu gagner, mais ne perd en fait pas d'argent).
- **False Negative (FN):** Les Targets qui sont réellement vrais mais que nous avons prédit faux (Le prêt est accordé mais le client fait défaut : la banque perd de l'argent).

En quoi consiste la problématique 'métier' ?

On doit prendre en compte qu'un **FP** n'a pas le même coût qu'un **FN**. Un faux négatif est en effet 10 fois plus coûteux qu'un faux positif.



On doit adapter le modèle afin de minimiser **FN**

Pour y parvenir on peut appliquer 2 stratégies.

Modélisation. Optimisation du modèle sélectionné d'un point de vue métier

Strategies des minimization de perts

Optimisation des hyperparamètres via HyperOpt

- 1) fonction métier adaptée au projet qui permet d'attribuer plus de poids à la minimisation des FN

```
def gain_score(y_true, y_pred) :  
    (TN, FP, FN, TP) = confusion_matrix(y_true, y_pred).ravel()  
    N = TN + FP    # total negatives cases  
    P = TP + FN    # total positives cases  
  
    #FN weight = -10 # The loan is granted but the customer defaults : the bank loses money  
    #TN weight = 1  # The loan is reimbursed : the bank makes money  
    #TP weight = 0   # The loan is (rightly) refused : the bank neither wins nor loses money  
    #FP weight = -1  # Loan is refused by mistake : the bank loses money it could have made,  
  
    # calculate total gains  
    gain = TP*0 + TN*1 + FP*(-1) + FN*(-10)  
  
    # best score : all observations are correctly predicted  
    best_case = N*1 + P*0  
  
    # baseline : all observations are predicted = 0 (all the clients are predicted to be able to repay the loan)  
    baseline = N*1 + P*(-10)  
  
    # normalize to get score between 0 (baseline) and 1  
    score = (gain - baseline) / (best_case - baseline)  
  
    return score
```

- 1) Effectuer une nouvelle optimisation d'hyperparamètres via HyperOpt basée sur la fonction métier proposée (Hyperopt est une bibliothèque Python pour l'optimisation série et parallèle sur des espaces de recherche qui peuvent inclure des dimensions réelles, discrètes et conditionnelles). Les hyperparamètres seront choisis de manière à minimiser la perte d'argent pour la banque.

Adaptation de seuil (le meilleur f1_score)

La métrique ROC AUC donne une probabilité qu'un échantillon appartienne à une classe. Par défaut le seuil de décision est fixé à 0.5.



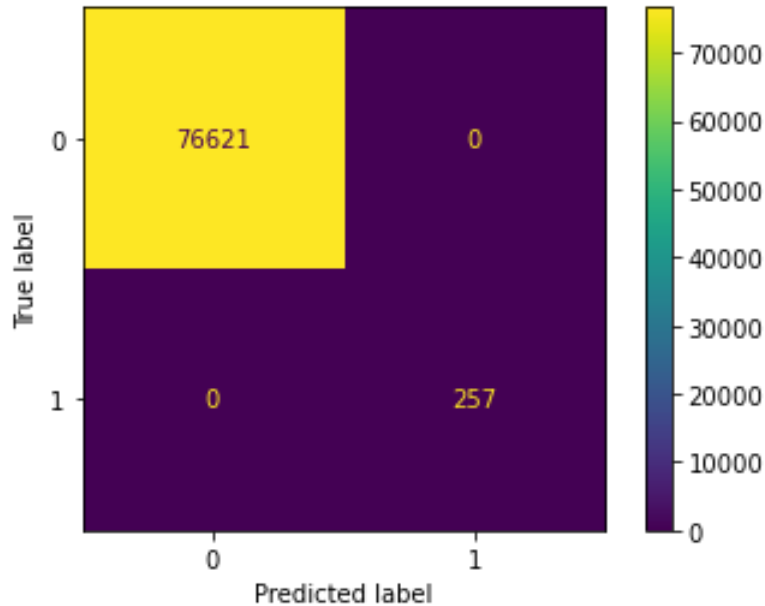
On doit adapter le seuil afin de minimiser la perte



