
NOTE METHODOLOGIQUE

Projet 7 OpenClassRooms – Octave Pouillot

Table des matières

1.	Sélection du modèle	2
1.1	Pre-processing	2
1.2	Fonction coût métier et métrique d'évaluation	3
1.3	Choix du modèle	3
1.4	Optimisation des modèles	4
2.	Les résultats	5
3.	Interprétabilité du modèle.....	6
3.1	Interprétabilité globale	6
3.2	Interprétabilité locale	6
4.	Limites et améliorations	7
5.	Analyse du Data Drift	8

1. Sélection du modèle

Suite à l'exploration et l'analyse des données, et après avoir nettoyé le dataset, il est question de trouver le modèle le plus adapté à la problématique. De manière générale, voici la méthode par laquelle je suis passé pour sélectionner le modèle :

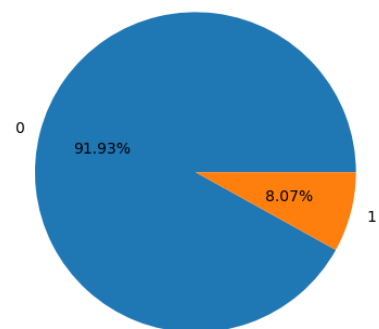
- **Première sélection** : Plusieurs modèles testés rapidement.
- **Sélection avec optimisation** : Sélection des modèles les plus prometteurs et optimisation des hyperparamètres.
- **Analyse des résultats** : Comparaison des scores pour chacun des modèles optimisés, et observation de l'interprétabilité de ceux-ci.
- **Calcul d'un score global** : A partir des métriques étudiées à l'étape précédente.
- **Choix du modèle et enregistrement**

Je vais donc présenter ici les étapes qui m'ont amenées à choisir ce modèle. L'ordre de présentation de ces étapes est arbitraire, car elles sont toutes liées, mais je le pense cohérent pour aborder les différentes notions en lien les unes aux autres.

1.1 Pre-processing

La première chose avant de lancer un modèle est de préparer le dataset. Pour cela, j'ai pris en compte l'EDA précédemment effectuée pour déterminer les étapes :

- Rééquilibrage des classes : On a constaté lors de l'analyse des données qu'il y a environ 90% de prêts accordés pour 10% de refusés. Il faut compenser cela pour que les modèles « voient » autant de positifs que de négatifs.
- Mise à l'échelle des données : Pour certains modèles il faut des données standardisées pour avoir de meilleurs résultats
- Pas d'imputation : Les données ayant été nettoyé précédemment, je sais que je n'ai plus de valeurs manquantes donc cette étape est inutile.



Pour le rééquilibrage, j'ai utilisé le paramètre « class_weight » intégré à certains modèles, et la librairie SMOTE pour les autres modèles, qui permet de faire de l'oversampling (= rajouter des échantillons fictifs des classes sous représentées, basés sur les échantillons existants).

```
# Apply SMOTE oversampling to the training data
smote = SMOTE()
X_train_smt, y_train_smt = smote.fit_resample(X_train, y_train)

# Scaling data
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train_smt)
X_test_scaled = scaler.transform(X_test)

# Train the model on the selected features
model.fit(X_train_scaled, y_train_smt)
```

1.2 Fonction coût métier et métrique d'évaluation

1.2.1 Score métier

Dans le contexte de ce projet, une fonction de coût métier a été définie. Elle a été construite en considérant qu'un faux négatif (FN) est dix fois plus coûteux qu'un faux positif (FP).

La fonction de coût métier est la suivante :

```
job_score = (10 * fnr) + fpr
```

Cette fonction utilise la matrice de confusion pour calculer le taux de faux négatifs (fnr) et le taux de faux positifs (fpr), qui sont ensuite combinés pour obtenir un score métier.

1.2.2 Autres métriques

En plus du score métier, parmi toutes les métriques possible, j'ai choisi de me concentrer sur les suivantes :

- Le ROC AUC (Area Under the « Receiver Operating Characteristic » Curve) : Métrique couramment utilisée pour évaluer la performance des modèles de classification binaire. Il mesure la capacité du modèle à distinguer les classes positives et négatives.
- Le temps de calcul : Il est important d'avoir un modèle qui ait un temps de calcul raisonnable, même si ce n'est pas la métrique qui a le plus de poids dans notre cas. Cette métrique n'a pas été utilisée pour la pré-sélection du modèle, mais dans le choix final.