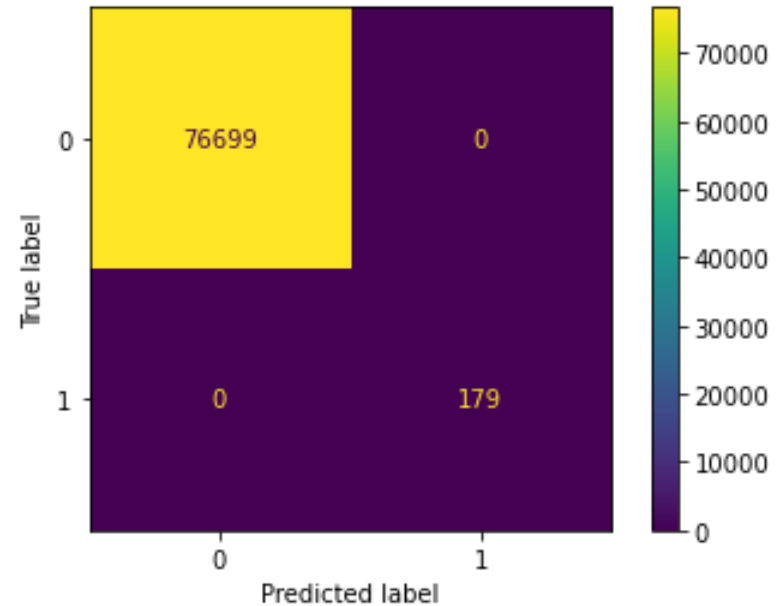
Les fonctions spéciales pour trouver le meilleur seuil compte tenu de la règle selon laquelle un faux négatif est en effet 10 fois plus coûteux qu'un faux positif.