1.2.3 Score Global

Enfin, **dans la phase d'évaluation des modèles**, j'ai adopté une approche globale pour sélectionner le modèle final en utilisant une fonction de scoring spécifique. Cette fonction procède comme suit :

```
# Normalisation des scores : entre 0 et 1, 1 étant le meilleur score
normalized_time_score = (model['best_time'] - max_time) / (min_time - max_time)
normalized_job_score = 1 - ( model['job_score'] / 10 )

score = normalized_time_score + 2 * roc_score + 2 * normalized_job_score
```

Le modèle final est sélectionné en fonction du score global le plus élevé.

Cette approche permet de prendre en compte à la fois les performances du modèle (ROC AUC), la fonction coût métier (job score) et les contraintes de temps de calcul dans le processus de sélection du modèle final. Cela garantit que le modèle retenu est à la fois performant et opérationnellement efficace.

1.3 Choix du modèle

Pour cela, je procède du plus simple au plus complexe, en mettant en place en amont un modèle de référence pour évaluer le pouvoir prédictif du modèle choisi.

- Modèle de référence :
 - o Dummy Classifier

- Modèles simples :
 - o Logistic Regression
 - o Decision Tree Classifier
 - o Linear Discriminant Analysis
- Modèles ensemblistes :
 - o Gradient Boosting Classifier
 - o XGBoost Classifier
 - o LGBM Classifier
 - o AdaBoost Classifier
 - o Random Forest Classifier

Pour les modèles simple, il a été appliqué le préprocessing tel que décrit dans §1.1. Pour les modèles complexe, l'utilisation du paramètre « class_weight » du modèle a été utilisé.

Pour cette pré-sélection, je me suis basé sur les métriques d'évaluations job_score et roc_auc (cf §1.2) pour déterminer les modèles qui méritent d'être optimisés avant de sélectionner le plus adapté.

Les modèles qui ont retenu mon attention sont :

- Logistic Regression
- XGBoost Classifier
- LGBM Classifier

1.4 Optimisation des modèles

Le processus d'entraînement du modèle consiste en plusieurs étapes clés :

1. **Définition des hyperparamètres** : Pour chaque modèle, une liste d'hyperparamètres a été définie. Pour les modèles ensemblistes, cela comprends la définition du paramètres class_weight comme défini en §1.1.
2. **Définition des pipelines de modèles** : Pour chaque modèles, le pipeline inclus le preprocessing ou non en fonction du modèle, puis le modèle en lui-même.
3. **Recherche sur grille** : Pour chaque modèle configuré, une recherche des meilleurs hyperparamètres est effectuée à l'aide d'une recherche aléatoire (RandomizedSearchCV) ainsi qu'une validation croisée (RepeatedStratifiedKFold). Cela a permis de trouver la combinaison d'hyperparamètres qui maximise le score ROC AUC.
4. **Sauvegarde des métriques** : Les modèles sont évalués sur les données de test pour mesurer leur performance. Les métriques évaluées incluent le score ROC AUC et le score métier défini précédemment.

L'ensemble de ces étapes a permis de sélectionner les modèles les mieux adaptés à la problématique spécifique de prédiction de la solvabilité des clients, en tenant compte des coûts associés aux erreurs de prédiction.

Suite à cette recherche des hyperparamètres optimisés pour les modèles sélectionnés, j'ai utilisé la fonction de calcul du score global (§1.2.3) pour déterminer le modèle qui sera utilisé pour les prédictions du projet.

Le modèle sélectionné par cette méthode est le LGBM Classifier.

2. Les résultats

Récapitulatif des scores des modèles de départs :

Model Name	Roc auc score	Job score	Time
DummyClassifier	0.5	10	11.9s
Logistic Regression	0.744	3.687	16.3s
Decision Tree	0.539	8.41	32.7s
Linear Discriminant Analysys	0.745	3.636	18.4s
Gradient Boosting	0.746	9.794	1.1min
XGBoost	0.729	4.905	53.3s
LightBoost	0.756	4.35	6.0s
Adaboost	0.743	9.88	1.4min
Random Forest	0.726	9.978	1.8min

Récapitulatif des scores des modèles sélectionnés :