Modélisation. Optimisation du modèle sélectionné d'un point de vue métier

Matrice de confusion avant l'optimisation



Matrice de confusion après l'optimisation



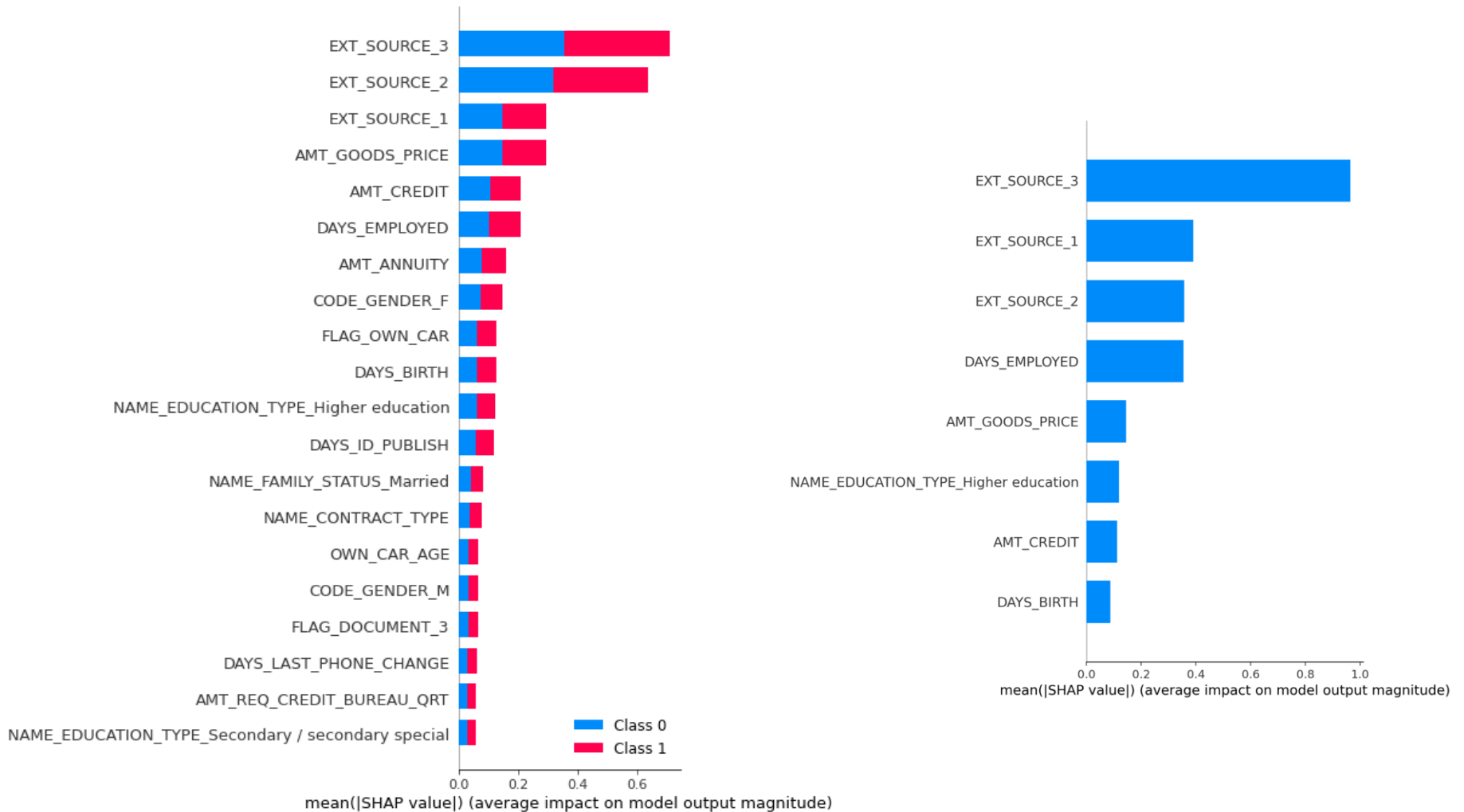
1. Le roc_auc_score a été légèrement amélioré (de 0.7581 à 0.7594)
2. On a augmenté le gain de l'argent:

$$\text{gain} = \text{TP} \cdot 0 + \text{TN} \cdot 1 + \text{FP} \cdot (-1) + \text{FN} \cdot (-10)$$

$$\begin{aligned} \text{gain_before} &= 257 \cdot 0 + 76621 \cdot 1 + 0 \cdot (-1) \\ &+ 0 \cdot (-10) = 76621 \end{aligned}$$


$$\begin{aligned} \text{gain_after} &= 179 \cdot 0 + 76699 \cdot 1 + 0 \cdot (-1) + 0 \cdot (-10) \\ &= 76699 \end{aligned}$$

Modélisation. Explicabilité (Feature importance)



On a expliqué les prédictions (les importances des features globales et locales (exemple - Client ID 100001)) du modèle choisi à l'aide de la bibliothèque SHAP. 3 variables EXT_SOURCE sont les plus importantes car elles prédisent la Target dans les deux cas.

Présentation du dashboard

 **Client analysis**

Client ID

Select Customer ID

100001 ▾

Actions

☐ View cliend info

☐ View comparison with other clients

☐ View credit decision

☐ View local SHAP features importances


☐ View global SHAP features importances

Dashboard - Scoring Credit

Credit decision support for customer relationship managers

What is this app for?