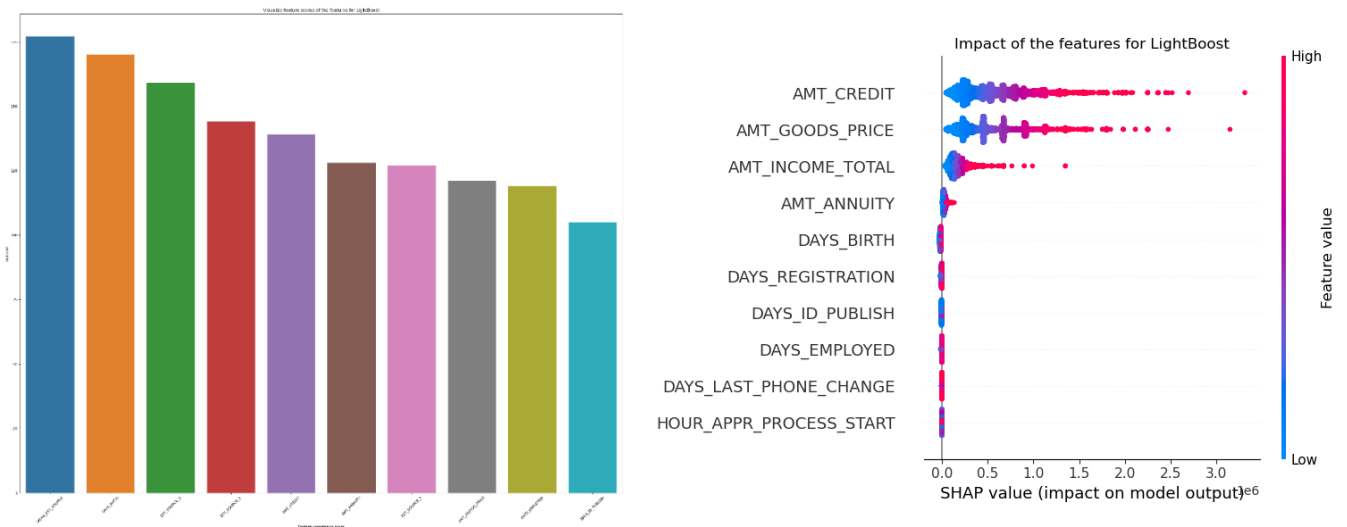
Model Name	Roc auc score	Job score	Time	Score Total
Logistic Regression	0.737	3.84	23.7	2.983
XGBoost	0.756	4.344	16.55	2.706
LightBoost	0.756	4.403	1.543	3.632

3. Interprétabilité du modèle

L'interprétabilité a été étudiée pour les 3 modèles sélectionnés, avant le choix final du modèle. La méthodologie étant la même pour les 3 modèles, je vais présenter ici uniquement le modèle final, à savoir LightBoost.

3.1 Interprétabilité globale

Pour étudier l'interprétabilité globale, j'ai utilisé dans un premier temps la features importance native au modèle, puis l'impact des features via le summary plot de la librairie shap.

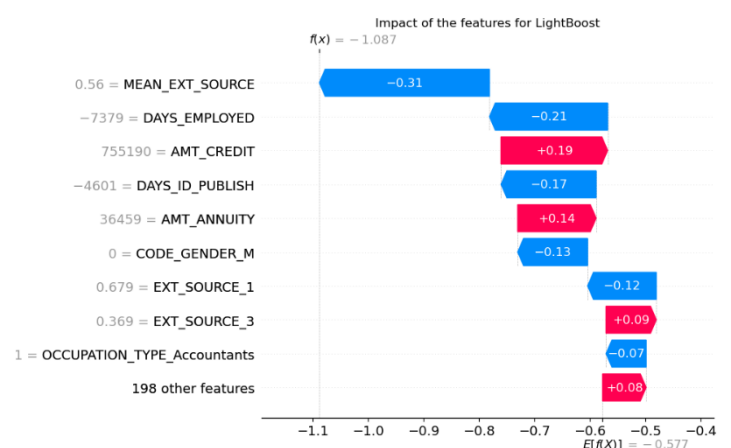


- Les features « EXT_SOURCE » semblent avoir une grande importance, mais ne sont pas présentes dans le top des features du summary plot. Cela peut signifier que leur influence sur les prédictions n'est pas uniforme, et qu'elle est forte que sur quelques observations.
- Outre cette différence, on retrouve les mêmes features en tête, à savoir l'âge, et des informations financières comme le montant du crédit, le revenu, et le patrimoine du client.
- L'impact global des features semble principalement positif (ou faiblement négatif) de par le Summary plot, et la plupart des prédictions sont fortement induites par un petit nombre de features.

3.2 Interprétabilité locale

Pour l'interprétabilité locale, j'ai utilisé le graphique waterfall de la librairie SHAP.

On peut constater que localement, les features qui ont le plus d'impact ne sont pas forcément celles vues globalement. Cela va dépendre des informations propres à chaque client. Ici, c'est majoritairement la valeur de MEAN_EXT_SOURCE qui oriente fortement le client vers l'accord du prêt.



4. Limites et améliorations

Voici quelques limites et améliorations qui ont été identifiées. Cette liste n'est pas exhaustive, mais souligne les points qui me semblent essentiels.

- Automatisation excessive :

La fonction de score global comprends des « angles morts » qu'il est important de mettre en lumière. Le fait d'automatiser intégralement le choix d'un modèle de machine learning et son enregistrement peut présenter des limites, car cela ne prend pas en compte certains aspects spécifiques du domaine ou des caractéristiques des données qui pourraient influencer le choix du modèle optimal. Cependant, c'est un exercice d'apprentissage très intéressant !

- Optimisation des hyperparamètres :

Les choix des hyperparamètres et surtout de leurs valeurs à tester reste arbitraire, et dépendant de la puissance de calcul disponible (et donc du temps de calcul). Il existe probablement de meilleures combinaisons d'hyperparamètres pour chacun des modèles testés.

- Scoring de la GridSearch :

Il aurait été envisageable de définir la stratégie d'évaluation de performance des modèles non pas sur le score ROC AUC, mais sur le job score ou bien de les évaluer sur les deux métriques en même temps. Le choix a été fait d'évaluer les métriques de manière séquentielle uniquement pour des raisons de gain de temps essentiellement.

- Optimisation du feature engineering :

Tout comme pour l'optimisation des hyperparamètres, il est toujours possible d'améliorer la qualité des données d'entrée. Dans l'optique d'améliorer les performances du modèle final, un processus itératif sur le dataset est nécessaire.











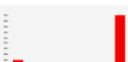
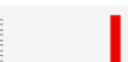








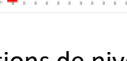
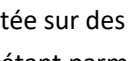
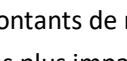
- Centralisation des tâches :

Dans une entreprise disposant de moyen, et dont la mission est aussi essentielle, comme une société financière, il est probablement plus judicieux de séparer le travail d'analyse exploratoire, de détermination du modèle, de déploiement et d'interface utilisateur entre différents acteurs, dont le cœur de métier serait spécifique à une phase particulière du processus, ce qui peut contribuer à une meilleure efficacité et expertise globale de l'entreprise. (Data Analyst, ML Engineer, Front-End Developer, ...)

5. Analyse du Data Drift

L'analyse du Data Drift a été faite avec la librairie evidently, entre les jeux de données train et test, dans leur version « nettoyées », soit avec 207 features. Pour des raisons de temps de calcul, j'ai généré le rapport sur les 10 000 premiers échantillons des deux datasets. Cela implique une relative réduction de la précision de l'analyse.

Une dérive a été détecté par evidently pour 5.8% du dataset, soit 12 features. Voici les features en question :

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
> DEBT_TO_INCOME_RATIO	num			Detected	Wasserstein distance (normed)	0.292271
> INCOME_TO_CREDIT	num			Detected	Wasserstein distance (normed)	0.278425
> AMT_GOODS_PRICE	num			Detected	Wasserstein distance (normed)	0.221456
> AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.218926
> FLAG_DOCUMENT	num			Detected	Jensen-Shannon distance	0.164015
> EXT_SOURCE_1	num			Detected	Wasserstein distance (normed)	0.157719
> NAME_CONTRACT_TYPE_Cash loans	num			Detected	Jensen-Shannon distance	0.151032
> NAME_CONTRACT_TYPE_Revolving loans	num			Detected	Jensen-Shannon distance	0.151032
> AMT_ANNUITY	num			Detected	Wasserstein distance (normed)	0.148694
> FLAG_EMAIL	num			Detected	Jensen-Shannon distance	0.128973
> DAYS_LAST_PHONE_CHANGE	num			Detected	Wasserstein distance (normed)	0.120622
> AMT_INCOME_TOTAL	num			Detected	Wasserstein distance (normed)	0.115021

Parmi celles-ci, on constate notamment les évolutions de niveau de vie et de pouvoir d'achat de la population. De la dérive a été détectée sur des montants de revenus annuels, de crédit demandés, ou encore de patrimoine. Ces features étant parmi les plus importants pour notre modèle, il est nécessaire de déterminer un seuil maximum, afin d'actualiser la modélisation par rapport à des variables comme l'inflation et le PIB.