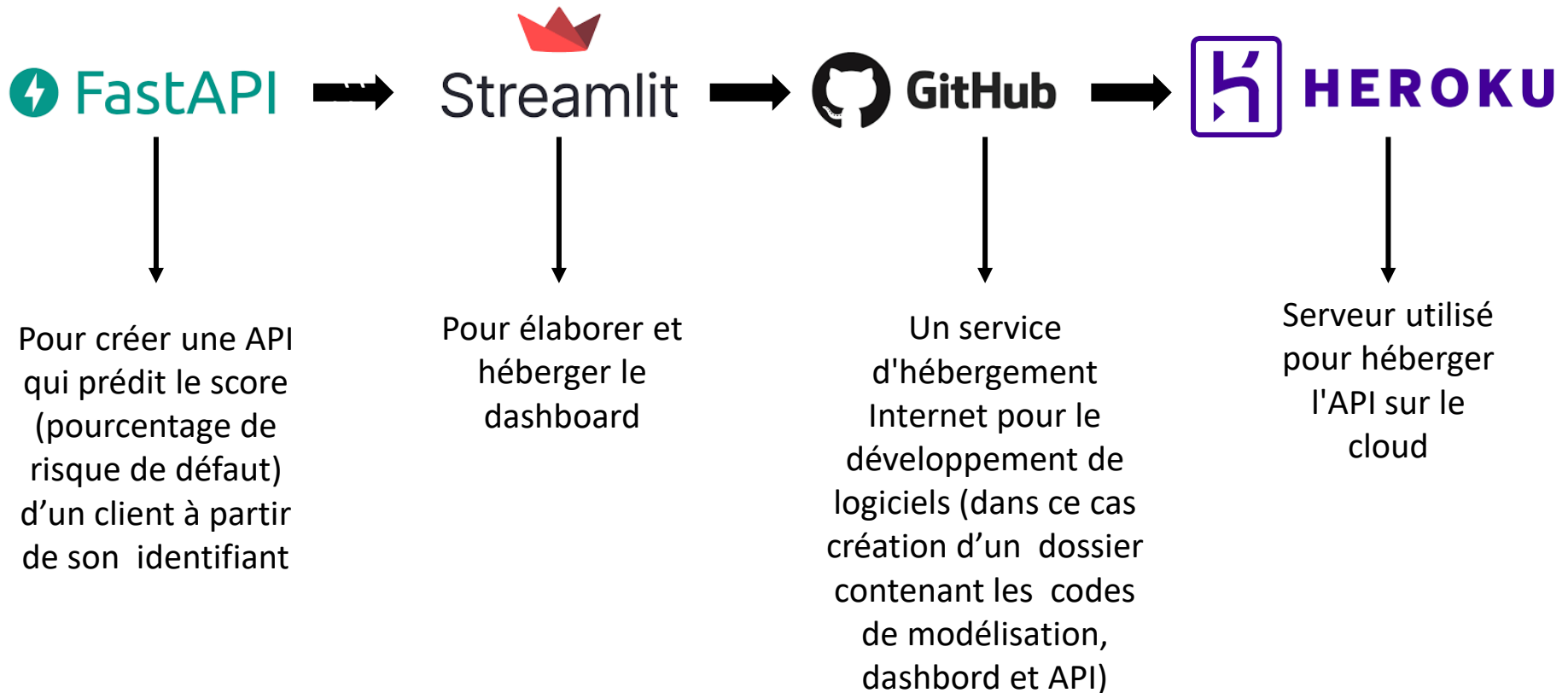
This app is used to predict the capacity of the client to repay the loan



Fonctionnalités du dashboard :

- Visualiser le score et l'interprétation de ce score pour chaque client de façon intelligible pour une personne non experte en data science.
- Visualiser des informations descriptives relatives à un client.
- Comparer les informations descriptives relatives à un client à l'ensemble des clients ou à un groupe de clients similaires.

Présentation du dashboard. Technologies utilisées (déploiement)



Présentation du dashboard

Le lien pour l'API :

https://fastapilemishko.herokuapp.com/docs#/default/credit_predict_get

Le lien pour dashboard :

<https://lemkot-1-app-j8ggqo.streamlit.app/>

Limites et améliorations possibles

- Améliorer la modélisation en affinant le travail de features engineering en collaboration avec les équipes métier.
- Utiliser la recherche d'expérience utilisateur pour adapter le dashboard aux besoins des utilisateurs.
- Chacun des modèles entraînés a été optimisé en testant plusieurs valeurs des 2-4 principaux hyperparamètres (car l'exécution de GridSearch prend beaucoup de temps). Elargir le nombre d'hyperparamètres pourrait peut-être permettre une amélioration des performances des différents modèles.